Portfolio Optimization

Theory and Application

Daniel P. Palomar

The Hong Kong University of Science and Technology

Cambridge University Press

2025

Latest update: October 29, 2026

Portfolio Optimization

This comprehensive guide to the world of financial data modeling and portfolio design is a must-read for anyone looking to understand and apply portfolio optimization in a practical context. It bridges the gap between mathematical formulations and the design of practical numerical algorithms.

It explores a range of methods, from basic time series models to cutting-edge financial graph estimation approaches. The portfolio formulations span from Markowitz's original 1952 mean–variance portfolio to more advanced formulations, including downside risk portfolios, drawdown portfolios, risk parity portfolios, robust portfolios, bootstrapped portfolios, index tracking, pairs trading, and deep-learning portfolios.

Enriched with a remarkable collection of numerical experiments and more than 200 figures, this is a valuable resource for researchers and finance industry practitioners. With slides, R and Python code examples, and exercise solutions available online, it serves as a textbook for portfolio optimization and financial data modeling courses, at advanced undergraduate and graduate level.

Daniel P. Palomar is a Professor at the Hong Kong University of Science and Technology. He is recognized as a EURASIP Fellow, IEEE Fellow, and Fulbright Scholar, and recipient of numerous research awards. His current research focus is on convex optimization applications in signal processing, machine learning, and finance. He is the author of many research articles and books, including *Convex Optimization in Signal Processing and Communications*.

"Professor Palomar's *Portfolio Optimization: Theory and Application* is a remarkable contribution to the field, bridging advanced optimization techniques with real-world portfolio design. Unlike traditional texts, it integrates heavy-tailed modeling, graph-based methods, and robust optimization with a practical, algorithmic focus. This book is an invaluable resource for those seeking a cutting-edge, computationally sound approach to portfolio management."

— Marcos Lopez de Prado OMC PhD, Global Head of Quantitative R&D at Abu Dhabi Investment Authority, and Professor of Practice at Cornell University

"I highly recommend Palomar's book Portfolio Optimization to anyone interested in constructing good portfolios using computational optimization. It collects methods from an enormous literature into one volume, using common notation accessible to any STEM researcher, with clear explanations, discussion, and comparisons of different methods. It is required reading for all of my finance-curious students."

— Stephen Boyd, Stanford University

"An invaluable reference for single period portfolio optimization under heavy tails. Palomar emphasizes the connections between portfolio methods as well as their differences, and explores tools for ameliorating their flaws rather than glossing over them."

— Peter Cotton, author of Microprediction: Building an Open AI Network

"Daniel Palomar's book is a hands-on guide to portfolio optimization at the research frontier. By integrating financial data modeling, code, equations, and real-world data, it bridges theory and practice. A must-read for aspiring data-driven portfolio managers and researchers seeking to stay updated with the latest advancements."

— Kris Boudt, Ghent University, Vrije Universiteit Brussel and Vrije Universiteit Amsterdam

"Dan Palomar's book is a comprehensive treatment of portfolio optimization, covering the complete range from traditional optimization to more sophisticated methods of robust portfolio construction and machine learning algorithms. Directed toward graduate students and quantitative asset managers, any practitioner who builds financial portfolios would be well served by knowing everything in this book."

— Dev Joneja, Chief Risk Officer, ExodusPoint Capital Management

"Daniel P. Palomar's Portfolio Optimization: Theory and Application is a definitive guide at the nexus of financial data modeling and optimal portfolio design. This text is distinguished by its extensive and systematic exploration of a wide array of portfolio strategies, ranging from traditional mean-variance models to state-of-the-art graph-based and deep-learning methods. Palomar masterfully integrates rigorous optimization theory with practical numerical algorithms. This approach renders complex mathematical methods and algorithms accessible to a wide audience and applicable to real-world scenarios. With supplemental resources like slides, R and Python code examples, and exercise solutions available online, this book is indispensable for researchers and practitioners eager to translate theoretical concepts into practical, effective algorithm development."

— Gesualdo Scutari, Purdue University

A mis padres

Contents

Preface xiii

1 Introduction 1
1.1 What is Portfolio Optimization? 2
1.2 The Big Picture 4
1.3 Outline of the Book 5
1.4 Comparison with Existing Books 8
1.5 Reading Guidelines 9
1.6 Notation 10
1.7 Website for the Book 12
1.8 Code Examples 12
References 13

I Financial Data 17

2 Financial Data: Stylized Facts 19
2.1 Stylized Facts 19
2.2 Prices and Returns 20
2.3 Non-Gaussianity: Asymmetry and Heavy Tails 24
2.4 Temporal Structure 29
2.5 Asset Structure 33
2.6 Summary 36
Exercises 37
References 38

3 Financial Data: I.I.D. Modeling 39
3.1 I.I.D. Model 39
3.2 Sample Estimators 40
3.3 Location Estimators 42
3.4 Gaussian ML Estimators 46
3.5 Heavy-Tailed ML Estimators 48
3.6 Prior Information: Shrinkage, Factor Models, and Black–Litterman 58
3.7 Summary 70
Exercises 70
References 73

4 Financial Data: Time Series Modeling 79
4.1 Temporal Structure 79
4.2 Kalman Filter 81
4.3 Mean Modeling 84
4.4 Volatility/Variance Modeling 95
4.5 Summary 105
Exercises 105
References 107

5 Financial Data: Graphs 111
5.1 Graphs 111
5.2 Learning Graphs 115
5.3 Learning Structured Graphs 122
5.4 Learning Heavy-Tailed Graphs 129
5.5 Learning Time-Varying Graphs 132
5.6 Summary 134
Exercises 134
References 137

II Portfolio Optimization 141

6 Portfolio Basics 143
6.1 Fundamentals 143
6.2 Portfolio Constraints 151
6.3 Performance Measures 155
6.4 Heuristic Portfolios 163
6.5 Risk-Based Portfolios 166
6.6 Summary 171
Exercises 172
References 173

7 Modern Portfolio Theory 175
7.1 Mean–Variance Portfolio (MVP) 175
7.2 Maximum Sharpe Ratio Portfolio 183
7.3 Utility-Based Portfolios 187
7.4 Universal Algorithm 191
7.5 Drawbacks 196
7.6 Summary 199
Exercises 199
References 202

8 Portfolio Backtesting 205
8.1 A Typical Backtest 205
8.2 The Seven Sins of Quantitative Investing 207
8.3 The Dangers of Backtesting 214
8.4 Backtesting with Historical Market Data 219
8.5 Backtesting with Synthetic Data 226

8.6 Summary 230
Exercises 231
References 232

9 High-Order Portfolios 235
9.1 Introduction 235
9.2 High-Order Moments 238
9.3 High-Order Portfolio Formulations 245
9.4 Algorithms 252
9.5 Summary 260
Exercises 261
References 262

10 Portfolios with Alternative Risk Measures 267
10.1 Introduction 267
10.2 Alternative Risk Measures 268
10.3 Downside Risk Portfolios 274
10.4 Tail-Based Portfolios 276
10.5 Drawdown Portfolios 280
10.6 Summary 285
Exercises 285
References 289

11 Risk Parity Portfolios 291
11.1 Introduction 291
11.2 From Dollar to Risk Diversification 292
11.3 Risk Contributions 293
11.4 Problem Formulation 295
11.5 Naive Diagonal Formulation 297
11.6 Vanilla Convex Formulations 297
11.7 General Nonconvex Formulations 307
11.8 Summary 316
Exercises 317
References 318

12 Graph-Based Portfolios 321
12.1 Introduction 321
12.2 Hierarchical Clustering and Dendrograms 324
12.3 Hierarchical Clustering-Based Portfolios 327
12.4 Numerical Experiments 340
12.5 Summary 342
Exercises 343
References 345

13 Index Tracking Portfolios 347
13.1 Active vs. Passive Strategies 347
13.2 Sparse Regression 351
13.3 Sparse Index Tracking 355

13.4 Enhanced Index Tracking 364
13.5 Automatic Sparsity Control 368
13.6 Summary 371
Exercises 372
References 376

14 Robust Portfolios 379
14.1 Introduction 380
14.2 Robust Portfolio Optimization 380
14.3 Portfolio Resampling 390
14.4 Summary 397
Exercises 400
References 401

15 Pairs Trading Portfolios 405
15.1 Mean Reversion 405
15.2 Cointegration and Correlation 407
15.3 Pairs Trading 411
15.4 Discovering Cointegrated Pairs 415
15.5 Trading the Spread 423
15.6 Kalman Filtering for Pairs Trading 432
15.7 Statistical Arbitrage 440
15.8 Summary 445
Exercises 447
References 448

16 Deep Learning Portfolios 453
16.1 Machine Learning 454
16.2 Deep Learning 458
16.3 Deep Learning for Portfolio Design 471
16.4 Deep Learning Portfolio Case Studies 477
16.5 Summary 481
Exercises 482
References 485

Appendices 489

***Appendix A* Convex Optimization Theory** 491
A.1 Optimization Problems 492
A.2 Convex Sets 497
A.3 Convex Functions 500
A.4 Convex Optimization Problems 507
A.5 Taxonomy of Convex Problems 514
A.6 Lagrange Duality 522
A.7 Multi-objective Optimization 530
A.8 Summary 534
Exercises 534

References 537

Appendix B **Optimization Algorithms** 539
B.1 Solvers 539
B.2 Gradient Methods 545
B.3 Projected Gradient Methods 548
B.4 Interior-Point Methods 550
B.5 Fractional Programming Methods 557
B.6 Block Coordinate Descent (BCD) 560
B.7 Majorization–Minimization (MM) 563
B.8 Successive Convex Approximation (SCA) 570
B.9 Alternating Direction Method of Multipliers (ADMM) 576
B.10 Numerical Comparison 579
B.11 Summary 581
Exercises 582
References 585

Index 589

Preface

This book delves into the realm of practical portfolio optimization and financial data modeling, encompassing a wide range of formulations and algorithms. The book is not a trading manual. The central theme revolves around optimization, bridging the gap between mathematical formulations and the design of practical numerical algorithms. The text is enriched with an abundance of numerical experiments and a remarkable collection of over 200 figures.

The financial data modeling presented here departs from the conventional Gaussian assumption and adopts more realistic heavy-tailed distributions, exploring a gamut of methods from basic time series models and Kalman filtering techniques to cutting-edge financial graph estimation approaches. The portfolio formulations span from Markowitz's original 1952 mean–variance portfolio and the 1966 maximum Sharpe ratio portfolio to more advanced formulations, such as downside risk portfolios, semi-variance portfolios, CVaR portfolios, drawdown portfolios, risk parity portfolios, Kelly-based portfolios, utility-based portfolios, high-order portfolios, index tracking portfolios, robust portfolios, bootstrapped portfolios, bagged portfolios, graph-based portfolios, pairs trading portfolios, statistical arbitrage portfolios, and deep learning portfolios, among others. The primary focus of this book is on practical algorithms that can be readily implemented on a standard computer.

While numerous textbooks on portfolio design exist, most adopt a traditional approach, primarily concentrating on Markowitz's portfolio. Others may target specific topics, such as robust portfolios, risk parity portfolios, or index tracking. This book aims to encompass all types of portfolios under an optimization framework. Instead of dwelling on deriving closed-form solutions for simplistic formulations, the emphasis is on the convexity analysis of the formulations and the use of mature solvers available in all programming languages, as well as on developing tailored, efficient numerical algorithms. Each portfolio chapter is dedicated to a specific type of portfolio, starting with the mathematical formulation of the problem and culminating in a practical numerical algorithm. To facilitate understanding, the book showcases an extensive array of numerical experiments based on market data.

Birth of This Book

How long has this book been in the making? It's difficult to say, as it depends on how one counts, but it's somewhere between four and eight years. Let's delve into the story of the book . . .

My 2003 Ph.D. topic was on convex optimization methods in signal processing for wireless

communication systems (later, in 2012, I would be named IEEE Fellow for contributions in this area). Around 2008, a couple of years after I became an Assistant Professor at the Hong Kong University of Science and Technology, my interest in wireless communications was fading and I got introduced by serendipity to financial engineering. In 2010, I got tenured at the university and I decided to slowly switch my research area towards my new passion. It was not easy, I had to start from scratch. Surprisingly, the topics of wireless communications and portfolio optimization share a striking resemblance on a mathematical level and in terms of theoretical and practical tools, but that's another story. It wasn't until 2012 that I started timidly publishing research on modeling of financial data. However, the truth is that most of the Ph.D. students in my own research group were reluctant to join me in the exploration of this new direction and preferred to continue their research on wireless communications. They felt safe this way. I felt alone for a number of years. Later, in 2015, my research group finally started to slowly steer into this direction of optimization methods in finance. In 2017, my group joined the open-source software movement by creating packages and libraries for financial-related optimization methods based on our own research papers. That year, to my pleasant surprise, I was invited to teach a portfolio optimization course in the reputable Financial Mathematics M.Sc. program. In 2018, I started teaching the course after having spent a year preparing the course material from scratch. The course slides kept evolving over the years and I always made them available online. Much to my delight, I kept receiving many encouraging emails from practitioners in the financial industry. There seemed to be broad interest in the material I had prepared and it was proving useful to someone out there. And so, around the summer of 2020 I started writing this book. In the early stages, I wasn't sure whether I was actually writing a book or just jotting down some supplementary notes for the course. But somehow the book came to life. The writing period took approximately four long years. Ultimately, I had to put a stop to the endless revisions, or I would never finish the book (thankfully, my wise colleague Emilio Sanvicente reminded me that "perfect is the enemy of good"). During the final revisions of the book in 2024, I received the rewarding news of being named EURASIP Fellow "for contributions to optimization theory and algorithms with applications in communication systems and finance." In another piece of joyful news, my former Ph.D. student, Junxiao Song, now a Principal Researcher at DeepSeek AI (an AI research division of High-Flyer, a Chinese quantitative hedge fund specializing in algorithmic trading), made breakthroughs in deep learning that sent ripples across the entire AI landscape. Overall, the making of this book was a lengthy and arduous journey, marked by a number of difficult personal events, yet always uplifted by the two precious lights in my life, Gisela and Mireia.

Audience

This book is intended for several types of readers:

- undergraduate students, who may focus on basic concepts and practical coding;
- practitioners, who may prefer a stronger emphasis on the practical implementation of algorithms;
- M.Sc. students, who may need to explore both mathematical and coding aspects;

- Ph.D. students, who may wish to delve deeper into the theoretical aspects and explore the provided references further.

This book, along with the slides and code examples available on the companion website, can be used as a textbook for a variety of courses related to portfolio optimization and financial data modeling. Most of the chapters are self-contained, with little dependence on one another, making it easy to select chapters for one-semester or two-semester courses.

For instance, material from this book, together with the slides, code examples, and exercises, have been used in part in the following courses at the Hong Kong University of Science and Technology:

- M.Sc. course *Portfolio Optimization with R* (part of the M.Sc. program on Financial Mathematics);
- M.Sc. course *Optimization in FinTech* (part of the M.Sc. program on FinTech);
- undergraduate course *Data-Driven Portfolio Optimization* (with Python);
- Ph.D. course *Convex Optimization*.

Topics Not Covered

The book considers the modeling of financial data and portfolio design for various types of tradable assets in financial markets, such as stocks, bonds, commodities, currencies, exchange-traded funds (ETFs), and cryptocurrencies. Derivatives, such as options, are not covered; however, many books are available on derivatives.

Multi-asset modeling in this book is approached from a statistical perspective based on heavy-tailed multivariate distributions. The methodology based on copulas is not covered but is standard material in many textbooks.

Multi-period portfolio optimization is not covered in this book. It involves very different mathematical formulations, treatments, and numerical algorithms; although scarce, literature on this topic is available elsewhere.

High-frequency trading based on the limit order book requires a completely different treatment than what is covered in this book.

Additional Resources

The book is supplemented with a variety of additional materials, including slides, sample code, exercises with solutions, and videos. These supplementary resources can be accessed on the companion website at:

portfoliooptimizationbook.com

The citation for this book is:

Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application*.
Cambridge University Press.

Acknowledgments

I will always be grateful to Stephen Boyd, for he introduced me to the wonderful world of convex optimization during my Ph.D. stay at Stanford University in 2001. Since then, it's been a common theme in my research, whether it was applied to wireless communications, data analytics, or financial systems.

Special thanks go to a few colleagues and good friends. Looking back, I realize that I have been a part of all of their weddings and bachelor parties, if any. Francisco Rubio ignited my curiosity for finance back in 2008. Yiyong Feng was the first adventurous Ph.D. student to join me in transitioning from wireless communications to finance. Konstantinos Benidis helped initiate our participation in the open-source software movement. Vinícius de M. Cardoso kindly proofread many of the manuscript chapters, provided aesthetic comments, and helped with Python code. Jasin Machkour also proofread most of the manuscript and provided critical comments.

I would also like to express my gratitude for the help and feedback provided by Dany Cajas, who proofread Chapter 9 on high-order portfolios; Xiwen Wang, who assisted with some numerical experiments in Chapter 9; Vinícius de M. Cardoso, who helped with the plots of financial graphs in Chapter 5; and Jasin Machkour, who provided assistance with the index tracking experiments under false discovery rate in Chapter 13.

A heartfelt "thank you" is extended to my current and former students at the Hong Kong University of Science and Technology who have persevered through the journey of exploring various facets of financial data modeling and portfolio optimization. Together, we have delved into a broad range of topics, gained substantial knowledge, advanced the state of the art, published papers, and developed open-source code. They know who they are, my coauthors, too numerous to list here.

1

Introduction

En un lugar de La Mancha, de cuyo nombre no quiero acordarme . . .

— Miguel de Cervantes Saavedra, *Don Quixote*

Modern portfolio theory started with Harry Markowitz's 1952 seminal paper "Portfolio Selection" (Markowitz, 1952), for which he would later receive the Nobel Prize in Economic Sciences[1] in 1990. He put forth the idea that risk-averse investors should optimize their portfolio based on a combination of two objectives: expected return and risk. Until today, that idea has remained central to portfolio optimization. In practice, however, the vanilla Markowitz portfolio formulation does not perform as anticipated. Consequently, most practitioners either combine it with various heuristics or refrain from using it altogether.

Over the past 70 years, researchers and practitioners have reconsidered the Markowitz portfolio formulation, leading to numerous variations, enhancements, and alternatives. These include robust optimization methods, alternative risk measures, regularization through sparsity, improved covariance matrix estimators via random matrix theory, robust estimators for heavy tails, factor models, mean models, volatility clustering models, risk parity formulations, and more.

This book explores practical financial data modeling and portfolio optimization, covering a wide range of variations and extensions. It systematically starts with mathematical formulations and proceeds to develop practical numerical algorithms, supplemented with code examples to enhance understanding.

- The financial data modeling considered herein moves away from the unrealistic Gaussian assumption and delves into more realistic models based on heavy-tailed distributions. It encompasses an array of topics, ranging from basic time series models, making extensive use of Kalman filtering methods, to state-of-the-art techniques for estimating financial graphs.
- The portfolio formulations covered in this book span a wide range, from the original

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

[1] To be exact, what is usually referred to as the Nobel Prize in Economic Sciences is actually the Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel.

1952 Markowitz's mean–variance portfolio and 1966 maximum Sharpe ratio portfolio, to more sophisticated formulations such as Kelly-based portfolios, utility-based portfolios, high-order portfolios, downside risk portfolios, semi-variance portfolios, CVaR portfolios, drawdown portfolios, risk parity portfolios, graph-based portfolios, index tracking portfolios, robust portfolios, bootstrapped portfolios, bagged portfolios, pairs trading portfolios, statistical arbitrage portfolios, and deep learning portfolios, among others.

The primary focus and central theme of this book is on practical algorithms for portfolio formulations that can be effortlessly executed on a standard computer.

1.1 What is Portfolio Optimization?

Suppose you observe a random variable X with mean $\mu = \mathbb{E}[X]$ and variance $\sigma^2 = \mathbb{E}[(X-\mu)^2]$; for example, a normal (or Gaussian) random variable $X \sim \mathcal{N}(\mu, \sigma^2)$. The mean μ is the value you expect to obtain, whereas the variance σ^2 gives the variability or randomness around that value. The ratio μ/σ gives a measure of the deterministic-to-random balance. In finance, X may represent the *return* of an investment and the ratio μ/σ is called *Sharpe ratio*. In signal processing, it is more common to use the *signal-to-noise ratio (SNR)* measured in units of power and defined as μ^2/σ^2.

Now suppose that for each time t, a different (independent) value of the random variable is observed (called a random process or random time series): $X_t \sim \mathcal{N}(\mu, \sigma^2)$. In finance, these represent the returns of the investment, and the cumulative return is the accumulation of the previous returns, which reflects the accumulated wealth of the investment. Figure 1.1 shows a realization of such return random variables as well as the cumulative returns.

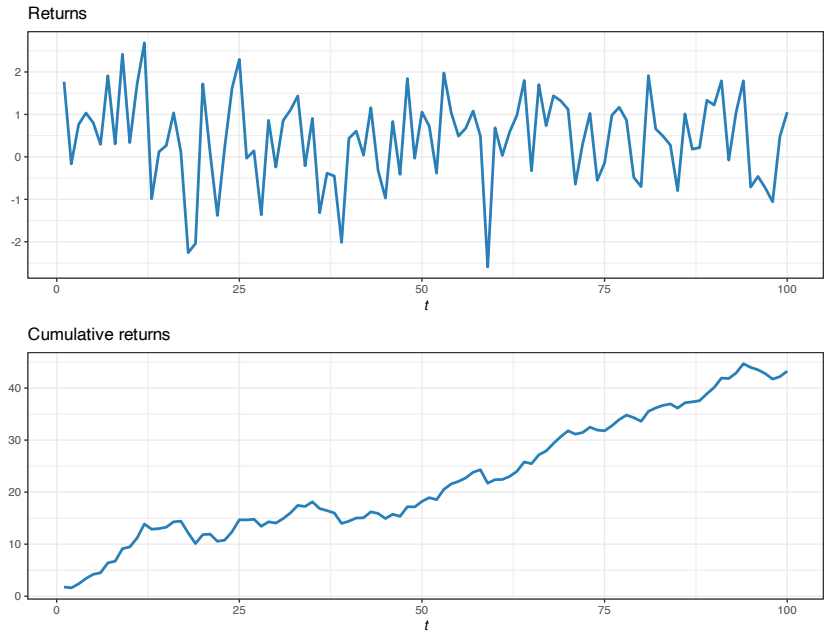


Figure 1.1 Illustration of random returns and cumulative returns.

The evolution of the cumulative returns or wealth over time, albeit random, is strongly determined by the value of the Sharpe ratio, μ/σ, as illustrated in Figure 1.2 for high and low values. If the Sharpe ratio is high, the cumulative return will have some fluctuations but with a consistent growth. On the other hand, if the ratio is low, the drift becomes small relative to the volatility and one may even end up losing everything, leading to bankruptcy. Note that Figure 1.2 assumes that the entire capital is invested in the asset; in practice, investors choose how much to invest, that is, the leverage, precisely to keep the growth of wealth positive and reduce the risk of ruin.

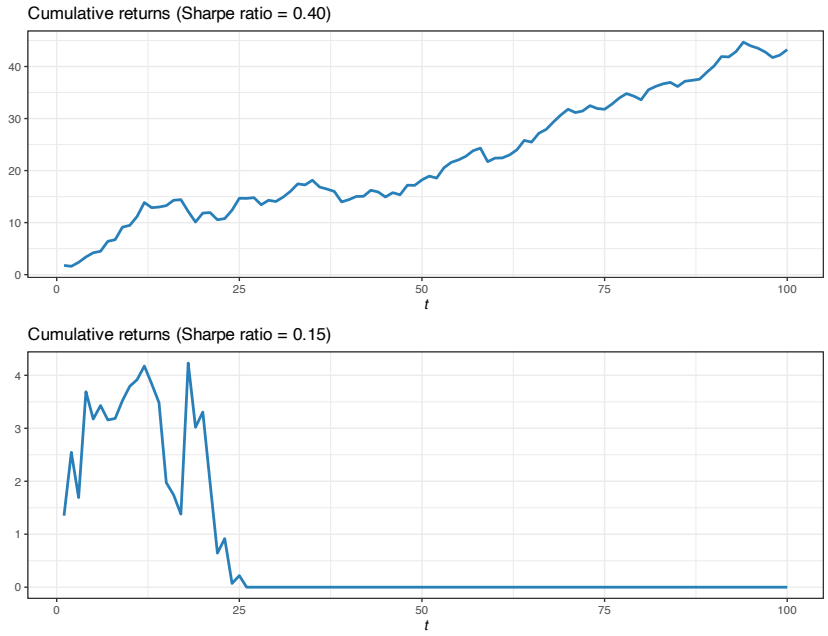


Figure 1.2 Illustration of cumulative returns with different values of Sharpe ratio.

What can an investor do to improve the cumulative return? While the random nature of the investment assets themselves cannot be changed, there are at least two dimensions that can be potentially exploited: the temporal dimension and the asset dimension.

- *Temporal dimension*: It may be the case that the distribution of the random return X_t changes with time, leading to time-varying μ_t and σ_t^2. In that case, a smart investor will adapt the size of the investment to the current value of μ_t/σ_t. In order to do that, one needs to develop an appropriate time series model, that is, a data model at time t given the past observations. This is called *data modeling* and it is explored in Part I of this book.

- *Asset dimension*: In general, an investor has a choice of N potential assets in which to invest, with returns X_i for $i = 1, \ldots, N$. Suppose they are all independent and identically distributed (i.i.d.): $X_i \sim \mathcal{N}(\mu, \sigma^2)$. It follows from basic probability that the average of such returns, $\frac{1}{N}\sum_{i=1}^{N} X_i$, preserves the mean μ but enjoys a reduced variance of σ^2/N. In finance, this average is achieved by distributing the capital equally over the N assets (the popular $1/N$ portfolio precisely implements this). In practice, however, the $1/N$ factor

in the reduction of the variance cannot be achieved because the random returns X_i are correlated among the assets, that is, the assumption of uncorrelation does not hold. Over the decades, academics and practitioners have proposed a multitude of ways to properly allocate the capital, as opposed to the baseline $1/N$ allocation, in order to try to circumvent the inherent correlation of the assets and minimize the risk or variance. This is called *portfolio optimization* (also known as *portfolio allocation* or *portfolio design*) and it is covered in detail in Part II of this book.

1.2 The Big Picture

The two main components for portfolio design are *data modeling* and *portfolio optimization*. Figure 1.3 illustrates these two building blocks for the case of mean–variance portfolios (i.e., based on the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$) to produce the optimal portfolio weights $\boldsymbol{w}$.

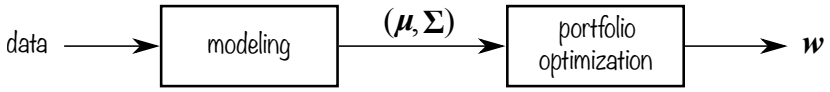


Figure 1.3 Block diagram of data modeling and portfolio optimization.

Part I of this book examines the data modeling component in Figure 1.3. The main purpose of this block is to characterize the statistical distribution of future returns, primarily in terms of the first- and second-order moments, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, which will be utilized by the portfolio optimization block later on.

Part II fully explores a wide variety of formulations for the portfolio optimization component in Figure 1.3. These portfolio formulations can be classified according to different criteria leading to a diverse *taxonomy of portfolios* as follows.

- Taxonomy according to the data used:
  - *second-order portfolios*: portfolios based on the mean and the variance, such as Markowitz mean–variance portfolio, maximum Sharpe ratio portfolio, index tracking portfolios, and volatility-based risk parity portfolios;
  - *high-order portfolios*: portfolios based directly on high-order moments as well as approximations of utility-based portfolios; and
  - *raw-data portfolios*: these include portfolios that require the raw data, such as downside risk portfolios, semi-variance portfolios, conditional variance-at-risk (CVaR) portfolios, drawdown portfolios, graph-based portfolios, and deep learning portfolios.
- Taxonomy according to the view on the *efficient-market hypothesis*:[2]
  - *active portfolios*: most of the portfolio formulations that attempt to beat the market; and
  - *passive portfolios*: index tracking portfolios which simply track the market, avoiding frequent portfolio rebalancing.

[2] The efficient-market hypothesis states that asset prices reflect all information and, therefore, it should be impossible to outperform the overall market through expert stock selection or market timing.

- Taxonomy according to the myopic nature of the portfolio formulation:
  - *single period portfolios*: most of the formulations considered here are based on a single step into the future; and
  - *multi-period portfolios*: more involved formulations that consider several steps into the future so that the long-term effect of current actions is better taken into account; this is not covered in this book, see (Boyd et al., 2017) for a monograph on multi-period portfolio optimization.

1.3 Outline of the Book

This book is organized into two main parts, comprising a total of 16 chapters, along with two appendices at the end. The content of each of the chapters is outlined next.

Part I. *Financial Data*: This part focuses on financial data modeling, which is a necessary component before the portfolio design.

- Chapter 2 discusses *stylized facts* in financial data. These unique characteristics differentiate financial data from other types of data. Some of these characteristics include lack of stationarity, volatility clustering, heavy-tailed distributions, and strong asset correlation. This chapter provides a concise and visual overview of these stylized facts to help readers better understand and analyze financial data.

- Chapter 3 focuses on *i.i.d. modeling* in financial data. Although the i.i.d. model is a simplistic approximation, it is still widely used in practice. However, challenges arise due to non-Gaussian distributions and noise, which are often ignored in financial literature. To address these challenges, robust and heavy-tailed estimators for the mean vector and the covariance matrix are necessary, and this chapter provides detailed explanations for these estimators. Furthermore, incorporating prior information through techniques such as shrinkage, factor modeling, and Black–Litterman fusion can significantly improve the accuracy of estimates. Due to the breadth of topics covered in this chapter, the length is rather long, but it provides readers with a comprehensive understanding of i.i.d. modeling for financial data.

- Chapter 4 explores the application of *time series models* to financial data to capture temporal dependencies for both mean modeling and variance modeling. While mean models provide debatable improvement over the i.i.d. approach, variance models, including GARCH-related models and stochastic volatility models, have been shown to be effective in capturing the volatility of financial data (the latter showing improved results but at a higher computational cost). This chapter presents a unified modeling approach through state-space modeling with special emphasis on the use of the efficient Kalman filter, which notably allows the approximation of stochastic volatility models with low computational cost.

- Chapter 5 focuses on *financial graphs* and their applications in financial data analysis. While graphical modeling of financial data originated in 1999, many methods have since been proposed. Among these methods, sparse Gaussian models are suitable for providing basic insights, low-rank formulations can be used to cluster assets, and heavy-tailed models

are appropriate for accounting for non-Gaussian data. Graph-based techniques can provide valuable visual and analytical tools for financial data analysis. This chapter provides an overview of cutting-edge techniques for graph modeling of financial assets, allowing readers to gain a deeper understanding of the applications and benefits of financial graphs in data analysis.

Part II. *Portfolio Optimization*: This part contains a wide range of chapters covering various portfolio formulations with corresponding algorithms and examples.

- Chapter 6 provides a comprehensive introduction to *portfolio basics*. The chapter covers fundamental topics such as portfolio notation, cumulative return calculation, transaction costs, portfolio rebalancing, practical constraints, measures of performance, simple heuristic portfolios, and basic risk-based portfolios. While the chapter covers the basics, it also includes an interesting nugget on the interpretation of the heuristic quintile portfolio, widely used by practitioners, as a formally derived robust portfolio. This chapter serves as an excellent starting point for readers new to portfolio management, providing them with the foundational knowledge necessary to understand and build portfolios.

- Chapter 7 delves into the topic of *modern portfolio theory*, which is the main focus of the majority of textbooks on portfolio design. In this book, this chapter serves as a starting point for exploring a wide range of different portfolio formulations. The chapter begins with an introduction to the basic mean–variance portfolio and then moves on to the often-ignored maximum Sharpe ratio portfolio, for which several practical numerical algorithms are presented in detail (such as bisection, Dinkelbach, and Schaible transform-based methods). The Kelly portfolio and utility-based portfolios are also introduced. The chapter concludes with a discussion of a recently proposed universal algorithm that can be utilized to solve portfolios based on any trade-off between the mean and variance. Overall, this chapter provides readers with a comprehensive understanding of modern portfolio theory and its practical applications.

- Chapter 8 focuses on *portfolio backtesting*, which is essential in strategy evaluation. Many biases, such as overfitting, can invalidate backtesting results, making it a challenging task. As a consequence, published backtests should not be trusted blindly. This chapter delves into the common pitfalls and dangers of backtesting, which are often ignored in textbooks, and puts forward the approach of multiple randomized backtests to help mitigate risks. The chapter also discusses the benefits of stress testing with resampled data to complement the backtesting results. By providing readers with a comprehensive understanding of the challenges of backtesting and suggesting practical solutions to overcome them, this chapter serves as an essential guide for portfolio assessment.

- Chapter 9 explores *high-order portfolios*, which introduce high-order moments in the mean–variance formulation. This idea dates back to the beginning of modern portfolio theory, but until recently it was impractical due to difficulties in parameter estimation, excessive memory requirements, and the complexity of optimization methods for a realistic number of assets. This chapter covers all the basics of high-order portfolios and introduces recent advances that make this approach practical.

- Chapter 10 considers *portfolios with alternative measures of risk*. While variance is the

most commonly used measure of risk in portfolio optimization, many advanced risk measures, such as downside risk, semi-variance, CVaR, and drawdown, can also be incorporated. These measures can be formulated in convex form, allowing for the use of efficient algorithms. This chapter provides an overview of these sophisticated alternatives to Markowitz's seminal mean–variance formulation.

- Chapter 11 presents *risk parity portfolios*, which aim to diversify risk allocation beyond equal capital allocation. These portfolios were proposed by practitioners and rely on using granular asset risk contributions rather than overall portfolio risk. This chapter presents risk parity portfolios progressively, starting from a naive diagonal formulation and progressing to sophisticated convex and nonconvex formulations. It also covers a wide range of numerical algorithms, including newly proposed techniques.

- Chapter 12 gives an overview of *graph-based portfolios*, which utilize graphical representations of asset relationships learned from data to improve the portfolio design. Graph-based portfolios enable hierarchical clustering and novel formulations that account for asset interconnectivity, enhancing portfolio construction. This chapter provides a comprehensive overview of all existing graph-based portfolios, presenting a unified view of the different approaches.

- Chapter 13 covers *index tracking portfolios*, which are designed to mimic an index under the assumption that the market is efficient and cannot be beaten. Sparse index tracking further improves this approach by using few assets, posing a sparse regression problem. This chapter provides a state-of-the-art overview of the existing methodologies and introduces new formulations for index tracking portfolios. It also includes a cutting-edge algorithm that automatically selects the right level of sparsity, making index tracking more efficient and effective.

- Chapter 14 gives an overview of *robust portfolios*, which aim to address the inevitable parameter estimation errors that can lead to meaningless or catastrophic results if ignored. While optimal portfolio solutions may seem ideal in theory, practical implementation requires techniques like robust optimization and resampling methods. This chapter covers these standard techniques, providing readers with a comprehensive understanding of robust portfolios and how to optimize them.

- Chapter 15 explores *pairs trading* or *statistical arbitrage portfolios*, which are market-neutral strategies designed to be orthogonal to the market trend. These strategies trade on the oscillations among different assets, making them a popular technique in advanced portfolio management. This chapter provides an overview of the basics of pairs trading and statistical arbitrage, as well as exploring the more sophisticated use of Kalman filtering.

- Chapter 16 presents the concept of *deep learning portfolios*, which utilize deep learning techniques to analyze financial time series data and optimize portfolios. While deep learning has revolutionized fields like natural language processing and computer vision, its potential in finance remains uncertain due to challenges such as limited availability of nonstationary data and the weakness of the signal buried in noise. This chapter provides a standalone account of deep learning and the current efforts in the financial arena, acknowledging the risk of becoming quickly obsolete but still providing a good starting point.

Appendices A and B. *Preliminaries on Optimization*: This final part provides an overview of basic concepts in optimization theory (Appendix A) and a concise account of practical algorithms (Appendix B) used throughout the book.

1.4 Comparison with Existing Books

The financial literature on data modeling and portfolio design is extensive and diverse. This book aims to provide a unique perspective on these topics, and it is instructive to compare it with some of the existing textbooks.

- *Financial data modeling*: Many excellent textbooks cover financial data modeling, such as Campbell et al. (1997), Meucci (2005), Tsay (2010), Ruppert and Matteson (2015), Lütkepohl (2007), Tsay (2013), Fabozzi et al. (2007), Fabozzi et al. (2010), and Feng and Palomar (2016). In this book, Chapters 3 and 4 provide a succinct overview of i.i.d. models and models with temporal structure, respectively. Particular emphasis is placed on heavy-tailed models and estimators (as opposed to the more traditional methods based on the Gaussian assumption), stochastic volatility models (usually not receiving their deserved attention), and the use of state-space models with Kalman filtering as a unified approach with efficient algorithms.

- *Modern portfolio theory*: Traditional books that focus primarily on portfolio foundations and mean–variance portfolios include Grinold and Kahn (2000), Meucci (2005), Cornuejols and Tütüncü (2006), Fabozzi et al. (2007), Prigent (2007), Michaud and Michaud (2008), Bacon (2008), and Fabozzi et al. (2010). In this book, Chapters 6 and 7 cover this material with an optimization perspective, including utility-based portfolios, a recent derivation of the otherwise heuristic quintile portfolio as a robust solution, and particularly delving in detail into the nonconvex formulation of the maximum Sharpe ratio portfolio. It also provides a recently proposed universal algorithm for all these portfolios based on different trade-offs of the mean and variance.

- *Risk parity portfolios*: Roncalli's book (Roncalli, 2013) provides a detailed mathematical treatment (see also Feng and Palomar (2016)), while Qian's book (Qian, 2016) covers the fundamentals. In this book, Chapter 11 covers risk parity portfolios from an optimization perspective, progressively covering the naive solution, the vanilla convex formulations, and the more practical and general nonconvex formulations, with emphasis on the numerical algorithms.

- *Backtesting*: López de Prado's book (López de Prado, 2018) covers backtesting and its dangers in great detail from the perspective of machine learning, while Pardo (2008) focuses on the walk-forward backtest. In this book, Chapter 8 explores the many dangers of backtesting and the different forms of executing backtesting based on market data, as well as synthetic data, with abundant figures.

- *Index tracking*: The topic of index tracking is treated in detail in Prigent (2007) and Benidis et al. (2018), with shorter treatments in Cornuejols and Tütüncü (2006) and Feng and Palomar (2016). In this book, Chapter 13 provides a concise yet broad state-of-the-art

exposure, offering new formulations and a cutting-edge algorithm that automatically selects the right level of sparsity.

- *Robust portfolios*: Robust optimization is widely explored within the context of portfolio design, with standard references including Fabozzi et al. (2007) and Cornuejols and Tütüncü (2006) (see also Feng and Palomar (2016)). In this book, Chapter 14 gives a concise presentation of these techniques for obtaining robust portfolios with illustrative numerical experiments.

- *Pairs trading*: The standard reference to this topic is Vidyamurthy (2004); see also Feng and Palomar (2016). In this book, Chapter 15 provides full coverage of the basics and presents a more sophisticated use of the Kalman filter for better adaptability over time.

- *High-frequency trading*: High-frequency data and trading based on the limit order book require a completely different treatment than what is covered in this book. Some key references include Abergel et al. (2016), Lehalle and Laruelle (2018), Bouchaud et al. (2018), and Kissell (2020).

- *Machine learning in finance*: Recent textbooks that give a broad account of the use of machine learning in financial systems include López de Prado (2018) and Dixon et al. (2020). In this book, Chapter 16 briefly discusses machine learning and deep learning techniques in the context of portfolio design.

1.5 Reading Guidelines

This book has been written under the premise that each chapter can be read independently. For example, a reader who is already familiar with portfolio optimization can jump directly to Chapter 16 on deep learning portfolios or to Chapter 15 on pairs trading.

Some suggested ways to read the book include the following approaches:

- A "reader in a rush" can go directly to Chapter 6 for portfolio basics and Chapter 7 for modern portfolio theory, perhaps also taking a quick look at Chapter 2 on stylized facts of financial data, and then jump to any other chapter, for example Chapter 14 on robust portfolios or Chapter 9 on high-order portfolios.

- A "reader with a bit more time," apart from the basic Chapters 2, 6, and 7, could also read Chapter 3 on i.i.d. data modeling and Chapter 8 on portfolio backtesting to get a better grasp of the fundamentals.

- For full coverage of all the different portfolio designs, a reader can go over any chapter in Part II; that is, apart from the fundamental Chapters 6–8, one can explore (in any particular order):

  – high-order portfolios (Chapter 9);
  – portfolios with alternative risk measures (Chapter 10);
  – risk parity portfolios (Chapter 11);
  – graph-based portfolios (Chapter 12);
  – index tracking portfolios (Chapter 13);

  - robust portfolios (Chapter 14);
  - pairs trading or statistical arbitrage portfolios (Chapter 15); and
  - deep learning portfolios (Chapter 16).

- To complete the financial data modeling, one should go over all the chapters in Part I: apart from Chapters 2 and 3; Chapter 4 covers time series modeling, and Chapter 5 explores the more recent topic of graph modeling of financial assets.

- In order to gain a more solid understanding of the portfolio optimization formulations and algorithms, a reader may want to go over Appendices A and B, that is, the basics of convex optimization theory in Appendix A and optimization algorithms in Appendix B.

1.6 Notation

Notation differs depending on the research area and on the personal taste of the author. This book mainly follows the notation widely accepted in the statistics, signal processing, and operations research communities.

To differentiate the dimensionality of quantities we employ lowercase for scalars, boldface lowercase for (column) vectors, and boldface uppercase for matrices, for example, x, $\boldsymbol{x}$, and $\boldsymbol{X}$, respectively. The ith entry of vector $\boldsymbol{x}$ is denoted by x_i and the (i, j)th element of matrix $\boldsymbol{X}$ by $X_{i,j}$. The elementwise product (also termed the Hadamard product) and elementwise division are denoted by $\odot$ and $\oslash$, respectively, e.g., $\boldsymbol{x} \odot \boldsymbol{y}$ and $\boldsymbol{x} \oslash \boldsymbol{y}$ ($\boldsymbol{x}/\boldsymbol{y}$ abusing notation); similarly, the Kronecker product is denoted by $\otimes$. The transpose of a vector $\boldsymbol{x}$ or a matrix $\boldsymbol{X}$ are denoted by $\boldsymbol{x}^\mathsf{T}$ and $\boldsymbol{X}^\mathsf{T}$, respectively. The inverse, trace, and determinant of matrix $\boldsymbol{X}$ are denoted by $\boldsymbol{X}^{-1}$, $\mathrm{Tr}(\boldsymbol{X})$, and $|\boldsymbol{X}|$ (or $\det(\boldsymbol{X})$), respectively. The norm of a vector is written as $\|\boldsymbol{x}\|$, which can be further specified as the ℓ_2-norm $\|\boldsymbol{x}\|_2$ (also termed the Euclidean norm), the ℓ_1-norm $\|\boldsymbol{x}\|_1$, and the ℓ_∞-norm $\|\boldsymbol{x}\|_\infty$. The operator $(\boldsymbol{x})^+$ denotes the projection onto the nonnegative orthant, that is, $(\boldsymbol{x})^+ \triangleq \max(\boldsymbol{0}, \boldsymbol{x})$. We denote by $\boldsymbol{I}$ the identity matrix of appropriate dimensions.

For random variables, $\Pr[\cdot]$ denotes probability, and the operators $\mathbb{E}[\cdot]$, $\mathrm{Std}[\cdot]$, $\mathrm{Var}[\cdot]$, and $\mathrm{Cov}[\cdot]$ denote expected value, standard deviation, variance, and covariance matrix, respectively.

The set of real numbers is denoted by $\mathbb{R}$ (nonnegative real numbers by $\mathbb{R}_+$ and positive real numbers by $\mathbb{R}_{++}$). The set of $m \times n$ matrices is denoted by $\mathbb{R}^{m \times n}$, the set of symmetric $n \times n$ matrices by $\mathbb{S}^n$, and the set of positive semidefinite $n \times n$ matrices by $\mathbb{S}^n_+$. By $\boldsymbol{a} \geq \boldsymbol{b}$ we denote elementwise inequality (i.e., $a_i \geq b_i$). The matrix inequalities $\boldsymbol{A} \succeq \boldsymbol{B}$ and $\boldsymbol{A} \succ \boldsymbol{B}$ denote that $\boldsymbol{A} - \boldsymbol{B}$ is positive semidefinite and positive definite, respectively. The indicator function is denoted by $1\{\cdot\}$ or $I(\cdot)$.

Table 1.1 lists the most common abbreviations used throughout the book, and Table 1.2 provides some key financial mathematical symbols.

Table 1.1 Common abbreviations used in the book.

Abbreviation	Meaning
AI	Artificial intelligence
AR	Autoregressive
ARCH	Autoregressive conditional heteroskedasticity
ARIMA	Autoregressive integrated moving average
ARMA	Autoregressive moving average
B&H portfolio	Buy and hold portfolio
BCD	Block coordinate descent
CAPM	Capital asset pricing model
CCC	Constant conditional correlation
CP	Conic problem/program
CVaR	Conditional value-at-risk
DCC	Dynamic conditional correlation
DD	Drawdown
DL	Deep learning
DR	Downside risk
ES	Expected shortfall
EWMA	Exponentially weighted moving average
EWP	Equally weighted portfolio (a.k.a. $1/N$ portfolio)
FP	Fractional problem/program
FX	Foreign exchange
GARCH	Generalized autoregressive conditional heteroskedasticity
GICS	Global Industry Classification Standard
GMRP	Global maximum return portfolio
GMVP	Global minimum variance portfolio
GP	Geometric problem/program
HRP	Hierarchical risk parity
i.i.d.	independent and identically distributed
IPM	Interior-point method
IVarP	Inverse variance portfolio
IVolP	Inverse volatility portfolio
LFP	Linear fractional problem/program
LP	Linear problem/program
LPM	Lower partial moment
LS	Least squares
MA	Moving average
MDecP	Maximum decorrelation portfolio
MDivP	Most diversified portfolio
ML	Maximum likelihood or machine learning (depending on context)
MM	Majorization–minimization
MSRP	Maximum Sharpe ratio portfolio
MVolP	Mean–volatility portfolio
MVP	Mean–variance portfolio
MVSK	Mean–variance–skewness–kurtosis
NAV	Net asset value
P&L	Profit and loss
QCQP	Quadratically–constrained quadratic problem/program
QP	Quadratic problem/program
QuintP	Quintile portfolio
RPP	Risk parity portfolio

Table 1.1 Common abbreviations used in the book. *(continued)*

Abbreviation	Meaning
S&P 500	Standard & Poor's 500
SCA	Successive convex approximation
SDP	Semidefinite problem/program
SOCP	Second-order cone problem/program
SR	Sharpe ratio
SV	Stochastic volatility
TE	Tracking error
VaR	Value-at-risk
VARMA	Vector autoregressive moving average
VECM	Vector error correction model

Table 1.2 Mathematical notation used in the book.

Term	Meaning
$\boldsymbol{w}$	Normalized portfolio weight vector
$\boldsymbol{w}^{\text{cap}}$	Portfolio capital allocation vector (e.g., in units of US dollar)
$\boldsymbol{w}^{\text{units}}$	Portfolio unit allocation vector (e.g., in units of shares for stocks)
$\boldsymbol{p}_t$	Price vector of assets at time t
$\boldsymbol{y}_t$	Log-price vector of assets at time t
$\boldsymbol{r}_t$ $(\boldsymbol{x}_t)$	Return vector of assets at time t (linear or log-returns, depending on context)
$\boldsymbol{r}_t^{\text{lin}}$	Linear returns vector of assets at time t
$\boldsymbol{r}_t^{\text{log}}$	Log-returns vector of assets at time t
$\boldsymbol{\mu}_t$	Vector of expected value of returns $\boldsymbol{r}_t$
$\boldsymbol{\Sigma}_t$	Covariance matrix of returns $\boldsymbol{r}_t$
N	Number of financial assets in the considered universe
T	Number of temporal observations $t = 1, \ldots, T$
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Normal or Gaussian multivariate distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

1.7 Website for the Book

The book is supplemented with a variety of additional materials, including slides, sample code, exercises with solutions, and videos. These supplementary resources can be accessed on the companion website at:

portfoliooptimizationbook.com

1.8 Code Examples

This book is supplemented with a large number of code examples in R and Python that can reproduce all the figures in the book. These supplementary resources are available on the companion website for the book.

Generally speaking, the resolution of all the portfolio optimization formulations covered in the book can be approached in a variety of ways, namely:

- Use a software package or library specifically designed to optimize portfolios under a wide variety of formulations and constraints. Examples include the popular R package `fPortfolio` (Wuertz et al., 2023) and the Python packages `skfolio` (Delatte et al., 2026; Nicolini et al., 2025), `Riskfolio-Lib` (Cajas, 2026), and `PyPortfolioOpt` (Martin, 2021).
- Utilize a modeling framework like CVX, which automatically calls upon a solver behind the scenes, available for programming languages including Python, R, and Julia (Fu et al., 2020, 2022; Grant & Boyd, 2008, 2014).
- Directly invoke an appropriate solver.
- Develop ad hoc efficient algorithms for specific formulations, as done in the packages developed by the ConvexFi group.[3]

References

Abergel, F., Anane, M., Chakraborti, A., Jedidi, A., & Toke, I. M. (2016). *Limit Order Books*. Cambridge University Press.

Bacon, C. (2008). *Practical Portfolio Performance Measurement and Attribution* (2nd ed.). John Wiley & Sons.

Benidis, K., Feng, Y., & Palomar, D. P. (2018). Optimization methods for financial index tracking: From theory to practice. *Foundations and Trends in Optimization, Now Publishers*, *3*(3), 171–279.

Bouchaud, J.-P., Bonart, J., Donier, J., & Gould, M. (2018). *Trades, Quotes and Prices: Financial Markets Under the Microscope*. Cambridge University Press.

Boyd, S., Busseti, E., Diamond, S., Kahn, R., Koh, K., Nystrup, P., & Speth, J. (2017). Multi-period trading via convex optimization. *Foundations and Trends in Optimization, Now Publishers*, *3*(1), 1–76.

Cajas, D. (2026). Riskfolio-Lib [Python library]. https://github.com/dcajasn/Riskfolio-Lib

Campbell, J. Y., Lo, A. W., & MacKinlay, A. C. (1997). *The Econometrics of Financial Markets*. Princeton University Press.

Cornuejols, G., & Tütüncü, R. (2006). *Optimization Methods in Finance*. Cambridge University Press.

Delatte, H., Nicolini, C., & Manzi, M. (2026). skfolio [Computer software]. https://doi.org/10.5281/zenodo.16148630

Dixon, M. F., Halperin, I., & Bilokon, P. (2020). *Machine Learning in Finance*. Springer.

Fabozzi, F. J., Focardi, S. M., & Kolm, P. N. (2010). *Quantitative Equity Investing: Techniques and Strategies*. John Wiley & Sons.

[3] Convex Optimization in Finance group: https://github.com/convexfi

Fabozzi, F. J., Kolm, P. N., Pachamanova, D. A., & Focardi, S. M. (2007). *Robust Portfolio Optimization and Management*. John Wiley & Sons.

Feng, Y., & Palomar, D. P. (2016). A signal processing perspective on financial engineering. *Foundations and Trends in Signal Processing, Now Publishers*, *9*(1–2), 1–231.

Fu, A., Narasimhan, B., & Boyd, S. (2020). CVXR: An R package for disciplined convex optimization. *Journal of Statistical Software*, *94*(14), 1–34.

Fu, A., Narasimhan, B., Kang, D. W., Diamond, S., Miller, J., Boyd, S., & Rosenfield, P. K. (2022). *CVXR: Disciplined Convex Optimization* [R package]. https://cran.r-project.org/package=CVXR

Grant, M., & Boyd, S. (2008). Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, & H. Kimura (Eds.), *Recent Advances in Learning and Control* (pp. 95–110). Springer.

Grant, M., & Boyd, S. (2014). *CVX: Matlab Software for Disciplined Convex Programming*. http://cvxr.com/cvx

Grinold, R. C., & Kahn, R. N. (2000). *Active Portfolio Management* (2nd ed.). McGraw Hill.

Kissell, R. (2020). *Algorithmic Trading Methods: Applications using Advanced Statistics, Optimization, and Machine Learning Techniques* (2nd ed.). Academic Press.

Lehalle, C.-A., & Laruelle, S. (Eds.). (2018). *Market Microstructure in Practice* (2nd ed.). World Scientific Publishing Co. Pte. Ltd.

López de Prado, M. (2018). *Advances in Financial Machine Learning*. John Wiley & Sons.

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer.

Markowitz, H. M. (1952). Portfolio selection. *The Journal of Finance*, *7*(1), 77–91.

Martin, R. A. (2021). PyPortfolioOpt: Portfolio optimization in Python. *Journal of Open Source Software*, *6*(61), 1–5.

Meucci, A. (2005). *Risk and Asset Allocation*. Springer.

Michaud, R. O., & Michaud, R. O. (2008). *Efficient Asset Management: A Practical Guide to Stock Portfolio Optimization and Asset Allocation* (2nd ed.). Oxford University Press.

Nicolini, C., Manzi, M., & Delatte, H. (2025). skfolio: Portfolio optimization in Python. *Proceedings of the International Research and Industry Symposium on AI (IRIS-AI 25)*, 35–41. https://doi.org/10.65236/iris-ai/proceedings/2025/article/3

Pardo, R. (2008). *The Evaluation and Optimization of Trading Strategies* (2nd ed.). John Wiley & Sons.

Prigent, J. L. (2007). *Portfolio Optimization and Performance Analysis*. CRC Press.

Qian, E. (2016). *Risk Parity Fundamentals*. CRC Press.

Roncalli, T. (2013). *Introduction to Risk Parity and Budgeting*. Chapman & Hall/CRC.

Ruppert, D., & Matteson, S. D. (2015). *Statistics and Data Analysis for Financial Engineering: With R Examples* (2nd ed.). Springer.

Tsay, R. S. (2010). *Analysis of Financial Time Series* (3rd ed.). John Wiley & Sons.

Tsay, R. S. (2013). *Multivariate Time Series Analysis: With R and Financial Applications*. John Wiley & Sons.

Vidyamurthy, G. (2004). *Pairs Trading: Quantitative Methods and Analysis*. John Wiley & Sons.

Wuertz, D., Setz, T., Chalabi, Y., Chen, W., & Theussl, S. (2023). *fPortfolio: Rmetrics – Portfolio Selection and Optimization* [R package]. https://cran.r-project.org/package=fPortfolio

Part I

Financial Data

2

Financial Data: Stylized Facts

"If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck."

— James Whitcomb Riley

Different domains in science and engineering are deeply rooted on the specifics of the data. For instance, Sir Isaac Newton formulated the laws of motion and universal gravitation based on observations of the motion of planets and other objects on Earth. As another example, Claude Shannon – the "father of information theory" and inventor of the concept of a "bit" as a measure of information – developed a groundbreaking mathematical theory of communication based on probabilistic models of wireless and wireline transmission channels.

Likewise, the first step in any endeavor in finance or financial engineering should be to understand financial data. The study and characterization of financial data started flourishing as early as the 1960s (Fama, 1965; Mandelbrot, 1963) and it is now a mature topic in which academics and practitioners have exposed some particularities of the data commonly referred to as "stylized facts." This chapter takes us on a rather visual exploratory analysis of financial data based on empirical market data.

2.1 Stylized Facts

Stylized facts are properties common across many instruments, markets, and time periods that have been observed by independent studies (Cont, 2001; McNeil et al., 2015). Some notable examples include:

- *Lack of stationarity*: The statistics of financial time series change over time (past returns do not necessarily reflect future performance).
- *Volatility clustering*: Large price changes tend to be followed by large price changes (ignoring the sign), whereas small price changes tend to be followed by small price changes (Fama, 1965; Mandelbrot, 1963).
- *Absence of autocorrelations*: Autocorrelations of returns are often insignificant (Ding & Granger, 1996), which can be explained by the efficient-market hypothesis (Fama, 1970).

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

- *Heavy tails*: Gaussian distributions generally do not hold in financial data; instead, distributions typically exhibit so-called heavy tails.
- *Gain/loss asymmetry*: The distribution of the returns is not symmetric.
- *Positive correlation of assets*: Returns are often positively correlated since assets typically move together with the market.

It is worth noting that different data frequency regimes may exhibit a variation of characteristics:

- *Low frequency (weekly, monthly, quarterly)*: Gaussian distributions may fit reasonably well after correcting for volatility clustering (except for the asymmetry), but the scarcity of data is a big issue in a statistical sense.
- *Medium frequency (daily)*: Heavy tails cannot be ignored (even after correcting for volatility clustering) and the amount of data may be acceptable for statistical significance provided the models do not contain too many parameters (or overfitting will be inevitable).
- *High frequency (intraday, 30 min, 5 min, tick-data)*: Large amounts of data are available, which makes this regime more amenable to data analytics and machine learning techniques. Furthermore, as the frequency of the data increases, the influence of microstructure noise becomes more prominent, which requires alternative models.

2.2 Prices and Returns

The price of an asset is arguably the most obvious quantity one can observe in financial markets. We denote it by p_t, where t is the discrete time index (a continuous time index can also be used) corresponding to arbitrary periods such as minutes, hours, days, weeks, months, quarters, or years.

When it comes to modeling, it turns out that the logarithm of the prices,

$$y_t \triangleq \log p_t,$$

is mathematically more convenient. In addition, using the logarithm has the advantage that a much wider dynamic range of the signal can be naturally represented (i.e., tiny values are amplified and large values are attenuated).

Some accessible textbooks that cover financial data modeling are Meucci (2005), Cowpertwait and Metcalfe (2009), Tsay (2010), Ruppert and Matteson (2015), with more emphasis on the multi-asset case in Lütkepohl (2007) and Tsay (2013).

The simplest model for the log-prices is the *random walk*:

$$y_t = \mu + y_{t-1} + \epsilon_t, \tag{2.1}$$

where μ is the drift and ϵ_t is the i.i.d. random noise. Figure 2.1 shows the daily prices of the stock index S&P 500[1] over a span of more than a decade and Figure 2.2 shows the daily

[1] The Standard and Poor's 500, or simply the S&P 500, is a stock market index of 500 of the largest companies listed on stock exchanges in the United States. It is one of the most commonly followed equity indices.

prices of Bitcoin[2] over six years, both on a logarithmic scale (which is equivalent to plotting the log-prices on a linear scale).

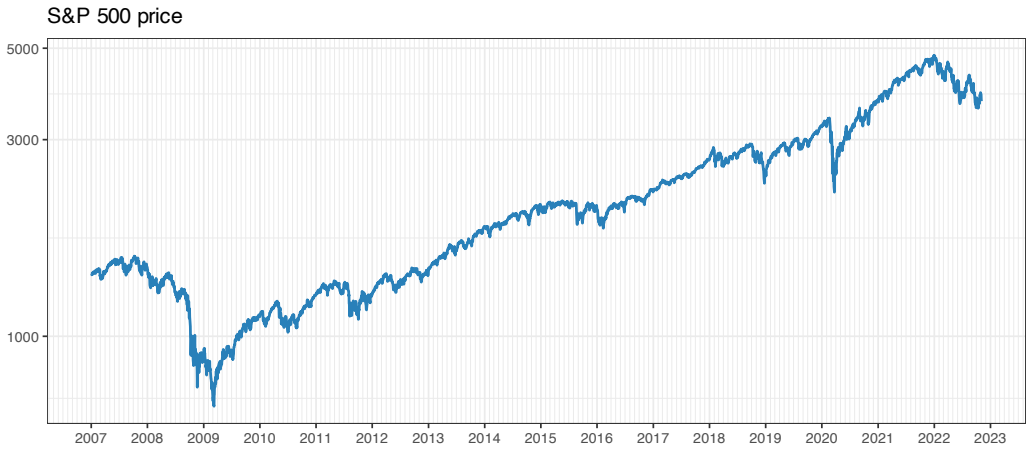


Figure 2.1 Price time series of S&P 500.



Figure 2.2 Price time series of Bitcoin.

Another key quantity is the price change, also called the return, which, unlike the absolute price, exhibits some degree of stationarity and may be more convenient for mathematical modeling. Among the different definitions of returns (Ruppert & Matteson, 2015; Tsay, 2010), we will focus on two types that have important additive properties along two different domains:

- The *linear return* (a.k.a. simple or net return) is defined as

$$r_t^{\text{lin}} \triangleq \frac{p_t - p_{t-1}}{p_{t-1}} = \frac{p_t}{p_{t-1}} - 1$$

[2] Bitcoin (BTC) is the first and most popular cryptocurrency. It was invented in 2008 by an unknown person or group of people using the name Satoshi Nakamoto. The currency began its use in 2009.

and has the property that it is additive among the assets (i.e., the overall linear return when investing in several assets equals the sum of the returns of the assets weighted according to the percentage of the budget invested in each). Thus, linear returns are key when dealing with the return of a portfolio of several assets (refer to Chapter 6 for details).

- The *log-return* (a.k.a. continuously compounded return) is defined as

$$r_t^{\log} \triangleq y_t - y_{t-1} = \log\left(\frac{p_t}{p_{t-1}}\right)$$

and has the property that it is additive along the time domain (i.e., the log-return of a long period equals the sum of the log-returns of the basic periods within the long period). Thus, log-returns are preferred when it comes to mathematical modeling of time series (refer to Chapters 3–4). For example, according to the previous random walk model in (2.1), the log-return is stationary:

$$r_t^{\log} = y_t - y_{t-1} = \mu + \epsilon_t .$$

Interestingly, the simple return and log-return are related as

$$r_t^{\log} = \log\left(1 + r_t^{\text{lin}}\right),$$

which leads to the convenient approximation $r_t^{\log} \approx r_t^{\text{lin}}$, when r_t^{lin} is small, as illustrated in Figure 2.3. The approximation is almost perfect when the magnitude of the return is less than 0.05 or 5%, then it slowly degrades.

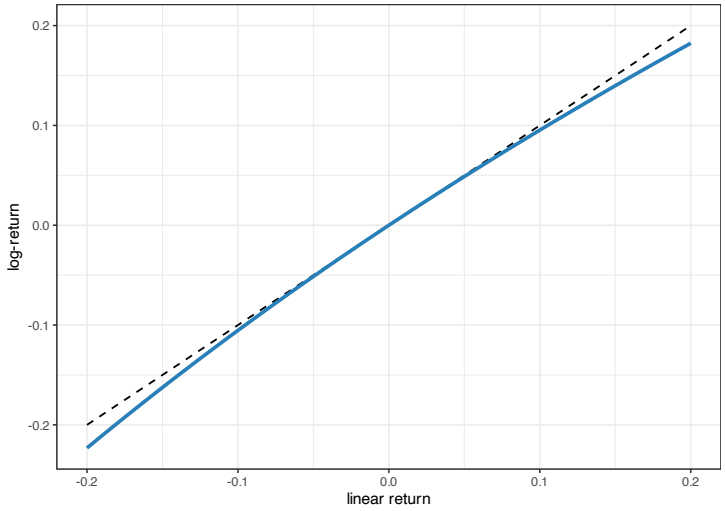


Figure 2.3 Approximation of log-return vs. linear return.

Figure 2.4 shows the daily log-returns of the S&P 500 stock index over a span of more than a decade. One can easily observe the high-volatility period during the global financial crisis in 2008, as well as the high peak in volatility in early 2020 due to the COVID-19 pandemic.

Figure 2.5 shows the daily log-returns of Bitcoin over six years. Observe the Bitcoin flash crash on March 12, 2020, with a drop close to 50% in a single day.[3]

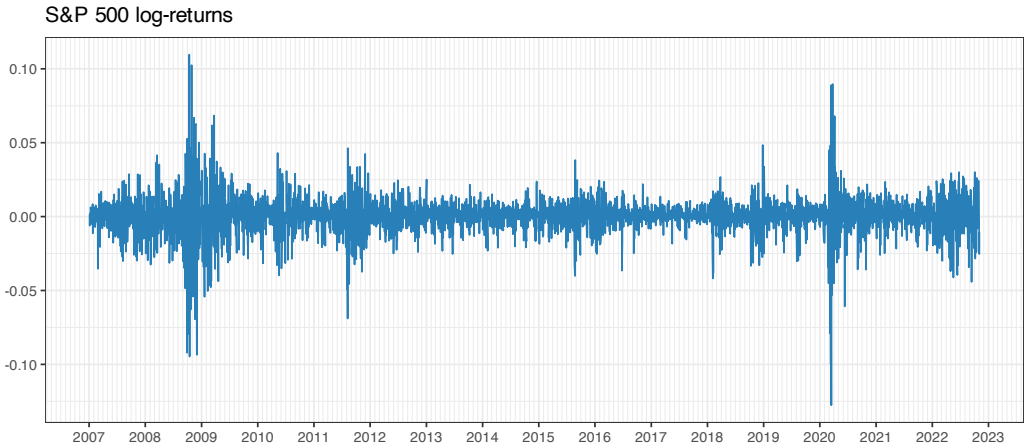


Figure 2.4 Daily log-return time series of S&P 500.

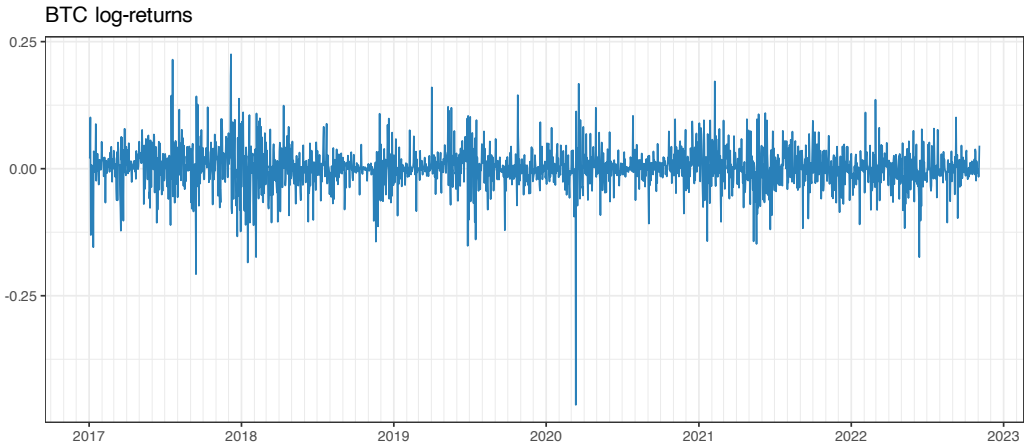


Figure 2.5 Daily log-return time series of Bitcoin.

It is insightful to compare the volatility levels for the two classes of assets. The annualized volatility of the S&P 500 stock index is on the order of 21%, whereas for Bitcoin it is around 78%. Generally speaking, a value of anywhere between 12% and 20% is considered low, whereas above 30% it is considered extremely volatile. Thus, the S&P 500 has a low volatility whereas Bitcoin is extremely volatile.

[3] The flash crash of March 12, 2020, was triggered by COVID-19, since, just one day earlier, the World Health Organization had announced that "COVID-19 can be characterized as a pandemic."

2.3 Non-Gaussianity: Asymmetry and Heavy Tails

The *Gaussian* or *normal* distribution is one of the most commonly used for continuous random variables due to its ease of mathematical manipulation. It is characterized by two parameters: the mean and the variance (first- and second-order moments). Its probability distribution function (pdf) reads:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2},$$

where μ is the mean and σ^2 is the variance.

In many areas, the Gaussian distribution may in fact be appropriate and easily justified from physical principles (like the pervasive thermal noise in electronic circuits that hinders communication systems). However, in other domains, like radar and financial systems, the random quantities are often not Gaussian distributed and higher-order moments are necessary for a proper characterization (Jondeau et al., 2007). Two new aspects enter the picture for a proper characterization:

- the *skewness* as a measure of the asymmetry in the distribution, and
- the *kurtosis* as a measure of the thickness of the tails (i.e., whether the tails decay faster or slower than the exponential decay of the Gaussian distribution).

Figure 2.6 illustrates the effect of skewness and kurtosis on the pdf. Financial data typically exhibits negative skewness and large kurtosis (with potential huge losses due to the heavy left tail).

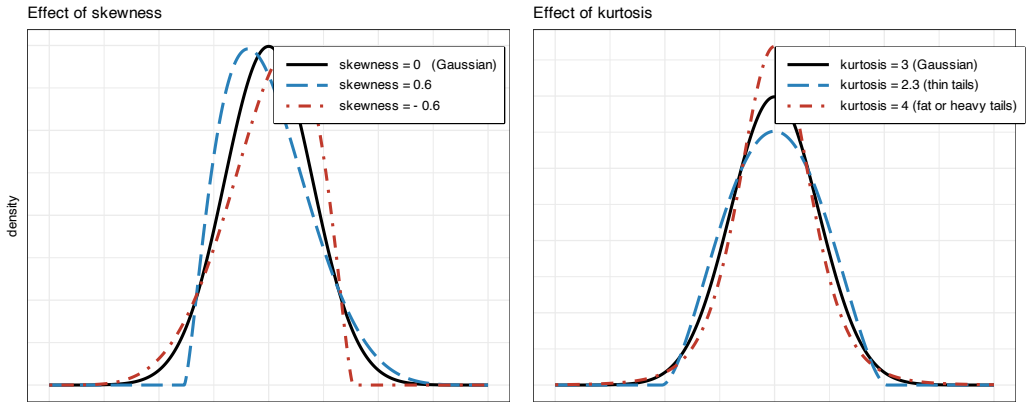


Figure 2.6 Effect of skewness and kurtosis on the probability distribution function.

The combination of skewness and kurtosis makes it more likely for highly negative returns to occur (with the obvious consequences for an investor who has bought the asset). This is illustrated in Figure 2.7, where the left tail of a typical distribution of financial data is clearly shown to be much fatter or heavier than that of a Gaussian distribution. This is why distributions with tails decaying slower than the exponential (decay of the Gaussian) are called heavy tails, fat tails, or thick tails.

Figure 2.8 shows histograms of the S&P 500 log-returns at different frequencies (namely,

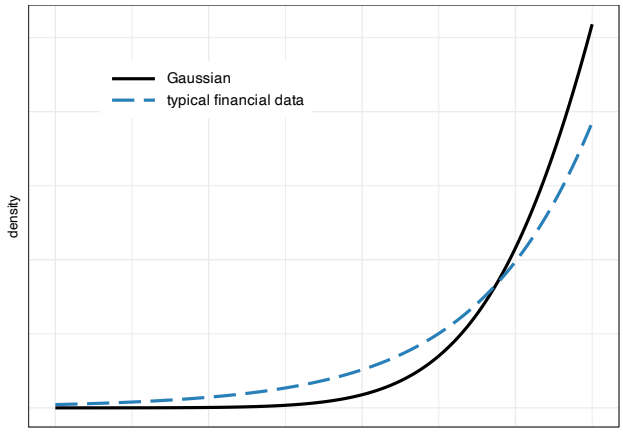


Figure 2.7 Left tail of Gaussian and typical financial data distributions.

daily, monthly, and quarterly). It can be seen that the tails of the histograms are significantly heavier or thicker than that of a Gaussian and that the histogram is not symmetric. Figure 2.9 shows histograms of Bitcoin log-returns, again with clear heavy tails, although the asymmetry seems less pronounced than in the S&P 500 case.

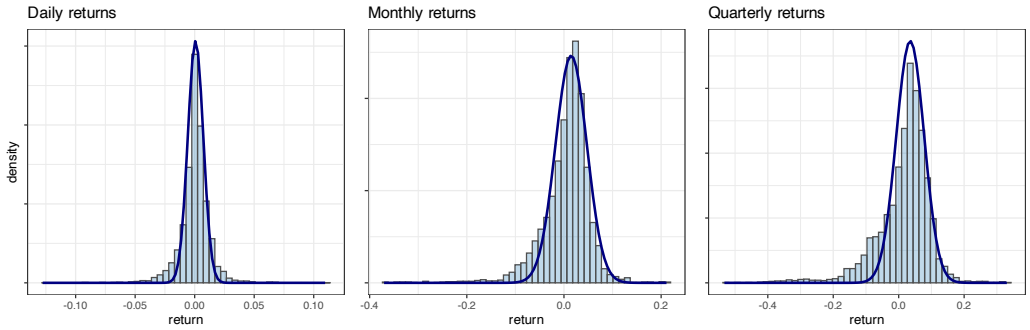


Figure 2.8 Histogram of S&P 500 log-returns at different frequencies (with inappropriate Gaussian fit).

While histograms provide a quick visual inspection of the whole distribution, there are other more convenient types of plot that allow for a clearer characterization of the level of asymmetry and heavy-tailness.

2.3.1 Asymmetry or Skewness

Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. Zero skewness implies a symmetric distribution; negative skew commonly indicates that the thick tail is on the left side of the distribution, whereas positive skew indicates that the thick tail is on the right. The skewness of a random variable X is defined as the third standardized moment $\mathbb{E}\left[\left((X-\mu)/\sigma\right)^3\right]$.

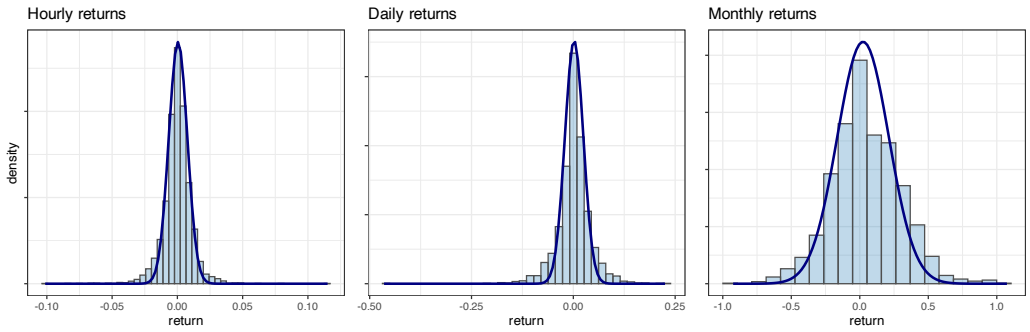


Figure 2.9 Histogram of Bitcoin log-returns at different frequencies (with inappropriate Gaussian fit).

Figure 2.10 plots the skewness of the S&P 500 returns during 2007–2022 as a function of the period of the returns. As the period increases from one day to ten days, the skewness decreases rapidly; then it saturates. Figure 2.11 shows the same for Bitcoin during 2017–2022, with similar results. As previously observed from the histograms, the skewness of Bitcoin is closer to zero than that of the S&P 500. Thus, as a first approximation, cryptocurrencies seem to be more symmetric than stocks.

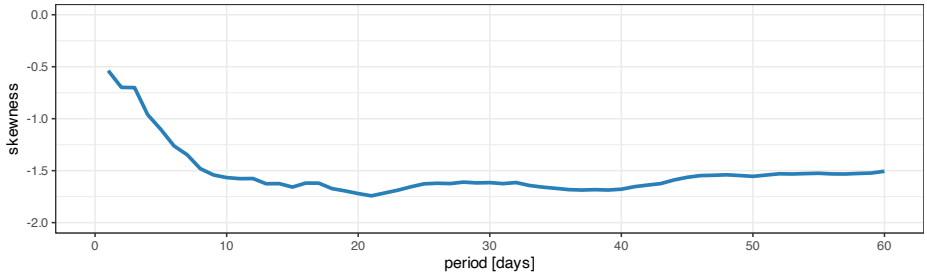


Figure 2.10 Skewness of S&P 500 log-returns.

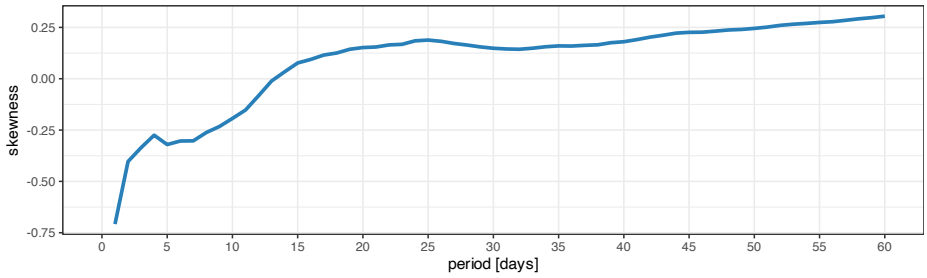


Figure 2.11 Skewness of Bitcoin log-returns.

2.3.2 Heavy-Tailness or Kurtosis

Q–Q (quantile–quantile) plots allow for a clearer assessment of the degree of heavy tails as compared to the exponential one of the Gaussian case. Figures 2.12 and 2.13 show Q–Q plots corresponding to the S&P 500 and Bitcoin log-returns, respectively. The deviation of both the left and right tails in all cases clearly indicates heavy tails.

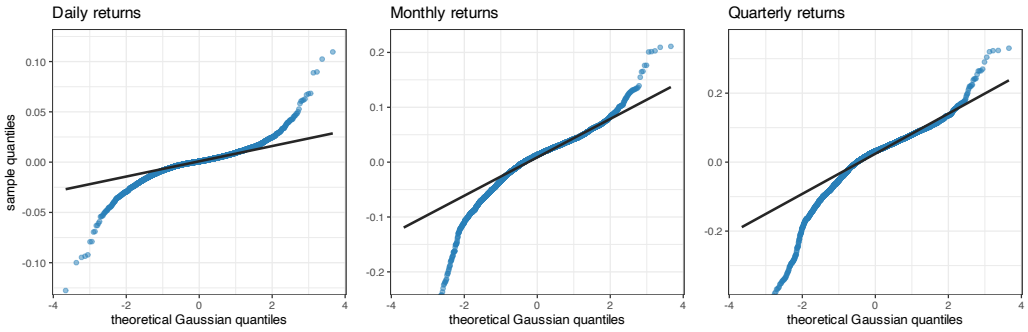


Figure 2.12 Q–Q plots of S&P 500 log-returns at different frequencies.

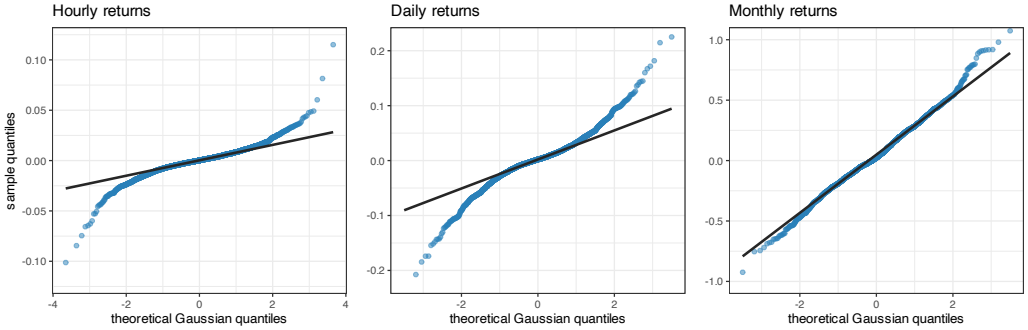


Figure 2.13 Q–Q plots of Bitcoin log-returns at different frequencies.

Kurtosis is a measure of the "tail heavyness" of the probability distribution of a real-valued random variable. Like skewness, kurtosis describes the shape of a probability distribution and there are different ways of quantifying it. The kurtosis of a Gaussian distribution is 3. Higher kurtosis values correspond to greater extremity of deviations (or outliers), hence the names heavy/fat/thick tails. It is common practice to use an adjusted version of the kurtosis, the excess kurtosis, which is the kurtosis minus 3. The standard definition of the kurtosis of a random variable X is the fourth standardized moment $\mathbb{E}\left[\left((X-\mu)/\sigma\right)^4\right]$.

Figure 2.14 plots the kurtosis of the S&P 500 returns during 2007–2022 as a function of the period of the returns. As the period increases from one day to three days, the excess kurtosis decreases very rapidly and then it saturates at around 6 to 8. Similarly, Figure 2.15 shows the kurtosis of Bitcoin returns during 2017–2022. As the period increases from one to three days, the excess kurtosis decreases very rapidly to less then 3. Interestingly, the kurtosis of Bitcoin seems to be smaller than that of the S&P 500. Thus, as a first approximation, it may seem that

cryptocurrencies are less heavy-tailed or more Gaussian than stocks. However, this deserves a closer look by looking at the excess kurtosis during different periods:

- From 2017 to 2019: 5.41 for the S&P 500 and 3.46 for Bitcoin.
- During 2020: 8.51 for the S&P 500 and 50.87 for Bitcoin.
- From 2021 to 2022: 0.95 for the S&P 500 and 2.34 Bitcoin.

From these distinct periods, it is evident that the primary difference occurred in 2020, during which Bitcoin exhibited significantly more heavy-tailed behavior.

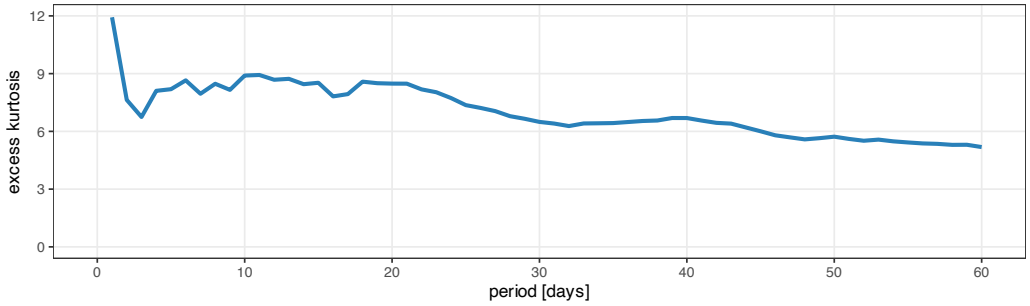


Figure 2.14 Excess kurtosis of S&P 500 log-returns.

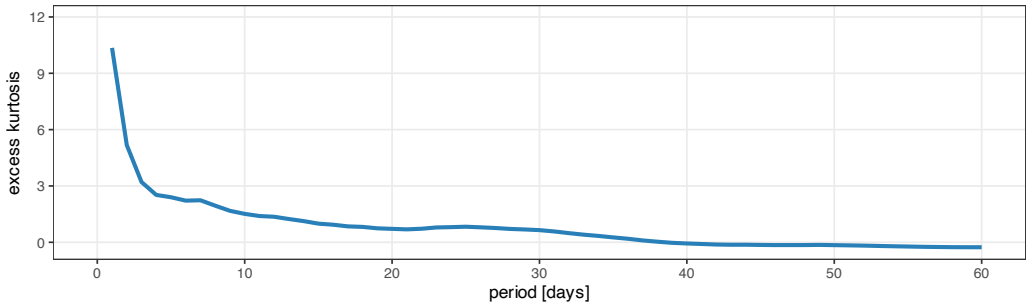


Figure 2.15 Excess kurtosis of Bitcoin log-returns.

2.3.3 Statistical Tests

From the previous analysis, financial data clearly show skewness and kurtosis. To assess whether these parameters, together with the mean and variance, are enough to characterize the data, we can resort to mathematically sound statistical tests.

The Anderson–Darling statistic measures how well the data follow a particular distribution (the better the distribution fits the data, the smaller this statistic will be). The hypotheses for the Anderson–Darling test are

- H0: the data follow a specified distribution (null hypothesis); and
- H1: the data do not follow a specified distribution (alternative hypothesis).

As is customary, we can use the p-value to determine whether the data come from the chosen distribution (if it is smaller than some threshold, typically 0.05 or so, then we can reject the null hypothesis that the data came from that distribution).[4]

Table 2.1 shows the results of the Anderson–Darling test for three distributions: the Gaussian, the Student t distribution (which models heavy tails), and the skewed t distribution (which accounts for both skewness and heavy tails). From these results we can conclude that the skewed t distribution provides a good fit to the S&P 500 during the period 2015–2020. For an additional visual inspection, Figure 2.16 shows Q–Q plots of the empirical data with respect to the three candidate distributions (Gaussian, Student t, and skewed t). We can again confirm that the skewed t distribution is a good fit.

Table 2.1 Results of Anderson–Darling test on financial data, supporting the skewed t distribution.

Distribution	Anderson–Darling test	p-value
Gaussian	55.315	4.17×10^{-7}
Student t	5.4503	0.001751
Skewed t	2.3208	0.06161

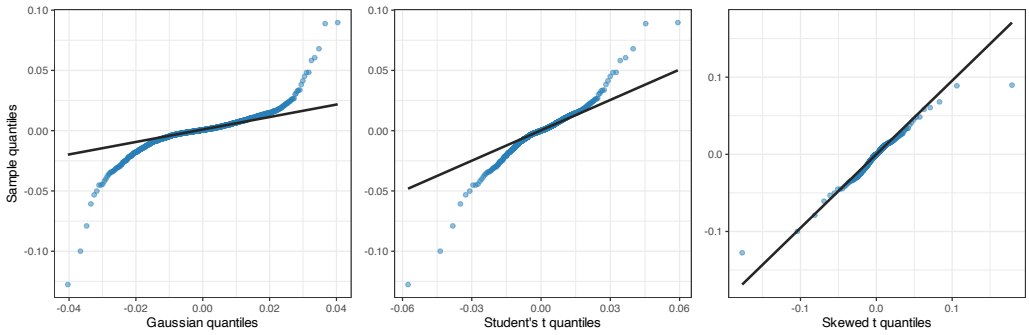


Figure 2.16 Q–Q plots of S&P 500 log-returns vs. different candidate distributions.

2.4 Temporal Structure

Are the returns i.i.d. or do they show some temporal structure? This is a key question in finance because it determines the problem of forecasting the returns or prices. In fact, this has been a highly debated topic in economics for decades.

The *efficient-market hypothesis* (EMH) states that share prices reflect all information and

[4] The p-value is the probability of obtaining the observed results under the assumption that the null hypothesis is correct. A small p-value means that there is strong evidence to reject the null hypothesis and accept the alternative hypothesis. Typical thresholds for determining whether a p-value is small enough are in the range 0.01–0.05.

consistent "alpha"[5] generation is impossible (Fama, 1970). Accordingly, stocks always trade at their fair value on exchanges, making it impossible for investors to purchase undervalued stocks or sell stocks for inflated prices. Therefore, it should be impossible to outperform the overall market through expert stock selection or market timing, and the only way an investor can obtain higher returns is by purchasing riskier investments.

Although it is a cornerstone of modern financial theory, the EMH is highly controversial and often disputed (Shiller, 1981). Believers argue it is pointless to search for undervalued stocks or to try to predict trends in the market through either fundamental or technical analysis. Theoretically, neither technical nor fundamental analysis can produce risk-adjusted excess returns (i.e., "alpha") consistently, and only insider information can result in outsized risk-adjusted returns. Although academics present a substantial body of evidence in support of it, there is an equal amount of dissent as well. For example, the fundamental-based investor Warren Buffett or the hedge fund Renaissance Technologies' Medallion Fund have consistently beaten the market over long periods, which by definition is impossible according to the EMH.

Proponents of the EMH conclude that, because of the randomness of the market, investors could do better by investing in a low-cost, passive portfolio. On the other hand, opponents insist on the possibility of designing portfolios that can beat the market.

Some accessible textbooks that cover temporal analysis include Tsay (2010), Cowpertwait and Metcalfe (2009), and Ruppert and Matteson (2015).

2.4.1 Linear Structure in Returns

The autocorrelation function (ACF) and partial autocorrelation function (PACF) are heavily used in time series analysis and forecasting. They measure the linear structure or dependency along the temporal domain, which, according to the EMH, should be insignificant (Ding & Granger, 1996). The autocorrelation is simply the correlation between the signal at a time and some previous time, whereas the partial autocorrelation eliminates the effect of the signal in between those two time instances.

Figure 2.17 indicates that the S&P 500 index exhibits almost no significant autocorrelation that can be exploited by models for forecasting (lags other than zero are basically within the statistical insignificant level). Figure 2.18 similarly illustrates that there is no significant autocorrelation to be exploited in Bitcoin (hourly returns similarly show no autocorrelations).

2.4.2 Nonlinear Structure in Returns

From the previous absence of significant autocorrelations, one may be tempted to conclude that there is no temporal structure to be exploited. However, that would be a wrong conclusion. A visual inspection of the returns suffices to see that clearly some structure is present in the volatility envelope of the signal (Ding & Granger, 1996), which measures the time-varying standard deviation of the signal along the time domain.

[5] In finance, the term "alpha" is commonly used to denote a signal or information that results in the outperformance of profits compared to a benchmark.

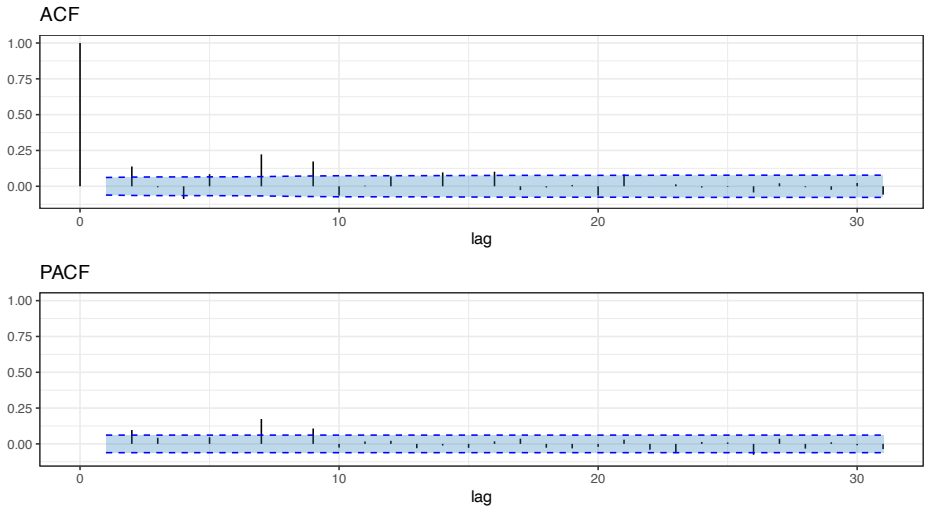


Figure 2.17 Autocorrelation of S&P 500 daily log-returns.

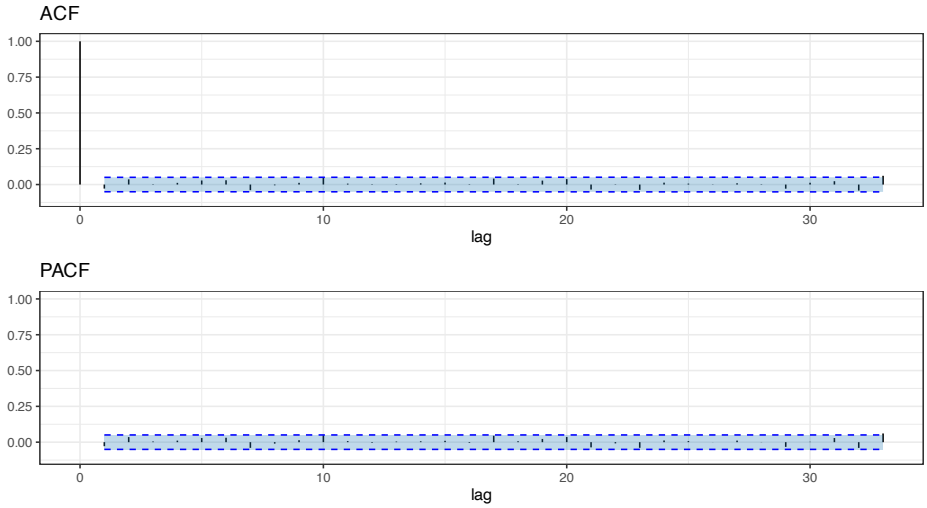


Figure 2.18 Autocorrelation of Bitcoin daily log-returns.

Figure 2.19 shows S&P 500 log-returns together with the volatility envelope, illustrating the phenomenon of the so-called *volatility clustering*. Clearly the volatility envelope changes slowly and can be easily forecast, as already pointed out in the 1990s (Ding & Granger, 1996). Figure 2.20 shows Bitcoin log-returns together with the volatility envelope, also exhibiting volatility clustering.

But how can it be that the returns show no significant autocorrelations but clearly there is some temporal structure?

The answer lies in the fact that autocorrelation measures only the linear dependency. Nonlinear

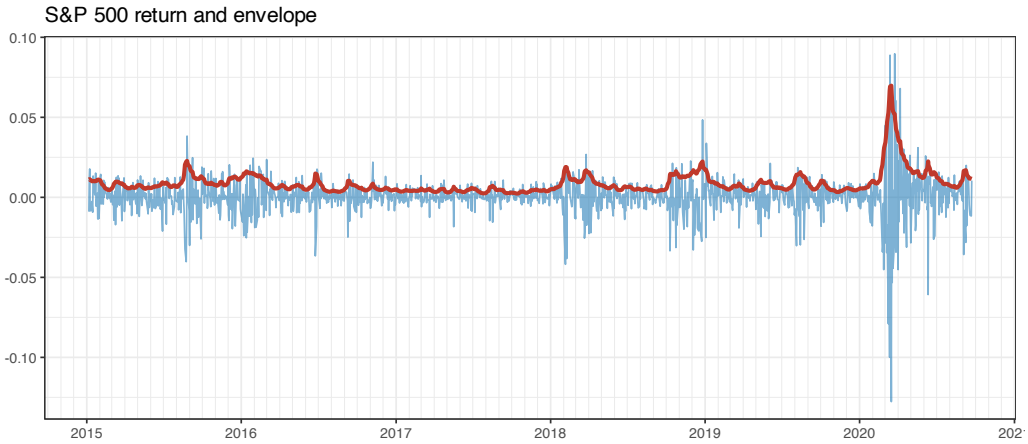


Figure 2.19 Volatility clustering in S&P 500.

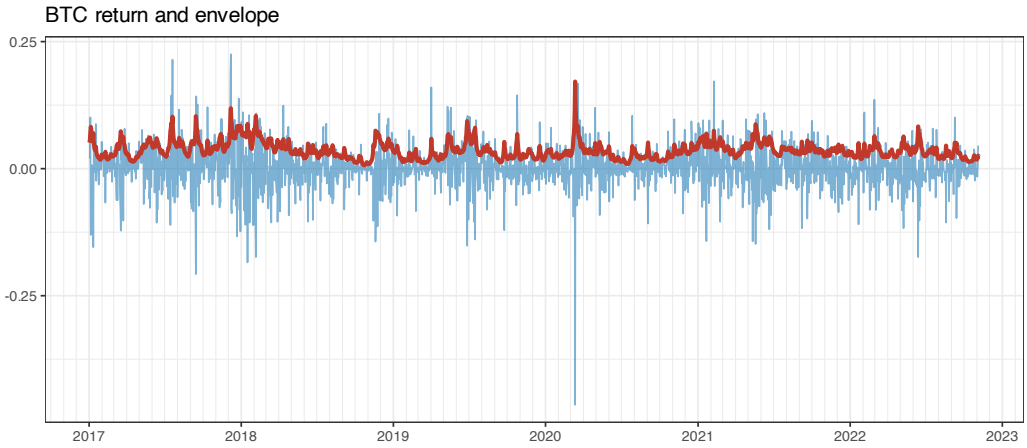


Figure 2.20 Volatility clustering in Bitcoin.

dependencies are more elusive to detect. In fact, machine learning is a potential tool to try to identify such nonlinear dependencies (López de Prado, 2018).

Since the envelope is basically a smooth version of the absolute value of the signal, one could calculate instead the autocorrelation of the absolute values of the returns. Figure 2.21 shows very significant autocorrelation values for the absolute values of the S&P 500 log-returns. Similarly, Figure 2.22 shows significant autocorrelation values for the absolute values of Bitcoin log-returns (albeit not as significant as in the case of the S&P 500).

It may be useful to factor out the volatility envelope from the returns (i.e., dividing the returns by the volatility) so as to obtain a time series without volatility clustering, termed *standardized returns*. Figures 2.23 and 2.24 illustrate such standardized returns for the S&P 500 and Bitcoin.

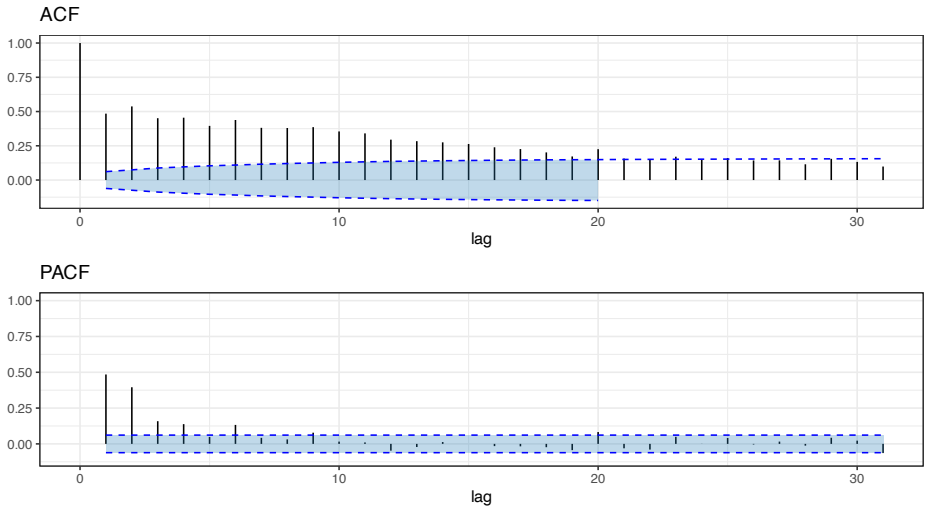


Figure 2.21 Autocorrelation of absolute value of S&P 500 daily log-returns.

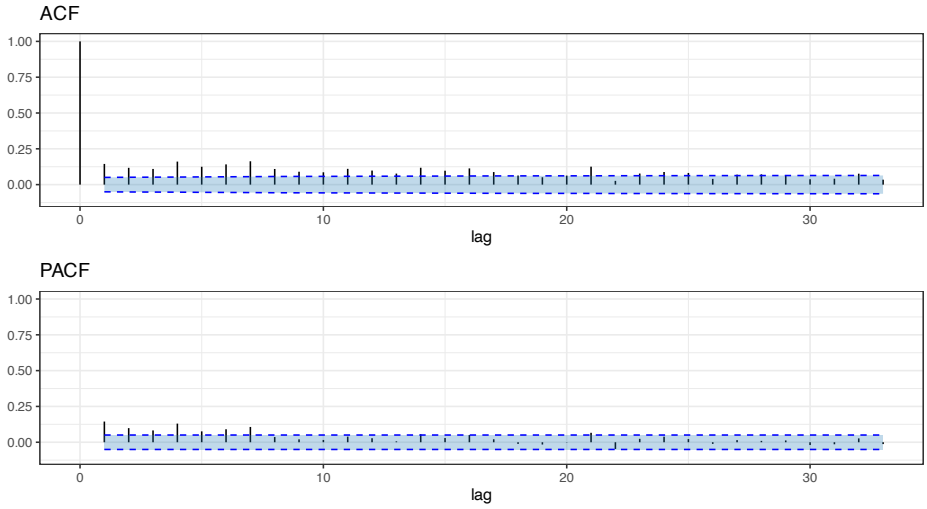


Figure 2.22 Autocorrelation of absolute value of Bitcoin daily log-returns.

2.5 Asset Structure

Apart from the structure along the temporal dimension, which can be used for modeling and forecasting, there is structure along the asset dimension (also referred to as cross-sectional structure). This means that rather than considering the assets one by one independently, they have to be jointly modeled.

This is particularly important when it comes to assessing the risk of a portfolio, since different stocks may have different correlations. For example, one may diversify an investment by allocating capital to several assets, but if they are strongly correlated it may not help in reducing the risk. Figure 2.25 illustrates the effect of asset correlation on the volatility of

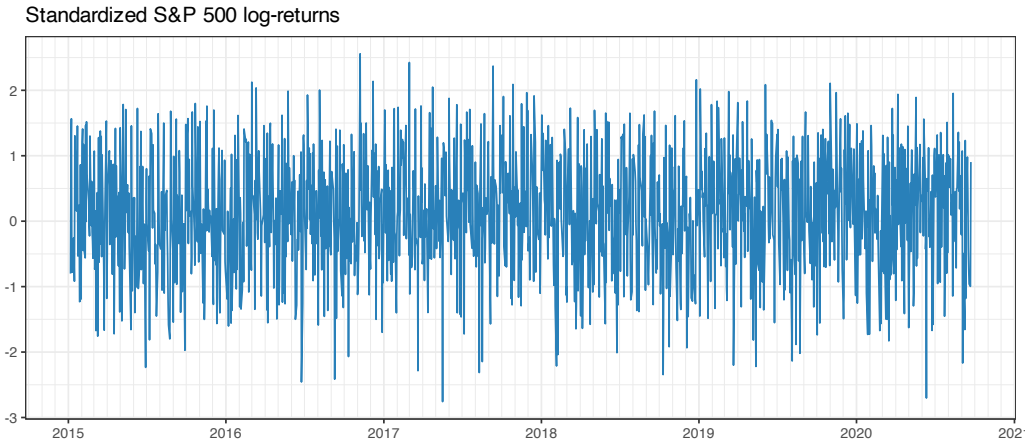


Figure 2.23 Standardized S&P 500 log-returns after factoring out the volatility envelope.

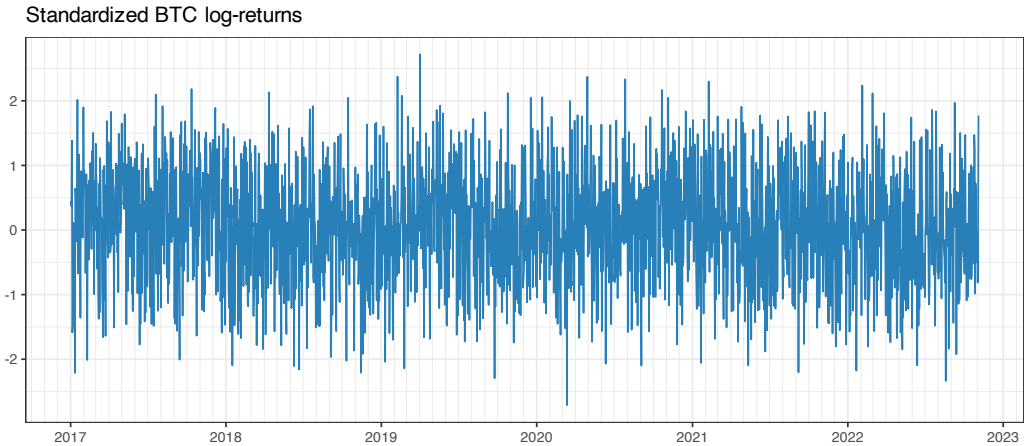


Figure 2.24 Standardized Bitcoin log-returns after factoring out the volatility envelope.

the equally weighted portfolio for the case of two assets with correlation ρ and each with variance 1. We can observe that for fully correlated assets, $\rho = 1$, the portfolio does not benefit from any diversity with respect to investing in a single asset and the variance remains at 1 (volatility of 1); for uncorrelated assets, $\rho = 0$, the portfolio gets a diversity benefit with the variance reduced to half (volatility of $\sqrt{0.5}$), and for negatively correlated assets, $\rho < 0$, the diversity benefit increases further. In reality, assets tend to have a high correlation close to 1 and finding uncorrelated assets or classes of assets is the "holy grail." The particular case of fully negatively correlated assets, $\rho = -1$, can be achieved with a synthetic asset created for the sole purpose of hedging another one to control the risk exposure.

Figure 2.26 shows heatmaps of the correlation matrix of some stocks from the S&P 500 (daily returns) and some cryptocurrencies (hourly returns). Compared to the diagonal elements

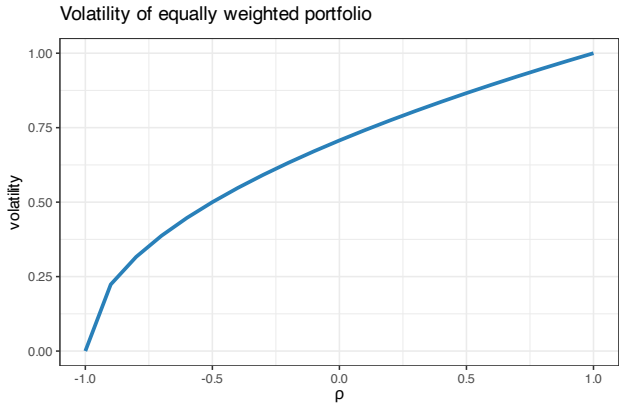


Figure 2.25 Effect of asset correlation on volatility for a two-asset portfolio.

(which are equal to 1), the off-diagonal components are much weaker (and no value lies in the negative range). In the case of cryptocurrencies, for example, there is a pair of names that are fully correlated: BTC and WBTC, which is expected since WBTC is by definition a wrapped BTC.

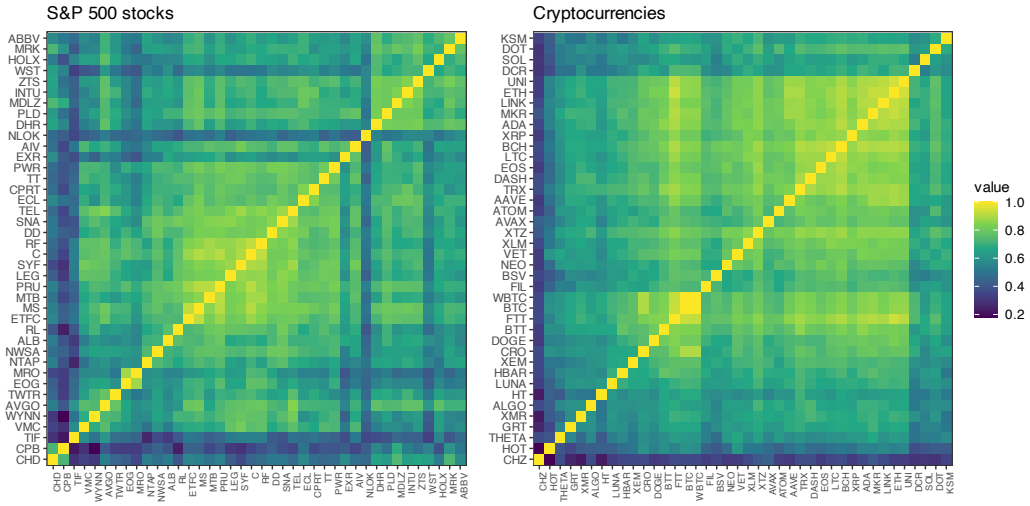


Figure 2.26 Correlation matrix of returns for stocks and cryptocurrencies.

We can confirm the previous observation that the cross-correlations are mostly nonnegative from the histograms depicted in Figure 2.27 for S&P 500 stocks (daily returns) and cryptocurrencies (hourly returns). In fact, this positive correlation among stocks and cryptocurrencies is not surprising since assets tend to move together with the market.

To explore the asset structure more deeply, it is worth inspecting how the eigenvalues of the correlation matrix tend to cluster into very few large ones and a big group of much smaller ones, the so-called *factor model* structure (Fama & French, 1992; Sharpe, 1964) (see Chapter 3

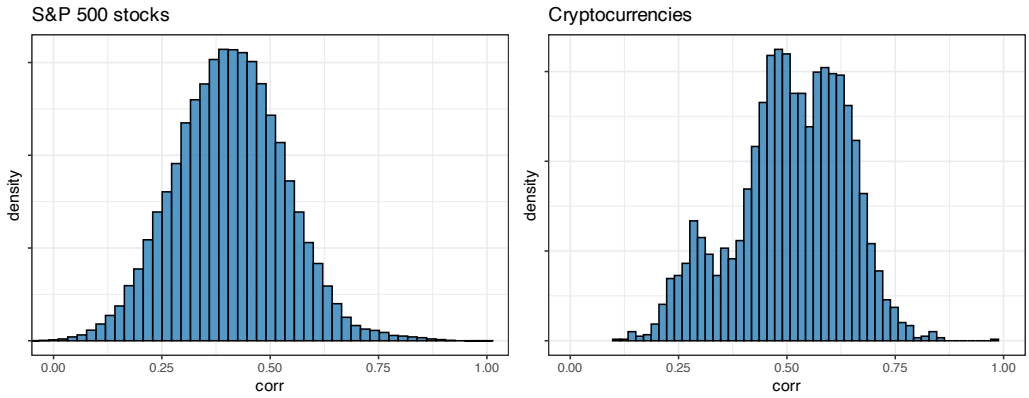


Figure 2.27 Histogram of correlations among returns of stocks and cryptocurrencies.

for details). Figure 2.28 depicts histograms of eigenvalues of the correlation matrix of daily returns for the S&P 500 stocks and hourly returns for the top 82 cryptocurrencies. In both cases, we can observe that a single eigenvalue is totally predominant (corresponding to the market index and constituting almost half of the total value of the eigenvalues), perhaps with one to four other nonnegligible eigenvalues, while the rest of them are orders of magnitude smaller (note that the horizontal axis follows a logarithmic scale).

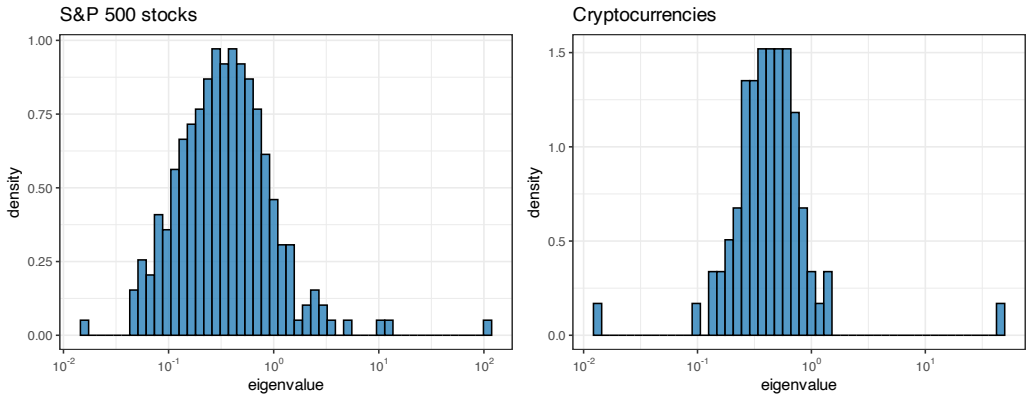


Figure 2.28 Histogram of correlation matrix eigenvalues of stocks and cryptocurrencies.

2.6 Summary

Financial data display unique characteristics known as stylized facts, with the most prominent ones including:

- *Lack of stationarity*: The statistics of financial data change over time significantly and any attempt at modeling will have to continuously adapt.
- *Volatility clustering*: This is perhaps the most visually apparent aspect of financial time

series. There are a myriad models in the literature that can be utilized for forecasting (covered in Chapter 4).

- *Heavy tails*: The distribution of financial data is definitely not Gaussian and this constitutes a significant departure from many traditional modeling approaches (covered in Chapter 3).
- *Strong asset correlation*: The goal in investing is to discover assets that are not strongly correlated, which is a daunting task due to the naturally occurring strong asset correlation.

Exercises

Choose one or several assets (e.g., stocks or cryptocurrencies) for the following exercises.

2.1 (Price time series) Choose one asset and plot the price time series using both a linear and a logarithmic scale. Compare the plots and comment.

2.2 (Return time series) Choose one asset and plot the linear returns and log-returns. Compare the plots and comment.

2.3 (Volatility envelope) Choose one asset and compute the volatility (square root of the average of the squared returns over k samples) on a rolling-window basis in two ways:

a. Left-aligned window: at each time t, use the samples $t-k+1,\dots,t$. Try different values of k, observe the effect, and discuss.
b. Centered window: at each time t, use the samples $t-\lfloor k\rfloor/2,\dots,t+\lceil k\rceil/2-1$. Try different values of k, observe the effect, and discuss.

Finally, compare the left-aligned and centered rolling-window approaches and discuss.

2.4 (Return distribution) Choose one asset and perform the following tasks:

a. Plot histograms of the log-returns at different frequencies. Compare the plots and comment.
b. Draw Q–Q plots to focus on the tail distribution. Do the returns follow a Gaussian distribution?
c. Compute the skewness and kurtosis to see if they correspond to a Gaussian distribution.

2.5 (Return autocorrelation) Choose one asset and perform the following tasks:

a. Plot the autocorrelation function of the log-returns at various frequencies. Compare the plots and comment.
b. Repeat the process using squared returns instead of log-returns. Compare these plots and comment.

2.6 (Asset correlation) Choose several stocks and perform the following tasks:

a. Compute the cross-correlations and plot a heatmap.
b. Compute the correlation between each of the stocks and the index. Discuss the results.
c. Compute the correlation between a stock and a cryptocurrency. Discuss the result and the implications.

References

Cont, R. (2001). Empirical properties of assets returns: Stylised facts and statistical issues. *Quantitative Finance*, *1*, 223–236.

Cowpertwait, P. S., & Metcalfe, A. V. (2009). *Introductory Time Series With R*. Springer.

Ding, Z., & Granger, C. (1996). Modeling volatility persistence of speculative returns: A new approach. *Journal of Econometrics*, *73*(1), 185–215.

Fama, E. F. (1965). The behavior of stock-market prices. *The Journal of Business*, *38*(1), 34–105.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, *25*(2), 383–417.

Fama, E. F., & French, K. R. (1992). The cross-section of expected stock returns. *The Journal of Finance*, *47*(2), 427–465.

Jondeau, E., Poon, S. H., & Rockinger, M. (2007). *Financial Modeling Under Non-Gaussian Distributions*. Springer.

López de Prado, M. (2018). *Advances in Financial Machine Learning*. John Wiley & Sons.

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer.

Mandelbrot, B. B. (1963). The variation of certain speculative prices. *The Journal of Business*, *36*(4), 394–419.

McNeil, A. J., Frey, R., & Embrechts, P. (2015). *Quantitative Risk Management*. Princeton University Press.

Meucci, A. (2005). *Risk and Asset Allocation*. Springer.

Ruppert, D., & Matteson, S. D. (2015). *Statistics and Data Analysis for Financial Engineering: With R Examples* (2nd ed.). Springer.

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, *19*(3), 425–442.

Shiller, R. J. (1981). Do stock prices move too much to be justified by subsequent changes in dividends? *American Economic Review*, *71*(3), 421–436.

Tsay, R. S. (2010). *Analysis of Financial Time Series* (3rd ed.). John Wiley & Sons.

Tsay, R. S. (2013). *Multivariate Time Series Analysis: With R and Financial Applications*. John Wiley & Sons.

3

Financial Data: I.I.D. Modeling

"All models are wrong, but some are useful."

— George E. P. Box

Under the efficient-market hypothesis (Fama, 1970), the price of a security is a good estimate of its intrinsic value. That is, any information about future prospects is already incorporated in the current price, so that the forecast is just the current price. This leads to modeling the prices as a random walk (Malkiel, 1973) and, equivalently, the returns as a sequence of independent and identically distributed (i.i.d.) random variables. In the case of multiple assets, the random variables denote the returns of all the assets, leading to a multivariate random variable. This is a simple and convenient model, which in fact was already employed in Markowitz's seminal paper on portfolio design (Markowitz, 1952).

Under the i.i.d. model, no temporal structure is incorporated and the returns at a given time are assumed to be independent from other time instances; in addition, the distribution of the random returns over time is assumed fixed. Hence the terminology "independent and identically distributed." This chapter explores the characterization of the multivariate i.i.d. distribution, from the simplest sample estimators to the more sophisticated robust non-Gaussian estimators that may include prior information in the form of shrinkage, factor modeling, or prior views.

3.1 I.I.D. Model

Chapter 2 gives an exploratory view of financial data and the important stylized facts. This chapter explores a simple but useful and popular way to model financial data. Suppose we have N securities or tradable assets – possibly from distinct asset classes such as bonds, equities, commodities, mutual funds, currencies, and cryptos – and let $\boldsymbol{x}_t \in \mathbb{R}^N$ (often denoted as $\boldsymbol{r}_t$) denote the random returns of the assets at time t. Note that the time index t can denote any arbitrary period such as minutes, hours, days, weeks, months, quarters, years, and so on.

Under the i.i.d. model, the returns are simply modeled as

$$\boldsymbol{x}_t = \boldsymbol{\mu} + \boldsymbol{\epsilon}_t, \tag{3.1}$$

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

where $\boldsymbol{\mu} \in \mathbb{R}^N$ denotes the expected return and $\boldsymbol{\epsilon}_t \in \mathbb{R}^N$ is the residual component with zero mean and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{N \times N}$. This model can be motivated by the efficient-market hypothesis (Fama, 1970).[1] Figure 3.1 shows an example of a synthetic univariate Gaussian i.i.d. time series.

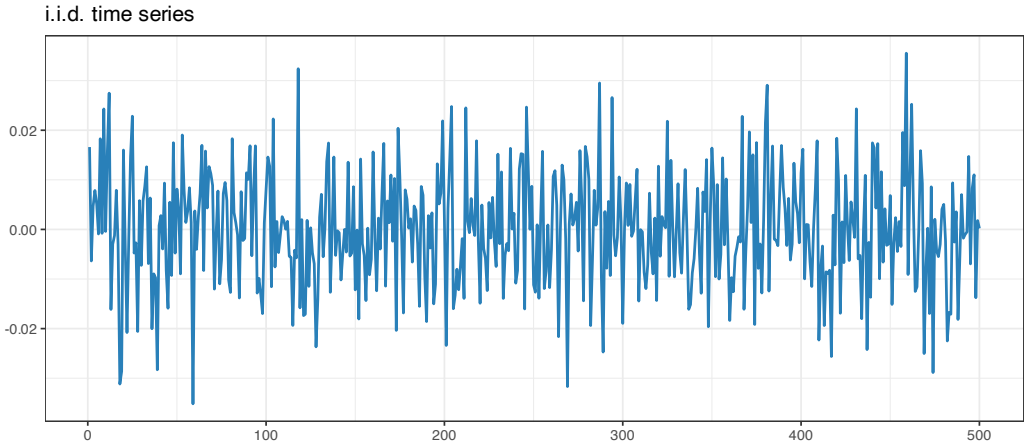


Figure 3.1 Example of a synthetic Gaussian i.i.d. time series.

To be more exact, the i.i.d. model in (3.1) corresponds to the *random walk* model (Campbell et al., 1997; Malkiel, 1973) on the log-prices $\boldsymbol{y}_t \triangleq \log \boldsymbol{p}_t$ (also referred to as the *geometric random walk* model on the prices):

$$\boldsymbol{y}_t = \boldsymbol{\mu} + \boldsymbol{y}_{t-1} + \boldsymbol{\epsilon}_t,$$

which leads to (3.1) when $\boldsymbol{x}_t$ denotes the log-returns: $\boldsymbol{x}_t = \boldsymbol{y}_t - \boldsymbol{y}_{t-1}$.

The i.i.d. model in (3.1) ignores any temporal structure or dependency in the data. Chapter 4 considers more sophisticated time series models that attempt to incorporate the time structure. Some accessible textbooks that cover financial data modeling are Meucci (2005), Tsay (2010), Ruppert and Matteson (2015), with more emphasis on the multivariate case in Lütkepohl (2007) and Tsay (2013).

3.2 Sample Estimators

In practice, the parameters of the i.i.d. model in (3.1), $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, are unknown and have to be estimated using historical data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ containing T past observations. The simplest estimators are the *sample mean*,

$$\hat{\boldsymbol{\mu}} = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{x}_t, \tag{3.2}$$

[1] Eugene F. Fama was awarded the 2013 Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel for his work on the efficient-market hypothesis. Ironically, Robert J. Shiller was co-awarded the same prize precisely for his work on the inefficiency of markets.

and the *sample covariance matrix*,

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{T-1}\sum_{t=1}^{T}(\boldsymbol{x}_t - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_t - \hat{\boldsymbol{\mu}})^{\mathsf{T}}, \tag{3.3}$$

where the "hat" notation " ˆ " denotes estimation.

One important property is that these estimators are *unbiased*, namely,

$$\mathbb{E}\left[\hat{\boldsymbol{\mu}}\right] = \boldsymbol{\mu}, \qquad \mathbb{E}\left[\hat{\boldsymbol{\Sigma}}\right] = \boldsymbol{\Sigma}.$$

In words, the random estimates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ are centered around the true values. In fact, the factor of $1/(T-1)$ in the sample covariance matrix is precisely chosen so that the estimator is unbiased; if, instead, the more natural factor $1/T$ is used, then the estimator is biased: $\mathbb{E}\left[\hat{\boldsymbol{\Sigma}}\right] = \left(1 - \frac{1}{T}\right)\boldsymbol{\Sigma}$.

Another important property is that these estimators are *consistent*, that is, from the law of large numbers (Anderson, 2003; Papoulis, 1991) it follows that

$$\lim_{T\to\infty} \hat{\boldsymbol{\mu}} = \boldsymbol{\mu}, \qquad \lim_{T\to\infty} \hat{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}.$$

In words, the sample estimates converge to the true quantities as the number of observations T grows. Figure 3.2 shows how the estimation error indeed goes to zero as T grows for synthetic Gaussian data of dimension $N = 100$ (in particular, the normalized error is used, defined as $100 \times \|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}\|/\|\boldsymbol{\mu}\|$ for the case of the sample mean, and similarly for the sample covariance matrix).

These sample estimators $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ are easy to understand, simple to implement, and cheap in terms of computational cost. However, they are only good estimators for a large number of observations T; otherwise, the estimation error becomes unacceptable. In particular, the sample mean is a very inefficient estimator, producing very noisy estimates (Chopra & Ziemba, 1993; Meucci, 2005). This can be observed in Figure 3.2, where the normalized error of $\hat{\boldsymbol{\mu}}$ for $N = 100$ and $T = 500$ is over 100%, which means that the error is as large as the true $\boldsymbol{\mu}$. In practice, however, there are two main reasons why T cannot be chosen large enough to produce good estimations:

- *Lack of available historical data*: For example, if we use use daily stock data (i.e., 252 observations per year) and the universe size is, say, $N = 500$, then a popular rule of thumb suggests that we should use $T \approx 10 \times N$ observations, which means 20 years of data. Generally speaking, 20 years of data are rarely available for the whole universe of assets.
- *Lack of stationarity*: Even if we had available all the historical data we wanted, because financial data is not stationary over long periods of time, it does not make much sense to use such data: the market behavior and dynamics 20 years ago are too different from the current ones (cf. Chapter 2).

As a consequence, in a practical setting, the amount of data that can be used is very limited. But then, the estimates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ will be very noisy, particularly the sample mean. This is, in fact, the "Achilles' heel" of portfolio optimization: $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ will inevitably contain estimation

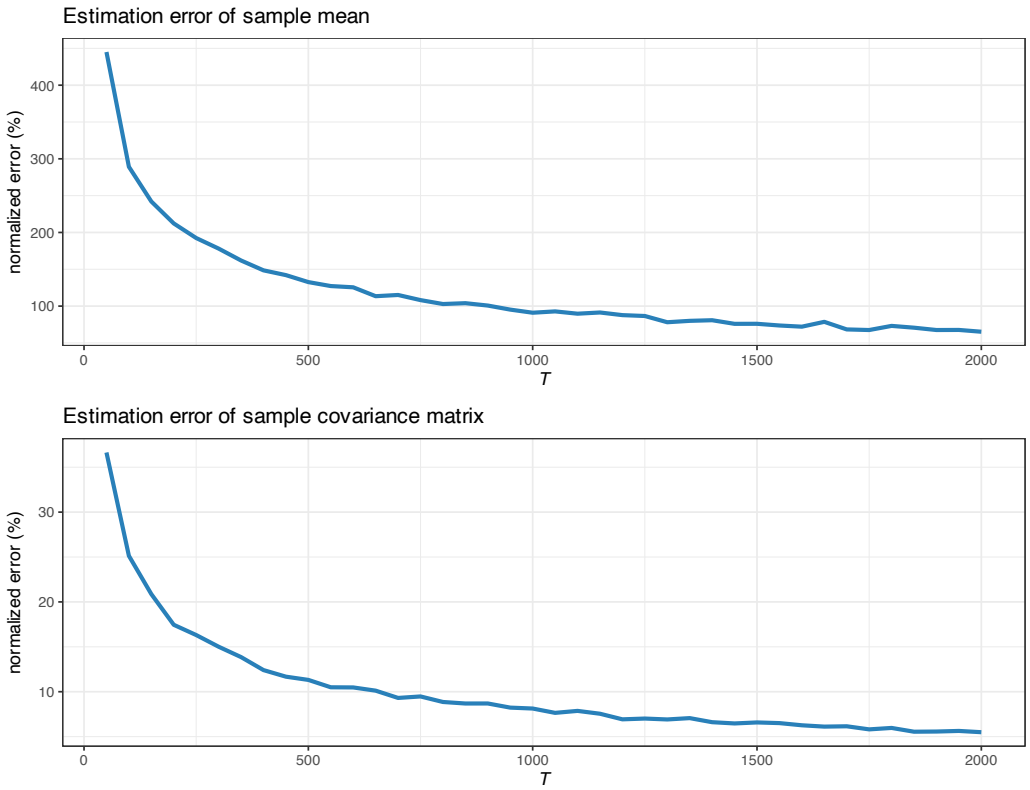


Figure 3.2 Estimation error of sample estimators vs. number of observations (for Gaussian data with $N = 100$).

noise which will lead to erratic portfolio designs. This is why Markowitz's portfolio has not been fully embraced by practitioners.

In the rest of this chapter, we will get a deeper perspective on why sample estimators perform so poorly with financial data and, then, we will explore a number of different ways to improve them.

3.3 Location Estimators

From the i.i.d. model in (3.1), the parameter $\boldsymbol{\mu}$ can be interpreted as the "center" or "location" around which the random points are distributed. Thus, it makes sense to try to estimate this location, which can be done in a variety of ways. The classical approach is based on least squares fitting, but it is very sensitive to extreme observations and missing values; thus, alternative robust multivariate location estimators have been thoroughly studied in the literature. In fact, this topic can be traced back to the 1960s (Huber, 1964; Maronna, 1976), as surveyed in Small (1990), Huber (2011), and Oja (2013).

3.3.1 Least Squares Estimator

The method of least squares (LS) dates back to 1795 when Gauss used it to study planetary motions. It deals with the linear model $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\epsilon}$ by solving the problem (Kay, 1993; Scharf, 1991)

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 ,$$

which has the closed-form solution $\boldsymbol{x}^\star = \left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^\mathsf{T}\boldsymbol{y}$.

Going back to the i.i.d. model in (3.1), we can now formulate the estimation of the center of the points, $\boldsymbol{x}_t = \boldsymbol{\mu} + \boldsymbol{\epsilon}_t$, as an LS problem:

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad \sum_{t=1}^{T} \|\boldsymbol{x}_t - \boldsymbol{\mu}\|_2^2,$$

whose solution, interestingly, coincides with the sample mean in (3.2):

$$\hat{\boldsymbol{\mu}} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t .$$

The sample mean is well known to suffer from lack of robustness against contaminated points or *outliers*, as well as against distributions with heavy tails (as will be empirically verified later in Figure 3.4).

3.3.2 Median Estimator

The median of a sample of points is the value separating the higher half from the lower half of the points. It may be thought of as the "middle" value of the points. The main advantage of the median compared to the mean (often described as the "average") is that it is not affected by a small proportion of extremely large or small values. Thus, it is a natural robust estimate and provides a better representation of a "typical" value. It is also possible to further improve on the median by using additional information from the data points such as the sample size, range, and quartile values (Wan et al., 2014).

In the multivariate case, there are multiple ways to extend the concept of median, as surveyed in Small (1990), Huber (2011), and Oja (2013), to obtain a natural robust estimate of the center of the distribution or set of points. One straightforward extension consists of employing the univariate median elementwise.

In the context of the i.i.d. model in (3.1), it turns out that this elementwise median can be formulated as the solution to the following problem:

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad \sum_{t=1}^{T} \|\boldsymbol{x}_t - \boldsymbol{\mu}\|_1,$$

where now the ℓ_1-norm is the measure of error instead of the squared ℓ_2-norm as in the case of the sample mean.

3.3.3 Spatial Median Estimator

Another way to extend the univariate median to the multivariate case is the so-called *spatial median* or *geometric median* (also known as the L_1 median) obtained as the solution to the problem

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad \sum_{t=1}^{T} \|\boldsymbol{x}_t - \boldsymbol{\mu}\|_2,$$

where now the ℓ_2-norm or Euclidean distance is the measure of error instead of the ℓ_1-norm or the squared ℓ_2-norm. Interestingly, the lack of the squaring operator has a huge effect; for example, the estimator for each element is no longer independent of the other elements (as is the case with the sample mean and elementwise median). The spatial median has the desired property that for $N = 1$ it reduces to the univariate median.

The spatial median is the solution to a convex second-order cone problem (SOCP) and can be solved with an SOCP solver. Alternatively, very efficient iterative algorithms can be derived by solving a sequence of LS problems via the majorization–minimization (MM) method (Sun et al., 2017); see also Section B.7 in Appendix B for details.

3.3.4 Numerical Experiments

Figure 3.3 illustrates the different location estimators in a two-dimensional ($N = 2$) case. Since this is just one realization, we cannot conclude which method is preferable.

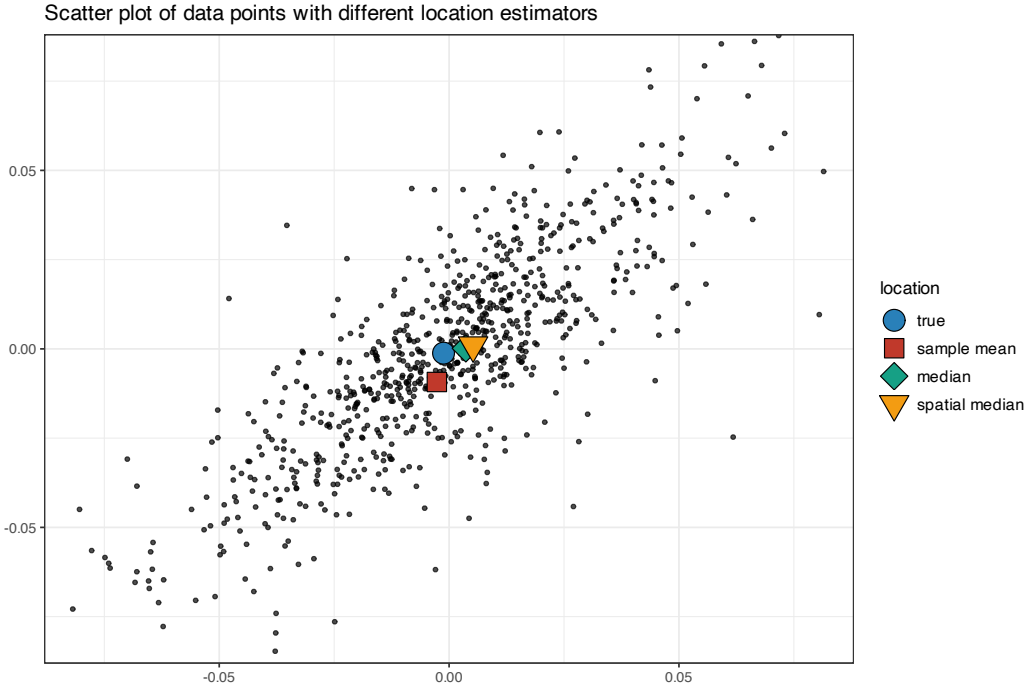


Figure 3.3 Illustration of different location estimators.

Figure 3.4 shows the estimation error of different location estimators as a function of the number of observations T for synthetic Gaussian and heavy-tailed data (the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ are taken as measured in typical stock market data). For Gaussian data, the sample mean is the best estimator (as further analyzed in Section 3.4), with the spatial median almost identical, and the elementwise median being the worst. For heavy-tailed data, the sample mean is as bad as the median and the spatial median remains the best. Overall, the spatial median seems to be the best option as it is robust to outliers and does not underperform under Gaussian data.

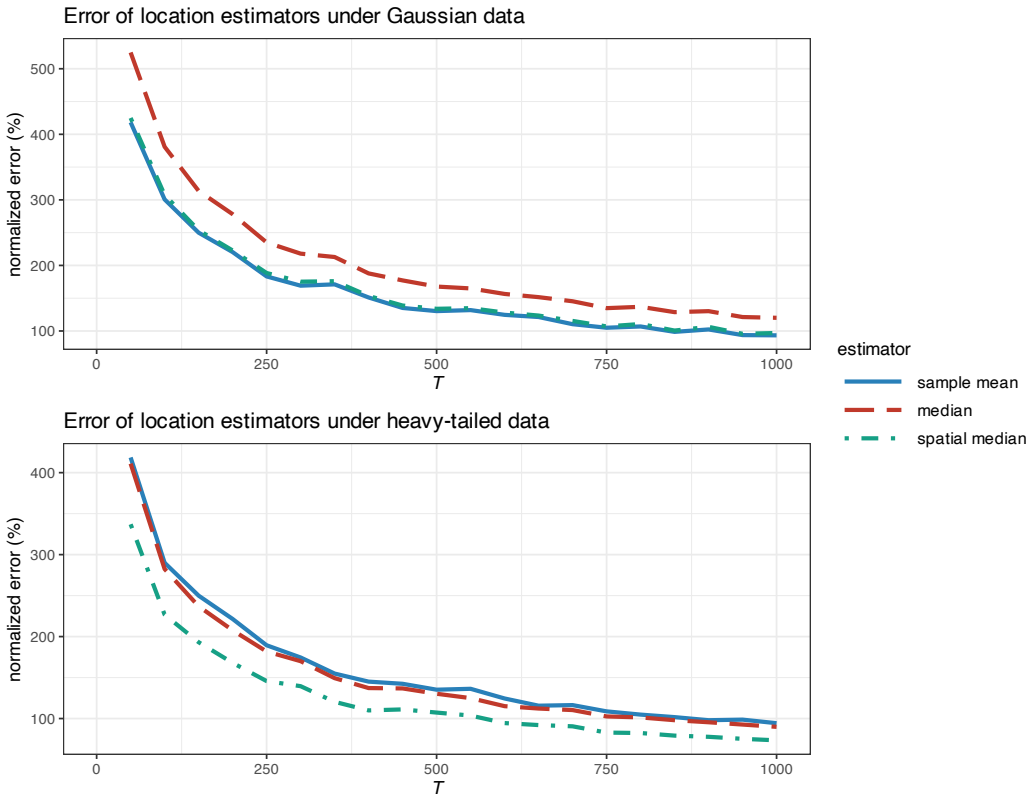


Figure 3.4 Estimation error of location estimators vs. number of observations (with $N = 100$).

Figure 3.5 examines in more detail the effect of the heavy tails in the estimation error for synthetic data following a t distribution with degrees of freedom ν. Financial data typically corresponds to ν on the order of somewhere between 4 and 5, which is significantly heavy-tailed, whereas a large value of ν corresponds to a Gaussian distribution. We can observe that the error for the sample mean remains similar regardless of ν. Interestingly, the spatial median is similar to the sample mean for large ν (Gaussian regime) whereas it becomes better for heavy tails. Again, this illustrates the robustness of the spatial median.

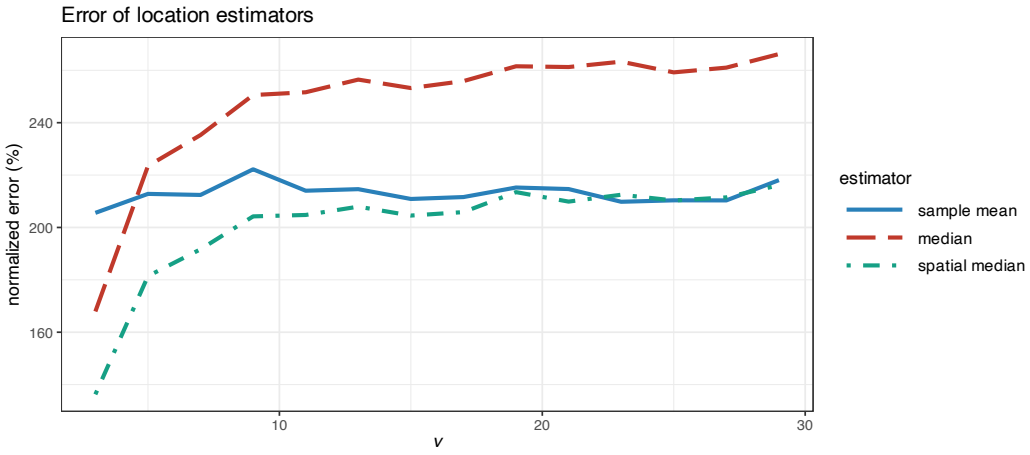


Figure 3.5 Estimation error of location estimators vs. degrees of freedom in a t distribution (with $T = 200$ and $N = 100$).

3.4 Gaussian ML Estimators

3.4.1 Preliminaries on ML Estimation

Maximum likelihood (ML) estimation is an important technique in *estimation theory* (Anderson, 2003; Kay, 1993; Scharf, 1991). The idea is very simple. Suppose the probability of a random variable $\boldsymbol{x}$ can be "measured" by the probability distribution function (pdf) f.[2] Then the probability of a series of T independent given observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ can be measured by the product $f(\boldsymbol{x}_1) \times \cdots \times f(\boldsymbol{x}_T)$. Suppose now that we have to guess which of two possible distributions, f_1 and f_2, is more likely to have produced these observations. It seems reasonable to choose as most likely the one that gives the largest probability of observing these observations. Now, suppose we have a family of possible distributions parameterized by the parameter vector $\boldsymbol{\theta}$: $f_{\boldsymbol{\theta}}$, which is called *likelihood function* when viewed as a function of $\boldsymbol{\theta}$ for the given observations. Again, it seems reasonable to choose as the most likely $\boldsymbol{\theta}$ the one that gives the largest probability of observing these observations. This is precisely the essence of ML estimation and can be written as the following optimization problem:

$$\underset{\boldsymbol{\theta}}{\text{maximize}} \quad f_{\boldsymbol{\theta}}(\boldsymbol{x}_1) \times \cdots \times f_{\boldsymbol{\theta}}(\boldsymbol{x}_T),$$

where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ denote the T given observations. For mathematical convenience, ML estimation is equivalently formulated as the maximization of the log-likelihood function:

$$\underset{\boldsymbol{\theta}}{\text{maximize}} \quad \sum_{t=1}^{T} \log f_{\boldsymbol{\theta}}(\boldsymbol{x}_t).$$

The ML estimator (MLE) enjoys many desirable theoretical properties that make it an *asymptotically optimal* estimator (asymptotic in the number of observations T). More specifically, the MLE is asymptotically unbiased and it asymptotically attains the Cramér–Rao

[2] To be precise, the term $f(\boldsymbol{x}_0)d\boldsymbol{x}$ gives the probability of observing $\boldsymbol{x}$ in a region centered around $\boldsymbol{x}_0$ with volume $d\boldsymbol{x}$.

bound (which gives the lowest possible variance attainable by an unbiased estimator); in other words, it is *asymptotically efficient* (Kay, 1993; Scharf, 1991). In practice, however, the key question is how large T has to be for the asymptotic properties to effectively hold.

3.4.2 Gaussian ML Estimation

The pdf corresponding to the i.i.d. model (3.1), assuming that the residual follows a multivariate normal or Gaussian distribution, is

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right), \tag{3.4}$$

from which we can see that the parameters of the model are $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Given T observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$, the MLE can then be formulated (ignoring irrelevant constant terms) as

$$\underset{\boldsymbol{\mu},\boldsymbol{\Sigma}}{\text{minimize}} \quad \log\det(\boldsymbol{\Sigma}) + \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{x}_t-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_t-\boldsymbol{\mu}),$$

where $\log\det(\cdot)$ denotes the logarithm of the determinant of a matrix.

Setting the gradient of the objective function with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}^{-1}$ to zero leads to

$$\frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{x}-\boldsymbol{\mu}) = \mathbf{0},$$
$$-\boldsymbol{\Sigma} + \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{x}_t-\boldsymbol{\mu})(\boldsymbol{x}_t-\boldsymbol{\mu})^\mathsf{T} = \mathbf{0},$$

which results in the following estimators for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$:

$$\begin{aligned}\hat{\boldsymbol{\mu}} &= \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t,\\ \hat{\boldsymbol{\Sigma}} &= \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{x}_t-\boldsymbol{\mu})(\boldsymbol{x}_t-\boldsymbol{\mu})^\mathsf{T}.\end{aligned} \tag{3.5}$$

Interestingly, these estimators coincide with the sample estimators in (3.2) and (3.3), apart from the factor $1/T$ instead of the factor $1/(T-1)$. The ML estimator of the covariance matrix is biased since $\mathbb{E}\left[\hat{\boldsymbol{\Sigma}}\right] = \left(1-\frac{1}{T}\right)\boldsymbol{\Sigma}$; however, it is asymptotically unbiased as $T \to \infty$ (as already expected from the asymptotic optimality of ML).

At first, it may seem that we have not achieved anything new, since we had already derived the same estimators from the perspective of sample estimators and from the perspective of least squares. However, after more careful thought, we have actually learned that the sample estimators are optimal when the data is Gaussian distributed, but not otherwise. That is, when the data has a different distribution, the optimal ML estimators can be expected to be different, as explored later in Section 3.5.

3.4.3 Numerical Experiments

Figure 3.6 shows the estimation error of the ML estimators for the mean and covariance matrix as a function of the number of observations T for synthetic Gaussian and heavy-tailed data. We can observe how the estimation of the covariance matrix $\boldsymbol{\Sigma}$ becomes much worse for heavy-tailed data, whereas the estimation of the mean $\boldsymbol{\mu}$ remains similar.

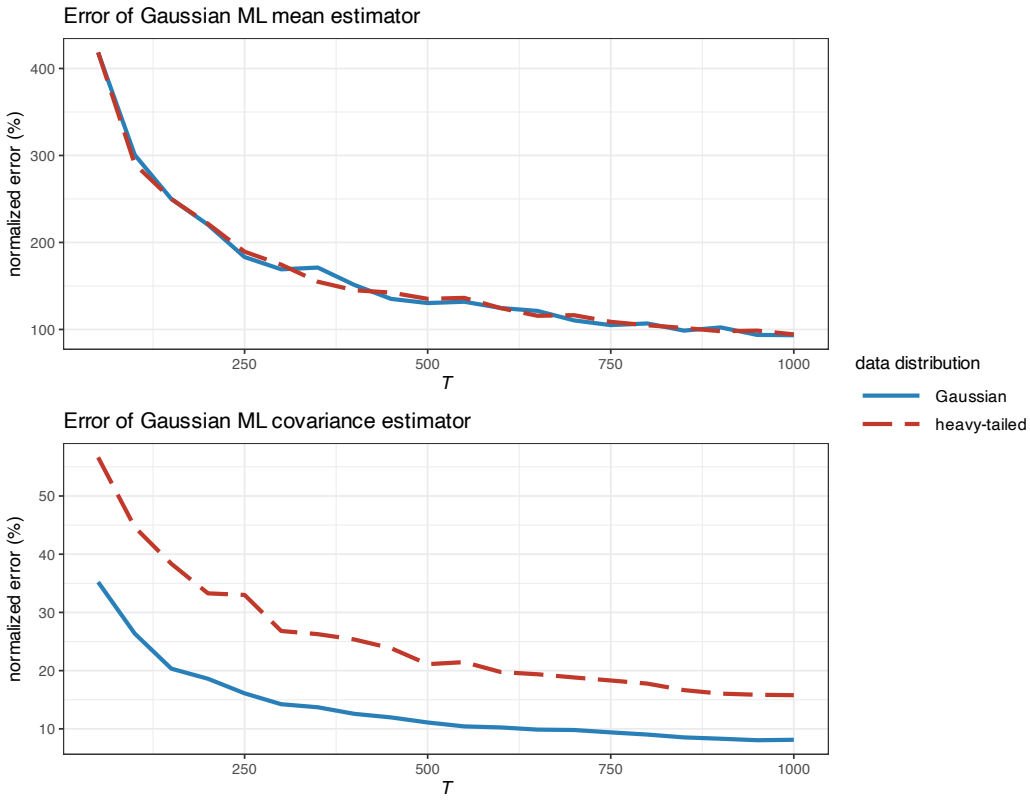


Figure 3.6 Estimation error of Gaussian ML estimators vs. number of observations (with $N = 100$).

Figure 3.7 examines in more detail the effect of the heavy tails in the estimation error for synthetic data following a t distribution with degrees of freedom ν. We can confirm that the error in the estimation of the mean remains the same, whereas the estimation of the covariance matrix becomes worse as the distribution exhibits heavier tails. Nevertheless, we should not forget that the size of the error is one order of magnitude larger for the mean than for the covariance matrix.

3.5 Heavy-Tailed ML Estimators

Gaussian ML estimators are optimal, in the sense of maximizing the likelihood of the observations, whenever data follow the Gaussian distribution. However, if data follow a heavy-tailed distribution – like it is the case with financial data – then we need some further

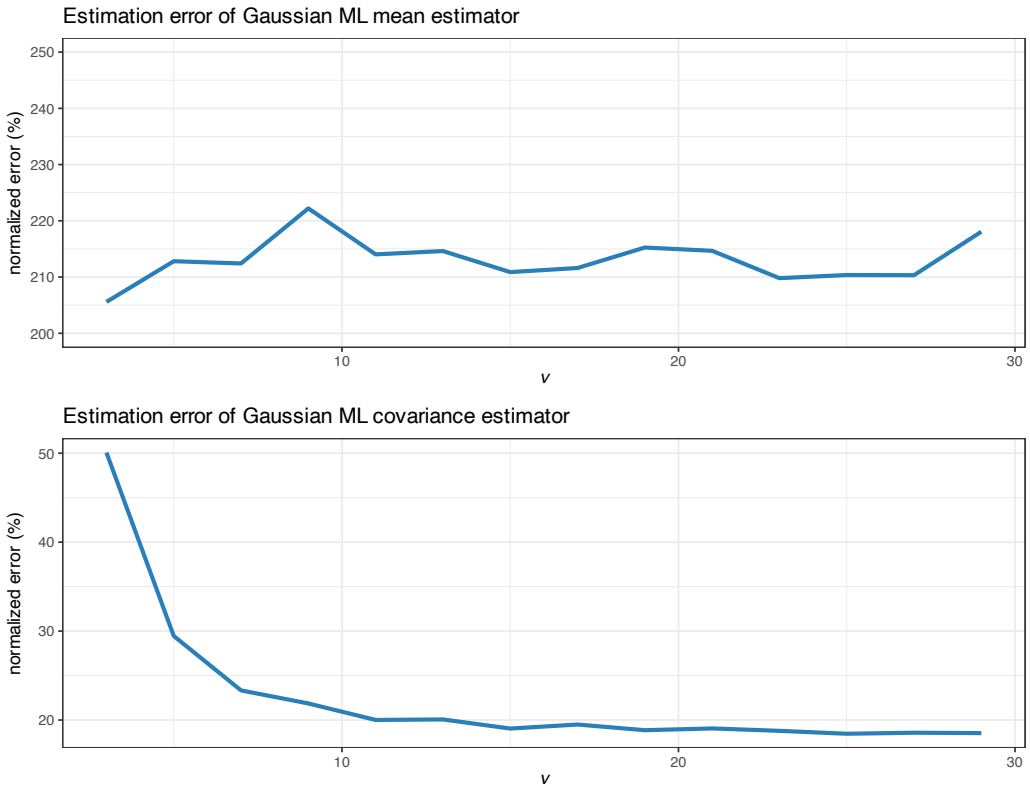


Figure 3.7 Estimation error of Gaussian ML estimators vs. degrees of freedom in a t distribution (with $T = 200$ and $N = 100$).

understanding of the potentially detrimental effect. On the one hand, since the ML estimators coincide with the sample estimators in Section 3.2, we know they are unbiased and consistent, which are desirable properties. But, on the other hand, is this sufficient or can we do better?

3.5.1 The Failure of Gaussian ML Estimators

As explored in Section 3.4, Figure 3.6 demonstrates that the effect of heavy tails in the estimation of the covariance matrix is significant, whereas it is almost nonexistent for the estimation of the mean. Figure 3.7 further shows the error as a function of how heavy the tails are (small ν represents heavy-tailed distributions whereas large ν tends to a Gaussian distribution).

To further understand the detrimental effect of heavy tails, Figure 3.8 illustrates this effect, as well as the effect of outliers in otherwise Gaussian data. In this example, $T = 10$ data points are used for the estimation of the two-dimensional covariance matrix, which satisfies the ratio $T/N = 5$. It is very clear that, while for Gaussian data the Gaussian ML estimator (or sample covariance matrix) follows the true covariance matrix very closely, once we include a single outlier in the Gaussian data or we use heavy-tailed data, the estimation is disastrous. Still, most practitioners and academics adopt the sample covariance matrix due to its simplicity.

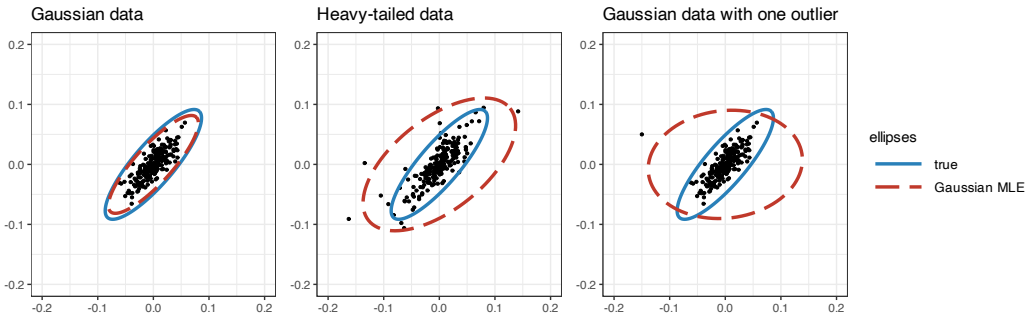


Figure 3.8 Effect of heavy tails and outliers in the Gaussian ML covariance matrix estimator.

3.5.2 Heavy-Tailed ML Estimation

We have empirically observed that the Gaussian MLE for the covariance matrix significantly degrades when the data distribution exhibits heavy tails (see Figures 3.6, 3.7, and 3.8). But this is not unexpected since the sample covariance matrix is optimal only under the Gaussian distribution as derived in Section 3.4. Thus, to derive an improved estimator, we should drop the Gaussian assumption and instead consider a heavy-tailed distribution in the derivation of optimal ML estimators. There are many families of distributions with heavy tails (McNeil et al., 2015) and, for simplicity, we will focus our attention on the Student t distribution or, simply, t distribution, which is widely used to model heavy tails via the parameter ν. It is worth noting that using any other heavy-tailed distribution shows little difference.

The pdf corresponding to the i.i.d. model (3.1), assuming now that the residual term follows a multivariate t distribution, is

$$f(\boldsymbol{x}) = \frac{\Gamma((\nu+N)/2)}{\Gamma(\nu/2)\sqrt{(\nu\pi)^N|\boldsymbol{\Sigma}|}}\left(1+\frac{1}{\nu}(\boldsymbol{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)^{-(\nu+N)/2},$$

where $\Gamma(\cdot)$ is the gamma function,[3] $\boldsymbol{\mu}$ is the location parameter (which coincides with the mean vector for $\nu > 1$ as in the Gaussian case and is undefined otherwise), $\boldsymbol{\Sigma}$ denotes the *scatter matrix* (not to be confused with the covariance matrix, which can now be obtained as $\frac{\nu}{\nu-2}\boldsymbol{\Sigma}$ if $\nu > 2$ and is undefined otherwise), and ν denotes the "degrees of freedom" that controls how heavy or thick the tails are ($\nu \approx 4$ corresponds to significantly heavy tails, whereas $\nu \to \infty$ corresponds to the Gaussian distribution). Thus, the parameters of this model are $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu)$, which includes the extra scalar parameter ν compared to the Gaussian case in (3.4). This parameter can either be fixed to some reasonable value (financial data typically satisfies $\nu \approx 4$ or $\nu \approx 5$) or can be estimated together with the other parameters.

[3] The gamma function is defined as $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}\,\mathrm{d}t$. For a positive integer n, it corresponds to the factorial function $\Gamma(n) = (n-1)!$.

The MLE can then be formulated, given T observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$, as

$$\underset{\boldsymbol{\mu},\boldsymbol{\Sigma},\nu}{\text{minimize}} \quad \log\det(\boldsymbol{\Sigma}) + \frac{\nu+N}{T}\sum_{t=1}^{T}\log\left(1+\frac{1}{\nu}(\boldsymbol{x}_t-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_t-\boldsymbol{\mu})\right) \\ +2\log\Gamma(\nu/2) + N\log(\nu) - 2\log\Gamma\left(\frac{\nu+N}{2}\right).$$

For simplicity we will fix the parameter $\nu = 4$, and then only the first two terms in the problem formulation become relevant in the estimation of the remaining parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Setting the gradient of the objective function with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}^{-1}$ to zero leads to

$$\frac{1}{T}\sum_{t=1}^{T}\frac{\nu+N}{\nu+(\boldsymbol{x}_t-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_t-\boldsymbol{\mu})}(\boldsymbol{x}_t-\boldsymbol{\mu}) = \mathbf{0},$$
$$-\boldsymbol{\Sigma} + \frac{1}{T}\sum_{t=1}^{T}\frac{\nu+N}{\nu+(\boldsymbol{x}_t-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_t-\boldsymbol{\mu})}(\boldsymbol{x}_t-\boldsymbol{\mu})(\boldsymbol{x}_t-\boldsymbol{\mu})^\top = \mathbf{0},$$

which can be conveniently rewritten as the following fixed-point equations for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$:

$$\begin{aligned}\boldsymbol{\mu} &= \frac{\frac{1}{T}\sum_{t=1}^{T} w_t(\boldsymbol{\mu},\boldsymbol{\Sigma})\times\boldsymbol{x}_t}{\frac{1}{T}\sum_{t=1}^{T} w_t(\boldsymbol{\mu},\boldsymbol{\Sigma})},\\ \boldsymbol{\Sigma} &= \frac{1}{T}\sum_{t=1}^{T} w_t(\boldsymbol{\mu},\boldsymbol{\Sigma})\times(\boldsymbol{x}_t-\boldsymbol{\mu})(\boldsymbol{x}_t-\boldsymbol{\mu})^\top,\end{aligned} \tag{3.6}$$

where we define the weights

$$w_t(\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{\nu+N}{\nu+(\boldsymbol{x}_t-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_t-\boldsymbol{\mu})}. \tag{3.7}$$

It is important to remark that (3.6) are fixed-point equations because the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ appear on both sides of the equalities, which makes their calculation not trivial. Existence of a solution is guaranteed if $T > N + 1$ (Kent & Tyler, 1991); conditions for existence and uniqueness with optional shrinkage were derived in Sun et al. (2015).

Nevertheless, these fixed-point equations have a beautiful interpretation: if one assumes the weights w_t to be known, then the expressions in (3.6) become a weighted version of the Gaussian MLE expressions in (3.5). Interestingly, the weights w_t have the insightful interpretation that they become smaller as the point $\boldsymbol{x}_t$ is further away from the center $\boldsymbol{\mu}$, which means that they automatically down-weight the outliers. Thus, we can expect these estimators to be robust to outliers, unlike the Gaussian ML estimators in (3.5) whose performance can be severely degraded in the presence of outliers. Observe that as $\nu \to \infty$, i.e., as the distribution becomes Gaussian, the weights in (3.7) tend to 1 (unweighted case) as expected.

In practice, a simple way to solve the fixed-point equations in (3.6) is via an iterative process whereby the weights are fixed to the most current value and then $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are updated.[4]

[4] The R package `fitHeavyTail` contains the function `fit_mvt()` to solve the fixed-point equations (3.6)-(3.7) iteratively via (3.8) (Palomar et al., 2023).

Specifically, the iterations for $k = 0, 1, 2, \ldots$ are given as follows:

$$\begin{aligned} \boldsymbol{\mu}^{k+1} &= \frac{\frac{1}{T}\sum_{t=1}^{T} w_t(\boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k) \times \boldsymbol{x}_t}{\frac{1}{T}\sum_{t=1}^{T} w_t(\boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k)}, \\ \boldsymbol{\Sigma}^{k+1} &= \frac{1}{T}\sum_{t=1}^{T} w_t(\boldsymbol{\mu}^{k+1}, \boldsymbol{\Sigma}^k) \times (\boldsymbol{x}_t - \boldsymbol{\mu}^{k+1})(\boldsymbol{x}_t - \boldsymbol{\mu}^{k+1})^{\mathsf{T}}. \end{aligned} \tag{3.8}$$

The iterative updates in (3.8) can be formally derived from the ML formulation and shown to converge to the optimal solution by means of the MM algorithmic framework. For details on MM, the reader is referred to Sun et al. (2017) and Section B.7 in Appendix B. For the specific derivation of (3.8), together with technical conditions for the convergence of the algorithm, details can be found in Kent et al. (1994) and in Sun et al. (2015) from the MM perspective. For convenience, this MM-based method is summarized in Algorithm 3.1. In addition, the estimation of ν is further considered in Liu and Rubin (1995) and acceleration methods for faster convergence are derived in Liu et al. (1998).

Algorithm 3.1: MM-based method to solve the heavy-tailed ML fixed-point equations in (3.6).

1: Choose initial point $(\boldsymbol{\mu}^0, \boldsymbol{\Sigma}^0)$;
2: Set $k \leftarrow 0$;
3: **repeat**
4: Iterate the weighted sample mean and sample covariance matrix as

$$\begin{aligned} \boldsymbol{\mu}^{k+1} &= \frac{\frac{1}{T}\sum_{t=1}^{T} w_t(\boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k) \times \boldsymbol{x}_t}{\frac{1}{T}\sum_{t=1}^{T} w_t(\boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k)}, \\ \boldsymbol{\Sigma}^{k+1} &= \frac{1}{T}\sum_{t=1}^{T} w_t(\boldsymbol{\mu}^{k+1}, \boldsymbol{\Sigma}^k) \times (\boldsymbol{x}_t - \boldsymbol{\mu}^{k+1})(\boldsymbol{x}_t - \boldsymbol{\mu}^{k+1})^{\mathsf{T}}, \end{aligned}$$

where the weights are defined in (3.7);
5: $k \leftarrow k + 1$;
6: **until** convergence;

Figure 3.9 illustrates the robustness of the heavy-tailed ML estimator, compared with the Gaussian ML estimator, under data with heavy tails and outliers. The difference observed is quite dramatic and should serve as a red flag to practitioners for using the sample covariance matrix when dealing with heavy-tailed data.

3.5.3 Robust Estimators

What happens if the true distribution that generates the data deviates slightly from the assumed – typically Gaussian – one? Estimators that are not very sensitive to outliers or distribution contamination are generally referred to as *robust estimators*.

The robustness of estimators can be objectively measured in different ways; notably, with the

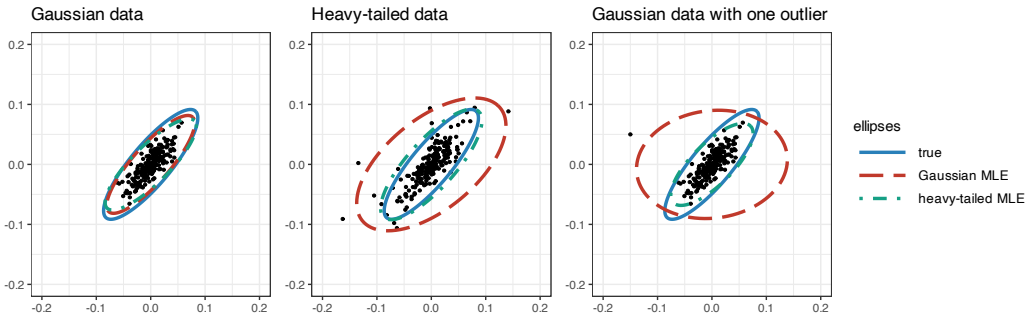


Figure 3.9 Effect of heavy tails and outliers in heavy-tailed ML covariance matrix estimator.

influence function, which measures the effect when the true distribution slightly deviates from the assumed one, and the *breakdown point*, which is the minimum fraction of contaminated data points that can render the estimator useless.

As already discussed in Section 3.4, sample estimators or Gaussian ML estimators are not robust against deviations from the Gaussian distribution. It is well known that they are very sensitive to the tails of the distribution (Huber, 1964; Maronna, 1976). In fact, their influence function is unbounded, meaning that an infinitesimal point mass contamination can have an arbitrarily large influence. In addition, a single contaminated point can ruin the sample mean or the sample covariance matrix, that is, the breakdown point is $1/T$. For reference, the median has a breakdown of around 0.5, that is, one needs more than 50% of the points contaminated to ruin it. On the other hand, as will be further elaborated next, the heavy-tailed ML estimators from Section 3.5.2 can be shown to be robust estimators.

Some classical early references on robust estimation are Huber (1964) for the univariate case and Maronna (1976) for the multivariate case, whereas more modern surveys include Maronna et al. (2006), Huber (2011), Wiesel and Zhang (2014), and Chapter 4 in Zoubir et al. (2018).

M-Estimators

The term M-estimators for robust estimation goes back to the 1960s (Huber, 1964). In a nutshell, M-estimators for the location and scatter parameters, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, are defined by the following fixed-point equations:

$$\begin{aligned} \frac{1}{T}\sum_{t=1}^{T} u_1\left(\sqrt{(\boldsymbol{x}_t-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_t-\boldsymbol{\mu})}\right)(\boldsymbol{x}_t-\boldsymbol{\mu}) &= \mathbf{0}, \\ \frac{1}{T}\sum_{t=1}^{T} u_2\left((\boldsymbol{x}_t-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_t-\boldsymbol{\mu})\right)(\boldsymbol{x}_t-\boldsymbol{\mu})(\boldsymbol{x}_t-\boldsymbol{\mu})^{\mathsf{T}} &= \boldsymbol{\Sigma}, \end{aligned} \tag{3.9}$$

where $u_1(\cdot)$ and $u_2(\cdot)$ are weight functions satisfying some conditions (Maronna, 1976; Maronna et al., 2006).

M-estimators are a generalization of the maximum likelihood estimators and can be regarded

as the weighted sample mean and the weighted sample covariance matrix. In terms of robustness, they have a desirable bounded influence function, although the breakdown point is still relatively low (Maronna, 1976; Maronna et al., 2006). Other estimators, such as the minimum volume ellipsoid and minimum covariance determinant, have higher breakdown points.

The Gaussian ML estimators can be obtained from the M-estimators in (3.9) with the trivial choice of weight functions $u_1(s) = u_2(s) = 1$.

A notable common choice to obtain robust estimators is to choose the weight functions based on Huber's ψ-function $\psi(z, k) = \max(-k, \min(z, k))$, where k is a positive constant that caps the argument z from above and below, as follows (Maronna, 1976):

$$\begin{aligned} u_1(s) &= \psi(z, k)/s, \\ u_2(s) &= \psi(z, k^2)/(\beta s), \end{aligned}$$

where β is a properly chosen constant.

Interestingly, the M-estimators in (3.9) particularize to the heavy-tailed ML estimators derived in (3.7) for the choice

$$u_1(s) = u_2(s^2) = \frac{\nu + N}{\nu + s^2}.$$

Tyler's Estimator

In 1987, Tyler proposed an estimator for the scatter matrix (which is proportional to the covariance matrix) for heavy-tailed distributions (Tyler, 1987). The idea is very simple and ingenious, as described next. Interestingly, Tyler's estimator can be shown to be the most robust version of an M-estimator.

Suppose the random variable $\boldsymbol{x}$ follows a zero-mean elliptical distribution – which means that the distribution depends on $\boldsymbol{x}$ through the term $\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$. If the mean is not zero, then it has to be estimated with some location estimator, as described in Section 3.3, and then subtracted from the observations so that they have zero mean.

The key idea is to normalize the observations

$$\boldsymbol{s}_t = \frac{\boldsymbol{x}_t}{\|\boldsymbol{x}_t\|_2}$$

and then use ML based on these normalized points. The surprising fact is that the pdf of the normalized points can be analytically derived – known as angular distribution – as

$$f(\boldsymbol{s}) \propto \frac{1}{\sqrt{|\boldsymbol{\Sigma}|}} \left(\boldsymbol{s}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{s}\right)^{-N/2},$$

which is independent of the shape of the tails and still contains the parameter $\boldsymbol{\Sigma}$ which we wish to estimate.

The MLE can then be formulated, given T observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ and noting that

$\left(\boldsymbol{s}_t^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{s}_t\right)^{-N/2} \propto \left(\boldsymbol{x}_t^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_t\right)^{-N/2}$, as

$$\underset{\boldsymbol{\Sigma}}{\text{minimize}} \quad \log\det(\boldsymbol{\Sigma}) + \frac{N}{T}\sum_{t=1}^{T}\log\left(\boldsymbol{x}_t^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_t\right).$$

Setting the gradient with respect to $\boldsymbol{\Sigma}^{-1}$ leads to the following fixed-point equation (which has the same form as the one for the heavy-tailed MLE in (3.6)):

$$\boldsymbol{\Sigma} = \frac{1}{T}\sum_{t=1}^{T} w_t(\boldsymbol{\Sigma}) \times \boldsymbol{x}_t\boldsymbol{x}_t^\mathsf{T},$$

where the weights are now given by

$$w_t(\boldsymbol{\Sigma}) = \frac{N}{\boldsymbol{x}_t^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_t}. \tag{3.10}$$

Observe that these weights behave similarly to those in (3.7), in the sense of down-weighting the outliers and making the estimator robust. Existence of a solution is guaranteed if $T > N$ (Tyler, 1987); conditions for existence and uniqueness with optional shrinkage were derived in Chen et al. (2011), Wiesel (2012), and Sun et al. (2014).

This fixed-point equation can be solved iteratively like in the case of the heavy-tailed MLE in Section 3.5.2, see Tyler (1987) and Sun et al. (2014). However, it is important to realize that Tyler's method can only estimate the parameter $\boldsymbol{\Sigma}$ up to a scaling factor, as can be observed from the invariance of the likelihood function with respect to a scaling factor in $\boldsymbol{\Sigma}$. In fact, this should not be surprising since the normalization of the original points destroys the information of such a scaling factor. In practice, the scaling factor κ can be recovered with some simple heuristic to enforce

$$\operatorname{diag}(\kappa \times \boldsymbol{\Sigma}) \approx \hat{\boldsymbol{\sigma}}^2,$$

where $\hat{\boldsymbol{\sigma}}^2$ denotes some robust estimation of the variances of the assets; for example,

$$\kappa = \frac{1}{N}\mathbf{1}^\mathsf{T}\left(\hat{\boldsymbol{\sigma}}^2/\operatorname{diag}(\boldsymbol{\Sigma})\right).$$

3.5.4 Numerical Experiments

We now proceed to a final comparison of the presented robust estimators benchmarked against the traditional and popular Gaussian-based estimator. In particular, the following estimators are considered for the mean and covariance matrix:

- Gaussian MLE
- heavy-tailed MLE
- Tyler's estimator for the covariance matrix (with spatial median for the location).

Figure 3.10 shows the estimation error as a function of the number of observations T for synthetic heavy-tailed data (t distribution with $\nu = 4$). The heavy-tailed MLE is the best, followed closely by the Tyler estimator (with the spatial median estimator for the mean), while the Gaussian MLE is the worst by far.

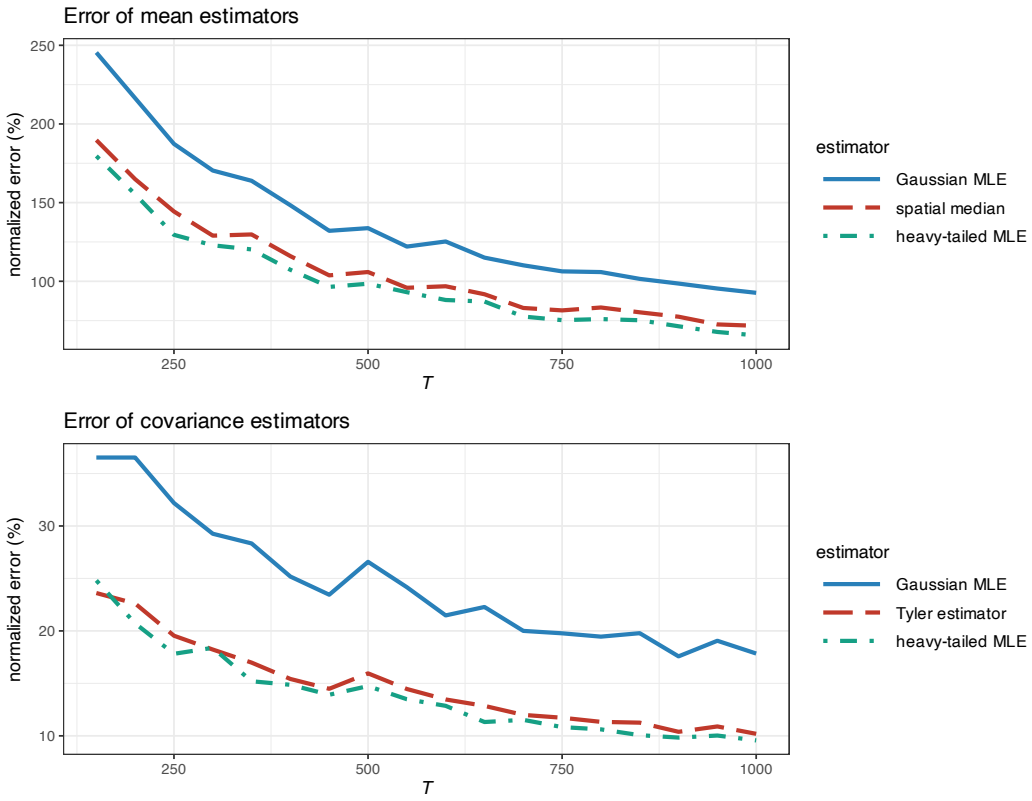


Figure 3.10 Estimation error of different ML estimators vs. number of observations (for t-distributed heavy-tailed data with $\nu = 4$ and $N = 100$).

Figure 3.11 examines in more detail the effect of the heavy tails in the estimation error for synthetic data following a t distribution with degrees of freedom ν. We can confirm that for Gaussian tails, the Gaussian MLE is similar to the heavy-tailed MLE; however, as the tails become heavier (smaller values of ν), the difference becomes quite significant in favor of the robust heavy-tailed MLE.

The final conclusion could not be more clear: financial data is heavy-tailed and one must necessarily use robust heavy-tailed ML estimators (like the one summarized in Algorithm 3.1). Interestingly, the computational cost of the robust estimators is not much higher than the traditional sample estimators because the algorithm converges in just a few iterations (each iteration has a cost comparable to the sample estimators). Indeed, Figure 3.12 indicates that the algorithm converges in three to five iterations in this particular example with heavy-tailed data (following a t distribution with $\nu = 4$) with $T = 200$ observations and $N = 100$ assets.

Nevertheless, it is important to emphasize that the errors in the estimation of the mean vector $\boldsymbol{\mu}$ based on historical data are extremely large (see Figures 3.10 and 3.11), to the point that such estimations may become useless in practice. This is precisely why practitioners typically obtain factors from data providers (at a high premium) and then use them to estimate $\boldsymbol{\mu}$ via regression. Alternatively, many portfolio designs that ignore any estimation of $\boldsymbol{\mu}$ are quite

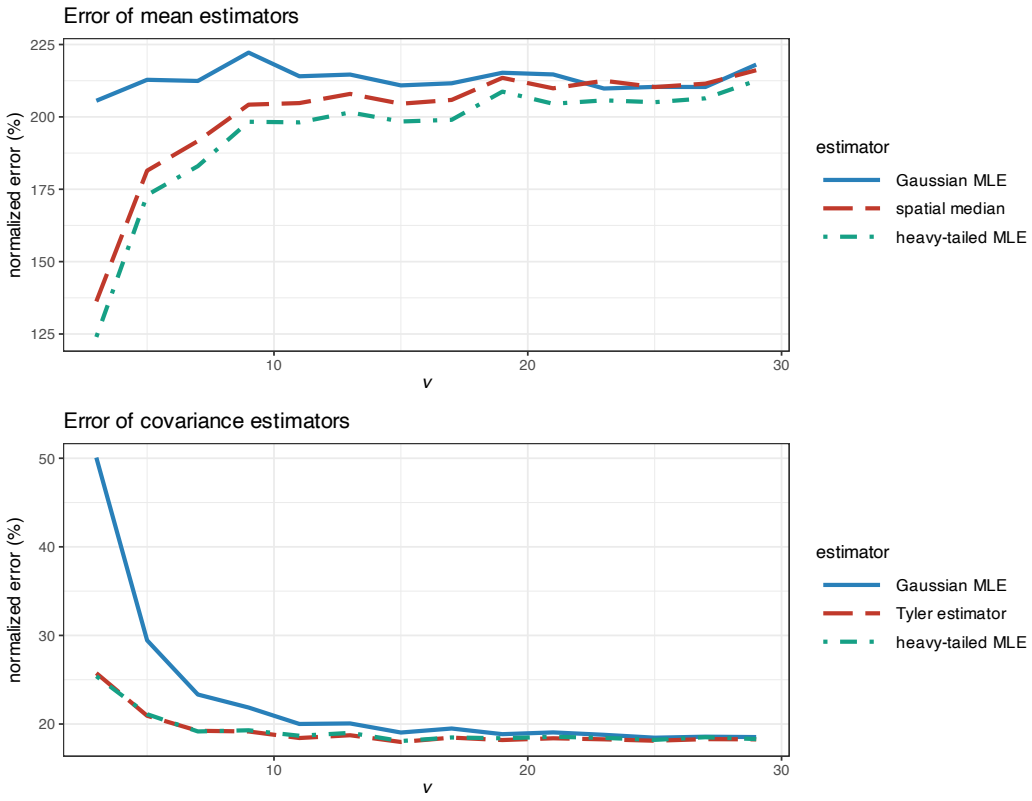


Figure 3.11 Estimation error of different ML estimators vs. degrees of freedom in a t distribution (with $T = 200$ and $N = 100$).

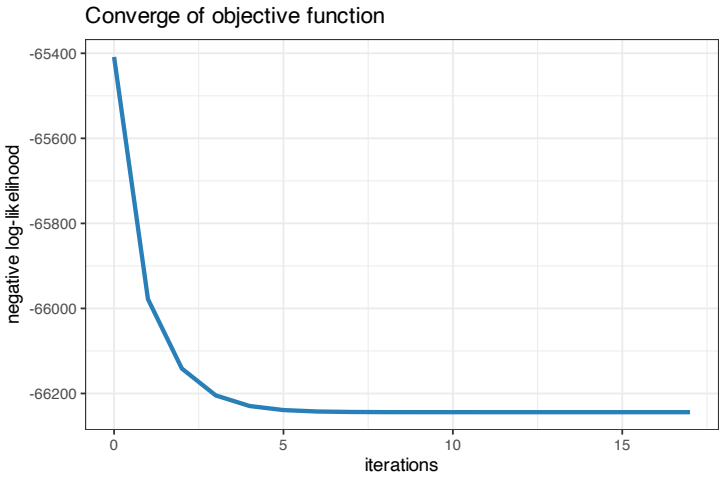


Figure 3.12 Convergence of robust heavy-tailed ML estimators.

common, for example the global minimum variance portfolio (see Chapter 6) and the risk parity portfolio (see Chapter 11).

3.6 Prior Information: Shrinkage, Factor Models, and Black–Litterman

All the estimators thus far surveyed in this chapter are based purely on the T historical data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$. Unfortunately, in many practical settings, as discussed in Section 3.2, the number of observations is not large enough to achieve proper estimation of the model parameters with a sufficiently small error. Researchers and practitioners have spent decades trying to deal with this issue and devising a variety of mechanisms to improve the estimators. The basic recipe is to somehow incorporate any prior information that one may have available. We will next give an overview of three notable ways to incorporate prior information into the parameter estimation process:

- *shrinkage*: to incorporate prior knowledge on the parameters in the form of targets;
- *factor modeling*: to incorporate structural information; and
- *Black–Litterman*: to combine the data with discretionary views.

Each of these approaches deserves a whole chapter – if not a whole book – but we will merely scratch the tip of the iceberg here, while providing key references for the interested reader to probe further.

3.6.1 Shrinkage

Shrinkage is a popular technique to reduce the estimation error by introducing a *bias* in the estimator. In statistics, this idea goes back to 1955 with Stein's seminal publication (Stein, 1955). In the financial area, it was popularized by its application to shrinkage in the covariance matrix in the early 2000s (Ledoit & Wolf, 2004) and it is now covered in many textbooks (Meucci, 2005) and surveys (Bun et al., 2017).

The mean squared error (MSE) of an estimator can be separated into two terms: the bias and the variance. This is a basic concept in estimation theory referred to as the *bias–variance trade-off* (Kay, 1993; Scharf, 1991). Mathematically, for a given parameter $\boldsymbol{\theta}$ and an estimator $\hat{\boldsymbol{\theta}}$, the bias–variance trade-off reads:

$$\begin{aligned}\mathrm{MSE}(\hat{\boldsymbol{\theta}}) &\triangleq \mathbb{E}\left[\left\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\right\|^2\right] \\ &= \mathbb{E}\left[\left\|\hat{\boldsymbol{\theta}} - \mathbb{E}\left[\hat{\boldsymbol{\theta}}\right]\right\|^2\right] + \left\|\mathbb{E}\left[\hat{\boldsymbol{\theta}}\right] - \boldsymbol{\theta}\right\|^2 \\ &= \mathrm{Var}\left(\hat{\boldsymbol{\theta}}\right) + \mathrm{Bias}^2\left(\hat{\boldsymbol{\theta}}\right).\end{aligned}$$

In the small-sample regime (i.e., when the number of observations is small), the main source of error comes from the variance of the estimator (since the estimator is based on a small number of random samples). In the large-sample regime, on the other hand, one may expect the variance of the estimator to be reduced and the bias to dominate the overall error.

Traditionally, unbiased estimators had always been desirable. To the surprise of the statistical community, Stein proved in 1955 in a seminal paper (Stein, 1955) that it might be advantageous

to allow for some bias in order to achieve a smaller overall error. This can be implemented by shrinking the estimator to some known target value. Mathematically, for a given estimator $\hat{\boldsymbol{\theta}}$ and some target $\boldsymbol{\theta}^{\text{tgt}}$ (i.e., the prior information), a shrinkage estimator is given by

$$\hat{\boldsymbol{\theta}}^{\text{sh}} = (1 - \rho)\, \hat{\boldsymbol{\theta}} + \rho\, \boldsymbol{\theta}^{\text{tgt}},$$

where ρ (with $0 \leq \rho \leq 1$) is the shrinkage trade-off parameter or shrinkage factor.

In practice, there are two important issues when implementing shrinkage:

- the choice of the target $\boldsymbol{\theta}^{\text{tgt}}$, which represents the prior information and, in a financial context, may come from discretionary views on the market;
- the choice of the shrinkage factor ρ, which may look like a trivial problem, but the reality is that tons of ink have been devoted to this topic in the literature.

While the choice of the target is important, it is perhaps surprising that the most critical part is the choice of the shrinkage factor ρ. The reason is that, no matter how poorly chosen the target is, a proper choice of the shrinkage factor can always weight the target in the right amount. There are two main philosophies when it comes to choosing the shrinkage factor ρ:

- *empirical choice* based on cross-validation, and
- *analytical choice* based on sophisticated mathematics.

In our context of financial data, the parameter $\boldsymbol{\theta}$ may represent the mean vector $\boldsymbol{\mu}$ or the covariance matrix $\boldsymbol{\Sigma}$, and the estimator $\hat{\boldsymbol{\theta}}$ may be, for example, the sample mean or the sample covariance matrix.

Shrinking the Mean Vector

Consider the mean vector $\boldsymbol{\mu}$ and the sample mean estimator in (3.2):

$$\hat{\boldsymbol{\mu}} = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{x}_t.$$

As we know from Section 3.2, the sample mean $\hat{\boldsymbol{\mu}}$ is an unbiased estimator, $\mathbb{E}\,[\hat{\boldsymbol{\mu}}] = \boldsymbol{\mu}$. In addition, from the central limit theorem, we can further characterize its distribution as

$$\hat{\boldsymbol{\mu}} \sim \mathcal{N}\left(\boldsymbol{\mu}, \frac{1}{T}\boldsymbol{\Sigma}\right)$$

and its MSE as

$$\mathbb{E}\left[\left\|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}\right\|^2\right] = \frac{1}{T}\text{Tr}(\boldsymbol{\Sigma}).$$

The startling result proved by Stein in 1955 is that, in terms of MSE, this approach is suboptimal and it is better to allow some bias in order to reduce the overall MSE, which can be achieved in the form of shrinkage.

The so-called *James–Stein estimator* (W. James & Stein, 1961) is

$$\hat{\boldsymbol{\mu}}^{\text{JS}} = (1 - \rho)\, \hat{\boldsymbol{\mu}} + \rho\, \boldsymbol{\mu}^{\text{tgt}},$$

where $\boldsymbol{\mu}^{\text{tgt}}$ is the target and ρ the shrinkage factor. To be precise, regardless of the chosen target $\boldsymbol{\mu}^{\text{tgt}}$, one can always improve the MSE,

$$\mathbb{E}\left[\left\|\hat{\boldsymbol{\mu}}^{\text{JS}} - \boldsymbol{\mu}\right\|^2\right] \leq \mathbb{E}\left[\left\|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}\right\|^2\right],$$

with a properly chosen ρ, such as (Jorion, 1986)

$$\rho = \frac{(N+2)}{(N+2) + T \times (\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^{\text{tgt}})^{\top} \boldsymbol{\Sigma}^{-1} (\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^{\text{tgt}})},$$

where, in practice, $\boldsymbol{\Sigma}$ can be replaced by an estimation $\hat{\boldsymbol{\Sigma}}$. The choice of ρ was also considered under heavy-tailed distributions in Srivastava and Bilodeau (1989).

It is worth emphasizing that this result holds regardless of the choice of the target $\boldsymbol{\mu}^{\text{tgt}}$. In other words, one can choose the target as bad as desired, but still the MSE can be reduced (a phenomenon often referred to as the Stein paradox). This happens because the value of ρ automatically adapts in the following ways:

- $\rho \to 0$ as T increases, that is, the more observations, the stronger the belief in the original sample mean estimator;
- $\rho \to 0$ as the target disagrees with the sample mean (built-in safety mechanism under wrongly chosen targets).

While it is true that the MSE can be reduced no matter the choice of the target, the size of the improvement will obviously depend on how good and informative the target is. Any available prior information can be incorporated in the target. Some reasonable and common choices are (Jorion, 1986):

- *zero*: $\boldsymbol{\mu}^{\text{tgt}} = \mathbf{0}$;
- *grand mean*: $\boldsymbol{\mu}^{\text{tgt}} = \frac{\mathbf{1}^{\top}\hat{\boldsymbol{\mu}}}{N} \times \mathbf{1}$; and
- *volatility-weighted grand mean*: $\boldsymbol{\mu}^{\text{tgt}} = \frac{\mathbf{1}^{\top}\hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{\mu}}}{\mathbf{1}^{\top}\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{1}} \times \mathbf{1}$.

Shrinking the Covariance Matrix

Consider the covariance matrix $\boldsymbol{\Sigma}$ and the sample covariance matrix estimator in (3.3):

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{T-1} \sum_{t=1}^{T} (\boldsymbol{x}_t - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_t - \hat{\boldsymbol{\mu}})^{\top}.$$

As we know from Section 3.2, the sample covariance matrix $\hat{\boldsymbol{\Sigma}}$ is an unbiased estimator, $\mathbb{E}\left[\hat{\boldsymbol{\Sigma}}\right] = \boldsymbol{\Sigma}$. We will now introduce some bias in the form of shrinkage.

The shrinkage estimator for the covariance matrix has the form

$$\hat{\boldsymbol{\Sigma}}^{\text{sh}} = (1-\rho)\,\hat{\boldsymbol{\Sigma}} + \rho\,\boldsymbol{\Sigma}^{\text{tgt}},$$

where $\boldsymbol{\Sigma}^{\text{tgt}}$ is the target and ρ the shrinkage factor.

The idea of shrinkage of the covariance matrix had been already used in the 1980s, for example in wireless communications systems under the term "diagonal loading" (Abramovich, 1981).

In finance, it was popularized in the early 2000s (Ledoit & Wolf, 2003, 2004) and more mature tools developed in recent decades (Bun et al., 2017).

Common choices for the covariance matrix target are:

- *scaled identity*: $\boldsymbol{\Sigma}^{\text{tgt}} = \frac{1}{N}\text{Tr}(\hat{\boldsymbol{\Sigma}}) \times \boldsymbol{I}$;
- *diagonal matrix*: $\boldsymbol{\Sigma}^{\text{tgt}} = \text{Diag}(\hat{\boldsymbol{\Sigma}})$, which is equivalent to using the identity matrix as the correlation matrix target, that is, $\boldsymbol{C}^{\text{tgt}} = \boldsymbol{I}$; and
- *equal-correlation matrix*: $\boldsymbol{C}^{\text{tgt}}$ with off-diagonal elements all equal to the average cross-correlation of assets.

The shrinkage factor ρ can be determined empirically via cross-validation or analytically via a mathematically sophisticated approach such as random matrix theory (RMT) (Bun et al., 2006, 2017). This was made popular in the financial community by Ledoit and Wolf in the early 2000s (Ledoit & Wolf, 2003, 2004). The idea is simple: choose ρ to form the shrinkage estimator $\hat{\boldsymbol{\Sigma}}^{\text{sh}}$ in order to minimize the error measure $\mathbb{E}\left[\left\|\hat{\boldsymbol{\Sigma}}^{\text{sh}} - \boldsymbol{\Sigma}\right\|_{\text{F}}^2\right]$, where $\|\cdot\|_{\text{F}}$ denotes the Frobenius norm. Of course, this problem is ill-posed since precisely the true covariance matrix $\boldsymbol{\Sigma}$ is unknown; otherwise, the so-called "oracle" solution is obtained as

$$\rho = \frac{\mathbb{E}\left[\left\|\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}\right\|_{\text{F}}^2\right]}{\mathbb{E}\left[\left\|\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}^{\text{tgt}}\right\|_{\text{F}}^2\right]}.$$

This is where the magic of RMT truly shines: asymptotically for large T and N, one can derive a consistent estimator that does not require knowledge of $\boldsymbol{\Sigma}$ as

$$\rho = \min\left(1, \frac{\frac{1}{T}\sum_{t=1}^{T}\left\|\hat{\boldsymbol{\Sigma}} - \boldsymbol{x}_t\boldsymbol{x}_t^{\mathsf{T}}\right\|_{\text{F}}^2}{\left\|\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}^{\text{tgt}}\right\|_{\text{F}}^2}\right). \tag{3.11}$$

The usefulness of RMT is that the results are extremely good even when T and N are not very large, that is, the asymptotics kick in very fast in practice.

One important way to extend shrinkage is to consider heavy-tailed distributions (introduced in Section 3.5) and derive an appropriate choice for ρ under heavy tails (Chen et al., 2011; Ollila & Raninen, 2019; Ollila et al., 2021, 2023).

It is worth pointing out that the shrinkage factor in (3.11) is derived to minimize the error measure $\mathbb{E}\left[\left\|\hat{\boldsymbol{\Sigma}}^{\text{sh}} - \boldsymbol{\Sigma}\right\|_{\text{F}}^2\right]$. Alternatively, one can consider more meaningful measures of error better suited to the purpose of portfolio design, with the drawback that the mathematical derivation of ρ becomes more involved and cannot be obtained in a nice closed-form expression such as (3.11). Some examples include the following (cf. Feng and Palomar (2016)):

- Portfolios designed based on $\boldsymbol{\Sigma}$ are more directly related to the inverse covariance matrix $\boldsymbol{\Sigma}^{-1}$ (see Chapter 7). Thus, it makes sense to measure the error in terms of the inverse covariance matrix instead, $\mathbb{E}\left[\left\|\left(\hat{\boldsymbol{\Sigma}}^{\text{sh}}\right)^{-1} - \boldsymbol{\Sigma}^{-1}\right\|_{\text{F}}^2\right]$ (Zhang, Rubio, & Palomar, 2013).
- Portfolios are typically evaluated in terms of the Sharpe ratio, which is related to the term $\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ (see Chapter 7). Thus, it has more practical meaning to choose ρ to maximize the achieved Sharpe ratio (Ledoit & Wolf, 2017; Zhang, Rubio, Palomar, & Mestre, 2013).

The covariance shrinkage estimator is a linear combination of the estimate $\hat{\boldsymbol{\Sigma}}$ and the target $\boldsymbol{\Sigma}^{\text{tgt}}$. Another way to extend this idea is to consider a nonlinear shrinkage in terms of eigenvalues of the covariance matrix, which again leads to an increased sophistication in the required mathematics, for example, Ledoit and Wolf (2017), Bun et al. (2006), Bun et al. (2017), Bartz (2016), and Tyler and Yi (2020).

Numerical Experiments

Figure 3.13 shows the estimation error of shrinkage estimators as a function of the number of observations T for synthetic Gaussian data. We can observe that a clear improvement is achieved in the estimation of the mean vector, whereas only a modest improvement is achieved in the estimation of the covariance matrix. As expected, the benefit of shrinkage diminishes as the number of observations grows large.

Interestingly, shrinkage to zero seems to produce the best results. This is not unexpected since, according to the efficient-market hypothesis (Fama, 1970), the prices are expected to contain all the current information of the assets (including future prospects). Thus, a reasonable forecast for the prices are just the current prices or, equivalently in terms of returns, the zero return vector.

It is important to emphasize that these numerical results are obtained in terms of the mean squared error of the estimators. However, in the context of portfolio optimization, the mean squared error may not be the best measure of errors. Thus, these results should be taken with a grain of salt and more appropriate measures of error should probably be considered (Ledoit & Wolf, 2017).

3.6.2 Factor Models

Factor modeling is standard material in finance and can be found in many textbooks, such as Campbell et al. (1997), Fabozzi et al. (2010), Tsay (2010), Ruppert and Matteson (2015), Lütkepohl (2007), and Tsay (2013).

The idea is to introduce prior information into the basic i.i.d. model of the returns in (3.1), $\boldsymbol{x}_t = \boldsymbol{\mu} + \boldsymbol{\epsilon}_t$, in the form of a more sophisticated asset structure. For example, the simplest case is the single-factor model

$$\boldsymbol{x}_t = \boldsymbol{\alpha} + \boldsymbol{\beta} f_t^{\text{mkt}} + \boldsymbol{\epsilon}_t, \tag{3.12}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^N$ and $\boldsymbol{\beta} \in \mathbb{R}^N$ are the so-called "alpha" and "beta" of the N assets, respectively, the scalar f_t^{mkt} is the market factor (or market index), and $\boldsymbol{\epsilon}_t$ is the zero-mean residual component. The "beta" refers to how sensitive the asset is to the overall market, whereas the "alpha" indicates the excess return of the asset. As explored in Chapter 6, one common constraint in portfolio design is to be market neutral, which refers precisely to the portfolio being orthogonal to $\boldsymbol{\beta}$.

The single-factor model in (3.12) has some connection with the capital asset pricing model (CAPM) introduced by Sharpe (1964). In particular, the CAPM is concerned with the expected excess returns and assumes a zero "alpha",

$$\mathbb{E}[x_i] - r_{\text{f}} = \beta_i \left(\mathbb{E}\left[f^{\text{mkt}} \right] - r_{\text{f}} \right), \tag{3.13}$$

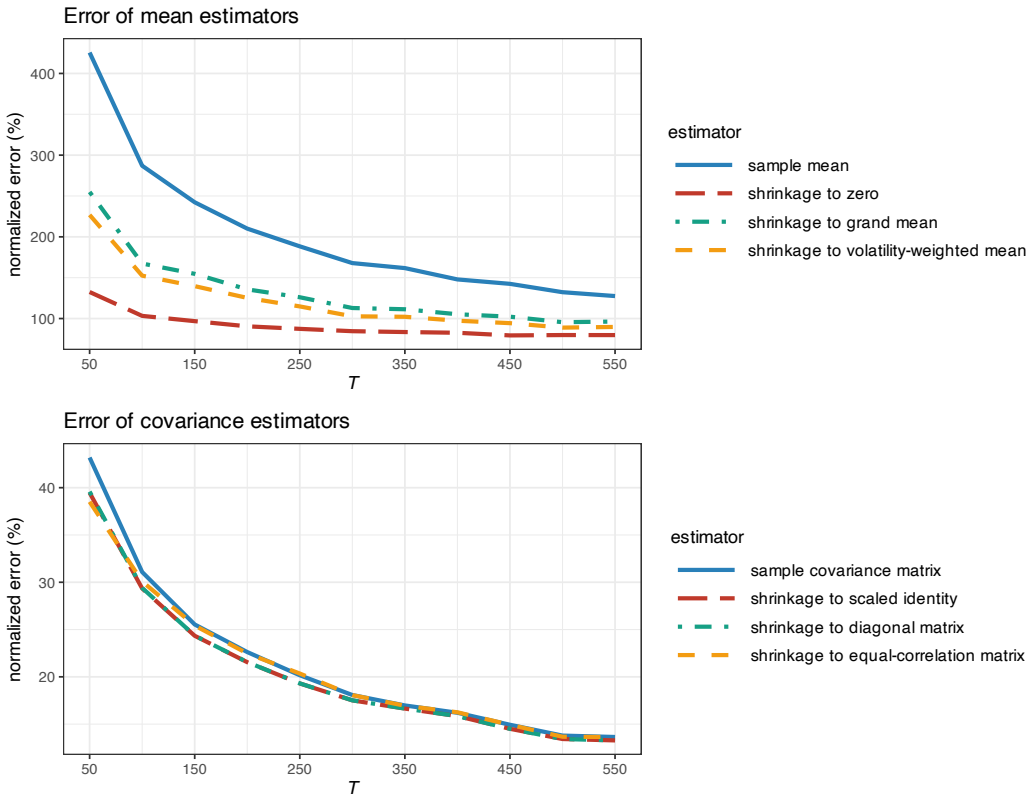


Figure 3.13 Estimation error of different shrinkage estimators vs. number of observations (for Gaussian data with $N = 100$).

where r_{f} denotes the risk-free asset return.

More generally, the multi-factor model is

$$\boldsymbol{x}_t = \boldsymbol{\alpha} + \boldsymbol{B}\boldsymbol{f}_t + \boldsymbol{\epsilon}_t, \tag{3.14}$$

where now $\boldsymbol{f}_t \in \mathbb{R}^K$ contains K factors (also termed *risk factors*)—typically with $K \ll N$—and the matrix $\boldsymbol{B} \in \mathbb{R}^{N\times K}$ contains along its columns the betas (also called *factor loadings*) for each of the factors. It is also possible to include time dependency in the factors, leading to the so-called *dynamic factor models* (Fabozzi et al., 2010).

The residual term $\boldsymbol{\epsilon}_t$ in (3.12) and (3.14) – typically referred to as the *idiosyncratic component* – is assumed to have independent elements or, in other words, a diagonal covariance matrix $\boldsymbol{\Psi} = \mathrm{Diag}(\psi_1, \ldots, \psi_N) \in \mathbb{R}^{N\times N}$. The rationale is that the correlation among the assets is already modeled by the other term via the factors.

One key realization in factor modeling is that the number of parameters of the model to be estimated is significantly reduced. For example, in the case of $N = 500$ assets and $K = 3$ factors, the number of parameters in the plain i.i.d. model in (3.1) is 125750 (N for $\boldsymbol{\mu}$ and $N(N+1)/2$ for the symmetric covariance matrix $\boldsymbol{\Sigma}$), whereas for the factor model in (3.14)

the number of parameters is 2500 (N for $\boldsymbol{\alpha}$, NK for $\boldsymbol{B}$, and N for the diagonal covariance matrix $\boldsymbol{\Psi}$).

The mean and covariance matrix according to the model (3.14) are given by

$$\begin{aligned} \boldsymbol{\mu} &= \boldsymbol{\alpha} + \boldsymbol{B}\boldsymbol{\mu}_f, \\ \boldsymbol{\Sigma} &= \boldsymbol{B}\boldsymbol{\Sigma}_f\boldsymbol{B}^\mathsf{T} + \boldsymbol{\Psi}, \end{aligned} \tag{3.15}$$

where $\boldsymbol{\mu}_f$ and $\boldsymbol{\Sigma}_f$ are the mean vector and covariance matrix, respectively, of the factors $\boldsymbol{f}_t$. Observe that the covariance matrix has effectively been decomposed into a low-rank component $\boldsymbol{B}\boldsymbol{\Sigma}_f\boldsymbol{B}^\mathsf{T}$ (with rank K) and a full-rank diagonal component $\boldsymbol{\Psi}$.

Essentially, factor models decompose the asset returns into two parts: the low-dimensional factor component, $\boldsymbol{B}\boldsymbol{f}_t$, and the idiosyncratic residual noise $\boldsymbol{\epsilon}_t$. Depending on the assumptions made on the factors $\boldsymbol{f}_t$ and "betas" in $\boldsymbol{B}$, factor models can be classified into three types:

- *Macroeconomic factor models*: Factors are observable economic and financial time series but the loading matrix $\boldsymbol{B}$ is unknown.
- *Fundamental factor models*: Some models construct the loading matrix $\boldsymbol{B}$ from asset characteristics with unknown factors, whereas others construct the factors from asset characteristics first.
- *Statistical factor models*: Both the factors and the loading matrix $\boldsymbol{B}$ are unknown.

Macroeconomic Factor Models

In macroeconomic factor models, the factors are observable time series such as the market index, the growth rate of the GDP,[5] interest rate, inflation rate, and so on. In the investment world, factors are computed in complicated proprietary ways from a variety of nonaccessible sources of data and, typically, investment funds pay a substantial premium to have access to them. Such expensive factors are not available to small investors, which have to rely on readily available sources of data.

Given the factors, the estimation of the model parameters can be easily formulated as a simple least squares regression problem,

$$\underset{\boldsymbol{\alpha},\boldsymbol{B}}{\text{minimize}} \quad \sum_{t=1}^{T} \|\boldsymbol{x}_t - (\boldsymbol{\alpha} + \boldsymbol{B}\boldsymbol{f}_t)\|_2^2,$$

from which $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ can then be obtained from (3.15).

Fundamental Factor Models

Fundamental factor models use observable asset-specific characteristics (termed fundamentals), such as industry classification, market capitalization, style classification (e.g., value, growth), to determine the factors. Typically, two approaches are followed in the industry:

[5] The gross domestic product (GDP) is a monetary measure of the market value of all the final goods and services produced and sold in a specific time period by a country.

- *Fama–French approach*: First, a portfolio is formed based on the chosen asset-specific characteristics to obtain the factors $\boldsymbol{f}_t$. Then, the loading factors $\boldsymbol{B}$ are obtained as in the macroeconomic factor models. The original model was composed of $K = 3$ factors, namely, the size of firms, book-to-market values, and excess return on the market (Fama & French, 1992), and was later extended to $K = 5$ factors (Fama & French, 2015).

- *Barra risk factor analysis approach* (developed by Barra Inc. in 1975): First, the loading factors in $\boldsymbol{B}$ are constructed from observable asset characteristics (e.g., based on industry classification). Then, the factors $\boldsymbol{f}_t$ can be estimated via regression (note that this is the opposite of macroeconomic factor models).

Statistical Factor Models

Statistical factor models work under the premise that both the factors $\boldsymbol{f}_t$ and the loading factor matrix $\boldsymbol{B}$ are unknown. At first, it may seem impossible to be able to fit such a model due to so many unknowns. After careful inspection, one realizes that the effect of the factor model structure in (3.14) is to introduce some structure in the parameters as in (3.15). Basically, now the covariance matrix $\boldsymbol{\Sigma}$ has a very specific structure in the form of a low-rank component $\boldsymbol{B}\boldsymbol{\Sigma}_f\boldsymbol{B}^\mathsf{T}$ and a diagonal matrix $\boldsymbol{\Psi}$. Since $\boldsymbol{\Sigma}_f$ is unknown, this decomposition has an infinite number of solutions because we can always right-multiply $\boldsymbol{B}$ and multiply $\boldsymbol{\Sigma}_f$ on both sides by an arbitrary invertible matrix. Thus, without loss of generality, we will assume that the factors are zero-mean and normalized (i.e., $\boldsymbol{\mu}_f = \boldsymbol{0}$ and $\boldsymbol{\Sigma}_f = \boldsymbol{I}$).

A heuristic formulation follows from first computing the sample covariance matrix $\hat{\boldsymbol{\Sigma}}$ and then approximating it with the desired structure (Sardarabadi & van der Veen, 2018):

$$\underset{\boldsymbol{B},\boldsymbol{\psi}}{\text{minimize}} \quad \|\hat{\boldsymbol{\Sigma}} - \left(\boldsymbol{B}\boldsymbol{B}^\mathsf{T} + \text{Diag}(\boldsymbol{\psi})\right)\|_\text{F}^2.$$

Alternatively, we can directly formulate the ML estimation of the parameters (similarly to Section 3.4) but imposing such a structure. Suppose that the returns $\boldsymbol{x}_t$, factors $\boldsymbol{f}_t$, and residuals $\boldsymbol{\epsilon}_t$ follow a Gaussian distribution, then the ML estimation can be formulated as

$$\begin{array}{ll} \underset{\boldsymbol{\alpha},\boldsymbol{\Sigma},\boldsymbol{B},\boldsymbol{\psi}}{\text{minimize}} & \log\det(\boldsymbol{\Sigma}) + \dfrac{1}{T}\displaystyle\sum_{t=1}^{T}(\boldsymbol{x}_t - \boldsymbol{\alpha})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_t - \boldsymbol{\alpha}) \\ \text{subject to} & \boldsymbol{\Sigma} = \boldsymbol{B}\boldsymbol{B}^\mathsf{T} + \text{Diag}(\boldsymbol{\psi}). \end{array} \tag{3.16}$$

Unfortunately, due to the nonconvex nature of the structural constraint, this problem is difficult to solve. Iterative algorithms were developed in Santamaria et al. (2017) and Khamaru and Mazumder (2019).

Even better, we can depart from the unrealistic Gaussian assumption and formulate the ML estimation under a heavy-tailed distribution, making the problem even more complicated. An iterative MM-based algorithm was proposed for this robust formulation in Zhou et al. (2020).

Another extension to this problem formulation comes from introducing additional structure commonly observed in financial data such as nonnegative asset correlation (see Section 2.5 in Chapter 2) as considered in Zhou et al. (2022).

Principal Component Analysis (PCA)

High-dimensional data can be challenging to analyze and model; as a consequence it has been widely studied by researchers in both statistics and signal processing. In most practical applications, high-dimensional data have most of their variation in a lower-dimensional subspace that can be found using dimension reduction techniques. The most popular one is *principal component analysis* (PCA), which can be used as an approximated way to solve the statistical factor model fitting in (3.16). We will now introduce the basics of PCA; for more information, the reader is referred to standard textbooks such as Jolliffe (2002), Hastie et al. (2009), and G. James et al. (2013).

PCA tries to capture the direction $\boldsymbol{u}$ of maximum variance of the vector-valued random variable $\boldsymbol{x}$ by maximizing the variance, $\text{Var}(\boldsymbol{u}^\top \boldsymbol{x}) = \boldsymbol{u}^\top \boldsymbol{\Sigma} \boldsymbol{u}$, whose solution is given by the eigenvector of $\boldsymbol{\Sigma}$ corresponding to the maximum eigenvalue. This procedure can be repeated by adding more directions of maximum variance provided they are orthogonal to the previously found ones, which reduces to an eigenvalue decomposition problem. Let $\boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^\top$ be the eigendecomposition of matrix $\boldsymbol{\Sigma}$, where $\boldsymbol{U}$ contains the (orthogonal) eigenvectors along the columns and $\boldsymbol{D}$ is a diagonal matrix containing the eigenvalues in decreasing order, $\lambda_1 \geq \cdots \geq \lambda_N$. Then, the best low-rank approximation of matrix $\boldsymbol{\Sigma}$ can be easily obtained using the strongest eigenvector-eigenvalues. That is, the best approximation with rank K is $\boldsymbol{\Sigma} \approx \boldsymbol{U}^{(K)} \boldsymbol{D}^{(K)} \boldsymbol{U}^{(K)\top}$, where matrix $\boldsymbol{U}^{(K)}$ contains the first K columns of $\boldsymbol{U}$ and $\boldsymbol{D}^{(K)}$ is a diagonal matrix containing the largest K diagonal elements. The larger the number of components K, the better the approximation will be but with the risk of overfitting. In practice, choosing K is critical, as a relatively small value of K may already capture a large percentage of the variance.

Using PCA to approximate the solution to (3.16) is simple. First, start by computing the sample covariance matrix (i.e., ignoring the structure). At this point, if we were to approximate this matrix with its K principal components $\boldsymbol{U}^{(K)} \boldsymbol{D}^{(K)} \boldsymbol{U}^{(K)\top}$, we would be missing the diagonal matrix component $\boldsymbol{\Psi}$. A simple heuristic approximates this diagonal matrix by a scaled identity matrix $\kappa \boldsymbol{I}$, where κ is the average of the $N - K$ smallest eigenvalues. Summarizing, the approximate solution is

$$\begin{aligned} \boldsymbol{B} &= \boldsymbol{U}^{(K)} \text{Diag}\left(\sqrt{\lambda_1 - \kappa}, \ldots, \sqrt{\lambda_K - \kappa}\right), \\ \boldsymbol{\Psi} &= \kappa \boldsymbol{I}, \end{aligned}$$

where $\kappa = \frac{1}{N-K} \sum_{i=K+1}^{N} D_{ii}$. This finally leads to the estimator

$$\hat{\boldsymbol{\Sigma}} = \boldsymbol{U} \text{Diag}\left(\lambda_1, \ldots, \lambda_K, \kappa, \ldots, \kappa\right) \boldsymbol{U}^\top.$$

Interestingly, what PCA has achieved in this case is some kind of noise averaging of the smallest eigenvalues. This idea of noise cleaning is reminiscent of the concept of shrinkage from Section 3.6.1.

Numerical Experiments

For illustration purposes, we now evaluate the estimation of the covariance matrix under the factor model assumption following the formulation in (3.16). It is important to emphasize that if the actual data do not comply with the assumed factor model structure, then the estimation

under the formulation in (3.16) may produce worse results than the plain sample covariance matrix. Therefore, the choice of employing the factor model structure in the estimation process has to be carefully made at the risk of the user. Trading strategies based on factor modeling are discussed in detail in Fabozzi et al. (2010).

Figure 3.14 shows the estimation error of the covariance matrix estimated under the factor model structure in (3.16) for different numbers of principal components, compared to the benchmark sample covariance matrix, as a function of the number of observations T for synthetic Gaussian data with a covariance matrix that complies with the factor model structure with $K = 3$. We can observe that when the estimation method uses the true value of K, then the estimation becomes slightly better than the sample covariance matrix (as expected). However, using the wrong choice of K can have drastic consequences (such as $K = 1$).

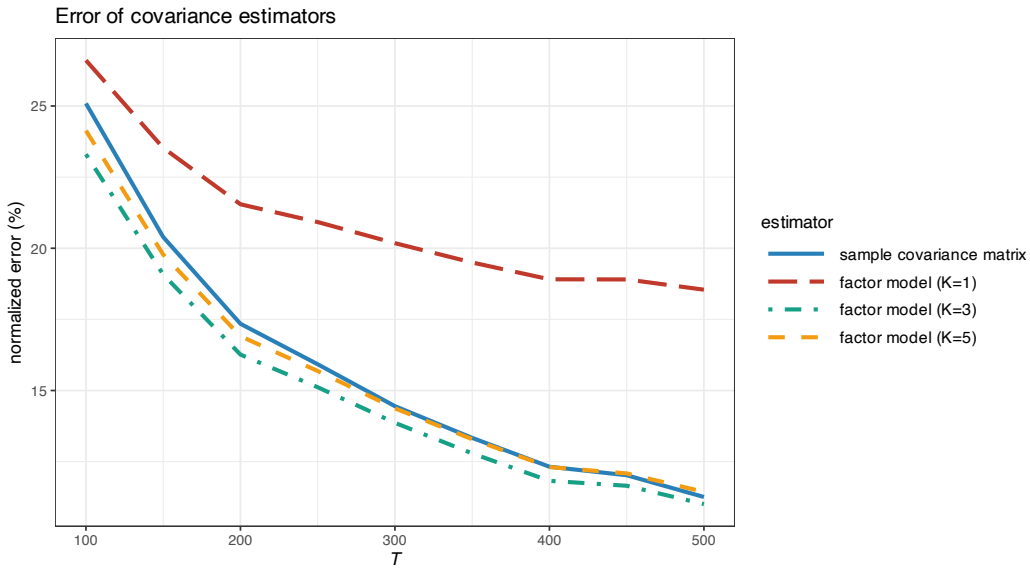


Figure 3.14 Estimation error of covariance matrix under factor modeling vs. number of observations (with $N = 100$).

3.6.3 Black–Litterman Model

The Black–Litterman model was proposed in 1991 (Black & Litterman, 1991) and has become standard material in finance as described in many textbooks, such as Fabozzi et al. (2010) and Meucci (2005). It is a mathematical technique that combines parameter estimation based on historical observation of the past T samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ with some prior information on these parameters.

The Black–Litterman model considers the following two components:

- *Market equilibrium*: One source of information for $\boldsymbol{\mu}$ is the market, for example, the sample mean $\hat{\boldsymbol{\mu}} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t$. We can explicitly write this estimate $\boldsymbol{\pi} = \hat{\boldsymbol{\mu}}$ in terms of the actual $\boldsymbol{\mu}$

and the estimation error:

$$\boldsymbol{\pi} = \boldsymbol{\mu} + \boldsymbol{\epsilon}, \tag{3.17}$$

where the error component $\boldsymbol{\epsilon}$ is assumed zero-mean and with covariance matrix $\tau\boldsymbol{\Sigma}$. The parameter τ can be selected via cross-validation or simply chosen as $\tau = 1/T$ (i.e., the more observations the less uncertainty on the market equilibrium).

- *Investor's views*: Suppose we have K views summarized from investors written in the form

$$\boldsymbol{v} = \boldsymbol{P}\boldsymbol{\mu} + \boldsymbol{e}, \tag{3.18}$$

where $\boldsymbol{v} \in \mathbb{R}^K$ and $\boldsymbol{P} \in \mathbb{R}^{K\times N}$ characterize the absolute or relative views, and the error term $\boldsymbol{e}$, which is assumed zero-mean with covariance $\boldsymbol{\Omega}$, measures the uncertainty of the views. Exactly how to obtain such views is the secret of each investor.

Example 3.1 (Quantitative investor's views) Suppose there are $N = 5$ stocks and two independent views on them (Fabozzi et al., 2010):

- Stock 1 will have a return of 1.5% with standard deviation of 1%.
- Stock 3 will outperform Stock 2 by 4% with a standard deviation of 1%.

Mathematically, we can express these two views as

$$\begin{bmatrix} 1.5\% \\ 4\% \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \end{bmatrix} \boldsymbol{\mu} + \boldsymbol{e},$$

with the covariance of $\boldsymbol{e}$ given by $\boldsymbol{\Omega} = \begin{bmatrix} 1\%^2 & 0 \\ 0 & 1\%^2 \end{bmatrix}$.

On some occasions, the investor may only have qualitative views (as opposed to quantitative ones), that is, only matrix $\boldsymbol{P}$ is specified while the views $\boldsymbol{v}$ and the uncertainty $\boldsymbol{\Omega}$ are undefined. Then, one can choose them as follows (Meucci, 2005):

$$v_i = (\boldsymbol{P}\boldsymbol{\pi})_i + \eta_i \sqrt{(\boldsymbol{P}\boldsymbol{\Sigma}\boldsymbol{P}^\mathsf{T})_{ii}} \qquad i = 1, \ldots, K,$$

where $\eta_i \in \{-\beta, -\alpha, +\alpha, +\beta\}$ defines "very bearish," "bearish," "bullish," and "very bullish" views, respectively (typical choices are $\alpha = 1$ and $\beta = 2$); as for the uncertainty,

$$\boldsymbol{\Omega} = \frac{1}{c}\boldsymbol{P}\boldsymbol{\Sigma}\boldsymbol{P}^\mathsf{T},$$

where the scatter structure of the uncertainty is inherited from the market volatilities and correlations, and $c \in (0, \infty)$ represents the overall level of confidence in the views.

An alternative to the previous market equilibrium $\boldsymbol{\pi} = \hat{\boldsymbol{\mu}}$ can be derived from the CAPM model in (3.13) as

$$\boldsymbol{\pi} = \boldsymbol{\beta}\left(\mathbb{E}\left[f^{\text{mkt}}\right] - r_\text{f}\right).$$

Merging the Market Equilibrium with the Views

The combination of the market equilibrium and the investor's views can be mathematically formulated in a variety of ways, ranging from least squares formulations, through maximum likelihood, and even different Bayesian formulations. Interestingly, the solution is the same (or very similar) as we briefly discuss next in three particular formulations.

- *Weighted least squares formulation*: First, we rewrite the equations (3.17) and (3.18) in a more compact way as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\mu} + \boldsymbol{n},$$

where $\boldsymbol{y} = \begin{bmatrix}\boldsymbol{\pi}\\ \boldsymbol{v}\end{bmatrix}$, $\boldsymbol{X} = \begin{bmatrix}\boldsymbol{I}\\ \boldsymbol{P}\end{bmatrix}$, and the covariance matrix of the noise term $\boldsymbol{n}$ is given by $\boldsymbol{V} = \begin{bmatrix}\tau\boldsymbol{\Sigma} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{\Omega}\end{bmatrix}$. Then we can formulate the problem as a weighted least squares problem (Feng & Palomar, 2016):

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu})^{\mathsf{T}} \boldsymbol{V}^{-1} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}),$$

with solution

$$\begin{aligned}\boldsymbol{\mu}_{\mathrm{BL}} &= \left(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{V}^{-1}\boldsymbol{X}\right)^{-1}\boldsymbol{V}^{-1}\boldsymbol{y}\\ &= \left((\tau\boldsymbol{\Sigma})^{-1} + \boldsymbol{P}^{\mathsf{T}}\boldsymbol{\Omega}^{-1}\boldsymbol{P}\right)^{-1}\left((\tau\boldsymbol{\Sigma})^{-1}\boldsymbol{\pi} + \boldsymbol{P}^{\mathsf{T}}\boldsymbol{\Omega}^{-1}\boldsymbol{v}\right),\end{aligned} \tag{3.19}$$

which has covariance matrix

$$\mathrm{Cov}(\boldsymbol{\mu}_{\mathrm{BL}}) = \left((\tau\boldsymbol{\Sigma})^{-1} + \boldsymbol{P}^{\mathsf{T}}\boldsymbol{\Omega}^{-1}\boldsymbol{P}\right)^{-1}. \tag{3.20}$$

- *Original Bayesian formulation*: The original formulation (Black & Litterman, 1991) assumes prior Gaussian distributions. In particular, the returns are modeled as $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is assumed known but now $\boldsymbol{\mu}$ is modeled as random with a Gaussian distribution:

$$\boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\pi}, \tau\boldsymbol{\Sigma}),$$

where $\boldsymbol{\pi}$ represents the best guess for $\boldsymbol{\mu}$ and $\tau\boldsymbol{\Sigma}$ is the uncertainty of this guess. Note that this implies that $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\pi}, (1+\tau)\boldsymbol{\Sigma})$. The views are also modeled as following a Gaussian distribution:

$$\boldsymbol{P}\boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{v}, \boldsymbol{\Omega}).$$

Then, the posterior distribution of $\boldsymbol{\mu}$ given $\boldsymbol{v}$ and $\boldsymbol{\Omega}$ follows from Bayes' formula as

$$\boldsymbol{\mu} \mid \boldsymbol{v}, \boldsymbol{\Omega} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\mathrm{BL}}, \boldsymbol{\Sigma}_{\mathrm{BL}}\right),$$

where the posterior mean is exactly like the previous LS solution in (3.19) and $\boldsymbol{\Sigma}_{\mathrm{BL}} = \mathrm{Cov}(\boldsymbol{\mu}_{\mathrm{BL}}) + \boldsymbol{\Sigma}$ with $\mathrm{Cov}(\boldsymbol{\mu}_{\mathrm{BL}})$ in (3.20). By using the matrix inversion lemma, we can further rewrite the Black–Litterman estimators for the mean and covariance matrix as

$$\begin{aligned}\boldsymbol{\mu}_{\mathrm{BL}} &= \boldsymbol{\pi} + \tau\boldsymbol{\Sigma}\boldsymbol{P}^{\mathsf{T}}\left(\tau\boldsymbol{P}\boldsymbol{\Sigma}\boldsymbol{P}^{\mathsf{T}} + \boldsymbol{\Omega}\right)^{-1}(\boldsymbol{v} - \boldsymbol{P}\boldsymbol{\pi}),\\ \boldsymbol{\Sigma}_{\mathrm{BL}} &= (1+\tau)\boldsymbol{\Sigma} - \tau^2\boldsymbol{\Sigma}\boldsymbol{P}^{\mathsf{T}}\left(\tau\boldsymbol{P}\boldsymbol{\Sigma}\boldsymbol{P}^{\mathsf{T}} + \boldsymbol{\Omega}\right)^{-1}\boldsymbol{P}\boldsymbol{\Sigma}.\end{aligned} \tag{3.21}$$

- *Alternative Bayesian formulation*: It is worth mentioning a variation of the original Bayesian formulation in Meucci (2005), where the views are modeled on the random returns, $\boldsymbol{v} = \boldsymbol{P}\boldsymbol{x} + \boldsymbol{e}$, unlike (3.18) where the views are on $\boldsymbol{\mu}$. In this case, one can similarly derive the posterior distribution of the returns with a result similar to (3.21).

As a final observation, it is insightful to consider the two extremes:

- $\tau = 0$: We give total accuracy to the market equilibrium and, as expected, we get

$$\boldsymbol{\mu}_{\mathrm{BL}} = \boldsymbol{\pi}.$$

- $\tau \to \infty$: We give no value at all to the market equilibrium and, therefore, the investor's views dominate:

$$\boldsymbol{\mu}_{\mathrm{BL}} = \left(\boldsymbol{P}^{\mathsf{T}}\boldsymbol{\Omega}^{-1}\boldsymbol{P}\right)^{-1}\left(\boldsymbol{P}^{\mathsf{T}}\boldsymbol{\Omega}^{-1}\boldsymbol{v}\right).$$

  In the general case with $0 < \tau < \infty$, $\boldsymbol{\mu}_{\mathrm{BL}}$ is a weighted combination of these two extreme cases, which is actually related to the concept of shrinkage explored in Section 3.6.1.

3.7 Summary

Countless models have been put forth in the literature for financial data. The i.i.d. model may be a rough approximation of reality, but it is functional and widely used by academics and practitioners. Some key points of the i.i.d. model for financial data include:

- *Sample estimators perform poorly*: This is not unexpected since the sample mean and sample covariance matrix are optimal estimators under the assumption of Gaussian-distributed data, which does not hold in practice.
- *Robust estimators are necessary*: The spatial median and Tyler estimator are examples of robust estimators against outliers for the mean vector and covariance matrix, respectively.
- *Heavy-tailed estimators are well suited to financial data*: Estimators derived under the assumption of heavy-tailed distributed data are naturally robust and fit financial data well. In addition, simple iterative algorithms can be used to compute them in practice.
- *Estimating the mean vector from historical data is extremely noisy*: Practitioners typically obtain factors from data providers (at a high premium) and then use them for regression; using just historical data is the "poor man's" substitute and it is not without its risks.
- *Prior information should be used when available*: This can be, among others, in the form of a shrinkage target, factor modeling, or information fusion via the Black–Litterman model (or similar).

Exercises

3.1 (Unbiasedness and consistency of sample mean estimator) Consider a univariate Gaussian-distributed i.i.d. time series with mean 0.01 and variance 1, $x_t \sim \mathcal{N}(0.01, 1)$, $t = 1, \ldots, T$.

a. Generate data for $T = 10$ and compute the sample mean. Repeat the experiment multiple times and plot the histogram of the estimated mean value. Confirm that the expected value of the histogram coincides with the true mean value.
b. Now repeat the experiment with $T = 20$ observations and compare the histograms (also compute the standard deviation of each histogram).
c. Finally, repeat the experiment multiple times, for different numbers of observations $T = 10, 20, \ldots, 100$, and plot the mean squared error of the estimation as a function of T.

3.2 (Bias of sample covariance matrix) Suppose we have T i.i.d. N-dimensional observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ distributed as $\boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

a. Derive the following expected value based on the true $\boldsymbol{\mu}$:

$$\mathbb{E}\left[\sum_{t=1}^{T}(\boldsymbol{x}_t - \boldsymbol{\mu})(\boldsymbol{x}_t - \boldsymbol{\mu})^\top\right].$$

b. Derive the following expected value based now on the sample mean $\hat{\boldsymbol{\mu}} = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t$:

$$\mathbb{E}\left[\sum_{t=1}^{T}(\boldsymbol{x}_t - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_t - \hat{\boldsymbol{\mu}})^\top\right].$$

c. Discuss the appropriate normalization factor, $1/(T-1)$ or $1/T$, to be used in the expression of the sample covariance matrix.

3.3 (Location estimators) Consider a two-dimensional ($N = 2$) Gaussian-distributed i.i.d. time series with zero mean and identity covariance matrix, $\boldsymbol{x}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}),\ t = 1, \ldots, T$.

a. Generate data for $T = 20$ and estimate the mean vector $\boldsymbol{\mu}$ via the sample mean, the median, and the spatial median. Visualize the results in a scatter plot.
b. Repeat the experiment multiple times, for different numbers of observations $T = 10, 20, \ldots, 100$, and plot the mean squared error as a function of T.

3.4 (Location estimators with outliers) Consider a two-dimensional ($N = 2$) Gaussian-distributed i.i.d. time series with zero mean and identity covariance matrix, $\boldsymbol{x}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}),\ t = 1, \ldots, T$.

a. Generate data for $T = 20$ and estimate the mean vector $\boldsymbol{\mu}$ via the sample mean, the median, and the spatial median. Visualize the results in a scatter plot. Repeat the experiment multiple times and compute the mean squared error of the estimators.
b. Then, add some small percentage of outliers in the observations, for example, distributed as $\boldsymbol{x}_t \sim \mathcal{N}(0.1 \times \mathbf{1}, \boldsymbol{I})$, and compute again the mean squared error of the estimators.
c. Finally, repeat the experiment multiple times and plot the estimation error as a function of the percentage of outliers. Observe the robustness of the three estimators against outliers and discuss.

3.5 (Derivation of sample mean as location estimator) Given the observations $\boldsymbol{x}_t,\ t = 1, \ldots, T$, the sample mean can be derived as the solution to the following optimization problem:

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad \sum_{t=1}^{T}\|\boldsymbol{x}_t - \boldsymbol{\mu}\|_2^2.$$

a. Is this problem convex? What class of optimization problem is it?
b. Derive the solution in closed form by setting the gradient with respect to $\boldsymbol{\mu}$ to zero.

3.6 (Computation of spatial median as location estimator) Given the observations $\boldsymbol{x}_t,\ t = 1, \ldots, T$, the spatial median can be derived as the solution to the following optimization

problem:

$$\underset{\boldsymbol{\mu}}{\text{minimize}} \quad \sum_{t=1}^{T} \|\boldsymbol{x}_t - \boldsymbol{\mu}\|_2.$$

a. Is this problem convex? What class of optimization problem is it?
b. Can a closed-form solution be obtained as in the case of the sample mean?
c. Develop an iterative algorithm to compute the spatial median by solving a sequence of weighted sample means. Hint: find a majorizer of the ℓ_2-norm in the form of a squared ℓ_2-norm and then employ the majorization–minimization framework.

3.7 (ML estimation of covariance matrix) Consider an N-dimensional i.i.d. time series with zero mean and identity covariance matrix, $\boldsymbol{x}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}),\ t = 1, \ldots, T$.

a. Generate Gaussian data for $N = 10$ and $T = 50$ and estimate the covariance matrix $\boldsymbol{\Sigma}$ via the Gaussian ML estimator and the heavy-tailed ML estimator. Run the experiment multiple times and compute the mean squared error of the estimators.

b. Now repeat the whole experiment but generating instead heavy-tailed data (e.g., following a t distribution) with the same mean and covariance matrix. Observe the robustness of the two estimators against heavy tails and discuss.

3.8 (Derivation of Gaussian ML estimators) Given T N-dimensional observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$, the Gaussian ML estimation for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ is formulated as

$$\underset{\boldsymbol{\mu}, \boldsymbol{\Sigma}}{\text{minimize}} \quad \log\det(\boldsymbol{\Sigma}) + \frac{1}{T}\sum_{t=1}^{T} (\boldsymbol{x}_t - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_t - \boldsymbol{\mu}).$$

Derive the estimators by setting the gradient of the objective function with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}^{-1}$ to zero.

3.9 (Derivation of heavy-tailed ML estimators) Given T N-dimensional observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$, the heavy-tailed ML estimation (under the t distribution with degrees of freedom ν) for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ is formulated as

$$\underset{\boldsymbol{\mu}, \boldsymbol{\Sigma}}{\text{minimize}} \quad \log\det(\boldsymbol{\Sigma}) + \frac{\nu + N}{T}\sum_{t=1}^{T} \log\left(1 + \frac{1}{\nu}(\boldsymbol{x}_t - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_t - \boldsymbol{\mu})\right).$$

Derive the fixed-point equations characterizing the estimators by setting the gradient of the objective function with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}^{-1}$ to zero.

3.10 (Shrinkage James–Stein estimator for the sample mean) Consider a Gaussian-distributed i.i.d. N-dimensional time series with zero mean and identity covariance matrix, $\boldsymbol{x}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}),\ t = 1, \ldots, T$.

a. Generate data for $N = 10$ and $T = 20$, and estimate the mean vector with the sample mean and with the shrinkage James–Stein estimator.
b. Run the experiment multiple times and compute the mean squared error of the estimators.
c. Finally, repeat the experiment multiple times, for different numbers of observations $T = 10, 20, \ldots, 100$, and plot the mean squared error as a function of T.

3.11 (Shrinkage sample covariance matrix estimator) Consider a Gaussian-distributed i.i.d. N-dimensional time series with zero mean and identity covariance matrix, $\boldsymbol{x}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}),\ t = 1, \ldots, T$.

a. Generate data for $N = 10$ and $T = 20$, and estimate the covariance matrix with the sample covariance matrix and with the shrinkage Ledoit–Wolf estimator.
b. Run the experiment multiple times and compute the mean squared error of the estimators.
c. Finally, repeat the experiment multiple times, for different numbers of observations $T = 10, 20, \ldots, 100$, and plot the mean squared error as a function of T.

3.12 (Factor model estimator) Consider a Gaussian-distributed i.i.d. N-dimensional time series with zero mean and covariance matrix with a single-factor structure $\boldsymbol{\Sigma} = \boldsymbol{\beta}\boldsymbol{\beta}^{\mathsf{T}} + \boldsymbol{I}$ (e.g., with $\boldsymbol{\beta} = \mathbf{1}$), $\boldsymbol{x}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}),\ t = 1, \ldots, T$.

a. Generate data for $N = 10$ and $T = 20$, and estimate the covariance matrix with the sample covariance matrix and with the single-factor model structure (e.g., with PCA).
b. Run the experiment multiple times and compute the mean squared error of the estimators.
c. Finally, repeat the experiment multiple times, for different numbers of observations $T = 10, 20, \ldots, 100$, and plot the mean squared error as a function of T.

References

Abramovich, Y. I. (1981). A controlled method for adaptive optimization of filters using the criterion of maximum signal-to-noise ratio. *Radio Engineering and Electronic Physics*, *25*(3), 87–95.

Anderson, T. W. (2003). *An Introduction to Multivariate Statistical Analysis* (3rd ed.). Wiley-Interscience.

Bartz, D. (2016). Cross-validation based nonlinear shrinkage. *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Black, F., & Litterman, R. (1991). Asset allocation: Combining investor views with market equilibrium. *The Journal of Fixed Income*, *2*(1), 7–18.

Bun, J., Bouchaud, J.-P., & Potters, M. (2006). Cleaning correlation matrices. *Risk Magazine*.

Bun, J., Bouchaud, J.-P., & Potters, M. (2017). Cleaning large correlation matrices: Tools from random matrix theory. *Physics Reports*, *666*, 1–109.

Campbell, J. Y., Lo, A. W., & MacKinlay, A. C. (1997). *The Econometrics of Financial Markets*. Princeton University Press.

Chen, Y., Wiesel, A., & Hero III, A. O. (2011). Robust shrinkage estimation of high-dimensional covariance matrices. *IEEE Transactions on Signal Processing*, *59*(9), 4097–4107.

Chopra, V., & Ziemba, W. (1993). The effect of errors in means, variances and covariances on optimal portfolio choice. *Journal of Portfolio Management*, *19*(2), 6–11.

Fabozzi, F. J., Focardi, S. M., & Kolm, P. N. (2010). *Quantitative Equity Investing: Techniques and Strategies*. John Wiley & Sons.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, *25*(2), 383–417.

Fama, E. F., & French, K. R. (1992). The cross-section of expected stock returns. *The Journal of Finance*, *47*(2), 427–465.

Fama, E. F., & French, K. R. (2015). A five-factor asset pricing model. *The Journal of Finance*, *116*(1), 1–22.

Feng, Y., & Palomar, D. P. (2016). A signal processing perspective on financial engineering. *Foundations and Trends in Signal Processing, Now Publishers*, *9*(1–2), 1–231.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer.

Huber, P. J. (1964). Robust estimation of a location parameter. *The Annals of Statistics*, *53*(1), 73–101.

Huber, P. J. (2011). *Robust Statistics*. Springer.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: With Applications in R*. Springer.

James, W., & Stein, C. (1961). Estimation with quadratic loss. *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability*, *Vol. 1*, (pp. 361–379).

Jolliffe, I. (2002). *Principal Component Analysis*. Springer.

Jorion, P. (1986). Bayes–Stein estimation for portfolio analysis. *Journal of Finance and Quantitative Analysis*, *21*(3), 279–292.

Kay, S. M. (1993). *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall.

Kent, J. T., & Tyler, D. E. (1991). Redescending M-estimates of multivariate location and scatter. *The Annals of Statistics*, *19*(4), 2102–2119.

Kent, J. T., Tyler, D. E., & Vard, Y. (1994). A curious likelihood identity for the multivariate t-distribution. *Communications in Statistics – Simulation and Computation*, *23*(2), 441–453.

Khamaru, K., & Mazumder, R. (2019). Computation of the maximum likelihood estimator in low-rank factor analysis. *Mathematical Programming*, *176*, 279–310.

Ledoit, O., & Wolf, M. (2003). Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of Empirical Finance*, *10*(5), 603–621.

Ledoit, O., & Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, *88*(2), 365–411.

Ledoit, O., & Wolf, M. (2017). Nonlinear shrinkage of the covariance matrix for portfolio selection. *The Review of Financial Studies*, *30*(12), 4349–4388.

Liu, C., & Rubin, D. B. (1995). ML estimation of the t-distribution using EM and its extensions, ECM and ECME. *Statistica Sinica*, *5*(1), 19–39.

Liu, C., Rubin, D. B., & Wu, Y. N. (1998). Parameter expansion to accelerate EM: The PX-EM algorithm. *Biometrika*, *85*(4), 755–770.

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer.

Malkiel, B. G. (1973). *A Random Walk Down Wall Street*. W. W. Norton.

Markowitz, H. M. (1952). Portfolio selection. *The Journal of Finance*, *7*(1), 77–91.

Maronna, R. A. (1976). Robust M-estimators of multivariate location and scatter. *The Annals of Statistics*, *4*(1), 51–67.

Maronna, R., Martin, D., & Yohai, V. (2006). *Robust Statistics: Theory and Methods*. John Wiley & Sons.

McNeil, A. J., Frey, R., & Embrechts, P. (2015). *Quantitative Risk Management*. Princeton University Press.

Meucci, A. (2005). *Risk and Asset Allocation*. Springer.

Oja, H. (2013). Multivariate median. In C. Becker, R. Fried, & S. Kuhnt (Eds.), *Robustness and Complex Data Structures*. Springer.

Ollila, E., Palomar, D. P., & Pascal, F. (2023). Affine equivariant Tyler's M-estimator applied to tail parameter learning of elliptical distributions. *IEEE Signal Processing Letters*, *30*, 1017–1021.

Ollila, E., Pascal, F., & Palomar, D. P. (2021). Shrinking the eigenvalues of M-estimators of covariance matrix. *IEEE Transactions on Signal Processing*, *69*, 256–269.

Ollila, E., & Raninen, E. (2019). Optimal shrinkage covariance matrix estimation under random sampling from elliptical distributions. *IEEE Transactions on Signal Processing*, *67*(10), 2707–2719.

Palomar, D. P., Zhou, R., Wang, X., Pascal, F., & Ollila, E. (2023). *fitHeavyTail: Mean and Covariance Matrix Estimation Under Heavy Tails* [R package]. https://CRAN.R-project.org/package=fitHeavyTail

Papoulis, A. (1991). *Probability, Random Variables, and Stochastic Processes* (3rd ed.). McGraw-Hill.

Ruppert, D., & Matteson, S. D. (2015). *Statistics and Data Analysis for Financial Engineering: With R Examples* (2nd ed.). Springer.

Santamaria, I., Scharf, L. L., Via, J., Wang, H., & Wang, Y. (2017). Passive detection of correlated subspace signals in two MIMO channels. *IEEE Transactions on Signal Processing*, *65*(20), 5266–5280.

Sardarabadi, A. M., & van der Veen, A. J. (2018). Complex factor analysis and extensions. *IEEE Transactions on Signal Processing*, *66*(4), 954–967.

Scharf, L. L. (1991). *Statistical Signal Processing*. Addison-Wesley.

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, *19*(3), 425–442.

Small, C. G. (1990). A survey of multidimensional medians. *International Statistical Review*, *58*(3), 263–277.

Srivastava, M. S., & Bilodeau, M. (1989). Stein estimation under elliptical distributions. *Journal of Multivariate Analysis*, *28*(2), 247–259.

Stein, C. (1955). Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. *Proceedings of the 3rd Berkeley Symposium on Probability and Statistics*, *Vol. 1*, (pp. 197–206).

Sun, Y., Babu, P., & Palomar, D. P. (2014). Regularized Tyler's scatter estimator: Existence, uniqueness, and algorithms. *IEEE Transactions on Signal Processing*, *62*(19), 5143–5156.

Sun, Y., Babu, P., & Palomar, D. P. (2015). Regularized robust estimation of mean and covariance matrix under heavy-tailed distributions. *IEEE Transactions on Signal Processing*, *63*(12), 3096–3109.

Sun, Y., Babu, P., & Palomar, D. P. (2017). Majorization–minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, *65*(3), 794–816.

Tsay, R. S. (2010). *Analysis of Financial Time Series* (3rd ed.). John Wiley & Sons.

Tsay, R. S. (2013). *Multivariate Time Series Analysis: With R and Financial Applications*. John Wiley & Sons.

Tyler, D. E. (1987). A distribution-free M-estimator of multivariate scatter. *The Annals of Statistics*, *15*(1), 234–251.

Tyler, D. E., & Yi, M. (2020). Lassoing eigenvalues. *Biometrika*, *107*(2), 397–414.

Wan, X., Wang, W., Liu, J., & Tong, T. (2014). Estimating the sample mean and standard deviation from the sample size, median, range and/or interquartile range. *BMC Medical Research Methodology*, *14*(135), 1–13.

Wiesel, A. (2012). Unified framework to regularized covariance estimation in scaled Gaussian models. *IEEE Transactions on Signal Processing*, *60*(1), 29–38.

Wiesel, A., & Zhang, T. (2014). Structured robust covariance estimation. *Foundations and Trends in Signal Processing, Now Publishers*, *8*(3), 127–216.

Zhang, M., Rubio, F., & Palomar, D. P. (2013). Improved calibration of high-dimensional precision matrices. *IEEE Transactions on Signal Processing*, *61*(6), 1509–1519.

Zhang, M., Rubio, F., Palomar, D. P., & Mestre, X. (2013). Finite-sample linear filter optimization in wireless communications and financial systems. *IEEE Transactions on Signal Processing*, *61*(20), 5014–5025.

Zhou, R., Liu, J., Kumar, S., & Palomar, D. P. (2020). Robust factor analysis parameter estimation. In R. Moreno-Díaz, F. Pichler, & A. Quesada-Arencibia (Eds.), *Computer Aided Systems Theory – EUROCAST 2019* (pp. 3–11). Springer.

Zhou, R., Ying, J., & Palomar, D. P. (2022). Covariance matrix estimation under low-rank factor model with nonnegative correlations. *IEEE Transactions on Signal Processing*, *70*, 4020–4030.

Zoubir, A. M., Koivunen, V., Ollila, E., & Muma, M. (2018). *Robust Statistics for Signal Processing*. Cambridge University Press.

4

Financial Data: Time Series Modeling

"It is very hard to predict, especially the future."

— Niels Bohr

The efficient-market hypothesis states that the price of a security already contains all the publicly available information about the future (Fama, 1970). From that, it makes sense to model a sequence of prices as a random walk (Malkiel, 1973) or, equivalently, to model returns as a sequence of independent and identically distributed (i.i.d.) random variables as explored in Chapter 3. This is a widely adopted model by practitioners and academics.

Nevertheless, another line of thought precisely supports the opposite view in favor of inefficient and irrational markets (Shiller, 1981) under so-called behavioral finance (Shiller, 2003). Indeed, it is undeniable that financial data exhibit some temporal structure that could be potentially modeled and exploited (Lo & Mackinlay, 2002). One of the most noticeable structural aspects is the volatility clustering (described in Chapter 2). This chapter examines the temporal structure in financial time series in the form of mean models and variance (or volatility) models with an emphasis on the Kalman filter.

4.1 Temporal Structure

From the exploratory analysis of financial data and stylized facts in Chapter 2, one can either assume the i.i.d. model, as pursued in detail in Chapter 3, or try to incorporate some temporal structure in the model, as this chapter attempts.

The i.i.d. model can be motivated by Fama's efficient-market hypothesis (Fama, 1970), which holds that one cannot forecast future prices since the price of a security already contains all the publicly available information. On the other hand, an equally popular line of thought in finance supports inefficient and irrational markets (Shiller, 1981) under behavioral finance (Shiller, 2003).[1] Under this premise, the market is predictable to some degree and perhaps

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

[1] It is somewhat interesting that both Robert J. Shiller and Eugene F. Fama were awarded the 2013 Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel, when they support opposing views on the nature of financial markets, namely, the inefficiency of markets and the efficient-market hypothesis, respectively.

it may move in trends so that the study of past prices can be used to forecast future price direction.

Thus, rather than assuming the random walk model (Malkiel, 1973), we will focus on nonrandom walk models (Lo & Mackinlay, 2002). Suppose we have N securities or tradable assets and let $\boldsymbol{x}_t \in \mathbb{R}^N$ be the random returns of the assets at time t. Instead of using the i.i.d. model for the returns, $\boldsymbol{x}_t = \boldsymbol{\mu} + \boldsymbol{\epsilon}_t$ as in (3.1), we will now transition to a more general model in which the returns, $\boldsymbol{x}_t$, are modeled conditioned on the past observations, denoted by $\mathcal{F}_{t-1} \triangleq \{\ldots, \boldsymbol{x}_{t-2}, \boldsymbol{x}_{t-1}\}$, and then write

$$\boldsymbol{x}_t = \boldsymbol{\mu}_t + \boldsymbol{\epsilon}_t, \tag{4.1}$$

where $\boldsymbol{\mu}_t \in \mathbb{R}^N$ is the *conditional expected return*

$$\boldsymbol{\mu}_t = \mathbb{E}\left[\boldsymbol{x}_t \mid \mathcal{F}_{t-1}\right]$$

and $\boldsymbol{\epsilon}_t \in \mathbb{R}^N$ denotes the error in the model (also called innovation or residual) with zero mean and *conditional covariance matrix*

$$\boldsymbol{\Sigma}_t = \mathbb{E}\left[(\boldsymbol{x}_t - \boldsymbol{\mu}_t)(\boldsymbol{x}_t - \boldsymbol{\mu}_t)^\mathsf{T} \mid \mathcal{F}_{t-1}\right].$$

The i.i.d. model in (3.1) from Chapter 3 can be obtained with the particular choice $\boldsymbol{\mu}_t = \boldsymbol{\mu}$ and $\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}$, which remain fixed over time.

Modeling the returns $\boldsymbol{x}_t$ conditioned on the historical data $\mathcal{F}_{t-1}$ is precisely one of the objectives in the field of *econometrics*, which integrates statistical and mathematical models to formulate theories or test existing hypotheses in economics, as well as to predict future trends based on historical data. Figure 4.1 shows an example of a synthetic univariate Gaussian AR(1) time series (model described later), where we can observe some temporal structure.

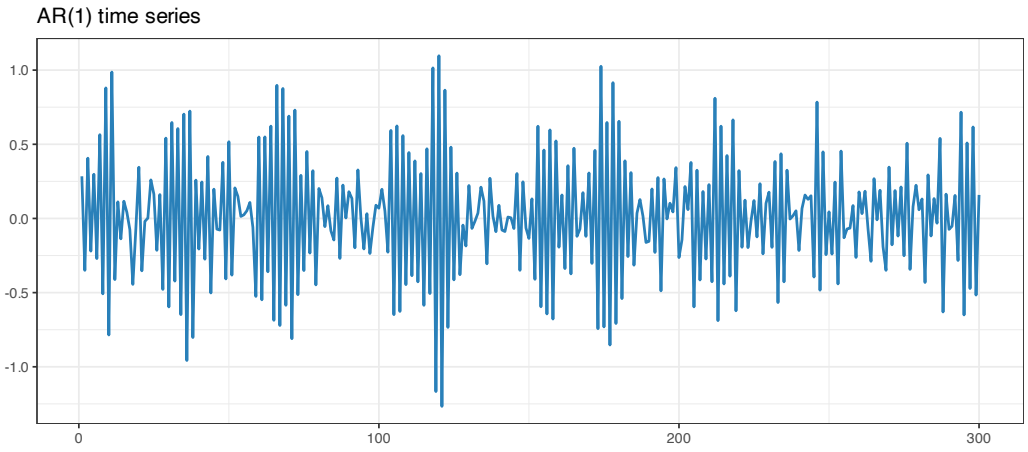


Figure 4.1 Example of a synthetic Gaussian AR(1) time series.

Some accessible textbooks that cover financial data modeling include Tsay (2010) and Ruppert and Matteson (2015), with more emphasis on the multivariate case in Lütkepohl (2007) and

Tsay (2013). Some excellent survey papers are also available, such as Bollerslev et al. (1992), Taylor (1994), and Poon and Granger (2003).

This chapter deviates slightly from the conventional econometric modeling approaches, which typically revolve around autoregressive models and "GARCH" volatility models. Instead, the emphasis is placed on the simplicity of models combined with the versatile Kalman filter (often underutilized in financial literature, although still covered in Tsay (2010) and Lütkepohl (2007)). Additionally, stochastic volatility modeling is emphasized here, which frequently does not receive the attention it deserves in favor of more popular "GARCH" models. Specifically, after an introduction to Kalman filtering in Section 4.2, the following two families of models are presented:

- *mean models*: for the conditional expected return $\boldsymbol{\mu}_t = \mathbb{E}\left[\boldsymbol{x}_t \mid \mathcal{F}_{t-1}\right]$ in Section 4.3; and
- *variance models*: for the conditional covariance matrix $\boldsymbol{\Sigma}_t = \mathbb{E}\left[(\boldsymbol{x}_t - \boldsymbol{\mu}_t)(\boldsymbol{x}_t - \boldsymbol{\mu}_t)^\top \mid \mathcal{F}_{t-1}\right]$ in Section 4.4.

4.2 Kalman Filter

State-space modeling provides a unified framework for treating a wide range of problems in time series analysis. It can be thought of as a universal and flexible modeling approach with a very efficient algorithm: the *Kalman filter*. The basic idea is to assume that the evolution of the system over time is driven by a series of unobserved or hidden values, which can only be measured indirectly through observations of the system output. This modeling can be used for filtering, smoothing, and forecasting.

The Kalman filter is named after Rudolf E. Kalman, who was one of the primary developers of the theory (Kalman, 1960). It is sometimes called the Kalman–Bucy filter or even Stratonovich–Kalman–Bucy filter, because Richard S. Bucy also contributed to the theory and Ruslan Stratonovich earlier proposed a more general nonlinear version. Arguably, some of the most comprehensive and authoritative classical references for state-space models and Kalman filtering include the textbooks Anderson and Moore (1979) and Durbin and Koopman (2012), which was originally published in 2001. Other textbook references on time series and the Kalman filter include Brockwell and Davis (2002), Shumway and Stoffer (2017), Harvey (1989) and, in particular, for financial time series, Zivot et al. (2004), Tsay (2010), Lütkepohl (2007), and Harvey and Koopman (2009).

The Kalman filter, which was employed by NASA during the 1960s in the Apollo program, now boasts a vast array of technological applications. It is commonly utilized in the guidance, navigation, and control of vehicles, including aircraft, spacecraft, and maritime vessels. It has also found applications in time series analysis, signal processing, and econometrics. More recently, it has become a key component in robotic motion planning and control, as well as trajectory optimization.

Currently, the software implementation of Kalman filtering is widespread and numerous libraries are available in most programming languages, for example Tusell (2011), Petris and Petrone (2011), Helske (2017), and Holmes et al. (2012) for the R programming language.

4.2.1 State-Space Model

Mathematically, the Kalman filter is based on the following linear Gaussian state-space model with discrete time $t = 1, \ldots, T$ (Durbin & Koopman, 2012):

$$\begin{aligned} \boldsymbol{y}_t &= \boldsymbol{Z}\boldsymbol{\alpha}_t + \boldsymbol{\epsilon}_t \quad \text{(observation equation)}, \\ \boldsymbol{\alpha}_{t+1} &= \boldsymbol{T}\boldsymbol{\alpha}_t + \boldsymbol{\eta}_t \quad \text{(state equation)}, \end{aligned} \tag{4.2}$$

where $\boldsymbol{y}_t$ denotes the observations over time with observation matrix $\boldsymbol{Z}$, $\boldsymbol{\alpha}_t$ represents the unobserved or hidden internal state with state transition matrix $\boldsymbol{T}$, and the two noise terms $\boldsymbol{\epsilon}_t$ and $\boldsymbol{\eta}_t$ are Gaussian distributed with zero mean and covariance matrices $\boldsymbol{H}$ and $\boldsymbol{Q}$, respectively, that is, $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{H})$ and $\boldsymbol{\eta}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$. The initial state can be modeled as $\boldsymbol{\alpha}_1 \sim \mathcal{N}(\boldsymbol{a}_1, \boldsymbol{P}_1)$. Mature and efficient software implementations of the model in (4.2) are readily available, for example Helske (2017).[2]

It is worth mentioning that an alternative notation widespread in the literature for the state-space model (4.2) is to shift the time index in the state equation by one: $\boldsymbol{\alpha}_t = \boldsymbol{T}\boldsymbol{\alpha}_{t-1} + \boldsymbol{\eta}_t$. This change in notation only has a slight effect on the initial point of the system, which is now $\boldsymbol{\alpha}_0$ (corresponding to the time before the first observation) instead of $\boldsymbol{\alpha}_1$ (corresponding to the same time as the first observation); other than that, it is just a notational preference.

The parameters of the state-space model in (4.2) (i.e., $\boldsymbol{Z}$, $\boldsymbol{T}$, $\boldsymbol{H}$, $\boldsymbol{Q}$, $\boldsymbol{a}_1$, and $\boldsymbol{P}_1$) can be provided by the user (if known). Otherwise, they can be inferred from the data with algorithms based on maximum likelihood estimation. Again, mature and efficient software implementations are available for parameter fitting (Holmes et al., 2012).[3] In order to build some intuition about state-space models, let us look at a simple yet illustrative example.

Example 4.1 (Tracking via state-space model) Suppose we want to track an object in one dimension over time, x_t, from noisy measurements $y_t = x_t + \epsilon_t$ measured at time intervals Δt. We provide four different ways to model this system, from the simplest to the most advanced, based on the state-space model in (4.2).

1. If we define the internal state simply as the position, $\alpha_t = x_t$, then (4.2) simply becomes

$$\begin{aligned} y_t &= x_t + \epsilon_t, \\ x_{t+1} &= x_t + \eta_t, \end{aligned}$$

   where it is tacitly assumed that the position x_t does not change much.

2. If now we incorporate the velocity v_t in the internal state, $\boldsymbol{\alpha}_t = \begin{bmatrix} x_t \\ v_t \end{bmatrix}$, then the state-space model becomes

$$\begin{aligned} y_t &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_t \\ v_t \end{bmatrix} + \epsilon_t, \\ \begin{bmatrix} x_{t+1} \\ v_{t+1} \end{bmatrix} &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ v_t \end{bmatrix} + \boldsymbol{\eta}_t, \end{aligned}$$

[2] The R package `KFAS` implements the Kalman filter for the model (4.2) (Helske, 2017). The Python package `filterpy` provides Kalman methods.

[3] The R package `MARSS` implements algorithms for fitting the unknown parameters of the state-space model (4.2) based on observed time series data (Holmes et al., 2012).

where now the position is better modeled thanks to also modeling the velocity.

3. We can further include the acceleration a_t in the internal state, $\boldsymbol{\alpha}_t = \begin{bmatrix} x_t \\ v_t \\ a_t \end{bmatrix}$, leading to the state-space model

$$y_t = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_t \\ v_t \\ a_t \end{bmatrix} + \epsilon_t,$$
$$\begin{bmatrix} x_{t+1} \\ v_{t+1} \\ a_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ v_t \\ a_t \end{bmatrix} + \boldsymbol{\eta}_t.$$

4. Finally, we can further improve the model, especially if the sampling rate is not high enough, by including the acceleration in the position equation, $x_{t+1} = x_t + \Delta t v_t + \frac{1}{2}\Delta t^2 a_t$, leading to

$$y_t = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_t \\ v_t \\ a_t \end{bmatrix} + \epsilon_t,$$
$$\begin{bmatrix} x_{t+1} \\ v_{t+1} \\ a_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ v_t \\ a_t \end{bmatrix} + \boldsymbol{\eta}_t.$$

It is important to emphasize that the state-space model in (4.2) is not the most general one. One trivial extension is to allow the parameters $\boldsymbol{Z}$, $\boldsymbol{T}$, $\boldsymbol{H}$, and $\boldsymbol{Q}$ to change over time: $\boldsymbol{Z}_t$, $\boldsymbol{T}_t$, $\boldsymbol{H}_t$, and $\boldsymbol{Q}_t$. More generally, one can relax the two key assumptions in (4.2), by (i) allowing nonlinear functions of $\boldsymbol{\alpha}_t$ (instead of the linear functions $\boldsymbol{Z}\boldsymbol{\alpha}_t$ and $\boldsymbol{T}\boldsymbol{\alpha}_t$) and (ii) not assuming the noise distributions to be Gaussian. This leads to extensions proposed in the literature such as the *extended Kalman filter*, the *unscented Kalman filter*, and even the more general (albeit more computationally demanding) *particle filtering* (Durbin & Koopman, 2012).

4.2.2 Kalman Filtering and Smoothing

The Kalman filter is a very efficient algorithm to optimally solve the state-space model in (4.2), which is linear and assumes Gaussian distributions for the noise terms. Its computational cost is manageable to the point that it was even used in the Apollo program by NASA in the 1960s: it was quite remarkable that it could be implemented in a tiny and rudimentary computer (2 KB of magnetic core RAM, 36 KB of core rope memory (ROM), CPU built from ICs with a clock speed under 100 kHz).

The objective of *Kalman filtering* is to characterize the distribution of the hidden state at time t, $\boldsymbol{\alpha}_t$, given the observations up to (and including) time t, $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$, that is, in a causal manner. Since the distribution of the noise terms is Gaussian, it follows that the conditional distribution of $\boldsymbol{\alpha}_t$ is also Gaussian; therefore, it suffices to characterize the conditional mean

and conditional covariance matrix:

$$\boldsymbol{a}_{t|t} \triangleq \mathbb{E}\left[\boldsymbol{\alpha}_t \mid (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)\right],$$
$$\boldsymbol{P}_{t|t} \triangleq \mathrm{Cov}\left[\boldsymbol{\alpha}_t \mid (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)\right].$$

For forecasting purposes, one is really interested in the distribution of the hidden state at time $t+1$, $\boldsymbol{\alpha}_{t+1}$, given the observations up to (and including) time t, denoted by $\boldsymbol{a}_{t+1|t}$ and $\boldsymbol{P}_{t+1|t}$. These filtering and forecasting quantities can be efficiently computed using a "forward pass" algorithm that goes from $t = 1$ to $t = T$ in a sequential way, so that it can operate in real time (Durbin & Koopman, 2012).

On the other hand, the objective of *Kalman smoothing* is to characterize the distribution of the hidden state at time t, $\boldsymbol{\alpha}_t$, given all the observations, $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T$, that is, in a noncausal manner. Such a distribution is also Gaussian and it is fully characterized by the following conditional mean and conditional covariance matrix:

$$\boldsymbol{a}_{t|T} \triangleq \mathbb{E}\left[\boldsymbol{\alpha}_t \mid (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T)\right],$$
$$\boldsymbol{P}_{t|T} \triangleq \mathrm{Cov}\left[\boldsymbol{\alpha}_t \mid (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T)\right].$$

Interestingly, these quantities can also be efficiently computed using a "backward pass" algorithm that goes from $t = T$ to $t = 1$ (Durbin & Koopman, 2012). Since this requires all observations, it is naturally a batch-processing algorithm rather than an online one.

Overall, the full characterization of the hidden states executes both forward and backward passes very efficiently. The choice of filtering vs. smoothing depends on whether the application requires a causal approach (real time) or noncausal (batch processing). Obviously, the hidden state characterization of smoothing is much better than filtering as it uses more information.

Figure 4.2 shows the result of Kalman filtering and Kalman smoothing for the four different state-space models in Example 4.1 (properly fitting the variances of the noise terms via maximum likelihood). In general, the more accurate the model, the better the performance. In this specific example, however, the differences are minimal. On the other hand, Kalman smoothing significantly outperforms Kalman filtering because it has access to all observations simultaneously.

4.3 Mean Modeling

In econometrics, we are interested in forecasting the future values of a financial time series given past observations. A first choice we need to make is whether to focus on the time series of the prices or the returns. Of course they are trivially related to each other, but it is not clear a priori which one is more manageable: the returns tend to be more constant and are easier to model, whereas the prices tend to present a trend. Recall that the log-prices are denoted as $\boldsymbol{y}_t = \log \boldsymbol{p}_t$ and the log-returns as $\boldsymbol{x}_t = \boldsymbol{y}_t - \boldsymbol{y}_{t-1}$ (see Chapter 3 for the random walk model).

Unless otherwise stated, we will focus on the univariate case (single asset) for simplicity. In fact, in most practical cases, the multivariate case can be treated on an asset-by-asset basis. Thus, the objective is to compute the expectation of the future value of the time series at time t based on the past observations $\mathcal{F}_{t-1}$ with the hope of extracting any structural information, that is, either $\mathbb{E}[y_t \mid \mathcal{F}_{t-1}]$ for the log-prices or $\mathbb{E}[x_t \mid \mathcal{F}_{t-1}]$ for the log-returns.

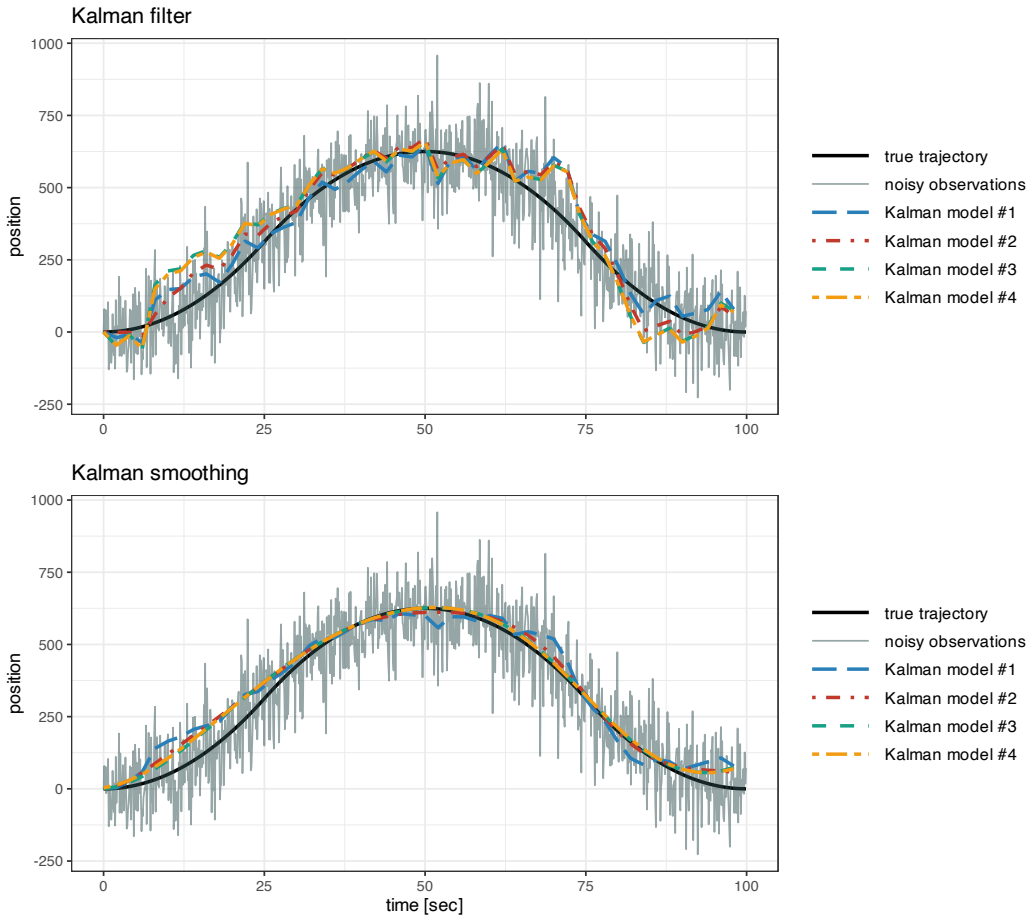


Figure 4.2 Example of Kalman position tracking under the four different models from Example 4.1.

Readers should be aware that there might not be any significant temporal structural information to leverage. In such cases, the i.i.d. model from Chapter 3 would suffice. In fact, this is partly supported by the exploratory data analysis in Section 2.4 of Chapter 2, where the autocorrelation of the returns appears to be insignificant. Ultimately, this depends on the nature of the data and, specifically, on the frequency of the data observations.

4.3.1 Moving Average (MA)

Inspired by the i.i.d. model for the returns $x_t = \mu + \epsilon_t$ in (3.1), the simplest way to estimate μ is by taking the average of several observations in order to reduce the effect of the noise or innovation component ϵ_t.

The *moving average* (MA) of order q, denoted by MA(q), is

$$\hat{x}_t = \frac{1}{q}\sum_{i=1}^{q} x_{t-i}, \tag{4.3}$$

where q denotes the lookback and determines the amount of averaging. Since this sample mean is computed for each value of t on a rolling-window basis, it is also commonly referred to as *rolling means*. A noncausal variation (appropriate for smoothing but not for forecasting) considers a centered version of the moving average by using past and future samples around x_t.

Observe that, under the i.i.d. model $x_t = \mu + \epsilon_t$, the moving average is indeed estimating μ by averaging the noise:

$$\hat{x}_t = \mu + \frac{1}{q}\sum_{i=1}^{q} \epsilon_{t-i},$$

where the averaged noise component has a variance reduced by a factor of q. If, instead of the i.i.d. model, μ_t is allowed to change over time, the effect of the moving average is to approximate this slowly changing value.

It is insightful to explore the interpretation of the moving average on the log-returns from the perspective of log-prices. Using $x_t = y_t - y_{t-1}$, the moving average in (4.3) can be rewritten as

$$\hat{x}_t = \frac{1}{q}(y_{t-1} - y_{t-q-1}).$$

This shows that it is effectively computing the trend of the log-prices in the form of a slope.

Alternatively, the moving average operation can be implemented on the log-prices instead of log-returns:

$$\hat{y}_t = \frac{1}{q}\sum_{i=1}^{q} y_{t-i}. \tag{4.4}$$

This is widely employed in the context of "charting" or "technical analysis" (Malkiel, 1973), where it is typically performed directly on the prices p_t (without the log operation). However, this does not seem to have any solid mathematical foundation apart from the visual interpretation of "smoothing" the noisy time series.

Figure 4.3 illustrates the effect of forecasting via moving averages performed on the log-prices and log-returns. As expected from the theoretical analysis, the MA on the log-prices is much worse, albeit being widely used in the "charting" community. Note that the difference of using prices instead of log-prices or using returns instead of log-returns is insignificant and not reported. Table 4.1 reports the mean squared error of the forecasting and it shows that, again, averaging the log-returns makes more sense (the lookback value q can be more carefully selected for improved performance).

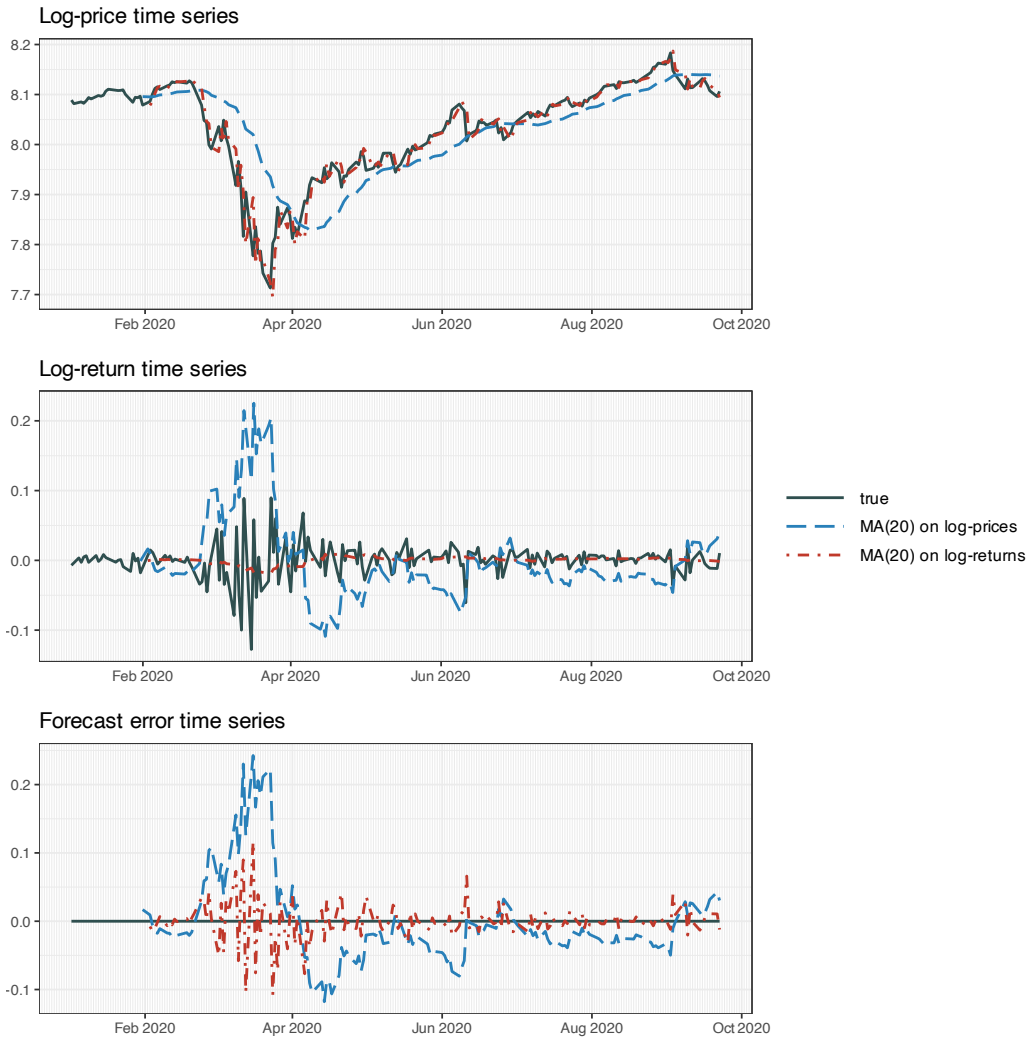


Figure 4.3 Forecasting with moving average.

Table 4.1 Comparison of moving average forecasting in terms of mean squared error.

	MSE
MA(20) on log-prices	0.004 032
MA(20) on log-returns	0.000 725

4.3.2 EWMA

The simple moving average in Section 4.3.1 essentially computes the average of the past observations. However, one can argue that the more recent observations should be weighted

more than the less recent ones. This can be conveniently achieved with the *exponentially weighted moving average* (EWMA), or simply *exponential moving average* (EMA), with the recursive computation

$$\hat{x}_t = \alpha x_{t-1} + (1-\alpha)\hat{x}_{t-1}, \tag{4.5}$$

where α (with $0 \leq \alpha \leq 1$) determines the exponential decay or memory.

We can easily see that this recursion is effectively implementing exponential weights (hence the name):

$$\hat{x}_t = \alpha x_{t-1} + \alpha(1-\alpha)x_{t-2} + \alpha(1-\alpha)^2 x_{t-3} + \alpha(1-\alpha)^3 x_{t-4} + \cdots$$

4.3.3 ARMA Modeling

The "bread and butter" modeling and forecasting techniques in finance are extensions of the basic MA and EMA models previously described. These classical methods are essentially autoregressive models that attempt to capture any existing linear structure in the return time series. These models have been extensively explored for the past five decades and have reached a good level of maturity. We now summarize the basic ideas without going into any level of detail; more information can be found in the many comprehensive textbooks (Lütkepohl, 2007; Ruppert & Matteson, 2015; Tsay, 2010, 2013).

The most basic *autoregressive* (AR) model is of order 1, denoted by AR(1), given as

$$x_t = \phi_0 + \phi_1 x_{t-1} + \epsilon_t,$$

where ϕ_0 and ϕ_1 (as well as the variance σ^2 of the noise term ϵ_t) are the parameters of the model to be determined via a fitting procedure on historical data. This model attemps to capture any linear dependency between the consecutive returns. More generally, an autoregressive model of order p, AR(p), is given by

$$x_t = \phi_0 + \sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon_t,$$

which now contains more parameters, $\phi_0, \ldots, \phi_p$ and σ^2, to be fitted (also, the order p has to be determined).

Similarly, we have the moving average models that attempt to exploit any linear dependency from past noise terms by averaging the last q values. Combining both components leads us to the popular *autoregressive moving average* (ARMA) models. In particular, an ARMA model of orders p and q, denoted by ARMA(p, q), is written as

$$x_t = \phi_0 + \sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon_t - \sum_{j=1}^{q} \psi_j \epsilon_{t-j}, \tag{4.6}$$

where now the parameters are $\phi_0, \ldots, \phi_p, \psi_1, \ldots, \psi_q$ and σ^2.

Under the ARMA model in (4.6), the forecast of x_t is given by the conditional expected return

$$\hat{x}_t \triangleq \mu_t = \mathbb{E}[x_t \mid \mathcal{F}_{t-1}] = \phi_0 + \sum_{i=1}^{p} \phi_i x_{t-i},$$

with conditional variance

$$\sigma_t^2 = \mathbb{E}[(x_t - \mu_t)^2 \mid \mathcal{F}_{t-1}] = \sigma^2,$$

where σ^2 is the (constant) variance of the noise term ϵ_t. Thus, ARMA modeling can properly deal with time-varying mean models; however, the variance is still constant. Time-varying variance models are considered in Section 4.4.

The ARMA model is defined on the log-returns x_t, which are stationary (the first and second moments are time invariant), as opposed to the log-prices y_t, which are nonstationary (e.g., a random walk). That is, the original log-price time series cannot be modeled directly and we have to compute the first-order difference to get $x_t = y_t - y_{t-1}$. In general, for other types of time series, we may need to take the dth-order difference before employing an ARMA model (Tsay, 2010). This is precisely what is termed the *autoregressive integrated moving average* (ARIMA) model, denoted by ARIMA(p, d, q), and is given by

$$x_t = \phi_0 + \sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon_t - \sum_{j=1}^{q} \psi_j \epsilon_{t-j},$$

where x_t is obtained by differencing the original time series y_t d times. Thus, an ARIMA model (with $d = 1$) on the log-prices is equivalent to an ARMA model on the log-returns.

ARMA models require the appropriate coefficients to properly fit the data. Mature and efficient software implementations are readily available in most programming languages for ARMA modeling, including the fitting process.[4]

All the previous models require order selection. The order of a model is characterized by the integers p and q that indicate the number of parameters. In practice, the order of a model is unknown and also has to be determined from historical data (Lütkepohl, 2007). The higher the order, the more parameters the model contains to better fit the data; however, this comes with the danger of overfitting, that is, fitting the historical data (including the noise) too well at the expense of not being able to fit the future data appropriately. The topic of overfitting is discussed in Chapter 8 in the context of backtesting. There are two main approaches to choosing the model order:

- *cross-validation*: based on splitting the historical data into a training part and a cross-validation part, the latter being used to test the model trained with different values of orders; and
- *penalization methods*: based on taking into account the number of parameters of the model with a penalty term in the assessment of the model, which has led to many methods, such as AIC, BIC, SIC, HQIC, and so on (Lütkepohl, 2007).

Figure 4.4 illustrates the effect of forecasting via ARMA modeling based on some specific choices of the orders, namely, i.i.d. model, AR(1), MA(1), and ARMA(1,1). Table 4.2 reports the mean squared error of the forecasts, from which we can infer that the i.i.d. modeling gives the best fit (not unexpected given the lack of strong autocorrelations in the returns).

[4] The R package `rugarch` implements algorithms for fitting a wide range of different ARMA models to time series data (Ghalanos, 2022). Many other packages are also available in R. The Python package `statsmodels` contains a number of statistical data modeling methods.

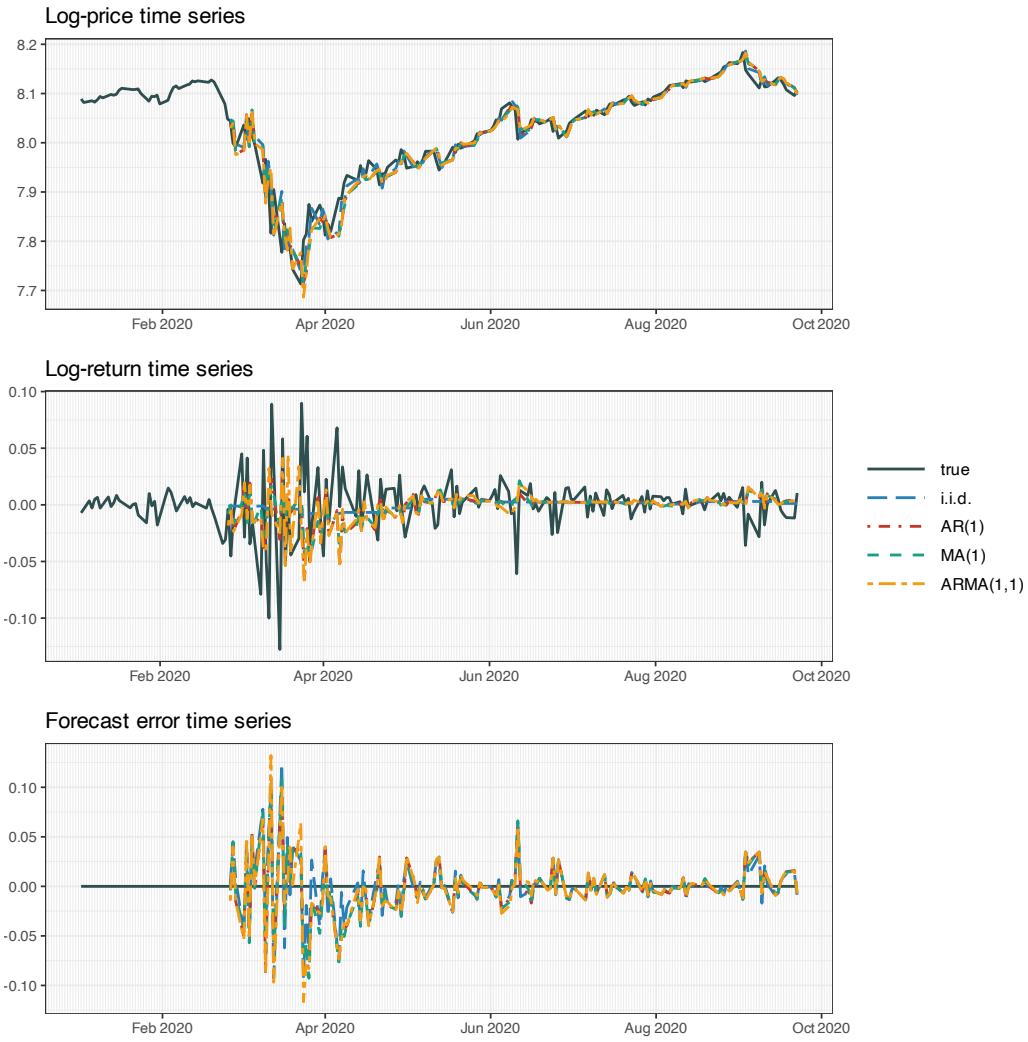


Figure 4.4 Forecasting with ARMA models.

Table 4.2 Comparison of ARMA forecasting in terms of mean squared error.

	MSE
i.i.d.	0.000 754
AR(1)	0.000 793
MA(1)	0.000 805
ARMA(1,1)	0.000 914

4.3.4 Seasonality Decomposition

In a *structural time series model*, the observed time series is viewed as a sum of unobserved components such as a trend, a seasonal component, and an irregular component (Durbin & Koopman, 2012; Lütkepohl, 2007). For example, the random walk model for the log-prices, $y_t = y_{t-1} + \mu + \epsilon_t$, can be extended to include the seasonal component γ_t as

$$y_t = \mu_t + \gamma_t + \epsilon_t,$$

where μ_t is the trend that can be modeled, for example, as $\mu_t = \mu_{t-1} + \eta_t$, and the seasonal component γ_t (with s seasons in a period) as $\gamma_t = -\sum_{j=1}^{s-1} \gamma_{t-j} + \omega_t$ so that the sum over a full period is approximately zero (ω_t is a small white noise term).

The topic of time series decomposition has received a lot of attention and a wide variety of models have been proposed since the 1950s (Hyndman et al., 2008). They are often referred to as *exponential smoothing methods* since they are sophisticated versions of the EWMA described in Section 4.3.2 combined with the idea of decomposition of the observed time series into a variety of terms such as trend, seasonality, cycle, and so on. These methods can be extremely useful if the time series indeed contains such seasonality and cyclic components. For example, intraday financial data clearly contains specific components that change with some pattern during the day, such as the so-called "volatility smile" pattern, which refers to the volatility being higher at the beginning and end of the day, while being low during the middle of the day. Interestingly, the intraday volatility decomposition can be efficiently modeled via a state-space representation and implemented with the Kalman algorithm (Chen et al., 2016; Xiu & Palomar, 2023).

4.3.5 Kalman Modeling

The random walk model for the log-prices, $y_t = y_{t-1} + \mu + \epsilon_t$ with $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$, is equivalent to the i.i.d. model for the log-returns, $x_t = \mu + \epsilon_t$, and the results of Chapter 3 can be used to fit the time series. Realistically, the drift parameter μ (as well as the volatility σ^2) will surely be time-varying. The state-space model (4.2) and the Kalman algorithm, described in Section 4.2, can be effectively used precisely to allow some time variation on the drift by treating it as a hidden state that evolves over time. In addition, it allows a more precise separation of the noise into observational noise and drift noise as described next.

The simple moving average used in (4.3) on the log-returns follows naturally from the i.i.d. model $x_t = \mu + \epsilon_t$ as a way to estimate μ. That is, we can reinterpret (4.3) in terms of estimating the drift μ based on the historical data up to time $t-1$,

$$\hat{\mu}_t = \frac{1}{m} \sum_{i=1}^{m} x_{t-i},$$

and then forecasting the value at time t as $\hat{x}_t = \hat{\mu}_t$. Instead, we can conveniently use a state-space model to allow the drift to slowly change over time by modeling the drift as the hidden state, $\alpha_t = \mu_t$, that evolves over time. This is the so-called *local level* model (Durbin

& Koopman, 2012):

$$\begin{aligned} x_t &= \mu_t + \epsilon_t \\ \mu_{t+1} &= \mu_t + \eta_t, \end{aligned} \tag{4.7}$$

where the noise term η_t allows μ_t to evolve over time. This model is expected to be more accurate than the simple moving average in (4.3). In addition, there is no parameter q to be chosen like in the MA(q) in (4.3). Of course, this model still requires the values of the variances of the noise terms, which can be chosen a priori or learned automatically via maximum likelihood.

Alternatively, the moving average was used in (4.4) on the log-prices in a rather heuristic way to smooth the otherwise noisy time series. We can again use a state-space model to improve the modeling. For example, if we define the hidden state α_t as a noiseless version of the log-prices, $\alpha_t = \tilde{y}_t$, we can write

$$\begin{aligned} y_t &= \tilde{y}_t + \epsilon_t, \\ \tilde{y}_{t+1} &= \tilde{y}_t + \mu + \eta_t, \end{aligned} \tag{4.8}$$

which allows for some observation noise via ϵ_t and some noise in the state transition via η_t. We can also allow the drift to be time-varying, μ_t, by augmenting the hidden state as $\boldsymbol{\alpha}_t = \begin{bmatrix} \tilde{y}_t \\ \mu_t \end{bmatrix}$, leading to the so-called *local linear trend* model (Durbin & Koopman, 2012):

$$\begin{aligned} y_t &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{y}_t \\ \mu_t \end{bmatrix} + \epsilon_t, \\ \begin{bmatrix} \tilde{y}_{t+1} \\ \mu_{t+1} \end{bmatrix} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{y}_t \\ \mu_t \end{bmatrix} + \boldsymbol{\eta}_t. \end{aligned} \tag{4.9}$$

Figure 4.5 illustrates the effect of forecasting via Kalman filtering based on the previous three models, namely, Kalman on log-returns (with dynamic drift μ_t), Kalman on log-prices (with static drift μ), and Kalman on log-prices (with dynamic drift μ_t). The difference is small. Table 4.3 reports the mean squared error of the forecasts, from which we can appreciate that as the models become more accurate the performance improves. Nevertheless, from a practical standpoint, one has to make a decision of complexity vs. meaningful performance. As a final remark, note that the performance of Kalman modeling is better than the previously explored models, namely, MA, EWMA, and ARMA.

Table 4.3 Comparison of Kalman forecasting in terms of mean squared error.

	MSE
Kalman on log-returns (dynamic)	0.000 632
Kalman on log-prices (static)	0.000 560
Kalman on log-prices (dynamic)	0.000 557

It is worth mentioning that Kalman filtering can be employed in the context of ARMA modeling from Section 4.3.3 simply by rewriting the ARMA model in (4.6), $x_t = \phi_0 +$

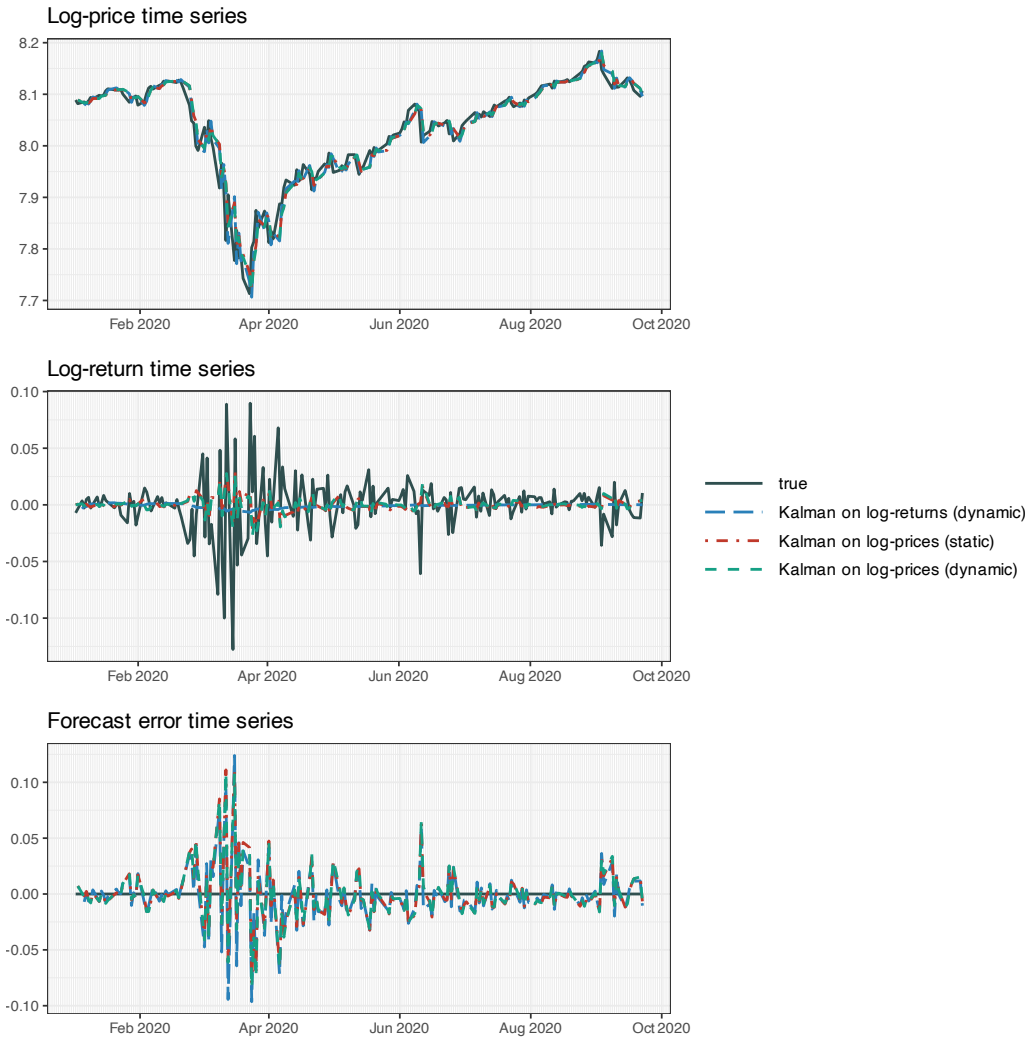


Figure 4.5 Forecasting with Kalman.

$\sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon_t - \sum_{j=1}^{q} \psi_j \epsilon_{t-j}$, in terms of the state-space model in (4.2). This can be done in a multitude of ways (Durbin & Koopman, 2012; Lütkepohl, 2007; Tsay, 2010; Zivot et al., 2004). For example, an AR(p) can be modeled by defining the hidden state as $\boldsymbol{\alpha}_t = \begin{bmatrix} x_t \\ \vdots \\ x_{t-p+1} \end{bmatrix}$,

leading to

$$x_t = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} \boldsymbol{\alpha}_t,$$
$$\boldsymbol{\alpha}_{t+1} = \begin{bmatrix} \phi_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} \phi_1 & \dots & \phi_{p-1} & \phi_p \\ 1 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 1 & 0 \end{bmatrix} \boldsymbol{\alpha}_t + \begin{bmatrix} \epsilon_t \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

4.3.6 Extension to the Multivariate Case

In practice, we typically deal with N assets, so the previous univariate mean modeling approaches need to be extended to the multivariate case, which is quite straightforward. In fact, the simple MA and EWMA models from Sections 4.3.1 and 4.3.2 can be directly applied to each of the assets separately. We will next briefly describe the extension for ARMA modeling and for the state-space representation of Kalman modeling.

VARMA

The ARMA model in Section 4.3.3 can be straightforwardly extended to the multivariate case by using matrix coefficients instead of scalar coefficients. Similarly to the ARMA(p,q) model in (4.6), a vector ARMA (VARMA) model of orders p and q, denoted by VARMA(p,q), becomes

$$\boldsymbol{x}_t = \boldsymbol{\phi}_0 + \sum_{i=1}^{p} \boldsymbol{\Phi}_i \boldsymbol{x}_{t-i} + \boldsymbol{\epsilon}_t - \sum_{j=1}^{q} \boldsymbol{\Psi}_j \boldsymbol{\epsilon}_{t-j}, \tag{4.10}$$

where now the parameters are $\boldsymbol{\phi}_0 \in \mathbb{R}^N$, $\boldsymbol{\Phi}_1, \dots, \boldsymbol{\Phi}_p \in \mathbb{R}^{N\times N}$, $\boldsymbol{\Psi}_1, \dots, \boldsymbol{\Psi}_q \in \mathbb{R}^{N\times N}$, and $\boldsymbol{\Sigma} \in \mathbb{R}^{N\times N}$ is the covariance matrix of $\boldsymbol{\epsilon}_t$.

While this extension looks trivial on paper, it quickly becomes impractical due to the exploding number of parameters. For the ARMA(p,q) model in (4.6) the number of parameters is simply $1+p+q+1$, whereas in the VARMA(p,q) case it increases to $N+(p+q)\times N^2+N(N-1)/2$, which grows as N^2. In other words, the number of parameters quickly explodes quadratically with the number of assets. The danger of fitting a model with such a large number of parameters is overfitting. This can be mitigated by either having a large number of observations in the historical data (which is rarely the case in financial applications) or by reducing the number of parameters, for example by forcing the matrix coefficients to be sparse. Note that if the matrix coefficients are forced to be diagonal matrices, then the model trivially reduces to asset-by-asset ARMA modeling.

VECM

Interestingly, in the multivariate case, a new model emerges based on the concept of *cointegration*, termed the *vector error correction model* (VECM) (Engle & Granger, 1987). The VECM model was proposed as a way to apply the ARMA model on the log-prices instead on the log-returns as in Section 4.3.3. The danger of modeling directly the log-prices is the lack of stationarity, which is why the safe approach is to differentiate first to obtain the log-returns prior to any modeling. However, the process of differentiating may potentially

destroy some structure in the time series. Applying a VAR(p) model on the log-prices $\boldsymbol{y}_t$ and using $\boldsymbol{x}_t = \boldsymbol{y}_t - \boldsymbol{y}_{t-1}$, the model can be finally written in terms of the log-returns as

$$\boldsymbol{x}_t = \boldsymbol{\phi}_0 + \boldsymbol{\Pi}\boldsymbol{y}_{t-1} + \sum_{i=1}^{p-1} \tilde{\boldsymbol{\Phi}}_i \boldsymbol{x}_{t-i} + \boldsymbol{\epsilon}_t,$$

where the matrix coefficients $\tilde{\boldsymbol{\Phi}}_i$ can be straightforwardly related to $\boldsymbol{\Phi}_i$ in (4.10) (Tsay, 2013).

Even though this model is written in terms of the log-returns $\boldsymbol{x}_t$, there is a new term that contains the log-prices: $\boldsymbol{\Pi}\boldsymbol{y}_{t-1}$. As it turns out, the matrix $\boldsymbol{\Pi}$ is of utmost importance: by decomposing it as $\boldsymbol{\Pi} = \boldsymbol{\alpha}\boldsymbol{\beta}^\mathsf{T}$, the matrix $\boldsymbol{\beta} \in \mathbb{R}^{N\times r}$, with r being the number of columns, reveals that the nonstationary log-prices $\boldsymbol{y}_t$ become stationary after multiplication with $\boldsymbol{\beta}^\mathsf{T}$, that is, they are cointegrated. This cointegration relationship is key in the design of mean-reverting time series in the context of *pairs trading* or *statistical arbitrage* considered in Chapter 15.

Multivariate Kalman Modeling

Finally, we can consider a state-space model for Kalman filtering, as in Section 4.3.5, properly extended to the vector case. For simplicity, we focus on the local trend model in (4.7). If we apply it on an asset-by-asset basis for $i = 1, \ldots, N$, we simply obtain

$$\begin{aligned} x_{i,t} &= \mu_{i,t} + \epsilon_{i,t}, \\ \mu_{i,t+1} &= \mu_{i,t} + \eta_{i,t}, \end{aligned}$$

where $\epsilon_{i,t} \sim \mathcal{N}(0, h_i)$ is the observation noise and $\eta_{i,t} \sim \mathcal{N}(0, q_i)$ is the drift noise. However, if we allow the noise terms for the different assets to be correlated, then we can write the more general vector model

$$\begin{aligned} \boldsymbol{x}_t &= \boldsymbol{\mu}_t + \boldsymbol{\epsilon}_t, \\ \boldsymbol{\mu}_{t+1} &= \boldsymbol{\mu}_t + \boldsymbol{\eta}_t, \end{aligned}$$

where now the observation noise vector is $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \boldsymbol{H})$ and the drift noise vector is $\boldsymbol{\eta}_{i,t} \sim \mathcal{N}(0, \boldsymbol{Q})$, both of which can model the asset correlation.

4.4 Volatility/Variance Modeling

Volatility clustering is a stylized fact of financial data that refers to the observation that large price changes tend to be followed by large price changes (ignoring the sign), whereas small price changes tend to be followed by small price changes (Fama, 1965; Mandelbrot, 1963). This phenomenon of volatility clustering, showcased in the exploratory data analysis in Section 2.4 of Chapter 2, clearly reveals temporal structure that can potentially be exploited through proper modeling.

Section 4.3 has focused on using the temporal structure for the purpose of mean modeling or modeling of the conditional expected return $\boldsymbol{\mu}_t$ in (4.1) under a constant conditional variance. We now explore a variety of models to exploit the temporal structure in the modeling of the conditional variance $\boldsymbol{\Sigma}_t$.

Unless otherwise stated, we will focus on the univariate case (single asset) for simplicity. Therefore, the objective is to compute the expectation of the future variance of the time series

at time t based on the past observations $\mathcal{F}_{t-1}$, that is, $\mathrm{Var}[\epsilon_t \mid \mathcal{F}_{t-1}] = \mathbb{E}[\epsilon_t^2]$ in (4.1), where $\epsilon_t = x_t - \mu_t$ is the forecasting error or innovation term. In practice, since the magnitude of μ_t is much smaller than that of the observed noisy data x_t, one can also focus, for simplicity and with a negligible effect in accuracy, on $\mathbb{E}[x_t^2 \mid \mathcal{F}_{t-1}]$. Note that the volatility is not a directly observable quantity (although meaningful proxies can be used) and defining an error measure requires some careful thought (Poon & Granger, 2003); alternatively, a visual inspection of the result is also of practical relevance.

The topic of volatility modeling or variance modeling is standard material in many textbooks (Lütkepohl, 2007; Ruppert & Matteson, 2015; Tsay, 2010), as well as some specific overview papers (Bollerslev et al., 1992; Poon & Granger, 2003; Taylor, 1994).

4.4.1 Moving Average (MA)

Under the i.i.d. model in (3.1), the residual is distributed as $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ and the simplest way to estimate its variance $\sigma^2 = \mathbb{E}[\epsilon_t^2]$ is by taking the average of the squared values. In practice, the variance and volatility are slowly time-varying, denoted by σ_t^2 and σ_t, respectively. The *volatility envelope* precisely refers to the time-varying volatility σ_t.

To allow for a slowly time-varying variance, we can use a moving average or rolling means (like in Section 4.3.1) on the squared residuals:

$$\hat{\sigma}_t^2 = \frac{1}{q} \sum_{i=1}^{q} \epsilon_{t-i}^2. \tag{4.11}$$

Figure 4.6 illustrates the volatility envelope computed via MAs of different lengths.

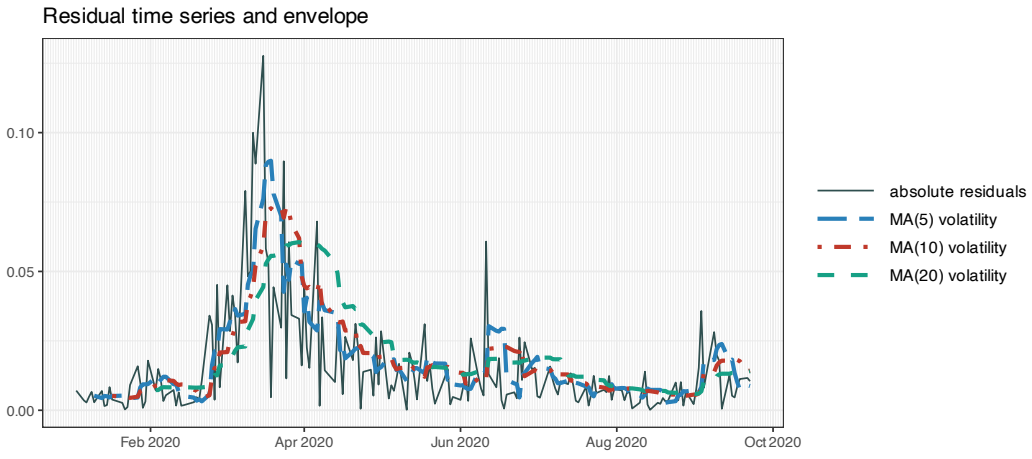


Figure 4.6 Volatility envelope with moving averages.

4.4.2 EWMA

Similarly to Section 4.3.2, we can put more weight on the recent observations. One option is to use exponential weights, which can be efficiently computed recursively as

$$\hat{\sigma}_t^2 = \alpha \epsilon_{t-1}^2 + (1-\alpha)\hat{\sigma}_{t-1}^2, \tag{4.12}$$

where α ($0 \leq \alpha \leq 1$) determines the exponential decay or memory.

Figure 4.7 shows the volatility envelope computed via EWMAs of different memory. Whenever there is a large residual spike, the subsequent exponential decay of the volatility can be observed.

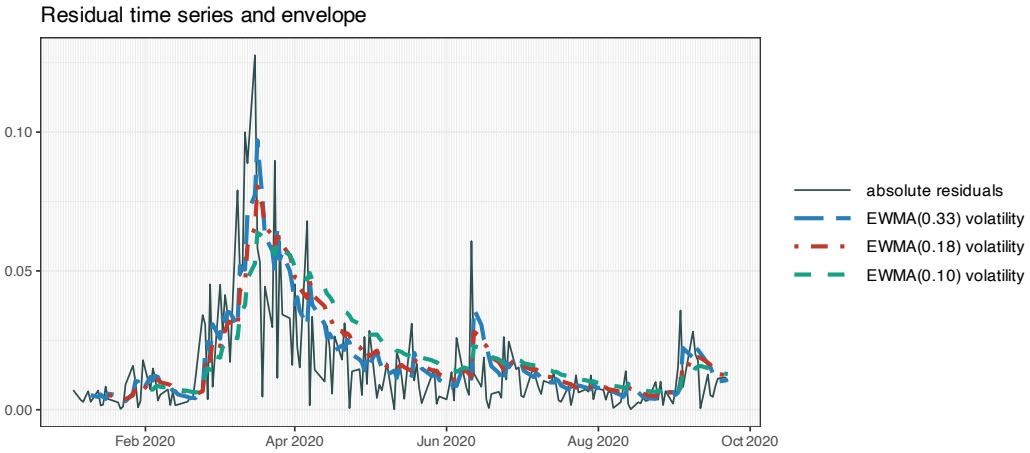


Figure 4.7 Volatility envelope with EWMA.

4.4.3 GARCH Modeling

Heteroskedasticity is the technical term that refers to the phenomenon of a time-varying variance. The *autoregressive conditional heteroskedasticity* (ARCH) model is one of the earliest models to deal with heteroskedasticity and, in particular, the volatility clustering effect.

The (linear) ARCH model of order q, denoted by ARCH(q), was proposed by Engle (1982)[5] and it is defined as

$$\begin{aligned} \epsilon_t &= \sigma_t z_t, \\ \sigma_t^2 &= \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2, \end{aligned} \tag{4.13}$$

where ϵ_t is the innovation to be modeled, z_t is an i.i.d. random variable with zero mean and unit variance, σ_t is the slowly time-varying volatility envelope (expressed as a moving average of the past squared residuals), and the parameters of the model are $\omega > 0$ and $\alpha_1, \ldots, \alpha_q \geq 0$.

[5] Robert F. Engle was awarded the 2003 Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel "for methods of analyzing economic time series with time-varying volatility (ARCH)."

An important limitation of the ARCH model is that the high volatility is not persistent enough (unless q is chosen very large) to capture the phenomenon of volatility clustering. This can be overcome by the *generalized ARCH* (GARCH) model proposed by Bollerslev (1986).

The (linear) GARCH model of orders p and q, denoted by GARCH(p, q), is

$$\begin{aligned} \epsilon_t &= \sigma_t z_t, \\ \sigma_t^2 &= \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2, \end{aligned} \tag{4.14}$$

where now the parameters are $\omega > 0$, $\alpha_1, \ldots, \alpha_q \geq 0$, and $\beta_1, \ldots, \beta_p \geq 0$ (one technical condition to guarantee stationarity is $\sum_{i=1}^{q} \alpha_i + \sum_{j=1}^{p} \beta_j < 1$). This is formally an ARMA model on ϵ_t^2. The recursive component in σ_t allows the volatility to be more persistent in time.

Since the ARCH and GARCH models were proposed in the 1980s, a wide range of extensions and variations, including nonlinear recursions and non-Gaussian distributed innovations, have appeared in the econometric literature, cf. Bollerslev et al. (1992), Lütkepohl (2007), Tsay (2010), and Ruppert and Matteson (2015). In the case of intraday financial data, the model is extended to include the intraday pattern, the so-called "volatility smile" (Engle & Sokalska, 2012; Xiu & Palomar, 2023).

GARCH models require the appropriate coefficients to properly fit the data and this is typically done in practice via maximum likelihood procedures. Mature and efficient software implementations are readily available in most programming languages.[6]

Figure 4.8 shows the volatility envelope computed via different ARCH and GARCH models. As in the EWMA case, whenever a large spike appears, a subsequent exponential decay of the volatility can be observed.

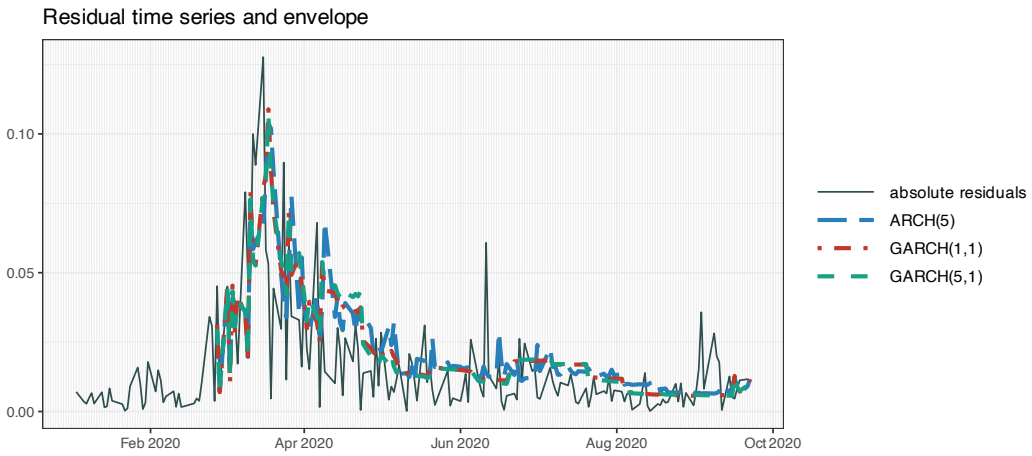


Figure 4.8 Volatility envelope with GARCH models.

[6] The R packages `rugarch` and `fGarch` implement algorithms for fitting a wide range of different GARCH models to time series data (Ghalanos, 2022; Wuertz et al., 2022).

Criticism of GARCH Models

GARCH models are extremely popular and well-researched in econometrics. Nevertheless, one has to admit that they are essentially "glorified" exponentially weighted moving averages. Indeed, taking the GARCH variance modeling in (4.14) and setting, for simplicity, $p = q = 1$, $\omega = 0$, and $\beta_1 = 1 - \alpha_1$, we obtain

$$\sigma_t^2 = \alpha_1 \epsilon_{t-1}^2 + (1 - \alpha_1)\sigma_{t-1}^2,$$

which looks exactly like the EWMA in (4.12). In words, GARCH models attempt to represent the volatility as a series of (unpredictable) spikes that decay exponentially over time. As a consequence, the volatility curve looks rugged and composed of overlapping exponential curves. Arguably this is not how a slowly time-varying envelope is expected to look, but nevertheless these models have enjoyed an unparalleled popularity.

In addition, the estimation of the GARCH parameters (model fitting) is extremely "data hungry." That is, unless the number of observations is really large, estimation of the parameters becomes unreliable as illustrated with the following numerical example.[7] Figure 4.9 shows a scatter plot with 1000 Monte Carlo random simulations of the parameter estimation values of a GARCH(1,1) model with $\omega = 0$, $\alpha_1 = 0.07$, and $\beta_1 = 0.925$ based on $T = 1000$ observations. The variation in the estimated parameters is around 0.1, which is quite large even though 1000 observations were used (4 years of daily data).

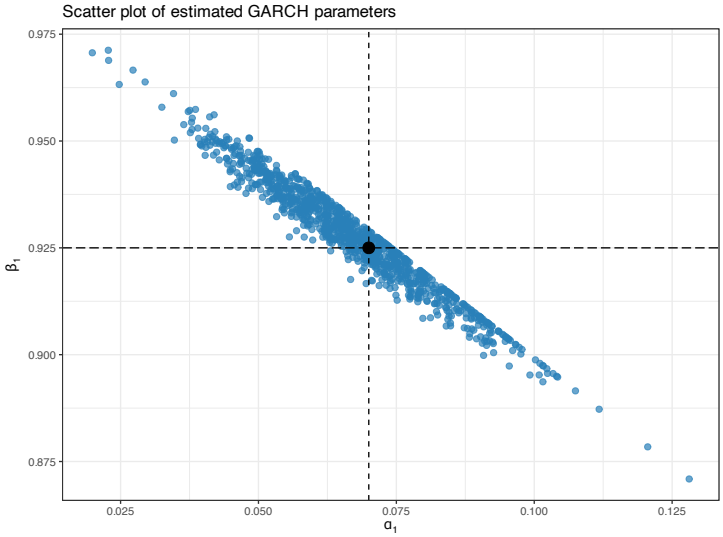


Figure 4.9 Instability in GARCH model fitting.

4.4.4 Stochastic Volatility Modeling

In 1982, Taylor proposed in a seminal work to model the volatility probabilistically via a state-space model termed the *stochastic volatility* (SV) model (Taylor, 1982). Even though

[7] Patrick Burns delivered an insightful presentation at the Imperial College Algorithmic Trading Conference on 8 December, 2012, exposing the limitations of GARCH models in favor of stochastic volatility models.

SV and GARCH were proposed in the same year, SV modeling has not enjoyed the same popularity as GARCH modeling in econometrics. One possible reason is that the fitting process is theoretically more involved and computationally demanding; in fact, a maximum likelihood optimization cannot be exactly formulated for SV (Kim et al., 1998; Poon & Granger, 2003). Nevertheless, as explored in later in Section 4.4.5, Kalman filtering can be efficiently used as a good practical approximation (Harvey et al., 1994; Ruiz, 1994).

The SV model is conveniently written in terms of the log-variance, $h_t = \log(\sigma_t^2)$, as

$$\begin{aligned} \epsilon_t &= \exp(h_t/2)z_t, \\ h_t &= \gamma + \phi h_{t-1} + \eta_t, \end{aligned} \tag{4.15}$$

where the first equation is equivalent to that in the ARCH and GARCH models, $\epsilon_t = \sigma_t z_t$, and the second equation models the volatility dynamics with parameters γ and ϕ in a stochastic way via the residual term η_t.

To gain more insight, we can rewrite the volatility state transition equation from (4.15) in terms of σ_t^2 as

$$\log(\sigma_t^2) = \gamma + \phi \log(\sigma_{t-1}^2) + \eta_t,$$

which allows a clearer comparison with a GARCH(1,1) model:

$$\sigma_t^2 = \omega + \beta_1 \sigma_{t-1}^2 + \alpha_1 \epsilon_{t-1}^2.$$

As we can observe, the main difference (apart from modeling the log variance instead of the variance, which has also been done in the exponential GARCH model) appears in the noise term η_t in the volatility state transition equation. This difference may seem tiny and insignificant, but it actually makes it fundamentally different to GARCH models where the time-varying volatility is assumed to follow a deterministic instead of stochastic evolution.

SV models, albeit not having received the same attention as GARCH models, are still covered in some excellent overview papers (Harvey et al., 1994; Kim et al., 1998; Poon & Granger, 2003; Ruiz, 1994; Taylor, 1994) and standard textbooks (Tsay, 2010).

SV modeling requires the appropriate coefficients to properly fit the data. Unlike GARCH models, this coefficient estimation process is mathematically more involved and computationally more demanding. Typically, computationally intensive Markov chain Monte Carlo (MCMC) algorithms are employed (Kim et al., 1998).[8] In the next section, we consider an efficient quasi-likelihood method via Kalman filtering (Harvey et al., 1994; Ruiz, 1994).

Figure 4.10 displays the volatility envelope calculated using the SV model. As can be observed, the envelope appears smooth, contrasting with the rugged overlap of decaying exponential curves observed in GARCH modeling.

4.4.5 Kalman Modeling

The SV model can be written with a state-space representation and approximately solved via the Kalman filtering as described next (Harvey et al., 1994; Ruiz, 1994; Zivot et al., 2004).

[8] The R package `stochvol` implements an MCMC algorithm for fitting SV models (Kastner, 2016). The Python package `PyMC` contains a MCMC methods that can be used for SV modeling.

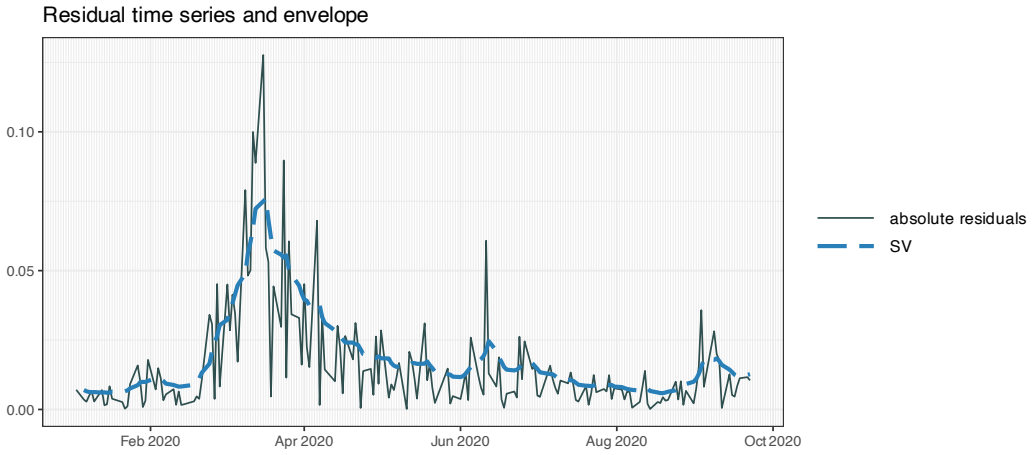


Figure 4.10 Volatility envelope with SV modeling via MCMC.

Taking the logarithm in the SV squared observation equation in (4.15), $\epsilon_t^2 = \exp(h_t)z_t^2$, gives

$$\log(\epsilon_t^2) = h_t + \log(z_t^2),$$

where $\log(z_t^2)$ is a non-Gaussian i.i.d. process. If a Gaussian distribution $z_t \sim \mathcal{N}(0,1)$ is assumed, then the mean and variance of $\log(z_t^2)$ are $\psi(1/2) - \log(1/2) \approx -1.27$ and $\pi^2/2$, respectively, where $\psi(\cdot)$ denotes the digamma function. Similarly, if a heavy-tailed t distribution is assumed with ν degrees of freedom, then the mean is approximately $-1.27 - \psi(\nu/2) + \log(\nu/2)$ and the variance $\pi^2/2 + \psi'(\nu/2)$, where $\psi'(\cdot)$ is the trigamma function.

Thus, under the Gaussian assumption for z_t, the SV model in (4.15) can be approximated as

$$\begin{aligned} \log(\epsilon_t^2) &= -1.27 + h_t + \xi_t, \\ h_t &= \gamma + \phi h_{t-1} + \eta_t, \end{aligned} \tag{4.16}$$

where ξ_t is a non-Gaussian i.i.d. process with zero mean and variance $\pi^2/2$, and the parameters of the model are γ, ϕ, and the variance σ_η^2 of η_t. This model fits the state-space representation in (4.2); however, since ξ_t is not Gaussian then Kalman filtering will not produce optimal results.

One particular case of (4.16) is the random walk plus noise model with $\gamma = 0$ and $\phi = 1$:

$$\begin{aligned} \log(\epsilon_t^2) &= -1.27 + h_t + \xi_t, \\ h_t &= h_{t-1} + \eta_t, \end{aligned} \tag{4.17}$$

with the single remaining parameter σ_η^2.

It is important to realize that the different choices of model will produce different volatility envelopes. Since the volatility is unobservable, it is not clear how to choose the best model. In some cases, however, it is possible to observe the volatility, called "realized volatility," and then the model can be fitted in a better way. One example is when higher-frequency data

is available and can be used to compute the realized volatility during the slower-frequency period, for example, hourly data can be used to estimate the daily volatility.

Figure 4.11 shows the volatility envelope forecast according to the SV model via Kalman filtering. If a noncausal envelope is allowed, then we can instead use Kalman smoothing as in Figure 4.12, which clearly produces much smoother and more accurate envelopes at the expense of using noncausal data.

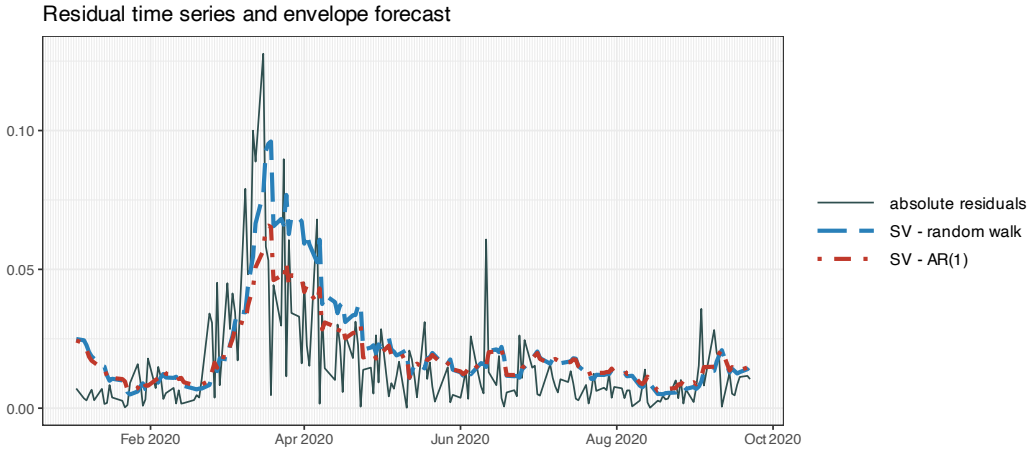


Figure 4.11 Volatility envelope with SV modeling via Kalman filter.

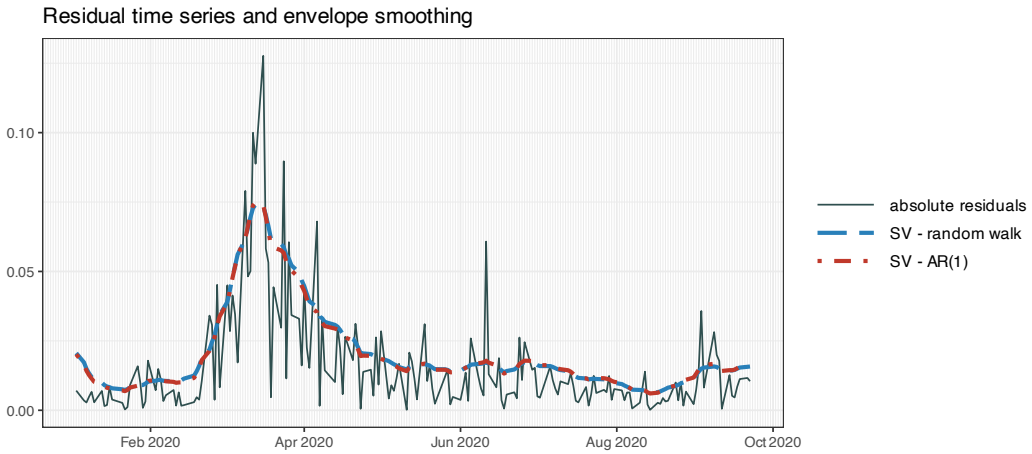


Figure 4.12 Volatility envelope with SV modeling via Kalman smoother.

4.4.6 Extension to the Multivariate Case

All the previous (univariate) volatility models can be applied to N assets on an asset-by-asset basis. Nevertheless, it may be advantageous to use a multivariate model that can better model the common volatility clustering observed in market data.

Multivariate EWMA

To start with, following the univariate EMWA volatility modeling from Section 4.4.2, we can easily extend the univariate estimation in (4.12) to the multivariate case as

$$\hat{\boldsymbol{\Sigma}}_t = \alpha \boldsymbol{\epsilon}_{t-1} \boldsymbol{\epsilon}_{t-1}^{\top} + (1-\alpha)\hat{\boldsymbol{\Sigma}}_{t-1},$$

where $\boldsymbol{\epsilon}_t = \boldsymbol{x}_t - \boldsymbol{\mu}_t \in \mathbb{R}^N$ is the forecasting error vector and α is the smoothing parameter that determines the exponential decay or memory. This model can easily be extended so that each asset has a different smoothing parameter (Tsay, 2013).

Multivariate GARCH

Numerous attempts have been made to extend the univariate GARCH models (see Section 4.4.3) to the multivariate case (Bollerslev et al., 1992; Lütkepohl, 2007; Tsay, 2013). Similarly to the univariate cases in (4.13) and (4.14), the forecasting error vector can be conveniently decomposed as

$$\boldsymbol{\epsilon}_t = \boldsymbol{\Sigma}_t^{1/2} \boldsymbol{z}_t,$$

where $\boldsymbol{z}_t \in \mathbb{R}^N$ is a zero-mean random vector with identity covariance matrix and the volatility is modeled by the matrix $\boldsymbol{\Sigma}_t^{1/2} \in \mathbb{R}^{N \times N}$, which is the square-root matrix of $\boldsymbol{\Sigma}_t$ (satisfying $\boldsymbol{\Sigma}_t^{\top/2} \boldsymbol{\Sigma}_t^{1/2} = \boldsymbol{\Sigma}_t$). In other words, $\boldsymbol{\Sigma}_t$ is the covariance matrix that generalizes the variance σ_t^2 in the univariate case and $\boldsymbol{\Sigma}_t^{1/2}$ is the matrix generalization of the volatility σ_t.

The complication arises when modeling the dynamics of the volatility matrix $\boldsymbol{\Sigma}_t^{1/2}$, particularly due to the significant increase in the number of parameters when transitioning from univariate to multivariate analysis. As always, the problem with a large number of parameters is that the model will inevitably suffer from overfitting in a practical setting. A wide range of models have been proposed in the literature trying to cope with this issue; see the overview in Bollerslev et al. (1992). A naive extension is the multivariate GARCH model, which simply follows from the univariate GARCH model by vectorizing all the matrix terms (each of dimension N^2) resulting in model coefficients in the form of huge matrices of dimension $N^2 \times N^2$. Subsequent proposals attempted to reduce the number of parameters by incorporating some structure in the matrix coefficients, such as enforcing diagonal matrices. However, the number of parameters remains on the order of N^2, which is still too large to prevent overfitting.

The *constant conditional correlation* (CCC) model addresses the dimensionality issue by modeling the heteroskedasticity in each asset with univariate models (asset by asset) combined with a constant correlation matrix for all the assets (Bollerslev, 1990). Mathematically, the model is

$$\boldsymbol{\Sigma}_t = \boldsymbol{D}_t \boldsymbol{C} \boldsymbol{D}_t,$$

where $\boldsymbol{D}_t = \mathrm{Diag}(\sigma_{1,t}, \ldots, \sigma_{N,t})$ represents the time-varying conditional volatilities of each of the assets and the matrix $\boldsymbol{C}$ is the constant correlation matrix. This model is very convenient in practice because it avoids the explosion of the number of parameters. Basically, the volatility envelope of each asset is first modeled individually and removed from the data as $\bar{\boldsymbol{\epsilon}}_t = \boldsymbol{D}_t^{-1} \boldsymbol{\epsilon}_t$, and then the correlation structure of the multivariate data $\bar{\boldsymbol{\epsilon}}_t$ (with approximately constant volatility) is obtained. A disadvantage of this model is the fact that the correlation structure is fixed.

The *dynamic conditional correlation* (DCC) model precisely addresses the drawback of the CCC model and allows the correlation matrix to change over time, but using a single scalar parameter to avoid overfitting (Engle, 2002). To be precise, the time-varying correlation matrix $\boldsymbol{C}_t$ is obtained via an exponentially weighted moving average of the data with removed volatility $\bar{\boldsymbol{\epsilon}}_t$,

$$\boldsymbol{Q}_t = \alpha \bar{\boldsymbol{\epsilon}}_{t-1} \bar{\boldsymbol{\epsilon}}_{t-1}^{\top} + (1-\alpha)\boldsymbol{Q}_{t-1},$$

with an additional normalization step in case a correlation matrix is desired (with diagonal elements equal to one):

$$\boldsymbol{C}_t = \text{Diag}(\boldsymbol{Q}_t)^{-1/2} \boldsymbol{Q}_t \text{Diag}(\boldsymbol{Q}_t)^{-1/2}.$$

One disadvantage of this model is that it forces all the correlation coefficients to have the same memory via the same α, which could be further relaxed at the expense of more parameters.

Thus, the recommended procedure for building DCC models is (Tsay, 2013):

1. Use any of the mean modeling techniques from Section 4.3 to obtain a forecast $\boldsymbol{\mu}_t$ and then compute the residual or error vector of the forecast $\boldsymbol{\epsilon}_t = \boldsymbol{x}_t - \boldsymbol{\mu}_t$.
2. Apply any of the univariate volatility models from Section 4.4 to obtain the volatility envelopes for the N assets $(\sigma_{1,t}, \ldots, \sigma_{N,t})$.
3. Standardize each of the series with the volatility envelope, $\bar{\boldsymbol{\epsilon}}_t = \boldsymbol{D}_t^{-1} \boldsymbol{\epsilon}_t$, so that a series with approximately constant envelope is obtained.
4. Compute either a fixed covariance matrix of the multivariate series $\bar{\boldsymbol{\epsilon}}_t$ or an exponentially weighted moving average version.

It is worth mentioning that copulas are another popular approach for multivariate modeling that can be combined with DCC models (Ruppert & Matteson, 2015; Tsay, 2013).

Multivariate SV

The multivariate extension of the SV observation equation $\epsilon_t = \exp(h_t/2) z_t$ in (4.15) is

$$\boldsymbol{\epsilon}_t = \text{Diag}\left(\exp(\boldsymbol{h}_t/2)\right) \boldsymbol{z}_t,$$

where now $\boldsymbol{h}_t = \log(\boldsymbol{\sigma}_t^2)$ denotes the log-variance vector and $\boldsymbol{z}_t$ is a random vector with zero mean and fixed covariance matrix $\boldsymbol{\Sigma}_z$. The covariance matrix modeled in this way can be expressed as $\boldsymbol{\Sigma}_t = \text{Diag}\left(\exp(\boldsymbol{h}_t/2)\right) \boldsymbol{\Sigma}_z \text{Diag}\left(\exp(\boldsymbol{h}_t/2)\right)$, which has the same form as the CCC model, that is, the covariance matrix can be decomposed into the time-varying volatilities and the fixed matrix that models the fixed correlations.

Similarly to (4.16), taking the logarithm of the observation equation leads to the following approximated state-space model (Harvey et al., 1994):

$$\begin{aligned} \log\left(\boldsymbol{\epsilon}_t^2\right) &= -1.27 \times \mathbf{1} + \boldsymbol{h}_t + \boldsymbol{\xi}_t, \\ \boldsymbol{h}_t &= \boldsymbol{\gamma} + \text{Diag}\left(\boldsymbol{\phi}\right) \boldsymbol{h}_{t-1} + \boldsymbol{\eta}_t, \end{aligned} \tag{4.18}$$

where now $\boldsymbol{\xi}_t$ is a non-Gaussian i.i.d. vector process with zero mean and a covariance matrix $\boldsymbol{\Sigma}_\xi$ that can be obtained from $\boldsymbol{\Sigma}_z$ (Harvey et al., 1994). In particular, if $\boldsymbol{\Sigma}_z = \boldsymbol{I}$ then $\boldsymbol{\Sigma}_\xi = \pi^2/2 \times \boldsymbol{I}$, which then reduces to an asset-by-asset model.

Similarly, the multivariate version of the random walk plus noise model in (4.17) is

$$\begin{aligned} \log\left(\boldsymbol{\epsilon}_t^2\right) &= -1.27 \times \mathbf{1} + \boldsymbol{h}_t + \boldsymbol{\xi}_t \\ \boldsymbol{h}_t &= \boldsymbol{h}_{t-1} + \boldsymbol{\eta}_t. \end{aligned} \tag{4.19}$$

Other extensions of SV, including common factors and heavy-tailed distributions, have also been considered (Harvey et al., 1994).

4.5 Summary

Hundreds of models have been proposed over the past decades for financial time series attempting to incorporate temporal structure, both for mean modeling, $\boldsymbol{\mu}_t$, and variance modeling, $\boldsymbol{\Sigma}_t$, with the following takeaways:

- *Mean models* range from simple moving averages to more sophisticated ARMA models (or even VECM). However, it is debatable whether they can outperform the simple i.i.d. model, particularly considering the small autocorrelation exhibited by typical financial time series. Nonetheless, the conclusion may greatly depend on the nature and frequency of the financial data.
- *Variance (or volatility) models* are undoubtedly practical, as financial data clearly displays a significant degree of temporal structure in variance (or volatility). Two main approaches exist: GARCH modeling, which is by far the most popular direction in econometrics, and stochastic volatility modeling, which arguably produces a more desirable volatility envelope. Interestingly, stochastic volatility has not gained the same popularity as GARCH models, perhaps due to its higher computational complexity (although this can be remedied via Kalman filtering).
- *State-space modeling* provides a general and convenient framework for financial time series. In fact, it embraces most of the common models for the mean and it approximates reasonably well the variance models, such as stochastic volatility modeling.
- The *Kalman filter* is an efficient algorithm for fitting financial time series that can be represented as a state-space model. Moreover, it enables time-varying modeling, which is essential for financial data. However, its usage does not seem to be as widespread as it deserves within the financial community, despite being covered in standard time series textbooks.

Exercises

Choose one or several assets (e.g., stocks or cryptocurrencies) for the following exercises.

Mean Modeling

4.1 (Autocorrelation function of returns) Choose one asset and plot the autocorrelation function of the log-returns at different frequencies.

4.2 (MA modeling) Choose one asset and try the MA(q) model on the log-returns and log-prices for different values of the lookback q. Compute the mean squared error of the forecast.

4.3 (EWMA modeling) Choose one asset and try the EWMA model on the log-returns and log-prices for different values of the memory α. Compute the mean squared error of the forecast.

4.4 (ARMA modeling) Choose one asset and experiment with ARMA(p, q) models with different values of p and q. Compute the mean squared error of the forecast.

4.5 (Kalman for mean modeling) Choose one asset and experiment with different state-space models together with Kalman filtering. Compute the mean squared error of the forecast.

4.6 (Kalman for ARMA modeling) Choose one asset and compare the results of a direct ARMA model with the corresponding state-space model via Kalman filtering.

4.7 (VARMA modeling) Choose several assets and compare the results of asset-by-asset ARMA modeling and VARMA modeling. Discuss the results.

4.8 (Kalman for multivariate mean modeling) Choose several assets and compare the results of asset-by-asset Kalman modeling and vector Kalman modeling. Discuss the results.

Volatility Envelope Modeling

4.9 (Autocorrelation function of absolute returns) Choose one asset and plot the autocorrelation function of the absolute value of the log-returns at different frequencies.

4.10 (MA volatility modeling) Choose one asset and try the MA(q) model on the squared log-returns for different values of the lookback q. Plot the volatility envelope.

4.11 (EWMA volatility modeling) Choose one asset and try the EWMA model on the squared log-returns for different values of the memory α. Plot the volatility envelope.

4.12 (ARCH volatility modeling) Choose one asset and experiment with ARCH(q) models with different values of q. Plot the volatility envelope.

4.13 (GARCH volatility modeling) Choose one asset and experiment with GARCH(p, q) models with different values of p and q. Plot the volatility envelope.

4.14 (SV modeling) Choose one asset and experiment with the SV model. Plot the volatility envelope and compare with the GARCH modeling.

4.15 (Kalman SV modeling) Choose one asset and experiment with the SV model via Kalman filtering. Try the AR(1) model and the random walk model. In addition, compare the models under the Gaussian distribution and the heavy-tailed t distribution.

4.16 (Multivariate GARCH modeling) Choose several assets and compare the results of asset-by-asset GARCH modeling and multivariate GARCH modeling via the constant conditional correlation model. Discuss the results.

4.17 (Kalman for multivariate SV modeling) Choose several assets and compare the results of asset-by-asset Kalman SV modeling and vector Kalman SV modeling (including correlation among assets). Discuss the results.

References

Anderson, B. D. O., & Moore, J. B. (1979). *Optimal Filtering*. Prentice Hall.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*(3), 307–327.

Bollerslev, T. (1990). Modelling the coherence in short-run nominal exchange rates: A multivariate generalized ARCH model. *The Review of Economics and Statistics*, *3*(72), 498–505.

Bollerslev, T., Chou, R. Y., & Kroner, K. F. (1992). ARCH modeling in finance: A review of the theory and empirical evidence. *Journal of Econometrics*, *52*, 5–59.

Brockwell, P. J., & Davis, R. A. (2002). *Introduction to Time Series and Forecasting*. Springer.

Chen, R., Feng, Y., & Palomar, D. P. (2016). Forecasting intraday trading volume: A Kalman filter approach. *SSRN Electronic Journal*. https://dx.doi.org/10.2139/ssrn.3101695

Durbin, J., & Koopman, S. J. (2012). *Time Series Analysis by State Space Methods* (2nd ed.). Oxford University Press.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica: Journal of the Econometric Society*, *50*(4), 987–1007.

Engle, R. F. (2002). Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models. *Journal of Business & Economic Statistics*, *20*(3), 339–350.

Engle, R. F., & Granger, C. W. J. (1987). Co-integration and error correction: Representation, estimation, and testing. *Econometrica: Journal of the Econometric Society*, *55*(2), 251–276.

Engle, R. F., & Sokalska, M. E. (2012). Forecasting intraday volatility in the US equity market. Multiplicative component GARCH. *Journal of Financial Econometrics*, *10*(1), 54–83.

Fama, E. F. (1965). The behavior of stock-market prices. *The Journal of Business*, *38*(1), 34–105.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, *25*(2), 383–417.

Ghalanos, A. (2022). *rugarch: Univariate GARCH Models* [R package]. https://CRAN.R-project.org/package=rugarch

Harvey, A. (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

Harvey, A., & Koopman, S. J. (2009). Unobserved components models in economics and finance: The role of the Kalman filter in time series econometrics. *IEEE Control Systems Magazine*, *29*(6), 71–81.

Harvey, A., Ruiz, E., & Shephard, N. (1994). Multivariate stochastic variance models. *The Review of Economic Studies*, *61*(2), 247–264.

Helske, J. (2017). KFAS: Exponential family state space models in R. *Journal of Statistical Software*, *78*(10), 1–39.

Holmes, E. E., Ward, E. J., & Wills, K. (2012). MARSS: Multivariate autoregressive state-space models for analyzing time-series data. *The R Journal*, *4*(1), 11–19.

Hyndman, R. J., Koehler, A. B., Ord, J. K., & Snyder, R. D. (2008). *Forecasting With Exponential Smoothing*. Springer.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *82*, 35–45.

Kastner, G. (2016). Dealing with stochastic volatility in time series using the R package stochvol. *Journal of Statistical Software*, *69*(5), 1–30.

Kim, S., Shephard, N., & Chib, S. (1998). Stochastic volatility: Likelihood inference and comparison with ARCH models. *Review of Economic Studies*, *65*, 361–393.

Lo, A. W., & Mackinlay, A. C. (2002). *A Non-Random Walk Down Wall Street*. Princeton University Press.

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer.

Malkiel, B. G. (1973). *A Random Walk Down Wall Street*. W. W. Norton.

Mandelbrot, B. B. (1963). The variation of certain speculative prices. *The Journal of Business*, *36*(4), 394–419.

Petris, G., & Petrone, S. (2011). State space models in R. *Journal of Statistical Software*, *41*(4), 1–25.

Poon, S.-H., & Granger, C. W. J. (2003). Forecasting volatility in financial markets: A review. *Journal of Economic Literature*, *XLI*, 478–539.

Ruiz, E. (1994). Quasi-maximum likelihood estimation of stochastic volatility models. *Journal of Econometrics*, *63*(1), 289–306.

Ruppert, D., & Matteson, S. D. (2015). *Statistics and Data Analysis for Financial Engineering: With R Examples* (2nd ed.). Springer.

Shiller, R. J. (1981). Do stock prices move too much to be justified by subsequent changes in dividends? *American Economic Review*, *71*(3), 421–436.

Shiller, R. J. (2003). From efficient markets theory to behavioral finance. *Journal of Economic Perspectives*, *17*(1), 83–104.

Shumway, R. H., & Stoffer, D. S. (2017). *Time Series Analysis and Its Applications* (4th ed.). Springer.

Taylor, S. J. (1982). Financial returns modelled by the product of two stochastic processes: A study of daily sugar prices, 1691–79. In O. D. Anderson (Ed.), *Time Series Analysis: Theory and Practice 1* (pp. 203–226). North-Holland.

Taylor, S. J. (1994). Modeling stochastic volatility: A review and comparative study. *Mathematical Finance*, *4*(2), 183–204.

Tsay, R. S. (2010). *Analysis of Financial Time Series* (3rd ed.). John Wiley & Sons.

Tsay, R. S. (2013). *Multivariate Time Series Analysis: With R and Financial Applications*. John Wiley & Sons.

Tusell, F. (2011). Kalman filtering in R. *Journal of Statistical Software*, *39*(2), 1–27.

Wuertz, D., Chalabi, Y., Setz, T., Maechler, M., Boudt, C., Chausse, P., Miklovac, M., & Boshnakov, G. N. (2022). *fGarch: Rmetrics – Autoregressive Conditional Heteroskedastic Modelling* [R package]. https://CRAN.R-project.org/package=fGarch

Xiu, S., & Palomar, D. P. (2023). Intraday volatility–volume joint modeling and forecasting: A state-space approach. *Proceedings of the European Signal Processing Conference (EUSIPCO)*.

Zivot, E., Wang, J., & Koopman, S. J. (2004). State space modeling in macroeconomics and finance using SsfPack for S+FinMetrics. In A. Harvey, S. J. Koopman, & N. Shephard (Eds.), *State Space and Unobserved Component Models: Theory and Applications* (pp. 284–335). Cambridge University Press.

5

Financial Data: Graphs

"Mankind invented a system to cope with the fact that we are so intrinsically lousy at manipulating numbers. It's called the graph."

— Charlie Munger

Graphs provide a convenient and compact way to represent data while highlighting the relationships between entities of a network. They constitute a powerful mathematical tool with broad applicability in numerous fields, such as biology, brain modeling, finance, statistical physics, management, behavioral modeling, machine learning, social networks, and data science in general. Given the recent availability of large amounts of data collected in a variety of application domains, graph-based analysis plays a fundamental role in understanding and analyzing the structure of large networks that generate data. In practical scenarios, the underlying graph structure that represents the network is often unknown and has to be inferred from the data. Many graph learning algorithms have been proposed in recent decades, with increased interest in the past few years. This chapter explores a broad range of graph estimation algorithms, emphasizing recent advances specifically tailored to financial data.

5.1 Graphs

Graphs have become a fundamental and ubiquitous mathematical tool for modeling data in a variety of application domains and for understanding the structure of large networks that generate data (Kolaczyk, 2009; Lauritzen, 1996). By abstracting the data as graphs, we can better capture the geometry of data and visualize high-dimensional data. Some iconic examples of graphs, illustrated in Figure 5.1, include the following:

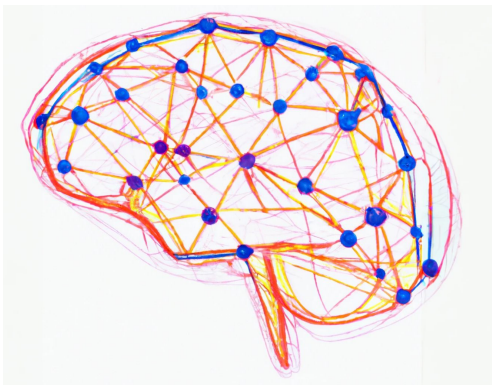
- *Social media graphs*: These model the behavioral similarity or influence between individuals, with the data consisting of online activities such as tagging, liking, purchasing, and so on.
- *Brain activity graphs*: These represent the correlation among sensors examining the brain, with the data being the measured brain activity known as fMRI (functional magnetic resonance imaging).

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

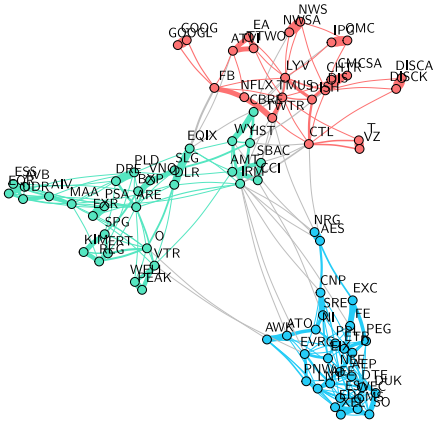
- *Financial stock graphs*: These capture the interdependencies among financial companies in stock markets, with the data consisting of measured economic quantities such as stock prices, volumes, and so on.
- *Financial currency graphs*: These summarize the interdependencies among currencies in foreign exchange markets, with the data comprising measured economic quantities like spot prices, volumes, and so on.
- *Financial cryptocurrency graphs*: Similarly to currency graphs, these model the interdependencies among cryptocurrencies in crypto markets.



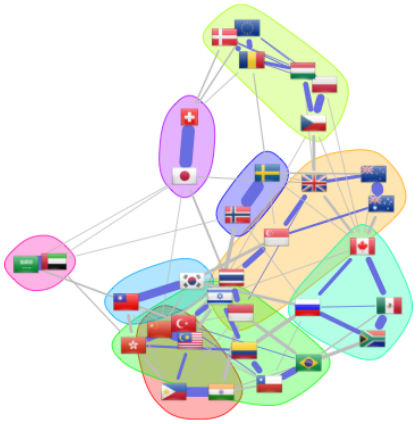
Social media graph



Brain activity graph



Financial stock graph



Financial currency graph

Figure 5.1 Examples of graphs in different applications.

5.1.1 Terminology

The basic elements of a graph (see Figure 5.2) are

- *nodes*: corresponding to the entities or variables; and
- *edges*: encoding the relationships between entities or variables.

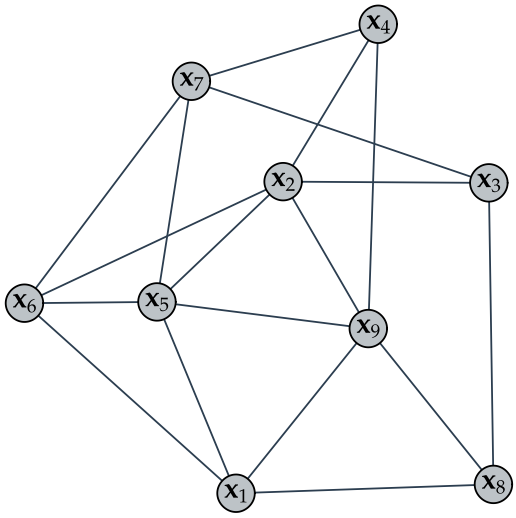


Figure 5.2 Illustration of a graph with nodes and edges.

A graph is a simple *mathematical structure* described by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$, where the set $\mathcal{V} = \{1, 2, 3, \ldots, p\}$ contains the indices of the nodes, the set of pairs $\mathcal{E} = \{(1,2), (1,3), \ldots, (i,j), \ldots\}$ contains the edges between any pair of nodes (i, j), and the weight matrix $\boldsymbol{W}$ encodes the strength of the relationships.

5.1.2 Graph Matrices

Several matrices are key in characterizing graphs.

- The *adjacency matrix* $\boldsymbol{W}$ is the most direct way to fully characterize a graph, where each element W_{ij} contains the strength of the connectivity, w_{ij}, between node i and node j:

$$[\boldsymbol{W}]_{ij} = \begin{cases} w_{ij} & \text{if } (i,j) \in \mathcal{E}, \\ 0 & \text{if } (i,j) \notin \mathcal{E}, \\ 0 & \text{if } i = j. \end{cases}$$

  We tacitly assume $W_{ij} \geq 0$ and $W_{ii} = 0$ (no self-loops). If $W_{ij} = W_{ji}$ (symmetric matrix $\boldsymbol{W}$), then the graph is called undirected.

- The *connectivity matrix* $\boldsymbol{C}$ is a particular case of the adjacency matrix containing elements that are either 0 or 1:

$$[\boldsymbol{C}]_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{E}, \\ 0 & \text{if } (i,j) \notin \mathcal{E}, \\ 0 & \text{if } i = j. \end{cases}$$

  It describes the connectivity pattern of the adjacency matrix. In fact, for some graphs the

adjacency matrix is already binary (with no measure of strength) and coincides with the connectivity matrix.

- The *degree matrix* $\boldsymbol{D}$ is defined as the diagonal matrix containing the degrees of the nodes $\boldsymbol{d} = (d_1, \ldots, d_p)$ along the diagonal, where the degree of a node i, d_i, represents its overall connectivity strength to all the nodes, that is, $d_i = \sum_j W_{ij}$. In other words,

$$\boldsymbol{D} = \text{Diag}(\boldsymbol{W}\mathbf{1}).$$

- The *Laplacian matrix* of a graph is defined as

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}.$$

  It is also referred to as the combinatorial Laplacian matrix to distinguish it from other variations in the definition of $\boldsymbol{L}$.

The Laplacian matrix may appear unusual at first; however, it possesses numerous valuable properties that make it a fundamental matrix in graph analysis. It satisfies the following mathematical properties:

- it is symmetric and positive semidefinite: $\boldsymbol{L} \succeq \mathbf{0}$;
- it has a zero eigenvalue with corresponding eigenvalue the all-one vector $\mathbf{1}$: $\boldsymbol{L}\mathbf{1} = \mathbf{0}$;
- the degree vector is found along its diagonal: $\text{diag}(\boldsymbol{L}) = \boldsymbol{d}$;
- the number of zero eigenvalues corresponds to the number of connected components of the graph (i.e., clusters);
- it defines a measure for the *smoothness* or *variance* of graph signals.

To further elaborate on the smoothness property of the Laplacian matrix, let $\boldsymbol{x} = (x_1, \ldots, x_p)$ denote a graph signal (i.e., one observation of the signal on all the nodes of the graph). Ignoring the graph, one way to measure the variance of this signal $\boldsymbol{x}$ is with the quantity $\sum_{i,j}(x_i - x_j)^2$. Now, to take into account the graph information in this computation, it makes sense to compute the variance between signals x_i and x_j only if these two nodes are connected, for example, by weighting the variance between each pair of nodes with the connectivity strength as $\sum_{i,j} W_{ij}(x_i - x_j)^2$. This finally leads us to the connection between the Laplacian matrix and a measure of smoothness or variance of the graph signal as

$$\boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x} = \frac{1}{2}\sum_{i,j} W_{ij}(x_i - x_j)^2. \tag{5.1}$$

Proof

$$\boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x} = \boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x} - \boldsymbol{x}^\top \boldsymbol{W}\boldsymbol{x} = \sum_i d_i x_i^2 - \sum_{i,j} w_{ij} x_i x_j = \sum_{i,j} w_{ij} x_i^2 - \sum_{i,j} w_{ij} x_i x_j.$$

□

Example 5.1 (Graph matrices for a toy example) Figure 5.3 shows a simple toy undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ with four nodes characterized by $\mathcal{V} = \{1, 2, 3, 4\}$, $\mathcal{E} = \{(1, 2), (2, 1), (1, 3), (3, 1), (2, 3), (3, 2), (2, 4), (4, 2)\}$, and weights $w_{12} = w_{21} = 2$, $w_{13} = w_{31} = 2$, $w_{23} = w_{32} = 3$, $w_{24} = w_{42} = 1$.

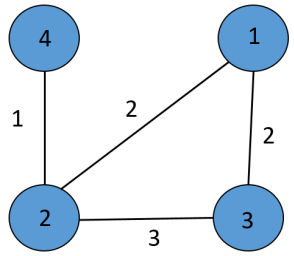


Figure 5.3 Toy graph.

The connectivity, adjacency, and Laplacian graph matrices are

$$\boldsymbol{C} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{W} = \begin{bmatrix} 0 & 2 & 2 & 0 \\ 2 & 0 & 3 & 1 \\ 2 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{L} = \begin{bmatrix} 4 & -2 & -2 & 0 \\ -2 & 6 & -3 & -1 \\ -2 & -3 & 5 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}.$$

5.2 Learning Graphs

In some applications, the graph structure can be readily obtained, such as in a social network where the nodes are the users and the connectivity can be measured by friendship relationships. In many other practical scenarios, however, the underlying graph structure is not directly observable and has to be inferred from the data, such as in a gene graph, brain activity graph, or a financial graph.

Numerous methods for learning graphs have been proposed in recent decades, ranging from heuristic techniques based on the physical interpretation of graphs to more statistically sound approaches that build upon well-established results from estimation theory.

The starting point in a graph learning method from data is the *data matrix*,

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_p] \in \mathbb{R}^{n \times p}, \tag{5.2}$$

where each column contains the signal of one variable or node, p is the number of variables or nodes, and n is the length of the signal or number of observations. In the context of financial time series, the number of observations is often denoted by T instead of n, and the number of nodes or assets is often denoted by N instead of p. Each row of matrix $\boldsymbol{X}$ represents one observation of the signal on the graph, called the *graph signal*.

The goal in graph learning is to transition from the data matrix $\boldsymbol{X}$ to the graph description $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ as illustrated in Figure 5.4. A simple example of graph learning from data is depicted in Figure 5.5, where the nodes are points in $\mathbb{R}^2$ sampled from the "two-moon" dataset, and the resulting graph clearly comprises two components corresponding to the two moons.

The field of graph learning has experienced significant growth in recent years, with numerous

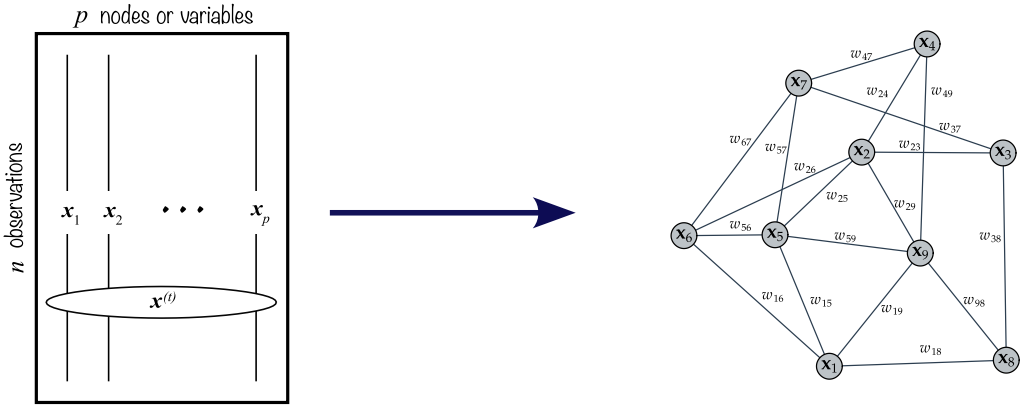


Figure 5.4 Learning a graph from data.

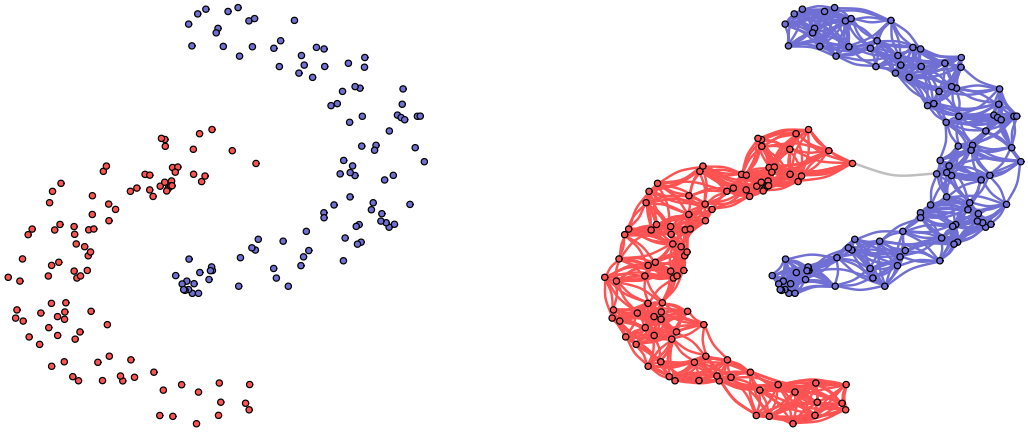


Figure 5.5 Illustration of graph learning for a toy example.

studies introducing enhanced graph estimation techniques in terms of both quality and computational efficiency.

- In what appears to be a pioneering effort, the seminal paper of Mantegna (1999) was the first to implement data-driven graphs in financial markets, employing a straightforward correlation graph.
- For a comprehensive understanding of graph theory, the standard textbooks Lauritzen (1996) and Kolaczyk (2009) provide excellent coverage.
- For introductory and overview articles on graph learning, refer to Mateos et al. (2019) and Dong et al. (2019).
- Basic graph learning algorithms are found in Lake and Tenenbaum (2010), Egilmez et al. (2017), and Zhao et al. (2019).
- Structured graph learning: A general approach via spectral constraints is proposed in Kumar

et al. (2019) and Kumar et al. (2020), graphs with sparsity are studied in Ying et al. (2020), and a convex formulation for bipartite graphs is developed in Cardoso et al. (2022b).

- Graph learning with financial data: General guidelines for financial time series are considered in Cardoso and Palomar (2020), learning under heavy tails is explored in Cardoso et al. (2021), and the application of bipartite-structured graphs for clustering is considered in Cardoso et al. (2022b); an overview is given in Cardoso et al. (2022a).

- A comprehensive overview of the literature on financial graphs in the past two decades can be found in Marti et al. (2021).

5.2.1 Learning Graphs from Similarity Measures

The simplest methods to infer a graph from data are based on computing each element of the adjacency matrix $\boldsymbol{W}$ (either weighted or 0–1 connectivity) by measuring the connectivity strength between each pair of nodes one by one. To measure the strength, a wide variety of similarity functions or scoring functions can be used (Kolaczyk, 2009), leading to totally different graphs.

For illustration purposes, a few simple methods are listed next based on the data matrix $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ defined in (5.2), where the ith column $\boldsymbol{x}_i \in \mathbb{R}^n$ corresponds to the signal at node i:

- *Thresholded distance graph*: Nodes i and j are connected ($w_{ij} = 1$) if the corresponding signals satisfy $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \leq \gamma$, where γ is a threshold; otherwise not connected ($w_{ij} = 0$).

- *Gaussian graph*: Set every pair of points $i \neq j$ as connected with the Gaussian weights

$$w_{ij} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right),$$

  where σ^2 controls the size of the neighborhood.

- *k-nearest neighbors (k-NN) graph*: Nodes i and j are connected ($w_{ij} = 1$) if $\boldsymbol{x}_i$ is one of the k closest points to $\boldsymbol{x}_j$ or vice versa; otherwise not connected ($w_{ij} = 0$).

- *Feature correlation graph*: Simply use pairwise feature correlation for $i \neq j$:

$$w_{ij} = \boldsymbol{x}_i^\mathsf{T}\boldsymbol{x}_j.$$

  It is worth noting that if the signals are normalized (i.e., $\|\boldsymbol{x}_i\|^2 = 1$), then the Euclidean distance used in the Gaussian weights is directly related to the correlation: $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = 2 \times (1 - \boldsymbol{x}_i^\mathsf{T}\boldsymbol{x}_j)$.

However, because the connectivity of each pair is measured independently of the others, these heuristic methods do not provide holistic measures and may not perform well in practice, especially for time series data. In principle, it is better to measure the connectivity of all pairs all at once in a joint manner, as explored in the next sections.

5.2.2 Learning Graphs from Smooth Signals

We will now derive a family of graph learning methods based on the measure of smoothness or variance of a graph signal defined in (5.1). To recall, given a p-dimensional graph signal $\boldsymbol{x}$ (defined on a graph with p nodes), a natural measure of its variance on the graph is $\boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x} = \frac{1}{2}\sum_{i,j} W_{ij}(x_i - x_j)^2$.

Suppose now we have n observations of graph signals contained in the data matrix $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ defined in (5.2), where the ith column $\boldsymbol{x}_i \in \mathbb{R}^n$ corresponds to the signal at node i and the t-th observation of the graph signal $\boldsymbol{x}^{(t)} \in \mathbb{R}^p$ is contained along the t-th row.

The overall variance corresponding to the n observations contained in the data matrix $\boldsymbol{X}$ can be written in terms of the Laplacian matrix $\boldsymbol{L}$ as

$$\sum_{t=1}^{n} (\boldsymbol{x}^{(t)})^\top \boldsymbol{L}\boldsymbol{x}^{(t)} = \mathrm{Tr}\left(\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^\top\right)$$

or, equivalently, in terms of the adjacency matrix $\boldsymbol{W}$ as

$$\sum_{t=1}^{n} \frac{1}{2}\sum_{i,j} W_{ij}\left(x_i^{(t)} - x_j^{(t)}\right)^2 = \frac{1}{2}\sum_{i,j} W_{ij}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = \frac{1}{2}\mathrm{Tr}(\boldsymbol{W}\boldsymbol{Z}),$$

where the matrix $\boldsymbol{Z}$ contains the squared Euclidean distances between signals: $Z_{ij} \triangleq \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$.

Now that we have expressed the variance of a collection of signals on a graph, we are ready to formulate the graph learning problem. The key observation is that if a signal has been generated by a graph, then it is expected to have a small variance as measured on that graph. This is a natural assumption because if two nodes are strongly connected, then the signals on these two nodes should be similar; alternatively, if two nodes are not connected, then the corresponding signals can be totally different.

Based on this assumption on the signal smoothness measured on the generating graph, suppose we collect some graph signals in the data matrix $\boldsymbol{X}$ with the hypothesis that they could have been generated either by graph $\mathcal{G}_1$ or $\mathcal{G}_2$, with corresponding Laplacian matrices $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$, respectively. Then, to determine which graph has generated the data, we simply have to compute the signal variance on each of the two graphs, $\mathrm{Tr}\left(\boldsymbol{X}\boldsymbol{L}_1\boldsymbol{X}^\top\right)$ and $\mathrm{Tr}\left(\boldsymbol{X}\boldsymbol{L}_2\boldsymbol{X}^\top\right)$, and choose the one with the smaller variance.

We can now take the previous problem of choosing a graph among a set of possible alternatives to the next level. Suppose again we collect some graph signals in the data matrix $\boldsymbol{X}$ and want to determine the graph $\mathcal{G}$ that best fits the data in the sense of producing a minimum signal variance. That is, we want to determine the graph (either in terms of $\boldsymbol{L}$ or $\boldsymbol{W}$) that gives the minimum signal variance on that graph. In addition, for practical purposes, we may want to include a regularization term on the estimated graph to control some other graph properties, such as sparsity, energy, or volume.

Thus, the simplest problem formulation in terms of the Laplacian matrix $\boldsymbol{L}$ is

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\boldsymbol{0}}{\text{minimize}} & \mathrm{Tr}\left(\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^\top\right) + \gamma h_L(\boldsymbol{L}) \\ \text{subject to} & \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j, \end{array} \tag{5.3}$$

where γ is a hyper-parameter to control the regularization level and $h_L(\boldsymbol{L})$ is a regularization function, e.g., $\|\boldsymbol{L}\|_1$, $\|\boldsymbol{L}\|_{\mathrm{F}}^2$, or volume$(\boldsymbol{L})$. Observe that this formulation incorporates as constraints the structural properties that the Laplacian matrix is supposed to satisfy (see Section 5.1.2).

Similarly, the simplest problem formulation in terms of the adjacency matrix $\boldsymbol{W}$ is

$$\begin{aligned} \underset{\boldsymbol{W}}{\text{minimize}} \quad & \tfrac{1}{2}\mathrm{Tr}(\boldsymbol{W}\boldsymbol{Z}) + \gamma h_W(\boldsymbol{W}) \\ \text{subject to} \quad & \mathrm{diag}(\boldsymbol{W}) = \boldsymbol{0}, \quad \boldsymbol{W} = \boldsymbol{W}^{\mathsf{T}} \geq \boldsymbol{0}, \end{aligned} \tag{5.4}$$

where $h_W(\boldsymbol{W})$ is a regularization function for the adjacency matrix. As before, this formulation incorporates as constraints the structural properties that the adjacency matrix is supposed to satisfy (see Section 5.1.2).

It is worth noting that these formulations are convex provided that the regularization terms are convex functions. It may seem that the formulation in terms of the Laplacian matrix in (5.3) has a higher complexity than (5.4) due to the positive semidefinite matrix constraint $\boldsymbol{L} \succeq \boldsymbol{0}$. However, this is not the case because $\boldsymbol{L} \succeq \boldsymbol{0}$ is implied by the other two sets of linear constraints, $\boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}$ and $L_{ij} = L_{ji} \leq 0$ for $i \neq j$ (Ying et al., 2020).

Controlling the Degrees of the Nodes

Controlling the degrees of the nodes in a graph is important to avoid the problem of unbalanced graphs or even isolated nodes in the graph. Recall from Section 5.1.2 that the degrees of the nodes in a graph are given by $\boldsymbol{d} = \boldsymbol{W}\boldsymbol{1}$.

Some examples of graph learning with degree control include:

- Sparse graphs with fixed degrees: The formulation in (5.4) was adopted in Nie et al. (2016) with the regularization term $\|\boldsymbol{W}\|_{\mathrm{F}}^2$ to control the sparsity of the graph and the constraint $\boldsymbol{W}\boldsymbol{1} = \boldsymbol{1}$ to control the degrees of the nodes.
- Sparse graphs with regularized degrees: An alternative to fixing the degrees as $\boldsymbol{W}\boldsymbol{1} = \boldsymbol{1}$ is to include a regularization term in the objective such as $-\boldsymbol{1}^{\mathsf{T}}\log(\boldsymbol{W}\boldsymbol{1})$ (Kalofolias, 2016).
- Robust graphs against noisy data: Since observations are often noisy, a robust version of the smoothness term $\mathrm{Tr}\left(\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^{\mathsf{T}}\right)$ in (5.3) was proposed in Dong et al. (2015) by combining the term $\mathrm{Tr}\left(\boldsymbol{Y}\boldsymbol{L}\boldsymbol{Y}^{\mathsf{T}}\right)$ with $\|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{F}}^2$, where $\boldsymbol{Y}$ attempts to remove the noise in $\boldsymbol{X}$.

5.2.3 Learning Graphs from Graphical Model Networks

In the previous sections, the data matrix $\boldsymbol{X}$ was assumed to contain the graph data without any statistical modeling. Alternatively, the graph learning process can be formulated in a more sound way as a statistical inference problem. We will now assume that the graph signals contained along the rows of $\boldsymbol{X}$, denoted by $\boldsymbol{x}^{(t)}$, $t = 1, \ldots, T$, where T is the number of observations, follow some multivariate distribution such as the Gaussian distribution,

$$\boldsymbol{x}^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ the covariance matrix of the observations. Later, in Section 5.4, more realistic heavy-tailed distributions will be considered.

Recall that, in practice, the covariance matrix $\boldsymbol{\Sigma}$ is typically estimated via the *sample covariance matrix*

$$\boldsymbol{S} = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{x}^{(t)} - \boldsymbol{\mu})(\boldsymbol{x}^{(t)} - \boldsymbol{\mu})^{\mathsf{T}} = \frac{1}{T}\big(\boldsymbol{X} - \bar{\boldsymbol{X}}\big)^{\mathsf{T}}\big(\boldsymbol{X} - \bar{\boldsymbol{X}}\big),$$

where the matrix $\bar{\boldsymbol{X}}$ contains $\boldsymbol{\mu}$ along each row (see Chapter 3 for more details on the estimation of covariance matrices).

The topic of estimation of graphical models goes back at least to the 1970s, when the inverse sample covariance matrix $\boldsymbol{S}^{-1}$ was proposed to determine a graph (Dempster, 1972). Some paradigmatic examples of network graph construction include the following:

- *Correlation networks*: The correlation between two random variables characterizes the similarity between them (at least in a linear sense). Therefore, it can be used as a way to measure how similar two nodes are and, hence, as a way to characterize a graph (Kolaczyk, 2009; Lauritzen, 1996). Nevertheless, a big drawback of using correlations is that two nodes may have a high correlation through a dependency on other nodes. For example, in the context of financial data, it is well known that all the stocks are significantly driven by a few factors. As a consequence, they all exhibit a high correlation that does not really characterize the similarity between stocks once the factors are accounted for.

- *Partial correlation networks*: The correlation measures the direct dependency between two nodes but ignores the other nodes. A more refined version is to measure the dependency but conditioned on the other nodes, that is, factoring out the effect of other nodes. For example, height and vocabulary of children are not independent, but they are conditionally independent conditioned on age.

  Interestingly, all the information of partial correlation and dependency conditioned on the rest of the graph is contained in the so-called *precision matrix* defined as $\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1}$, that is, the inverse covariance matrix. To be precise, the correlation between nodes i and j, conditioned on the rest of the nodes of the network, is equal to $-\Theta_{ij}/\sqrt{\Theta_{ii}\Theta_{jj}}$ (Kolaczyk, 2009; Lauritzen, 1996). As a consequence, two nodes i and j are conditionally independent if and only if $\Theta_{ij} = 0$.

  A graph defined by the precision matrix is called a *partial correlation network* or *conditional dependence graph*. In such a graph, nonzero off-diagonal entries of the precision matrix $\boldsymbol{\Theta}$ correspond to the edges of the graph.

- *Graphical LASSO (GLASSO)*: This method tries to estimate a sparse precision matrix (Banerjee et al., 2008; Friedman et al., 2008). Assuming a multivariate Gaussian distribution function for the data, the regularized maximum likelihood estimation of the precision matrix $\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1}$ can be formulated (refer to Section 3.4 in Chapter 3) as

$$\underset{\boldsymbol{\Theta}\succ\mathbf{0}}{\text{maximize}} \quad \log\det(\boldsymbol{\Theta}) - \text{Tr}(\boldsymbol{\Theta}\boldsymbol{S}) - \rho\|\boldsymbol{\Theta}\|_{1,\text{off}}, \tag{5.5}$$

  where $\|\cdot\|_{1,\text{off}}$ denotes the elementwise ℓ_1-norm of the off-diagonal elements and the hyper-parameter ρ controls the level of sparsity of the precision matrix. The regularization term $\|\boldsymbol{\Theta}\|_{1,\text{off}}$ enforces learning a sparse precision matrix.

- *Laplacian-structured GLASSO*: The precision matrix plays a role in graphs similar to the Laplacian matrix in the context of Gaussian Markov random fields (GMRFs) (Rue & Held, 2005). Under that setting, the GLASSO formulation in (5.5) can be reformulated to include the Laplacian constraints (Egilmez et al., 2017; Lake & Tenenbaum, 2010; Zhao et al., 2019) as[1]

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\mathbf{0}}{\text{maximize}} & \log \text{gdet}(\boldsymbol{L}) - \text{Tr}(\boldsymbol{L}\boldsymbol{S}) - \rho\|\boldsymbol{L}\|_{1,\text{off}} \\ \text{subject to} & \boldsymbol{L}\mathbf{1} = \mathbf{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j, \end{array} \tag{5.6}$$

  where gdet denotes the generalized determinant defined as the product of nonzero eigenvalues (this is necessary because, differently from $\boldsymbol{\Theta} \succ \mathbf{0}$ in (5.5), the Laplacian $\boldsymbol{L}$ is singular due to the constraint $\boldsymbol{L}\mathbf{1} = \mathbf{0}$).

- *Sparse GMRF graphs*: The Laplacian-structured GLASSO in (5.6) is an improvement over the vanilla GLASSO in (5.5). However, rather surprisingly, the ℓ_1-norm regularization term $\|\boldsymbol{L}\|_{1,\text{off}}$ produces dense graphs instead of sparse ones (Ying et al., 2020). Thus, a more appropriate formulation for sparse GMRF graphs is (Kumar et al., 2020; Ying et al., 2020) [2]

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\mathbf{0}}{\text{maximize}} & \log \text{gdet}(\boldsymbol{L}) - \text{Tr}(\boldsymbol{L}\boldsymbol{S}) - \rho\|\boldsymbol{L}\|_{0,\text{off}} \\ \text{subject to} & \boldsymbol{L}\mathbf{1} = \mathbf{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j. \end{array} \tag{5.7}$$

  The sparsity regularization term $\|\boldsymbol{L}\|_{0,\text{off}}$ is a difficult function to deal with, being nonconvex, nondifferentiable, and noncontinuous. In practice, it can be approximated with a smooth concave function $\sum_{ij} \phi(L_{ij})$, with ϕ concave, such as $\phi(x) = \log(\epsilon + |x|)$, where the parameter ϵ is a small positive number, and then this concave function can be successively approximated by a convex weighted ℓ_1-norm (via the majorization–minimization method (Sun et al., 2017), see Section B.7 in Appendix B for details), leading to the so-called reweighted ℓ_1-norm regularization method (Candès et al., 2008), which has been successfully employed for sparse graph learning (Cardoso et al., 2022a; Kumar et al., 2020; Ying et al., 2020).

5.2.4 Numerical Experiments

For the empirical analysis, we use three years' worth of stock price data (2016–2019) from the following three sectors of the S&P 500 index: Industrials, Consumer Staples, and Energy. The data matrix $\boldsymbol{X} \in \mathbb{R}^{T\times N}$ is created with the log-returns of the N assets.

Since different assets can show widely different volatilities, it is convenient to normalize them so that each has volatility one (normalizing the data is equivalent to using the correlation matrix in lieu of the covariance matrix). In fact, in machine learning it is almost always the case that data is normalized prior to the application of any method; this is to avoid problems arising from different dynamic ranges in the data or even different units of measurement.

[1] The R package `spectralGraphTopology` contains the function `learn_laplacian_gle_admm()` to solve problem (5.6) (Cardoso & Palomar, 2022).

[2] The R package `sparseGraph` contains the function `learn_laplacian_pgd_connected()` to solve problem (5.7).

It is worth pointing out that, as previously mentioned in Section 5.2.3, financial assets typically present a high correlation due to the market factor or other few factors (see Chapter 3). One may be tempted to remove the effect of these factors and then learn the graph based on the residual idiosynchratic component. However, the precision matrix (and the Laplacian matrix) have an interpretation of partial correlation, which means that the effect of common factors affecting other nodes is already removed.

Among all the methods explored in this section, the most appropriate for time series are the GMRF-based methods enforcing graph sparsity either via the ℓ_1-norm (the Laplacian-structured GLASSO formulation in (5.6)) or via the ℓ_0 penalty term (the sparse GMRF graph formulation (5.7)), which in practice is solved with a reweighted ℓ_1-norm iterative method. Figure 5.6 shows the graphs obtained with these two methods, demonstrating the superior performance of the reweighted ℓ_1-norm iterative method.

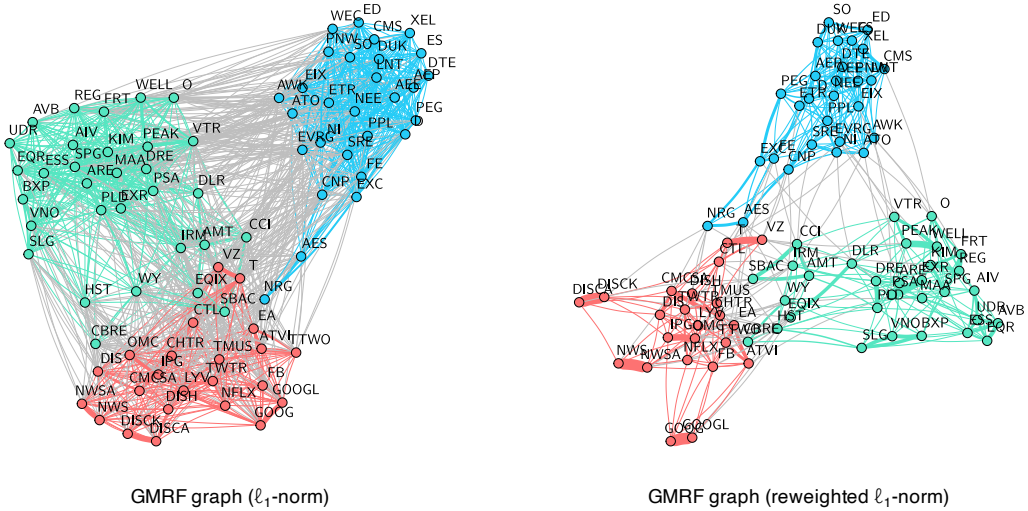


Figure 5.6 Effect of sparsity regularization term on financial graphs.

5.3 Learning Structured Graphs

The graph learning methods explored in Sections 5.2.2 and 5.2.3 can be successfully employed in different applications. Nevertheless, in some cases, some structural property of the unknown graph may be known and should be taken into account in the graph estimation process. Unfortunately, learning a graph with a specific structure is an NP-hard combinatorial problem for a general class of graphical models (Bogdanov et al., 2008) and, thus, designing a general algorithm is challenging.

Figure 5.7 illustrates different common types of graphs of interest, namely:

- *multi-component or k-component graph*: contains clusters, useful for classification;
- *regular graph*: where each node has the same number of neighbors (alternatively, the same degree), useful for balanced graphs;

- *modular graph*: satisfies some shortest path distance properties among triplets of nodes, useful for social network analysis;
- *bipartite graph*: containing two types of nodes with inter-connections and without intra-connections;
- *grid graph*: the nodes are distributed following a rectangular grid or two-dimensional lattice; and
- *tree graph*: undirected graph in which any two vertices are connected by exactly one path, resulting in a structure resembling a tree where connections branch out from nodes at each level.

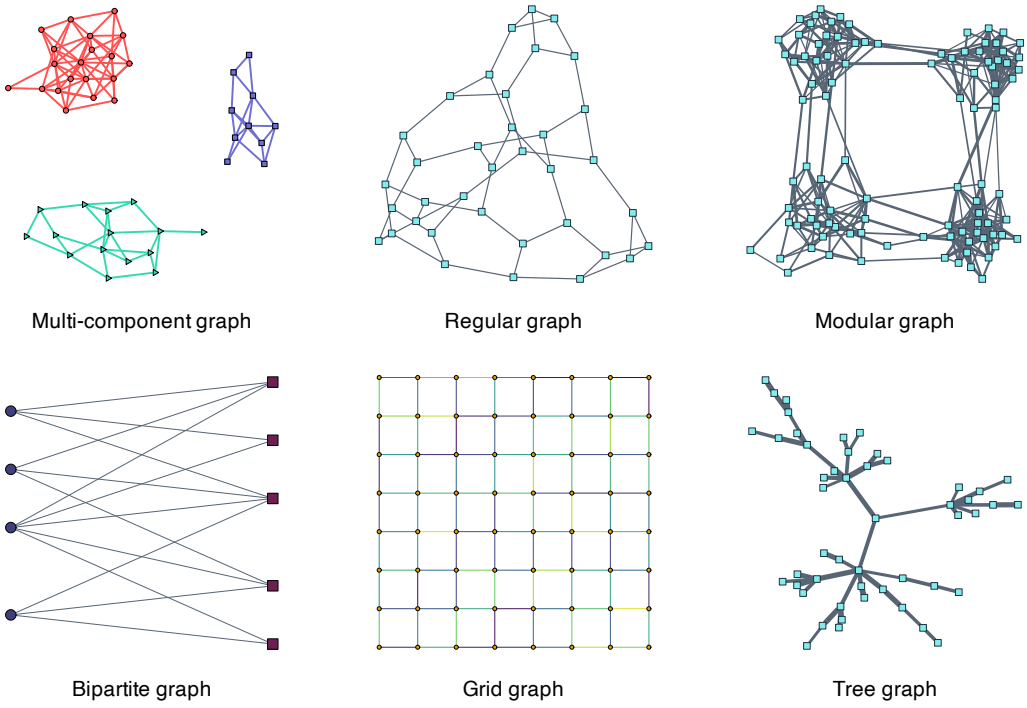


Figure 5.7 Types of structured graphs.

Some of the structural constraints are not difficult to control. For example, a grid graph simply means that each node can only be connected with a given neighborhood, that is, the adjacency and Laplacian matrices have many elements fixed to zero a priori. Another example is that of regular graphs where the degrees of the nodes, row sums of $\boldsymbol{W}$, are fixed. If, instead, the number of neighbors is to be controlled, then the cardinality of the rows of $\boldsymbol{W}$ has to be constrained, which is a nonconvex constraint.

However, other graph structural constraints are more complicated to control and can be characterized by *spectral properties* (i.e., properties on the eigenvalues) of the Laplacian and adjacency matrices (Chung, 1997). Such spectral properties can be enforced in the graph learning formulations (Kumar et al., 2019, 2020; Nie et al., 2016) as discussed next.

5.3.1 k-Component Graphs

A k-component graph (i.e., a graph with k clusters or components) is characterized by its Laplacian matrix being low rank with k zero eigenvalues (Chung, 1997).

More explicitly, consider the eigenvalue decomposition of the Laplacian matrix,

$$\boldsymbol{L} = \boldsymbol{U}\text{Diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)\boldsymbol{U}^\top,$$

where $\boldsymbol{U}$ contains eigenvectors columnwise and $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_p$ are the eigenvalues in increasing order. Then, a k-component graph has the k smallest eigenvalues equal to zero:

$$\lambda_1 = \cdots = \lambda_k = 0.$$

The opposite direction is also true: if the Laplacian matrix has k zero eigenvalues, then it is a k-component graph. Figure 5.8 illustrates this spectral property of k-component graphs.

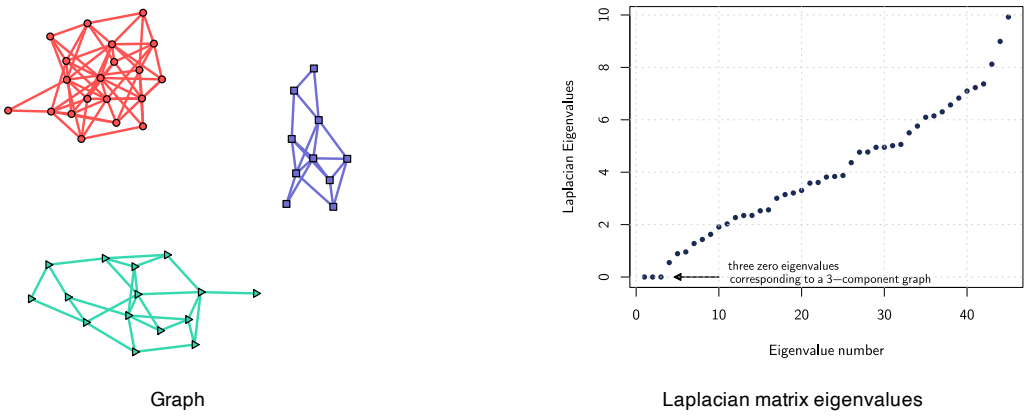


Figure 5.8 Example of a three-component graph (three clusters) with corresponding Laplacian matrix eigenvalues (three zero eigenvalues).

The low-rank property of the Laplacian matrix, $\text{rank}(\boldsymbol{L}) = p - k$, is a nonconvex and difficult constraint to handle in an optimization problem. In practice, it can be better handled by enforcing the sum of the k smallest eigenvalues to be zero, $\sum_{i=1}^{k} \lambda_i(\boldsymbol{L}) = 0$, via Ky Fan's theorem (Fan, 1949):

$$\sum_{i=1}^{k} \lambda_i(\boldsymbol{L}) = \min_{\boldsymbol{F} \in \mathbb{R}^{p \times k}: \boldsymbol{F}^\top \boldsymbol{F} = \boldsymbol{I}} \text{Tr}(\boldsymbol{F}^\top \boldsymbol{L} \boldsymbol{F}),$$

where the matrix $\boldsymbol{F}$ becomes a new variable to be optimized.

For convenience, the low-rank constraint can be relaxed and moved to the objective function as a regularization term. For instance, denoting a general objective function by $f(\boldsymbol{L})$, a regularized formulation could be

$$\begin{array}{ll} \underset{\boldsymbol{L}, \boldsymbol{F}}{\text{minimize}} & f(\boldsymbol{L}) + \gamma \text{Tr}(\boldsymbol{F}^\top \boldsymbol{L} \boldsymbol{F}) \\ \text{subject to} & \text{any constraint on } \boldsymbol{L}, \\ & \boldsymbol{F}^\top \boldsymbol{F} = \boldsymbol{I}, \end{array}$$

where now the optimization variables are $\boldsymbol{L}$ and $\boldsymbol{F}$, making the problem nonconvex due to the term $\boldsymbol{F}^\top \boldsymbol{L}\boldsymbol{F}$. To solve this problem we can conveniently use an alternate minimization between $\boldsymbol{L}$ and $\boldsymbol{F}$ (see Section B.6 in Appendix B for details). The optimization of $\boldsymbol{L}$ with a fixed $\boldsymbol{F}$ is basically the same problem without the low-rank structure, whereas the optimization of $\boldsymbol{F}$ with a fixed $\boldsymbol{L}$ is a trivial problem with a solution given by the eigenvectors corresponding to the k smallest eigenvalues of $\boldsymbol{L}$.

Some illustrative and specific examples of low-rank graph estimation problems include:

- *Low-rank graph approximation*: Suppose we are given a graph in the form of a Laplacian matrix $\boldsymbol{L}_0$; we can formulate the low-rank approximation problem as

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\boldsymbol{0},\boldsymbol{F}}{\text{minimize}} & \|\boldsymbol{L}-\boldsymbol{L}_0\|_{\text{F}}^2 + \gamma \text{Tr}\left(\boldsymbol{F}^\top \boldsymbol{L}\boldsymbol{F}\right) \\ \text{subject to} & \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j, \\ & \text{diag}(\boldsymbol{L}) = \boldsymbol{1}, \\ & \boldsymbol{F}^\top \boldsymbol{F} = \boldsymbol{I}. \end{array} \tag{5.8}$$

- *Low-rank graphs from sparse GMRFs*: Consider the formulation in (5.7) to learn a sparse Laplacian matrix under a GMRF framework. We can easily reformulate it to include a low-rank regularization term as

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\boldsymbol{0},\boldsymbol{F}}{\text{maximize}} & \log \text{gdet}(\boldsymbol{L}) - \text{Tr}(\boldsymbol{L}\boldsymbol{S}) - \rho\|\boldsymbol{L}\|_{0,\text{off}} - \gamma \text{Tr}\left(\boldsymbol{F}^\top \boldsymbol{L}\boldsymbol{F}\right) \\ \text{subject to} & \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j, \\ & \text{diag}(\boldsymbol{L}) = \boldsymbol{1}, \\ & \boldsymbol{F}^\top \boldsymbol{F} = \boldsymbol{I}. \end{array} \tag{5.9}$$

  An alternative formulation based on $\|\boldsymbol{L} - \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top\|_{\text{F}}^2$ as the regularization term can also be considered (Kumar et al., 2019, 2020).[3]

Enforcing a low-rank Laplacian matrix will generate a k-component graph, but it may have isolated nodes. To avoid such trivial graph solutions one can control the degrees of the nodes with the constraint $\text{diag}(\boldsymbol{L}) = \boldsymbol{1}$ (Cardoso & Palomar, 2020).

5.3.2 Bipartite Graphs

A bipartite graph is characterized by its adjacency matrix having symmetric eigenvalues around zero (Chung, 1997).

More explicitly, consider the eigenvalue decomposition of the adjacency matrix,

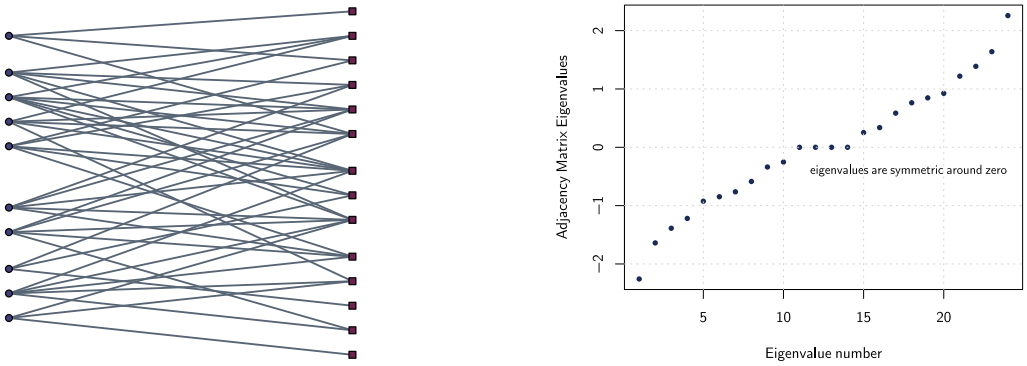
$$\boldsymbol{W} = \boldsymbol{V}\text{Diag}(\psi_1, \psi_2, \ldots, \psi_p)\boldsymbol{V}^\top,$$

where $\boldsymbol{V}$ contains the eigenvectors columnwise and $\psi_1 \leq \psi_2 \leq \cdots \leq \psi_p$ are the eigenvalues in increasing order. Then, a bipartite graph has symmetric eigenvalues around zero:

$$\psi_i = -\psi_{p-i}, \quad \forall i.$$

[3] The R package `spectralGraphTopology` contains several functions to solve graph formulations with spectral constraints (Cardoso & Palomar, 2022).

The opposite direction is also true: if the adjacency matrix has symmetric eigenvalues, then the graph is bipartite. Figure 5.9 illustrates this spectral property of bipartite graphs.



Graph

Adjacency matrix eigenvalues

Figure 5.9 Example of a bipartite graph with corresponding adjacency matrix eigenvalues.

Enforcing symmetric eigenvalues in the adjacency matrix $\boldsymbol{W}$ is a nonconvex and difficult constraint to handle in an optimization problem (Kumar et al., 2019, 2020). An alternative and more convenient characterization of a bipartite graph is via its Laplacian matrix, which has the following structure:[4]

$$\boldsymbol{L} = \begin{bmatrix} \mathrm{Diag}(\boldsymbol{B}\mathbf{1}) & -\boldsymbol{B} \\ -\boldsymbol{B}^\mathsf{T} & \mathrm{Diag}(\boldsymbol{B}^\mathsf{T}\mathbf{1}) \end{bmatrix}, \tag{5.10}$$

where $\boldsymbol{B} \in \mathbb{R}_+^{r\times q}$ contains the edge weights between the two types of nodes (with r and q denoting the number of nodes in each group). Note that, any Laplacian $\boldsymbol{L}$ constructed as in (5.10) already satisfies $\boldsymbol{L}\mathbf{1} = \mathbf{0}, L_{ij} = L_{ji} \leq 0, \forall i \neq j$.

Some illustrative and specific examples of low-rank graph estimation problems are:

- *Bipartite graph approximation*: Suppose we are given a graph in the form of a Laplacian matrix $\boldsymbol{L}_0$; we can find the closest bipartite graph approximation using (5.10) as

$$\begin{array}{ll} \underset{\boldsymbol{L},\boldsymbol{B}}{\text{minimize}} & \|\boldsymbol{L} - \boldsymbol{L}_0\|_\mathrm{F}^2 \\ \text{subject to} & \boldsymbol{L} = \begin{bmatrix} \mathrm{Diag}(\boldsymbol{B}\mathbf{1}) & -\boldsymbol{B} \\ -\boldsymbol{B}^\mathsf{T} & \mathrm{Diag}(\boldsymbol{B}^\mathsf{T}\mathbf{1}) \end{bmatrix}, \\ & \boldsymbol{B} \geq \mathbf{0}, \quad \boldsymbol{B}\mathbf{1} = \mathbf{1}. \end{array}$$

- *Bipartite graphs from sparse GMRFs*: Consider now the sparse GMRF formulation in (5.7),

[4] The R package `finbipartite` contains methods to solve problem (5.10) (Cardoso & Palomar, 2023a).

but enforcing the bipartite structure with (5.10) as follows (Cardoso et al., 2022b):

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\mathbf{0},\boldsymbol{B}}{\text{maximize}} & \log \operatorname{gdet}(\boldsymbol{L}) - \operatorname{Tr}(\boldsymbol{L}\boldsymbol{S}) - \rho\|\boldsymbol{L}\|_{0,\text{off}} \\ \text{subject to} & \boldsymbol{L} = \begin{bmatrix} \operatorname{Diag}(\boldsymbol{B}\mathbf{1}) & -\boldsymbol{B} \\ -\boldsymbol{B}^\top & \operatorname{Diag}(\boldsymbol{B}^\top\mathbf{1}) \end{bmatrix}, \\ & \boldsymbol{B} \geq \mathbf{0}, \quad \boldsymbol{B}\mathbf{1} = \mathbf{1}. \end{array} \tag{5.11}$$

5.3.3 Numerical Experiments

For the empirical analysis, we again use three years' worth of stock price data (2016–2019) from the following three sectors of the S&P 500 index: Industrials, Consumer Staples, and Energy.

Stocks are classified and grouped together into sectors and industries. This organization is convenient for investors in order to easily diversify their investment across different sectors (which presumably are less correlated than stocks within each sector). However, there are different criteria to classify stocks into sectors and industries, producing different classifications; for example:

- *production-oriented* approach: by the products they produce or use as inputs in the manufacturing process;
- *market-oriented* approach: by the markets they serve.

Table 5.1 lists some of the major sector classification systems in the financial industry:

- GICS (Global Industry Classification Standard): A system developed by Morgan Stanley Capital International (MSCI) and Standard & Poor's (S&P) in 1999 to classify companies and stocks into industry groups, sectors, and sub-industries based on their primary business activities.

- ICB (Industry Classification Benchmark): A classification system developed by Dow Jones and FTSE Group that categorizes companies and securities into industries, supersectors, sectors, and subsectors based on their primary business activities. It is used by investors, analysts, and researchers for consistent sector analysis and performance comparison.

- TRBC (Thomson Reuters Business Classification): A proprietary classification system developed by Thomson Reuters to categorize companies and securities into economic sectors, business sectors, and industries based on their primary business activities. It is used for research, analysis, and investment purposes.

In principle, each system is different and groups stocks in a different manner, although there is some degree of similarity.

Table 5.1 Major sector classification systems.

Level/System	GICS	ICB	TRBC
First	11 sectors	10 industries	10 economic sectors
Second	24 industry groups	19 supersectors	28 business sectors
Third	68 industries	41 sectors	56 industry groups
Fourth	157 sub-industries	114 subsectors	136 industries

Given that there are multiple stock classifications into sectors and industries, it is not clear which one is more relevant in terms of portfolio investment. In a more data-oriented world, one can ignore such human-made classification systems and instead learn the graph of stocks from data and, perhaps, even enforce a k-component graph to obtain a clustered graph automatically.

Two-Stage vs. Joint Design of k-Component Graphs

Suppose we want to learn a k-component graph; as described in Section 5.3.1, one can employ two approaches:

- *Two-stage approach*: First learn a connected graph using any formulation, such as the GMRF design in (5.6) or (5.7). Then perform a low-rank approximation to that graph as in (5.8).
- *Joint approach*: Enforce the low-rank property in the GMRF formulation as in (5.9), which clearly must be better than the two-stage approach.

Figure 5.10 shows the effect of employing a joint design vs. the two-stage design using the GMRF formulation ((5.7) for the two-stage case and (5.9) for the joint case). The difference is too obvious to require any comment on the fact that a joint approach is significantly superior. Note, however, that care has to be taken with controlling the degrees when enforcing a low-rank graph, as discussed next.

Isolated Nodes in Low-Rank Designs for k-Component Graphs

Isolated nodes is an artifact that may happen when imposing a low-rank structure in the Laplacian matrix to obtain a graph with clusters. Basically, if one enforces a low-rank structure to learn a k-component graph but the graph deviates from a k-component graph, the solution to this formulation may trivially leave some nodes without any connection to other nodes (i.e., isolated nodes) to artificially increase the number of clusters.

The solution to avoid isolated nodes is quite simple. As discussed in Section 5.3.1, control the degrees of the nodes so that they are nonzero; even better if they are balanced.

Figure 5.11 clearly shows the effect of isolated nodes for the low-rank GMRF design in (5.9) and with the additional degree control. The difference is staggering: always control the degrees.

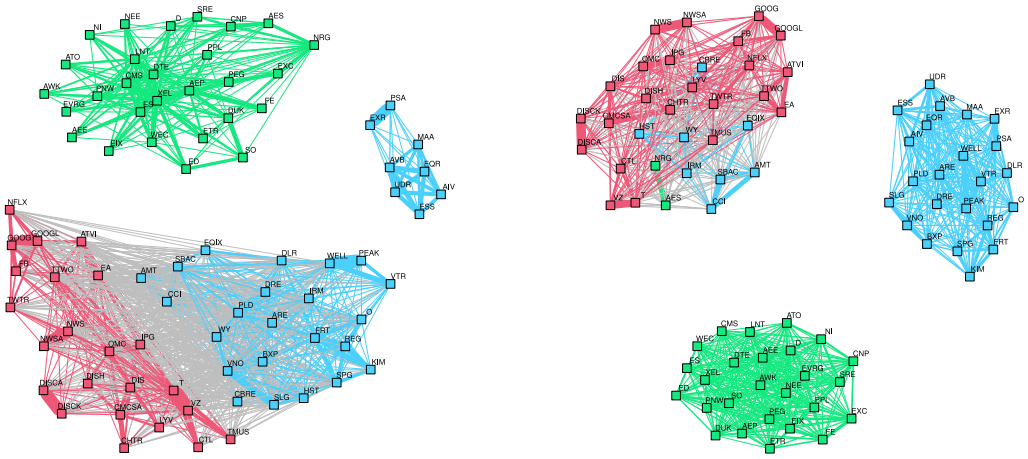


Figure 5.10 Effect of joint design vs. two-stage design on multi-component financial graphs.

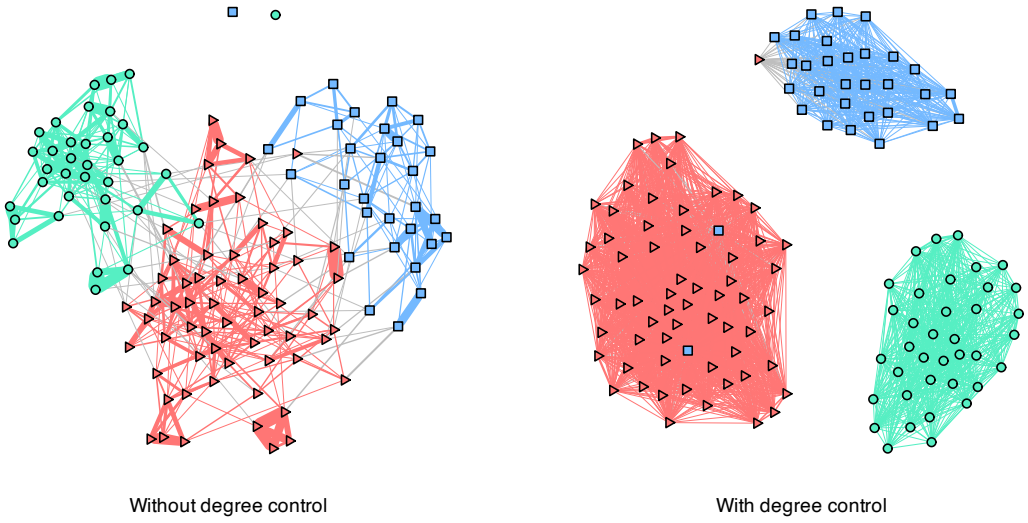


Figure 5.11 Effect of isolated nodes on low-rank (clustered) financial graphs.

5.4 Learning Heavy-Tailed Graphs

5.4.1 From Gaussian to Heavy-Tailed Graphs

The basic GMRF formulations in (5.6) or (5.7) are based on the assumption that data follow a Gaussian distribution,

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right),$$

leading to the maximum likelihood estimation of the Laplacian matrix $\boldsymbol{L}$ (which plays the role of the inverse covariance matrix $\boldsymbol{\Sigma}^{-1}$) formulated as

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\mathbf{0}}{\text{maximize}} & \log \operatorname{gdet}(\boldsymbol{L}) - \operatorname{Tr}(\boldsymbol{L}\boldsymbol{S}) \\ \text{subject to} & \boldsymbol{L}\mathbf{1} = \mathbf{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j. \end{array} \tag{5.12}$$

However, in many applications, data do not follow a Gaussian distribution. Instead, a better model for data may be a heavy-tailed distribution such as the Student t distribution characterized by

$$f(\boldsymbol{x}) \propto \frac{1}{\sqrt{|\boldsymbol{\Sigma}|}} \left(1 + \frac{1}{\nu}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)^{-(p+\nu)/2},$$

where the parameter $\nu > 2$ determines how heavy the tails are (for $\nu \to \infty$ it becomes the Gaussian distribution). This leads to

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\mathbf{0}}{\text{maximize}} & \log \operatorname{gdet}(\boldsymbol{L}) - \dfrac{p+\nu}{T}\displaystyle\sum_{t=1}^{T} \log\left(1 + \frac{1}{\nu}(\boldsymbol{x}^{(t)})^\top \boldsymbol{L}\boldsymbol{x}^{(t)}\right) \\ \text{subject to} & \boldsymbol{L}\mathbf{1} = \mathbf{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j, \end{array} \tag{5.13}$$

where the mean is assumed to be zero for simplicity.

This heavy-tailed formulation is nonconvex and difficult to solve directly. Instead, we can employ the majorization–minimization (MM) framework (Sun et al., 2017) to iteratively solve the problem (see Section B.7 in Appendix B for details on MM). In particular, the logarithm upper bound $\log(t) \leq \log(t_0) + \dfrac{t}{t_0} - 1$ leads to

$$\log\left(1 + \frac{1}{\nu}(\boldsymbol{x}^{(t)})^\top \boldsymbol{L}\boldsymbol{x}^{(t)}\right) \leq \log\left(1 + \frac{1}{\nu}(\boldsymbol{x}^{(t)})^\top \boldsymbol{L}_0\boldsymbol{x}^{(t)}\right) + \frac{\nu + (\boldsymbol{x}^{(t)})^\top \boldsymbol{L}\boldsymbol{x}^{(t)}}{\nu + (\boldsymbol{x}^{(t)})^\top \boldsymbol{L}_0\boldsymbol{x}^{(t)}} - 1.$$

Thus, the surrogate problem of (5.13) can be written similarly to the Gaussian case (Cardoso et al., 2021) as

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\mathbf{0}}{\text{maximize}} & \log \operatorname{gdet}(\boldsymbol{L}) - \dfrac{p+\nu}{T}\displaystyle\sum_{t=1}^{T} \frac{(\boldsymbol{x}^{(t)})^\top \boldsymbol{L}\boldsymbol{x}^{(t)}}{\nu + (\boldsymbol{x}^{(t)})^\top \boldsymbol{L}_0\boldsymbol{x}^{(t)}} \\ \text{subject to} & \boldsymbol{L}\mathbf{1} = \mathbf{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j. \end{array}$$

Summarizing, the MM algorithm iteratively solves the following sequence of Gaussianized problems[5] for $k = 1, 2, \ldots$:

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\mathbf{0}}{\text{maximize}} & \log \operatorname{gdet}(\boldsymbol{L}) - \operatorname{Tr}(\boldsymbol{L}\boldsymbol{S}^k) \\ \text{subject to} & \boldsymbol{L}\mathbf{1} = \mathbf{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j, \end{array} \tag{5.14}$$

[5] The R package `fingraph`, based on Cardoso et al. (2021), contains efficient algorithms to learn graphs from heavy-tailed formulations (Cardoso & Palomar, 2023b). In particular, the function `learn_regular_heavytail_graph()` solves problem (5.13) with an additional degree constraint; also, the function `learn_kcomp_heavytail_graph()` further allows the specification of a k-component constraint.

where $\boldsymbol{S}^k$ is conveniently defined as a weighted sample covariance matrix

$$\boldsymbol{S}^k = \frac{1}{T}\sum_{t=1}^{T} w_t^k \times \boldsymbol{x}^{(t)}(\boldsymbol{x}^{(t)})^\mathsf{T},$$

with weights $w_t^k = \dfrac{p+\nu}{\nu + (\boldsymbol{x}^{(t)})^\mathsf{T}\boldsymbol{L}^k\boldsymbol{x}^{(t)}}$.

In words, to solve a heavy-tailed graph learning problem, instead of solving the Gaussian formulation in (5.12), one simply needs to solve a sequence of Gaussianized formulations as in (5.14).

5.4.2 Numerical Experiments

From Gaussian to Heavy-Tailed Graphs

We use again three years worth of stock price data (2016-2019) from the following three sectors of the S&P 500 index: Industrials, Consumer Staples, and Energy.

Figure 5.12 shows the results for the MRF formulations under the Gaussian assumption and under a heavy-tailed model. For the Gaussian case, the GMRF with a concave sparsity regularizer in (5.7) is used. For the non-Gaussian case, the heavy-tailed MRF formulation in (5.13) is employed. The conclusion is clear: heavy-tailed graphs are more convenient for financial data.

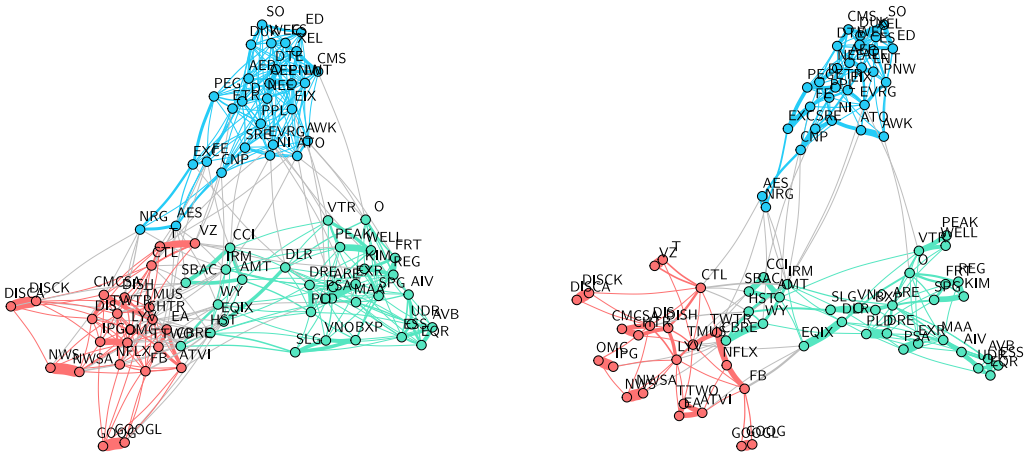


Figure 5.12 Gaussian vs. heavy-tailed graph learning with stocks.

k-Component Graphs

The effect of learning a k-component graph is that the graph will automatically be clustered. The following numerical example illustrates this point with foreign exchange (FX) market data (FX or forex is the trading of one currency for another). In particular, we use the 34

most traded currencies in the period from January 2, 2019 to December 31, 2020 (a total of $n = 522$ observations). The data matrix is composed by the log-returns of the currency prices with respect to the United States dollar (USD). Unlike for S&P 500 stocks, there is no classification standard for currencies.

To start with, Figure 5.13 shows the graphs learned with the GMRF formulations (with the ℓ_1-norm in (5.6) and concave sparsity regularizer in (5.7)) and with the heavy-tailed MRF formulation in (5.13). As expected, the GMRF graph with ℓ_1-norm regularizer does not give a sparse graph like the one with a concave sparsity regularizer. Also, the heavy-tailed MRF formulation produces a much cleaner and more interpretable graph; for instance, the expected correlation between currencies of locations geographically close to each other is more evident, for example, {Hong Kong SAR, China}, {Taiwan, South Korea}, and {Poland, Czech Republic}.

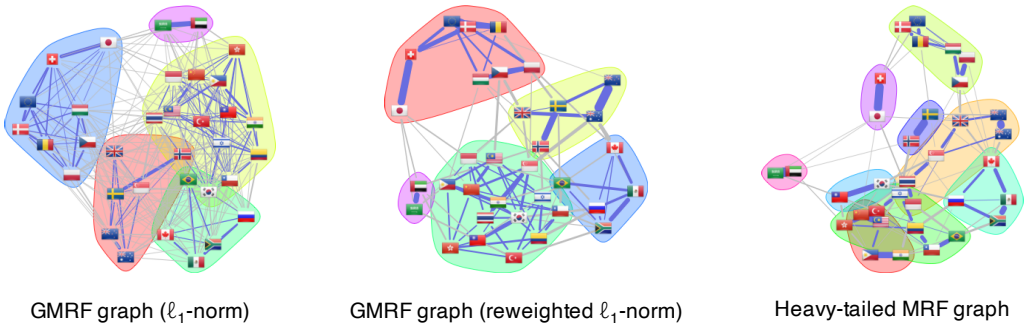


Figure 5.13 Gaussian vs. heavy-tailed graph learning with FX.

Turning now to k-component graphs, Figure 5.14 shows the equivalent graphs learned enforcing the low-rank structure to obtain clustered graphs (in particular, nine-component graphs). In this case, all the graphs are clearly clustered as expected. The heavy-tailed MRF graph provides a clearer interpretation with more reasonable clusters, such as {New Zealand, Australia} and {Poland, Czech Republic, Hungary}, which are not separated in the Gaussian-based graphs. Observe that the heavy-tailed MRF graph in (5.13) with low-rank structure controls the degrees of the nodes and isolated nodes are avoided (the GMRF formulations used do not control the degrees and the graphs present isolated nodes).

5.5 Learning Time-Varying Graphs

All the aforementioned graph learning frameworks are designed towards *static graphs*. However, many practical network-based systems are intrinsically dynamic in nature and static graphs would inherently neglect the time variations of the data (Kolaczyk, 2009, Section 8.6). For instance, financial systems are evidently dynamic and subject to various market regimes, such as bull markets, bear markets, economic crises, bubbles, and so on.

Learning time-varying or *dynamic graphs* is more cumbersome than static graphs. Not only does the formulation become more complicated but the number of variables increases significantly. As a consequence, few works can be found in the literature.

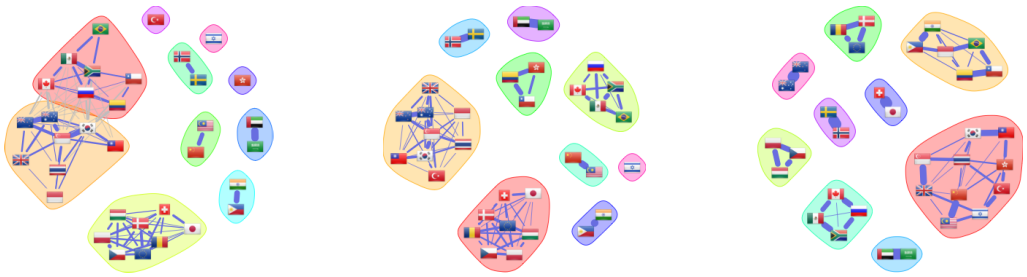


Figure 5.14 Gaussian vs. heavy-tailed multi-component graph learning with FX.

A naive approach would be to divide the available observations into T chunks, each with n_t samples, and then learn different graphs independently for each chunk: $\boldsymbol{L}_t$ for $t = 1, \ldots, T$. The drawback of this approach, apart from the fact that fewer observations are available to learn each graph, is that the graphs learned may lack time consistency, that is, they may change abruptly rather than smoothly.

In order to learn time-varying graphs that preserve the time consistency between consecutive graphs, a regularization term of the form $d(\boldsymbol{L}_{t-1}, \boldsymbol{L}_t)$ can be employed (Kalofolias et al., 2017); for example, the Frobenius norm, $d(\boldsymbol{L}_{t-1}, \boldsymbol{L}_t) = \|\boldsymbol{L}_{t-1} - \boldsymbol{L}_t\|_{\mathrm{F}}^2$, or the ℓ_1-norm, $d(\boldsymbol{L}_{t-1}, \boldsymbol{L}_t) = \|\boldsymbol{L}_{t-1} - \boldsymbol{L}_t\|_1$.

The following graph learning formulation follows from the GMRF framework in (5.6) or (5.7) but preserving the time consistency (Cardoso & Palomar, 2020):

$$\begin{array}{ll} \underset{\boldsymbol{L}_1,\ldots,\boldsymbol{L}_T}{\text{minimize}} & \sum_{t=1}^{T} n_t \times [\mathrm{Tr}(\boldsymbol{L}_t \boldsymbol{S}_t) - \log \mathrm{gdet}(\boldsymbol{L}_t)] + \delta \sum_{t=2}^{T} d(\boldsymbol{L}_{t-1}, \boldsymbol{L}_t) \\ \text{subject to} & \left\{ \boldsymbol{L}_t \succeq \boldsymbol{0}, \quad \boldsymbol{L}_t \boldsymbol{1} = \boldsymbol{0}, \quad (\boldsymbol{L}_t)_{ij} = (\boldsymbol{L}_t)_{ji} \leq 0, \quad \forall\, i \neq j \right\}_{t=1}^{T}, \end{array} \tag{5.15}$$

where $\boldsymbol{S}_t$ denotes the sample correlation matrix of chunk t and the hyper-parameter δ controls the level of time consistency (for $\delta = 0$ it becomes the naive approach without time consistency and for $\delta \to \infty$ tends to the static graph solution).

This problem admits two different resolution philosophies:

- *Offline approach*: The dynamic graph is conditioned on all the T chunks of observations (i.e., with look-ahead bias):

$$\hat{\boldsymbol{L}}_{t|T} \qquad t = 1, \ldots, T.$$

- *Online approach*: The dynamic graph is obtained causally, without look-ahead bias, using rolling-window estimation:

$$\hat{\boldsymbol{L}}_{t|t} \qquad t = 1, \ldots, T.$$

An alternative online approach based on a probabilistic temporal model for graph weights was proposed in Javaheri et al. (2025). Let $\boldsymbol{w}_t$ collect the edge weights in frame t, with

$\boldsymbol{L}_t = \mathcal{L}(\boldsymbol{w}_t)$ and $\mathcal{L}(\cdot)$ the Laplacian operator. A simplified version of the formulation in that paper is

$$\begin{array}{ll} \underset{\boldsymbol{w}_t \geq \boldsymbol{0}}{\text{minimize}} & \operatorname{Tr}(\boldsymbol{L}_t \boldsymbol{S}_t) - \log \operatorname{gdet}(\boldsymbol{L}_t) + \alpha \|\boldsymbol{w}_t - \boldsymbol{a} \odot \hat{\boldsymbol{w}}_{t-1}\|_1 \\ \text{subject to} & \boldsymbol{L}_t = \mathcal{L}(\boldsymbol{w}_t), \end{array} \tag{5.16}$$

where $\hat{\boldsymbol{w}}_{t-1}$ is the estimate from the previous time frame, $\boldsymbol{a}$ contains the nonnegative autoregressive coefficients, which can also be estimated, and α controls the temporal consistency. The original paper considers a more sophisticated formulation that accommodates heavy-tailed data and missing values, and incorporates graph sparsity and clustered structures.

5.6 Summary

While the topic of estimation of graphical models goes back at least to the 1970s, the first application in financial markets can only be found in the seminal paper of Mantegna (1999), where a simple correlation graph was employed. Since then, a myriad methods have been proposed as surveyed in Marti et al. (2021).

Among the many methods described in this chapter, only a few seem to be appropriate for financial time series and able to produce desirable graphs:

- Sparse GMRF graphs (Section 5.2.3) as formulated in (5.7) are a good start.
- Asset clustering is a data-driven alternative to handcrafted sectors or industries. For this purpose, k-component graphs (Section 5.3.1) can be readily obtained by imposing low-rank structure with degree control as in formulation (5.9).
- Lastly, owing to the inherent heavy-tailed nature of financial data, heavy-tailed graph models (Section 5.4) are undoubtedly more suitable than Gaussian ones. These models are formulated in (5.13) and can be practically solved iteratively using the simpler problems outlined in (5.14).

Exercises

5.1 (Graph matrices) Consider a graph described by the following adjacency matrix:

$$\boldsymbol{W} = \begin{bmatrix} 0 & 2 & 2 & 0 & 6 & 1 \\ 2 & 0 & 3 & 1 & 5 & 0 \\ 2 & 3 & 0 & 9 & 0 & 2 \\ 0 & 1 & 9 & 0 & 7 & 3 \\ 6 & 5 & 0 & 7 & 0 & 2 \\ 1 & 0 & 2 & 3 & 2 & 0 \end{bmatrix}.$$

a. Calculate the connectivity matrix.
b. Calculate the degree matrix.
c. Calculate the Laplacian matrix.
d. Plot the graph showing the nodes and indicating the connectivity weights.

5.2 (Laplacian matrix of a k-connected graph) Consider a graph described by the following adjacency matrix:

$$\boldsymbol{W} = \begin{bmatrix} 0 & 2 & 0 & 0 & 2 & 0 \\ 2 & 0 & 0 & 9 & 3 & 2 \\ 0 & 0 & 0 & 7 & 0 & 2 \\ 0 & 9 & 7 & 0 & 0 & 3 \\ 2 & 3 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 3 & 0 & 0 \end{bmatrix}.$$

a. Calculate the Laplacian matrix.
b. Plot the graph and describe the graph structure.
c. Compute the eigenvalue decomposition of the Laplacian matrix. What can be concluded from its eigenvalues?

5.3 (Adjacency matrix of a bipartite graph) Consider a graph described by the following adjacency matrix:

$$\boldsymbol{W} = \begin{bmatrix} 0 & 0 & 0 & 2 & 6 & 1 \\ 0 & 0 & 0 & 1 & 5 & 3 \\ 0 & 0 & 0 & 9 & 0 & 2 \\ 2 & 1 & 9 & 0 & 0 & 0 \\ 6 & 5 & 0 & 0 & 0 & 0 \\ 1 & 3 & 2 & 0 & 0 & 0 \end{bmatrix}.$$

a. Calculate the Laplacian matrix.
b. Plot the graph and describe the graph structure.
c. Compute the eigenvalue decomposition of the adjacency matrix. What can be concluded from its eigenvalues?

5.4 (Learning graphs from similarity measures) Consider the following graph:

$$\boldsymbol{W} = \begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 0 \\ 2 & 0 & 3 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 & 9 & 2 \\ 0 & 0 & 0 & 0 & 7 & 2 \\ 0 & 0 & 9 & 7 & 0 & 3 \\ 0 & 0 & 2 & 2 & 3 & 0 \end{bmatrix}.$$

a. Calculate the Laplacian matrix $\boldsymbol{L}$.

b. Generate $T = 100$ observations of a graph signal $\boldsymbol{x}^{(t)}$, $t = 1, \ldots, T$, by drawing each realization from a zero-mean Gaussian distribution with covariance matrix equal to the Moore–Penrose matrix inverse of the Laplacian matrix $\boldsymbol{L}^{\dagger}$ (which has inverse positive eigenvalues but keeps the same zero eigenvalues as $\boldsymbol{L}$), that is, $\boldsymbol{x}^{(t)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{L}^{\dagger})$.

c. Learn the following graphs based on similarity measures:

- thresholded distance graph
- Gaussian graph
- k-nearest neighbors (k-NN) graph

- feature correlation graph.

d. Compare the graphs in terms of Laplacian matrix error and with graph plots.

5.5 (Learning graphs from smooth signals) Consider the following graph:

$$\boldsymbol{W} = \begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 0 \\ 2 & 0 & 3 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 & 9 & 2 \\ 0 & 0 & 0 & 0 & 7 & 2 \\ 0 & 0 & 9 & 7 & 0 & 3 \\ 0 & 0 & 2 & 2 & 3 & 0 \end{bmatrix}.$$

a. Calculate the Laplacian matrix $\boldsymbol{L}$.

b. Generate $T = 100$ observations of a graph signal $\boldsymbol{x}^{(t)},\ t = 1, \ldots, T$, by drawing each realization from a zero-mean Gaussian distribution with covariance matrix equal to the Moore–Penrose matrix inverse of the Laplacian matrix $\boldsymbol{L}^{\dagger}$ (which has inverse positive eigenvalues but keeps the same zero eigenvalues as $\boldsymbol{L}$), that is, $\boldsymbol{x}^{(t)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{L}^{\dagger})$.

c. Learn the following graphs:

- sparse smooth graph:

$$\begin{array}{ll} \underset{\boldsymbol{W}}{\text{minimize}} & \frac{1}{2}\text{Tr}(\boldsymbol{W}\boldsymbol{Z}) + \gamma\|\boldsymbol{W}\|_{\text{F}}^2 \\ \text{subject to} & \text{diag}(\boldsymbol{W}) = \boldsymbol{0}, \quad \boldsymbol{W} = \boldsymbol{W}^{\top} \geq \boldsymbol{0}; \end{array}$$

- sparse smooth graph with hard degree control: same formulation but including the constraint $\boldsymbol{W}\boldsymbol{1} = \boldsymbol{1}$ to control the degrees of the nodes;

- sparse smooth graph with regularized degree control: same formulation again but now including the regularization term $-\boldsymbol{1}^{\top}\log(\boldsymbol{W}\boldsymbol{1})$ to control the degrees of the nodes.

d. Compare the graphs in terms of Laplacian matrix error and with graph plots.

5.6 (Learning k-component financial graphs from GRMF)

a. Download market data corresponding to N assets (e.g., stocks or cryptocurrencies) during a period with T observations, and form the data matrix $\boldsymbol{X} \in \mathbb{R}^{T \times N}$.

b. Learn a sparse GMRF graph:

$$\begin{array}{ll} \underset{\boldsymbol{L} \succeq \boldsymbol{0}}{\text{maximize}} & \log \text{gdet}(\boldsymbol{L}) - \text{Tr}(\boldsymbol{L}\boldsymbol{S}) - \rho\|\boldsymbol{L}\|_{0,\text{off}} \\ \text{subject to} & \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j. \end{array}$$

c. Learn a k-component sparse GMRF graph:

$$\begin{array}{ll} \underset{\boldsymbol{L} \succeq \boldsymbol{0}, \boldsymbol{F}}{\text{maximize}} & \log \text{gdet}(\boldsymbol{L}) - \text{Tr}(\boldsymbol{L}\boldsymbol{S}) - \rho\|\boldsymbol{L}\|_{0,\text{off}} - \gamma\text{Tr}\left(\boldsymbol{F}^{\top}\boldsymbol{L}\boldsymbol{F}\right) \\ \text{subject to} & \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j, \\ & \text{diag}(\boldsymbol{L}) = \boldsymbol{1}, \\ & \boldsymbol{F}^{\top}\boldsymbol{F} = \boldsymbol{I}. \end{array}$$

d. Plot the graphs and compare them.

5.7 (Learning heavy-tailed financial graphs)

a. Download market data corresponding to N assets (e.g., stocks or cryptocurrencies) during a period with T observations, and form the data matrix $\boldsymbol{X} \in \mathbb{R}^{T \times N}$.

b. Learn a sparse GMRF graph:

$$\begin{array}{ll} \underset{\boldsymbol{L} \succeq \boldsymbol{0}}{\text{maximize}} & \log \text{gdet}(\boldsymbol{L}) - \text{Tr}(\boldsymbol{L}\boldsymbol{S}) - \rho \|\boldsymbol{L}\|_{0,\text{off}} \\ \text{subject to} & \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j. \end{array}$$

c. Learn a heavy-tailed MRF graph by solving the following sequence of Gaussianized problems for $k = 1, 2, \ldots$:

$$\begin{array}{ll} \underset{\boldsymbol{L} \succeq \boldsymbol{0}}{\text{maximize}} & \log \text{gdet}(\boldsymbol{L}) - \text{Tr}(\boldsymbol{L}\boldsymbol{S}^k) \\ \text{subject to} & \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j, \end{array}$$

where $\boldsymbol{S}^k$ is a weighted sample covariance matrix,

$$\boldsymbol{S}^k = \frac{1}{T} \sum_{t=1}^{T} w_t^k \times \boldsymbol{x}^{(t)} (\boldsymbol{x}^{(t)})^\top,$$

with weights $w_t^k = \dfrac{p + \nu}{\nu + (\boldsymbol{x}^{(t)})^\top \boldsymbol{L}^k \boldsymbol{x}^{(t)}}$.

d. Plot the graphs and compare.

References

Banerjee, O., El Ghaoui, L., & d'Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research (JMLR)*, *9*, 485–516.

Bogdanov, A., Mossel, E., & Vadhan, S. (2008). The complexity of distinguishing Markov random fields. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques* (pp. 331–342). Springer.

Candès, E. J., Wakin, M. B., & Boyd, S. P. (2008). Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, *14*, 877–905.

Cardoso, J. V. M., & Palomar, D. P. (2020). Learning undirected graphs in financial markets. *Proceedings of the 54th Asilomar Conference on Signals, Systems and Computers*, 741–745.

Cardoso, J. V. M., & Palomar, D. P. (2022). *spectralGraphTopology: Learning Graphs From Data via Spectral Constraints* [R package]. https://CRAN.R-project.org/package=spectralGraphTopology

Cardoso, J. V. M., & Palomar, D. P. (2023a). *finbipartite: Learning Bipartite Graphs: Heavy Tails and Multiple Components* [R package]. https://cran.r-project.org/package=finbipartite

Cardoso, J. V. M., & Palomar, D. P. (2023b). *fingraph: Learning Graphs for Financial Markets* [R package]. https://CRAN.R-project.org/package=fingraph

Cardoso, J. V. M., Ying, J., & Palomar, D. P. (2021). Graphical models in heavy-tailed markets. *Proceedings of 35th International Conference on Neural Information Processing Systems (NeurIPS)*, 19989–20001.

Cardoso, J. V. M., Ying, J., & Palomar, D. P. (2022a). Nonconvex graph learning: Sparsity, heavy-tails, and clustering. In P. Diniz (Ed.), *Signal Processing and Machine Learning Theory* (pp. 1049–1072). Elsevier.

Cardoso, J. V. M., Ying, J., & Palomar, D. P. (2022b). Learning bipartite graphs: Heavy tails and multiple components. *Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS)*, 14044–14057.

Chung, F. R. K. (1997). *Spectral Graph Theory*. American Mathematical Society.

Dempster, A. P. (1972). Covariance selection. *Biometrics*, *28*(1), 157–175.

Dong, X., Thanou, D., Frossard, P., & Vandergheynst, P. (2015). Laplacian matrix learning for smooth graph signal representation. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 3736–3740.

Dong, X., Thanou, D., Rabbat, M., & Frossard, P. (2019). Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, *36*(3), 44–63.

Egilmez, H. E., Pavez, E., & Ortega, A. (2017). Graph learning from data under Laplacian and structural constraints. *IEEE Journal of Selected Topics in Signal Processing*, *11*(6), 825–841.

Fan, K. (1949). On a theorem of Weyl concerning eigenvalues of linear transformations I. *Proceedings of the National Academy of Sciences of the United States of America*, *35*(11), 652.

Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, *9*(3), 432–441.

Javaheri, A., Ying, J., Palomar, D. P., & Marvasti, F. (2025). Time-varying graph learning for data with heavy-tailed distribution. *IEEE Transactions on Signal Processing*, *73*, 3044–3060. https://doi.org/10.1109/TSP.2025.3588173

Kalofolias, V. (2016). How to learn a graph from smooth signals. *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 920–929.

Kalofolias, V., Loukas, A., Thanou, D., & Frossard, P. (2017). Learning time varying graphs. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2826–2830.

Kolaczyk, E. D. (2009). *Statistical Analysis of Network Data: Methods and Models*. Springer.

Kumar, S., Ying, J., Cardoso, J. V. M., & Palomar, D. P. (2019). Structured graph learning via Laplacian spectral constraints. *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)*, 11651–11663.

Kumar, S., Ying, J., Cardoso, J. V. M., & Palomar, D. P. (2020). A unified framework for structured graph learning via spectral constraints. *Journal of Machine Learning Research (JMLR)*, *21*(22), 1–60.

Lake, B., & Tenenbaum, J. (2010). Discovering structure by learning sparse graphs. *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*, 778–783.

Lauritzen, S. (1996). *Graphical Models*. Oxford University Press.

Mantegna, R. N. (1999). Hierarchical structure in financial markets. *The European Physical Journal B – Condensed Matter and Complex Systems*, *11*, 193–197.

Marti, G., Nielsen, F., Binkowski, M., & Donnat, P. (2021). A review of two decades of correlations, hierarchies, networks and clustering in financial markets. In F. Nielsen (Ed.), *Progress in Information Geometry* (pp. 245–274). Springer.

Mateos, G., Segarra, S., Marques, A. G., & Ribeiro, A. (2019). Connecting the dots. *IEEE Signal Processing Magazine*, *36*(3), 16–43.

Nie, F., Wang, X., Jordan, M., & Huang, H. (2016). The constrained Laplacian rank algorithm for graph-based clustering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 1969–1976.

Rue, H., & Held, L. (2005). *Gaussian Markov Random Fields: Theory And Applications*. Chapman & Hall/CRC.

Sun, Y., Babu, P., & Palomar, D. P. (2017). Majorization–minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, *65*(3), 794–816.

Ying, J., Cardoso, J. V. M., & Palomar, D. P. (2020). Nonconvex sparse graph learning under Laplacian constrained graphical model. *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)*, 7101–7113.

Zhao, L., Wang, Y., Kumar, S., & Palomar, D. P. (2019). Optimization algorithms for graph Laplacian estimation via ADMM and MM. *IEEE Transactions on Signal Processing*, *67*(16), 4231–4244.

Part II

Portfolio Optimization

6

Portfolio Basics

It is not the man who has too little, but the man who craves more, that is poor.

— Seneca, *Letters to Lucilius*

In this chapter, we introduce fundamental concepts related to portfolios, including the definition of portfolio weights and the notion of rebalancing. We also discuss common portfolio constraints and performance metrics, as well as a selection of prevalent heuristic and risk-based portfolios employed by practitioners. Examples of these portfolios include the equal-weighted $1/N$ portfolio, the quintile portfolio, and the global minimum variance portfolio.

6.1 Fundamentals

6.1.1 Data Modeling

Following Chapters 2–4 on financial data modeling, we denote the prices of a universe of N assets by $\boldsymbol{p}_t \in \mathbb{R}^N$, the *linear returns* by

$$\boldsymbol{r}_t^{\text{lin}} = \frac{\boldsymbol{p}_t - \boldsymbol{p}_{t-1}}{\boldsymbol{p}_{t-1}} = \frac{\boldsymbol{p}_t}{\boldsymbol{p}_{t-1}} - \mathbf{1},$$

where the division is elementwise (with some abuse of notation), and the *log-returns* by

$$\boldsymbol{r}_t^{\log} = \log \boldsymbol{p}_t - \log \boldsymbol{p}_{t-1},$$

where the time index t can denote any arbitrary period such as minutes, hours, days, weeks, months, quarters, years, ...

The goal of the different financial data econometric models in Chapter 3, for the i.i.d. case, and Chapter 4, for time series with temporal structure, is to form a forecast or model for the returns at time t based on the previous historical data up to time $t-1$, denoted by $\mathcal{F}_{t-1}$. This modeling is typically done in terms of the conditional first- and second-order moments, that

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

is, the conditional mean vector and covariance matrix:

$$\boldsymbol{\mu}_t^{\log} \triangleq \mathbb{E}\left[\boldsymbol{r}_t^{\log} \mid \mathcal{F}_{t-1}\right],$$
$$\boldsymbol{\Sigma}_t^{\log} \triangleq \mathrm{Cov}\left[\boldsymbol{r}_t^{\log} \mid \mathcal{F}_{t-1}\right] = \mathbb{E}\left[\left(\boldsymbol{r}_t^{\log} - \boldsymbol{\mu}_t^{\log}\right)\left(\boldsymbol{r}_t^{\log} - \boldsymbol{\mu}_t^{\log}\right)^{\mathsf{T}} \mid \mathcal{F}_{t-1}\right].$$

We remark that the majority of financial data models refer to the log-returns and not to the linear returns; for example, the random walk model for the log-prices in Chapter 3. However, to compute the return of a portfolio, from which its performance can be derived, it is the linear returns of the assets that are needed. This seems to create an impasse: we prefer to model the log-returns but what is really needed is the model of the linear returns.

In practice, since linear returns are very close to log-returns, $\boldsymbol{r}_t^{\mathrm{lin}} \approx \boldsymbol{r}_t^{\log}$, for small values of the returns (see Chapter 2), most practitioners and academics simply use the following approximation and otherwise ignore the distinction between log- and linear returns:

$$\boldsymbol{\mu}_t^{\mathrm{lin}} \approx \boldsymbol{\mu}_t^{\log} \qquad \text{and} \qquad \boldsymbol{\Sigma}_t^{\mathrm{lin}} \approx \boldsymbol{\Sigma}_t^{\log}.$$

Mathematically, using the relationship between log-returns and linear returns, $\boldsymbol{r}_t^{\log} = \log(\mathbf{1} + \boldsymbol{r}_t^{\mathrm{lin}})$, or, equivalently, $\boldsymbol{r}_t^{\mathrm{lin}} = \exp(\boldsymbol{r}_t^{\log}) - \mathbf{1}$, it follows that the mean vector and covariance matrix of the linear returns can be obtained from those of the log-returns as

$$\begin{aligned}\boldsymbol{\mu}_t^{\mathrm{lin}} &= \exp\left(\boldsymbol{\mu}_t^{\log} + \frac{1}{2}\mathrm{diag}\left(\boldsymbol{\Sigma}_t^{\log}\right)\right) - \mathbf{1},\\ \left[\boldsymbol{\Sigma}_t^{\mathrm{lin}}\right]_{ij} &= \left(\exp\left(\left[\boldsymbol{\Sigma}_t^{\log}\right]_{ij}\right) - 1\right)\exp\left(\left[\boldsymbol{\mu}_t^{\log}\right]_i + \left[\boldsymbol{\mu}_t^{\log}\right]_j + \frac{1}{2}\left(\left[\boldsymbol{\Sigma}_t^{\log}\right]_{ii} + \left[\boldsymbol{\Sigma}_t^{\log}\right]_{jj}\right)\right).\end{aligned} \tag{6.1}$$

However, due to the inherent noise in the estimation of the mean and covariance matrix, it is not clear that these more mathematically exact approximations in (6.1) give a practical advantage.

For the sake of notation, in the rest of the chapter we will drop the time dependency of the first- and second-order moments. In fact, in many cases the adopted econometric model is the i.i.d. one (see Chapter 3), which assumes no time dependency. In addition, we will refer to the linear moments by default:

$$\boldsymbol{\mu} = \boldsymbol{\mu}_t^{\mathrm{lin}} \qquad \text{and} \qquad \boldsymbol{\Sigma} = \boldsymbol{\Sigma}_t^{\mathrm{lin}}.$$

6.1.2 Portfolio Return and Net Asset Value (NAV)

A *portfolio* is simply an allocation of the available budget among N risky assets (the amount not invested is kept as cash). The most common way to define a portfolio at time t is via its *dollar allocation*[1] or *capital allocation* $\boldsymbol{w}_t^{\mathrm{cap}} \in \mathbb{R}^N$, where $w_{i,t}^{\mathrm{cap}}$ denotes the dollar or capital amount allocated to the ith asset. Typically we denote the cash as a separate scalar $c_t^{\mathrm{cap}} \in \mathbb{R}$, although it is also possible to include it as part of the portfolio vector $\boldsymbol{w}_t^{\mathrm{cap}}$ by considering it as one extra riskless asset (however, that would produce a singular covariance matrix, which

[1] The term "dollar" is an abuse of terminology and it can, of course, refer to any currency, such as the Euro, or even a cryptocurrency, such as Bitcoin.

may lead to numerical issues in the optimization part). Another way to represent a portfolio is via the number of units held for the assets $\boldsymbol{w}_t^{\text{units}} \in \mathbb{R}^N$; for example, in the case of a stock, a unit represents a share of a company and, in the case of a cryptocurrency, it represents a coin amount.

It is important to realize that if the unit amount is kept constant over time, $\boldsymbol{w}_t^{\text{units}} = \boldsymbol{w}^{\text{units}}$, then the dollar amount will change as the prices of the assets change: $\boldsymbol{w}_t^{\text{cap}} = \boldsymbol{w}^{\text{units}} \odot \boldsymbol{p}_t$.[2] To be exact, the change in the portfolio is

$$\boldsymbol{w}_t^{\text{cap}} = \boldsymbol{w}_{t-1}^{\text{cap}} \odot (\boldsymbol{p}_t \oslash \boldsymbol{p}_{t-1}) = \boldsymbol{w}_{t-1}^{\text{cap}} \odot \left(\mathbf{1} + \boldsymbol{r}_t^{\text{lin}}\right), \tag{6.2}$$

while the cash remains constant, $c_t^{\text{cap}} = c_{t-1}^{\text{cap}}$, unless there is some cash contribution or withdrawal (negative contribution), in which case it would have to be reflected as $c_t^{\text{cap}} = c_{t-1}^{\text{cap}} + \text{contribution}_{t-1}$. If a fixed dollar amount over time is desired, $\boldsymbol{w}_t^{\text{cap}} = \boldsymbol{w}^{\text{cap}}$, then the portfolio has to be regularly *rebalanced*, that is, the components of $\boldsymbol{w}_t^{\text{cap}}$ have to be adjusted (by buying or selling the assets) to bring them back to the originally designed positions, incurring in transaction costs. Figure 6.1 illustrates the change of the $1/N$ portfolio over time until a rebalance is executed.

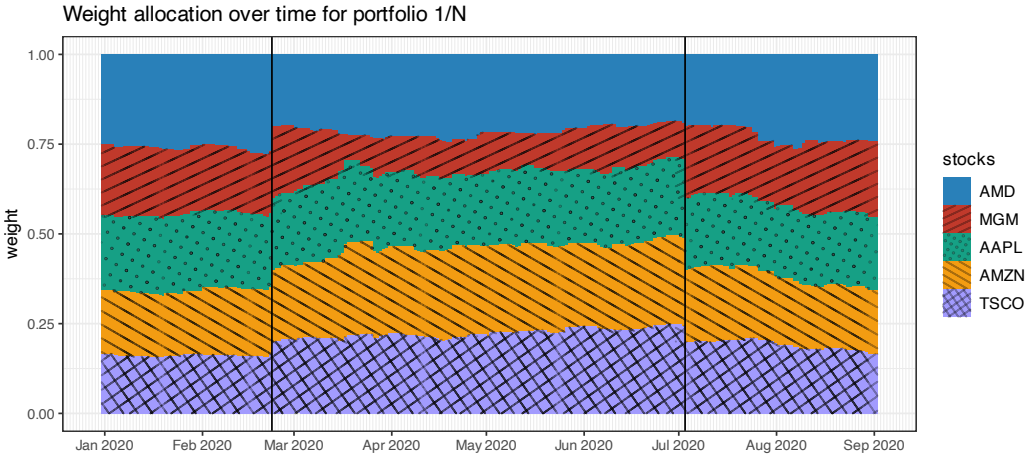


Figure 6.1 Evolution of the $1/N$ portfolio over time with effect of rebalancing (vertical lines).

NAV and Return

The portfolio *net asset value* (NAV), commonly referred to as *wealth*, is defined as the value of the portfolio at the current market valuation including cash:

$$\text{NAV}_t \triangleq \mathbf{1}^\top \boldsymbol{w}_t^{\text{cap}} + c_t^{\text{cap}}. \tag{6.3}$$

[2] For convenience of notation, we denote the elementwise product between vectors by $\odot$ and elementwise division by $\oslash$.

Plugging the portfolio time evolution (6.2) into (6.3) gives the NAV evolution

$$\begin{aligned} \mathrm{NAV}_t &= \mathbf{1}^\mathsf{T} \boldsymbol{w}_t^{\mathrm{cap}} + c_t^{\mathrm{cap}} \\ &= \mathbf{1}^\mathsf{T} \left(\boldsymbol{w}_{t-1}^{\mathrm{cap}} \odot \left(\mathbf{1} + \boldsymbol{r}_t^{\mathrm{lin}} \right) \right) + c_{t-1}^{\mathrm{cap}} \\ &= \mathrm{NAV}_{t-1} + \left(\boldsymbol{w}_{t-1}^{\mathrm{cap}} \right)^\mathsf{T} \boldsymbol{r}_t^{\mathrm{lin}}, \end{aligned} \tag{6.4}$$

which indicates that the change in NAV depends on the assets' returns and the portfolio via the term $\left(\boldsymbol{w}_{t-1}^{\mathrm{cap}} \right)^\mathsf{T} \boldsymbol{r}_t^{\mathrm{lin}}$.

The *portfolio return* is then[3]

$$R_t^{\mathrm{portf}} \triangleq \frac{\mathrm{NAV}_t - \mathrm{NAV}_{t-1}}{\mathrm{NAV}_{t-1}} = \boldsymbol{w}_{t-1}^\mathsf{T} \boldsymbol{r}_t^{\mathrm{lin}}, \tag{6.5}$$

where

$$\boldsymbol{w}_t = \boldsymbol{w}_t^{\mathrm{cap}} / \mathrm{NAV}_t \tag{6.6}$$

is the normalized portfolio with respect to the current NAV. In portfolio design or portfolio optimization, it is precisely the normalized version $\boldsymbol{w}_t$ that is obtained. The expression $\boldsymbol{w}_{t-1}^\mathsf{T} \boldsymbol{r}_t^{\mathrm{lin}}$ in (6.5) explains what is commonly referred to as the *asset additivity property* of the linear returns (as opposed to the *time additivity property* of the log-returns). Figure 6.2 shows the return and NAV of a portfolio over time.

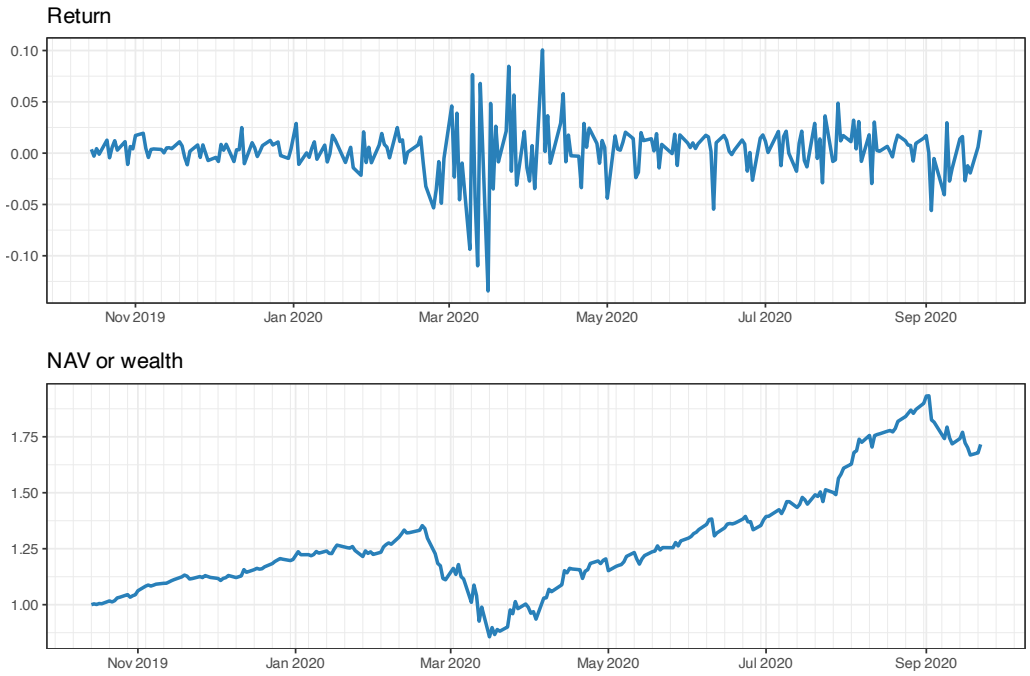


Figure 6.2 Return and NAV of the $1/N$ portfolio over time.

[3] Note that if there have been cash contributions, they have to be removed in the computation of the portfolio return: $R_t^{\mathrm{portf}} \triangleq (\mathrm{NAV}_t - \mathrm{NAV}_{t-1} - \mathrm{contribution}_{t-1}) / \mathrm{NAV}_{t-1} = \boldsymbol{w}_{t-1}^\mathsf{T} \boldsymbol{r}_t^{\mathrm{lin}}$.

Care has to be taken with the definition and implication of the time index, as illustrated in Figure 6.3. The portfolio at time t is $\boldsymbol{w}_t$, which has used information up to time $t-1$ (since that is the information used to estimate the moments $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$) and executed at time t.[4] At this point, one might be tempted to obtain the return of the portfolio by multiplying the portfolio $\boldsymbol{w}_t$ by the return at the same period $\boldsymbol{r}_t^{\text{lin}}$, but this would be incorrect as it would incur a look-ahead bias (refer to Chapter 8 for the dangers of backtesting). Since $\boldsymbol{w}_t$ is the portfolio executed and held at time t, with prices $\boldsymbol{p}_t$, its return can be computed upon observing the prices $\boldsymbol{p}_{t+1}$, that is, as $\boldsymbol{w}_t^{\mathsf{T}}\boldsymbol{r}_{t+1}^{\text{lin}}$. Note that some authors instead define the returns with a shift in the time index so that $\boldsymbol{p}_t$ can be multiplied by $\boldsymbol{r}_t^{\text{lin}}$.

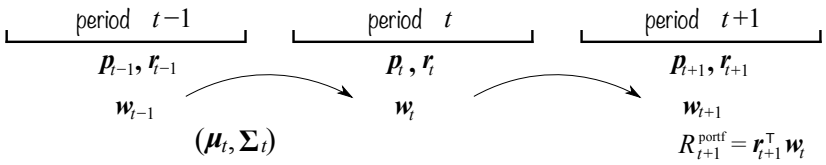


Figure 6.3 Portfolio notation over time periods.

The first- and second-order moments of the portfolio return are key quantities in portfolio optimization. For a given portfolio, the expected value and variance of the portfolio return in (6.5) are given by

$$\mathbb{E}\left[R_t^{\text{portf}}\right] = \boldsymbol{w}_{t-1}^{\mathsf{T}}\boldsymbol{\mu},$$
$$\text{Var}\left[R_t^{\text{portf}}\right] = \boldsymbol{w}_{t-1}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}_{t-1}.$$

6.1.3 Cumulative Return

The portfolio return at each given time t is a key quantity for assessing the portfolio performance. However, it is also invaluable to compute the *cumulative return* from inception to time t, also referred to as *cumulative profit and loss* (P&L).

Essentially, the cumulative return is equivalent to the NAV (or wealth), except that it is normalized and the initial NAV (or budget) is subtracted as $\text{NAV}_t/\text{NAV}_0 - 1$ (also, it does not account for contributions or redemptions related to the strategy). A plot of the cumulative return over time should start at zero, whereas a plot of the NAV or portfolio wealth starts at 1 (assuming it is normalized).

It is important to note that the cumulative return and NAV not only depend on the portfolio design at each period but also on the budget invested at each period (while some of the budget may remain as cash reserves). The amount of capital invested is often called *position size*. We now consider two extreme cases: full reinvesting and constant reinvesting.

- *Full reinvesting*: Suppose we design a normalized portfolio $\boldsymbol{w}_t$ and we fully reinvest the

[4] It could be argued that it might be possible to execute the portfolio instantaneously after observing the data at time $t-1$, although that would imply an instantaneous (or sufficiently fast) forecast, portfolio design, and market execution.

current NAV: $\boldsymbol{w}_t^{\text{cap}} = \text{NAV}_t \times \boldsymbol{w}_t$. Then, the NAV evolution in (6.4) leads to

$$\text{NAV}_t = \text{NAV}_{t-1} + \text{NAV}_{t-1} \times \boldsymbol{w}_{t-1}^{\mathsf{T}} \boldsymbol{r}_t^{\text{lin}} = \text{NAV}_{t-1} \times \left(1 + R_t^{\text{portf}}\right),$$

which corresponds to a *geometric growth*:

$$\text{NAV}_t = \text{NAV}_0 \times \left(1 + R_1^{\text{portf}}\right) \times \left(1 + R_2^{\text{portf}}\right) \times \cdots \times \left(1 + R_t^{\text{portf}}\right). \tag{6.7}$$

- *Constant reinvesting*: Now, suppose we design the same normalized portfolio $\boldsymbol{w}_t$ but we keep reinvesting the same initial NAV: $\boldsymbol{w}_t^{\text{cap}} = \text{NAV}_0 \times \boldsymbol{w}_t$. Then, the NAV evolution in (6.4) leads to

$$\text{NAV}_t = \text{NAV}_{t-1} + \text{NAV}_0 \times \boldsymbol{w}_{t-1}^{\mathsf{T}} \boldsymbol{r}_t^{\text{lin}} = \text{NAV}_{t-1} + \text{NAV}_0 \times R_t^{\text{portf}},$$

which corresponds to an *arithmetic growth*:

$$\text{NAV}_t = \text{NAV}_0 \times \left(1 + R_1^{\text{portf}} + R_2^{\text{portf}} + \cdots + R_t^{\text{portf}}\right). \tag{6.8}$$

Figure 6.4 illustrates the difference between arithmetic and geometric cumulative returns: the shape of the curves is similar except that the swings are bigger for the geometric case. As previously mentioned, these curves start at 1, so strictly speaking they refer to the normalized NAV or wealth, while the cumulative returns should be shifted down to 0. This abuse of notation is understood in the financial context and no further comment or distinction will be made.

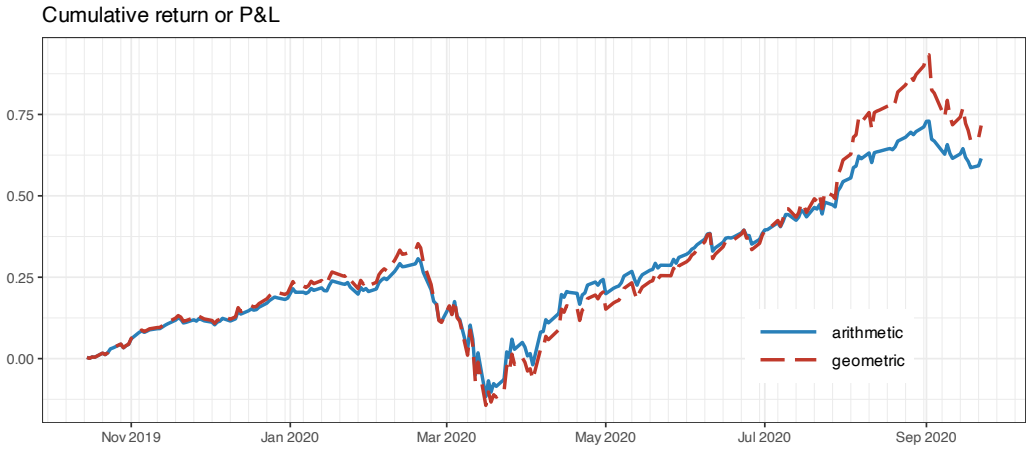


Figure 6.4 Comparison of arithmetic and geometric cumulative returns.

6.1.4 Transaction Costs

Every financial trade has an associated cost, called the *transaction cost*, that will diminish the overall return of the investment. Transaction costs consists of two terms: *commission fee* and *slippage*. The commission fee is the payment we make to our broker for completing the transaction. There is no universal scheme for commission fees: it depends on the country

and the specific broker. For example, the U.S. in general has lower commission fees than European and Asian countries. Slippage refers to the difference between the expected price of a trade and the price at which the trade is actually executed. It does not directly refer to a negative or positive movement, as any change between the expected and actual prices can qualify. In general, liquid[5] assets have smaller slippage than illiquid assets.

Suppose the current portfolio $\boldsymbol{w}_t^{\text{cap}}$ is rebalanced to $\boldsymbol{w}_t^{\text{cap,reb}}$. Then the portfolio NAV will decrease by the transaction cost, denoted by $\phi\big(\boldsymbol{w}_t^{\text{cap}} \rightarrow \boldsymbol{w}_t^{\text{cap,reb}}\big)$. The notation details of the NAV computation may differ depending on whether the transaction cost is accounted after the price change or before. Calculating the NAV after the rebalancing and the price change, equation (6.2) is modified to

$$\begin{aligned}
\text{NAV}_t &= \mathbf{1}^\top \boldsymbol{w}_t^{\text{cap}} + c_t^{\text{cap}} \\
&= \mathbf{1}^\top \left(\boldsymbol{w}_{t-1}^{\text{cap,reb}} \odot \left(\mathbf{1} + \boldsymbol{r}_t^{\text{lin}}\right)\right) + c_{t-1}^{\text{cap,reb}} - \phi\left(\boldsymbol{w}_{t-1}^{\text{cap}} \rightarrow \boldsymbol{w}_{t-1}^{\text{cap,reb}}\right) \\
&= \text{NAV}_{t-1} + \left(\boldsymbol{w}_{t-1}^{\text{cap,reb}}\right)^\top \boldsymbol{r}_t^{\text{lin}} - \phi\left(\boldsymbol{w}_{t-1}^{\text{cap}} \rightarrow \boldsymbol{w}_{t-1}^{\text{cap,reb}}\right),
\end{aligned}$$

with the portfolio return then given by

$$R_t^{\text{portf}} = \left(\boldsymbol{w}_{t-1}^{\text{reb}}\right)^\top \boldsymbol{r}_t^{\text{lin}} - \phi\left(\boldsymbol{w}_{t-1} \rightarrow \boldsymbol{w}_{t-1}^{\text{reb}}\right),$$

where the first term corresponds to the return due to the change of prices and the second term is the penalty due to transaction costs. Figure 6.5 shows the effect of transaction costs with daily rebalancing and 90 basis points (bps)[6] of fees.

The expected value and variance of the portfolio return in this case are given by

$$\begin{aligned}
\mathbb{E}\left[R_t^{\text{portf}}\right] &= \left(\boldsymbol{w}_{t-1}^{\text{reb}}\right)^\top \boldsymbol{\mu} - \phi\left(\boldsymbol{w}_{t-1} \rightarrow \boldsymbol{w}_{t-1}^{\text{reb}}\right), \\
\text{Var}\left[R_t^{\text{portf}}\right] &= \left(\boldsymbol{w}_{t-1}^{\text{reb}}\right)^\top \boldsymbol{\Sigma} \boldsymbol{w}_{t-1}^{\text{reb}}.
\end{aligned}$$

Let us focus on the form of the transaction cost function, which can be decomposed into fees and slippage:

$$\phi\left(\boldsymbol{w}_t \rightarrow \boldsymbol{w}_t^{\text{reb}}\right) = \phi^{\text{fees}}\left(\boldsymbol{w}_t \rightarrow \boldsymbol{w}_t^{\text{reb}}\right) + \phi^{\text{slippage}}\left(\boldsymbol{w}_t \rightarrow \boldsymbol{w}_t^{\text{reb}}\right).$$

The details of $\phi^{\text{fees}}(\cdot)$ depend on the specific broker, but it can be approximated as being proportional to the turnover, $\|\boldsymbol{w}_t^{\text{reb}} - \boldsymbol{w}_t\|_1 = \sum_{i=1}^N |w_{it}^{\text{reb}} - w_{it}|$, leading to

$$\phi^{\text{fees}}\left(\boldsymbol{w}_t \rightarrow \boldsymbol{w}_t^{\text{reb}}\right) \approx \tau^{\text{fee}} \|\boldsymbol{w}_t^{\text{reb}} - \boldsymbol{w}_t\|_1,$$

where the fee factor τ^{fee} is typically around $1 - -30$ bps. Keep in mind that some brokers include a minimum cost and may also have calculations based on shares instead of dollar amount.[7]

[5] Liquidity refers to the efficiency or ease with which an asset or security can be converted into ready cash without affecting its market price. The most liquid asset of all is cash itself.

[6] Basis points, denoted by bps, represents a factor of 10^{-4}. For example, 16 bps means 0.0016 or 0.16%.

[7] Fees of popular broker dealer Interactive Brokers: www.interactivebrokers.com

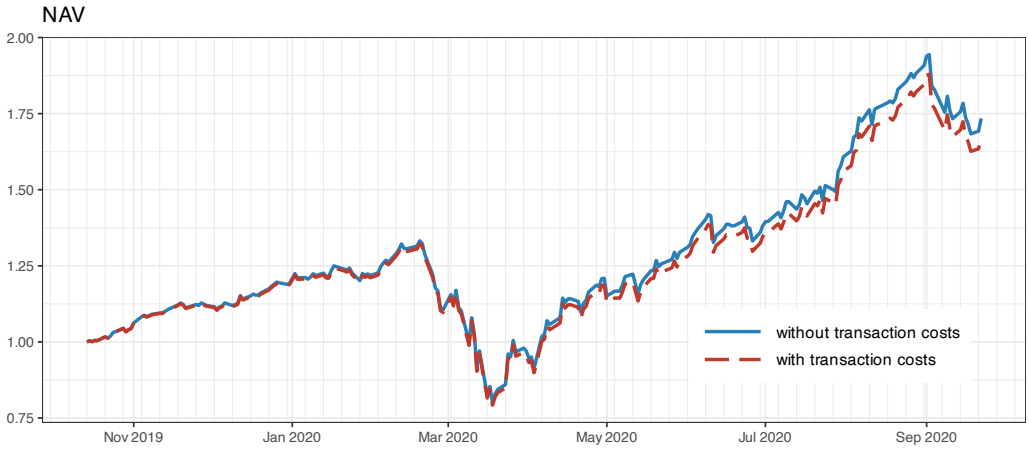


Figure 6.5 Effect of transaction costs on the NAV (with daily rebalancing and 90 bps of fees).

Let us consider the costs due to slippage, $\phi^{\text{slippage}}\left(\boldsymbol{w}_t \to \boldsymbol{w}_t^{\text{reb}}\right)$. Slippage can also be approximated as proportional to the turnover (assuming that the liquidity is enough to absorb the order executed, otherwise it can significantly increase due to market impact), but the slippage will be different for each asset:

$$\phi^{\text{slippage}}\left(\boldsymbol{w}_t \to \boldsymbol{w}_t^{\text{reb}}\right) \approx \sum_{i=1}^{N} \tau_i^{\text{slippage}} |w_{it}^{\text{reb}} - w_{it}|,$$

where τ_i^{slippage} is the asset-dependent slippage factor. In practice, the slippage factor can be estimated from the bid–ask spread of an asset. Although one can find various models to estimate slippage, a simple one is the following:

$$\tau^{\text{slippage}} = \frac{\text{half of spread}}{\text{middle of spread}} = \frac{0.5(\text{bid} - \text{ask})}{0.5(\text{bid} + \text{ask})} = \frac{\text{bid} - \text{ask}}{\text{bid} + \text{ask}}.$$

When assessing the performance of a portfolio via backtests, it is recommended to account for transaction costs to get a more realistic result, as shown in Figure 6.5. Since the transaction costs depend on the turnover, it also important to monitor the turnover of the strategy or related measures such as return on turnover.

6.1.5 Portfolio Rebalancing

Since rebalancing a portfolio entails transaction costs, there is a fundamental trade-off: we should rebalance the portfolio frequently (so as to match the currently held portfolio in the market with the desired one) but not that frequently in order to keep the transaction costs low.

The simplest way to rebalance a portfolio is with a regular calendar-based scheme, such as daily, weekly, or monthly rebalancing. There are other rebalancing schemes that decide whether to rebalance in an adaptive fashion; for example, rebalance $\boldsymbol{w}_t \to \boldsymbol{w}_t^{\text{reb}}$ only if their distance is above some small threshold, which is used to ignore unnecessary rebalancing due

to errors in the estimation and portfolio design process. Some common ways to measure the difference between portfolios include the turnover $\|\boldsymbol{w}_t^{\text{reb}} - \boldsymbol{w}_t\|_1$ or some measure of statistical difference such as the squared Mahalanobis distance $(\boldsymbol{w}_t^{\text{reb}} - \boldsymbol{w}_t)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{w}_t^{\text{reb}} - \boldsymbol{w}_t)$ (Scherer, 2002) or the tracking error squared distance $(\boldsymbol{w}_t^{\text{reb}} - \boldsymbol{w}_t)^\top \boldsymbol{\Sigma}(\boldsymbol{w}_t^{\text{reb}} - \boldsymbol{w}_t)$ widely used in asset management (Michaud & Michaud, 2008), where $\boldsymbol{\Sigma}$ denotes the covariance matrix of the assets. At the same time, the turnover $\|\boldsymbol{w}_t^{\text{reb}} - \boldsymbol{w}_t\|_1$ is typically upper-bounded or penalized in the portfolio design process to avoid frequent and large rebalancing.

The portfolio time-index notation illustrated in Figure 6.3 does not take into account the portfolio rebalancing process. To properly reflect that, the notation necessarily becomes more involved. A common way is to refine the notation by considering that each period t can be divided into the beginning of period (bop) and the end of period (eop); for example, the portfolio $\boldsymbol{w}_t$ can be more finely separated into $\boldsymbol{w}_t^{\text{bop}}$ and $\boldsymbol{w}_t^{\text{eop}}$, as shown in Figure 6.6.

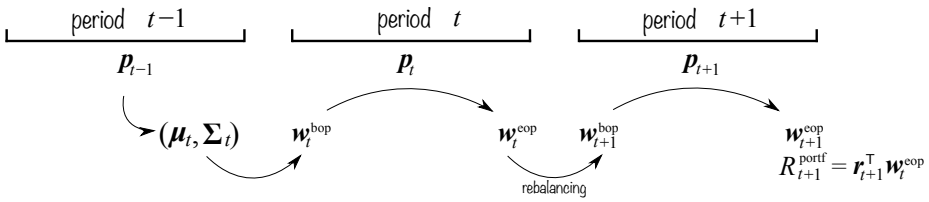


Figure 6.6 Portfolio notation over time periods with rebalancing.

According to this notation, the change from $\boldsymbol{w}_t^{\text{bop}}$ to $\boldsymbol{w}_t^{\text{eop}}$ is due to the price change as in (6.2),

$$\boldsymbol{w}_t^{\text{eop}} = \boldsymbol{w}_t^{\text{bop}} \odot \left(\mathbf{1} + \boldsymbol{r}_t^{\text{lin}}\right),$$

and the change from $\boldsymbol{w}_t^{\text{eop}}$ to $\boldsymbol{w}_{t+1}^{\text{bop}}$ is due to the rebalancing which incurs a transaction cost (if there is no rebalancing, then simply $\boldsymbol{w}_{t+1}^{\text{bop}} = \boldsymbol{w}_t^{\text{eop}}$). For example, suppose that we rebalance a fixed portfolio $\boldsymbol{w}$ at each period t and ignore the transaction cost; then, the portfolio after rebalancing is $\boldsymbol{w}_t^{\text{bop}} = \boldsymbol{w}$, after the prices change it becomes $\boldsymbol{w}_t^{\text{eop}} = \boldsymbol{w}_t^{\text{bop}} \odot \left(\mathbf{1} + \boldsymbol{r}_t^{\text{lin}}\right)$, and the portfolio return from (6.5) is

$$R_t^{\text{portf}} = \boldsymbol{w}^\top \boldsymbol{r}_t^{\text{lin}}. \tag{6.9}$$

Figure 6.7 illustrates the effect of different reoptimization and rebalancing schemes with transaction costs of 20 bps (rebalancing refers to executing the desired portfolio in the market, whereas reoptimization refers to collecting new data, estimating the mean and covariance matrix, and solving an optimization problem to design a new portfolio).

6.2 Portfolio Constraints

We now briefly describe the most commonly used constraints in portfolio design or optimization. Some constraints are imposed by the regulators or brokers (like shorting constraints, leverage constraints, and margin requirements), while others are optional depending on the investor's views (such as being market neutral or controlling the portfolio sparsity level or even enforcing diversity). For simplicity of notation, in the following we omit the time dependency

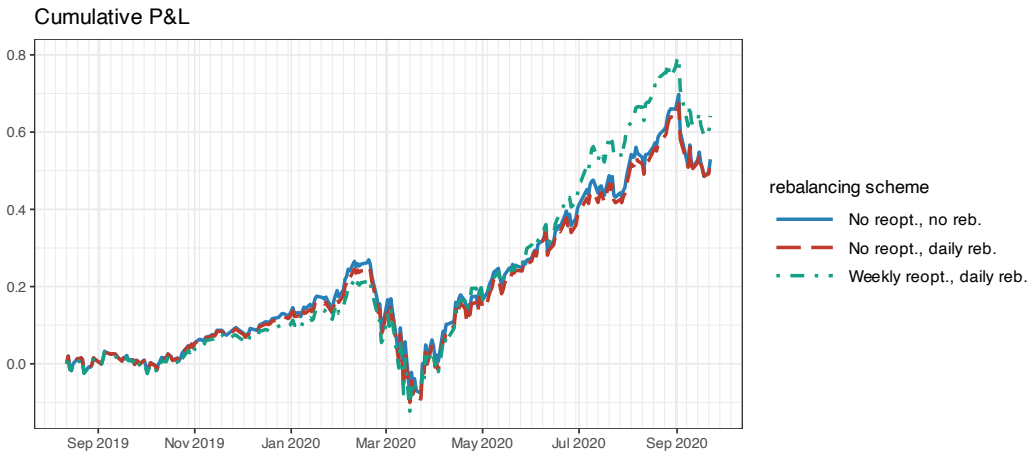


Figure 6.7 Backtest cumulative P&L of a portfolio with different reoptimization and rebalancing schemes.

and express a variety of constraints in terms of the (normalized) portfolio $\boldsymbol{w}$ defined in (6.6). It is important to recognize whether the constraints are convex, as that means that they can be efficiently handled later in the optimization process (see Appendix A).

6.2.1 Long-Only or No-Shorting Constraint

In financial markets, an investor or trader typically buys stocks taking what is called *long positions*. Interestingly, in some financial markets, a broker may allow the investor to *short sell* or *short* some stocks (i.e., to borrow some shares and sell them, with the commitment of buying them back later) for a corresponding borrowing fee. This means that the corresponding elements of $\boldsymbol{w}$ can be negative.

If shorting is not allowed, the constraint is

$$\boldsymbol{w} \geq \mathbf{0},$$

which is linear and, hence, convex.

6.2.2 Capital Budget Constraint

Assuming no shorting and no other kind of leverage, the portfolio must satisfy the budget constraint

$$\mathbf{1}^{\mathsf{T}}\boldsymbol{w} + c = 1,$$

which is linear and, hence, convex.

Alternatively, it can be rewritten without the cash variable as

$$\mathbf{1}^{\mathsf{T}}\boldsymbol{w} \leq 1.$$

If, instead, equality is imposed, the implication is that the cash is zero so that the portfolio is fully invested in the risky assets.

In practice, this constraint needs to be used in conjunction with some other constraint such as a no-shorting constraint or leverage constraint. Otherwise the positions may be unbounded resulting in unrealistic large positions.

6.2.3 Holding Constraints

Practitioners always set limits on maximum positions to avoid overexposure and ensure diversification, that is, upper bounds $\boldsymbol{u}$ on the portfolio elements. On occasions, they may also have minimum positions if they are sure they want to hold certain assets, that is, lower bounds $\boldsymbol{l}$.

Thus, holding constraints are imposed via lower and upper bounds:

$$\boldsymbol{l} \leq \boldsymbol{w} \leq \boldsymbol{u},$$

which are linear and, hence, convex.

6.2.4 Cardinality Constraint

While the universe of assets may be large (say 500 stocks), an investor typically wants to limit the number of active positions (i.e., positions with nonzero allocations) to simplify the logistics of the operation and reduce the rebalancing.

Limiting the number of active positions to K is mathematically equivalent to placing an upper bound on the cardinality of the portfolio vector:

$$\|\boldsymbol{w}\|_0 \leq K,$$

where $\|\cdot\|_0$ is the cardinality operator or ℓ_0-pseudo-norm (which counts the number of nonzero elements). This constraint is nonconvex and, therefore, difficult to handle.

6.2.5 Turnover Constraint

As discussed in Section 6.1.2, transaction costs are approximately proportional to the turnover. Therefore, it makes sense to control the turnover when designing the portfolio.

Suppose the currently held portfolio is $\boldsymbol{w}_0$. Then the turnover constraint is of the form

$$\|\boldsymbol{w} - \boldsymbol{w}_0\|_1 \leq u,$$

where u denotes the maximum turnover allowed. This constraint is convex because norms are convex functions (see Appendix A).

6.2.6 Market-Neutral constraint

According to factor modeling with the market as the single factor, the returns can be expressed as $\boldsymbol{r}_t = \boldsymbol{\alpha} + \boldsymbol{\beta} \cdot r_t^{\text{mkt}} + \boldsymbol{\epsilon}_t$ (see Chapter 3 for details).

Portfolio managers typically want to avoid exposure to the market, which means that they want the portfolio orthogonal to the "beta":

$$\boldsymbol{\beta}^\mathsf{T} \boldsymbol{w} = \mathbf{0},$$

which is linear and, hence, convex.

6.2.7 Dollar-Neutral Constraint

When shorting is allowed, one may want to balance the long and short positions in what is called a dollar-neutral position:

$$\mathbf{1}^\mathsf{T} \boldsymbol{w} = \mathbf{0}.$$

In practice, the dollar-neutral constraint should be used in conjunction with some leverage constraint. Some academic papers and capital asset pricing models assume, in effect, that one can sell a security short without limit and use the proceeds to buy securities long. This is a mathematically convenient assumption for hypothetical models of the economy, but it is unrealistic (Jacobs et al., 2005).

6.2.8 Diversification Constraint

Some portfolio designs tend to concentrate the allocation in a few assets. The quantity $\|\boldsymbol{w}\|_2^2 = \boldsymbol{w}^\mathsf{T}\boldsymbol{w} = \sum_{i=1}^N w_i^2$ can be used as a diversification measure (DeMiguel, Garlappi, Nogales, & Uppal, 2009; Goetzmann & Kumar, 2008). In fact, the quantity $\|\boldsymbol{w}\|_2^2$ is the Herfindahl index of the weights of the portfolio. A lower value is indicative of a higher level of diversification; it is lower-bounded by $1/N$ and upper-bounded by 1. Therefore, one can promote diversification with the constraint:

$$\|\boldsymbol{w}\|_2^2 \leq D,$$

where $1/N \leq D < 1$. This constraint is convex because norms are convex functions (see Appendix A).

6.2.9 Leverage Constraint

Regulations in countries typically limit the amount that can be borrowed for long and short positions. For example, in the U.S., the Federal Reserve System established Regulation T (Reg T) that limits the borrowing to be no greater than 50% of the securities purchase price. The 50% requirement is called the initial margin, but certain brokers may have stricter requirements.

The total amount of long and short positions can be measured with the ℓ_1-norm $\|\boldsymbol{w}\|_1 =$

$\sum_{i=1}^{N} |w_i|$ and the leverage constraint can be written mathematically as

$$\|\boldsymbol{w}\|_1 \leq 2,$$

where the factor of 2 indicates that 50% is being borrowed.

6.2.10 Margin Requirements

When borrowing is involved, apart from the leverage constraints imposed by regulators such as Reg T, brokers will impose margin requirements to reserve collateral[8] and ensure that the borrower can pay back the loan later.

The capital budget constraint $\mathbf{1}^\top \boldsymbol{w} \leq 1$ cannot be used in isolation when borrowing is involved since it would imply that an investor can use all the cash from short-selling to buy stocks, which is not the case in real life (Jacobs et al., 2005).

The broker will impose margin requirements in both long and short positions. For a long position, the broker may allow borrowing, say, 50% of the value of the position. For short positions, by definition the investor is borrowing stocks; the cash from short-selling is controlled by the broker as cash collateral but one is also required to have some equity as the "initial margin" to establish the positions; usually the overall collateral is about 105%.

It is convenient to separate the positions of the portfolio into longs and shorts:

$$\boldsymbol{w} = \boldsymbol{w}^+ - \boldsymbol{w}^-,$$

where $\boldsymbol{w}^+ \geq \mathbf{0}$ denotes the longs and $\boldsymbol{w}^- \geq \mathbf{0}$ the shorts. The margin requirement can then be expressed as

$$m^{\text{long}} \times \mathbf{1}^\top \boldsymbol{w}^+ + m^{\text{short}} \times \mathbf{1}^\top \boldsymbol{w}^- \leq 1,$$

where m^{long} and m^{short} control how much margin is required for longs and shorts, respectively. For example, a broker may impose $m^{\text{long}} = 0.5$ and $m^{\text{short}} = 0.05$.

6.3 Performance Measures

There are many ways to assess the performance of a portfolio. The key quantity in most of the performance measures is the portfolio return over time (6.5), which for the case of a fixed portfolio $\boldsymbol{w}$ can be written as in (6.9): $R_t^{\text{portf}} = \boldsymbol{w}^\top \boldsymbol{r}_t$ (for the sake of notation we use $\boldsymbol{r}_t$ to denote the linear returns $\boldsymbol{r}_t^{\text{lin}}$). We now list the most commonly used performance measures; for a detailed and extensive treatment, see Bacon (2008).

6.3.1 Expected Return

The expected return of the portfolio is simply given by

$$\mathbb{E}\left[R_t^{\text{portf}}\right] = \boldsymbol{w}^\top \boldsymbol{\mu},$$

[8] The term collateral refers to an asset that a lender accepts as security for a loan. The collateral acts as a form of protection for the lender. That is, if the borrower defaults on their loan payments, the lender can seize the collateral and sell it to recoup some or all of its losses.

which is actually the expected return over one period t. This measure quantifies the average benefit of the investment but ignores the risk.

In most cases, however, it is the *annualized return* that is reported. This simply requires scaling with a factor equal to the number of periods over a year, γ, leading to

$$\text{Annualized Return} = \gamma \times \boldsymbol{w}^\top \boldsymbol{\mu}.$$

Suppose the periods are days (i.e., daily prices); if we trade on the stock market then $\gamma = 252$ is typically the number of tradable days per year (excluding weekends and holidays, which depend on the country), whereas on cryptocurrency markets $\gamma = 365$ since they operate nonstop. Suppose now that the periods are hours (i.e., hourly prices); then the scaling factor in cryptocurrency markets would be $\gamma = 365 \times 24 = 8760$ (in stock markets it really depends on the country, since the hours per day differ and range from 5 to 9).

Given a sequence of T portfolio returns, the annualized return can be computed via the sample mean,

$$\frac{\gamma}{T} \sum_{t=1}^{T} R_t^{\text{portf}},$$

which implicitly assumes the use of arithmetic chaining to aggregate the returns. If, instead, one uses a geometric chaining to take into account compounding, the geometric mean should be used:

$$\left(\prod_{t=1}^{T} \left(1 + R_t^{\text{portf}} \right) \right)^{\gamma/T} - 1,$$

which is also termed *compound annual growth rate* (CAGR). Note that the geometric mean is always smaller than the arithmetic mean (i.e., with $\gamma = 1$); however, once the annualization with the factor γ is included, nothing can be said anymore. For example, an arbitrary sequence of returns has arithmetic mean 0.0047 and geometric mean 0.0040, but after annualizing with $\gamma = 100$ they become 0.47 and 0.49, respectively.

6.3.2 Volatility

The volatility is the standard deviation of the portfolio return:

$$\text{Std}\left[R_t^{\text{portf}} \right] = \sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}.$$

It is the simplest measure of risk of a portfolio. For convenience, one may also use the square of the volatility, that is, the variance of the portfolio return, as used by Markowitz in his seminal paper (Markowitz, 1952).

To obtain the annualized volatility we need to multiply by the square root of the number of periods per year γ:

$$\text{Annualized Volatility} = \sqrt{\gamma} \times \sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}.$$

6.3.3 Volatility-Adjusted Returns

Comparing portfolios by looking at the cumulative returns may be misleading since each portfolio may have a different volatility, since it is expected that a higher-volatility portfolio will also have a higher cumulative return (assuming a bull market). To get rid of this volatility artifact when comparing portfolios, some practitioners prefer to use the volatility-adjusted returns defined as

$$\bar{R}_t^{\text{portf}} = \boldsymbol{w}^\top \boldsymbol{r}_t \times \frac{\sigma^{\text{target}}}{\sigma_t},$$

where σ_t denotes the volatility of $\boldsymbol{w}^\top \boldsymbol{r}_t$ and σ^{target} is the desired target volatility for the portfolio.

Figure 6.8 compares the cumulative returns of two stocks (AMD and AMZN) with and without volatility adjustment. One can observe how from the non-adjusted version AMD may look better than AMZN, but this is misleading because it comes with much higher volatility. From the volatility-adjusted version, instead, it seems that AMZN has a better trade-off of final return and volatility. In fact, this trade-off is precisely captured by the Sharpe ratio described next.

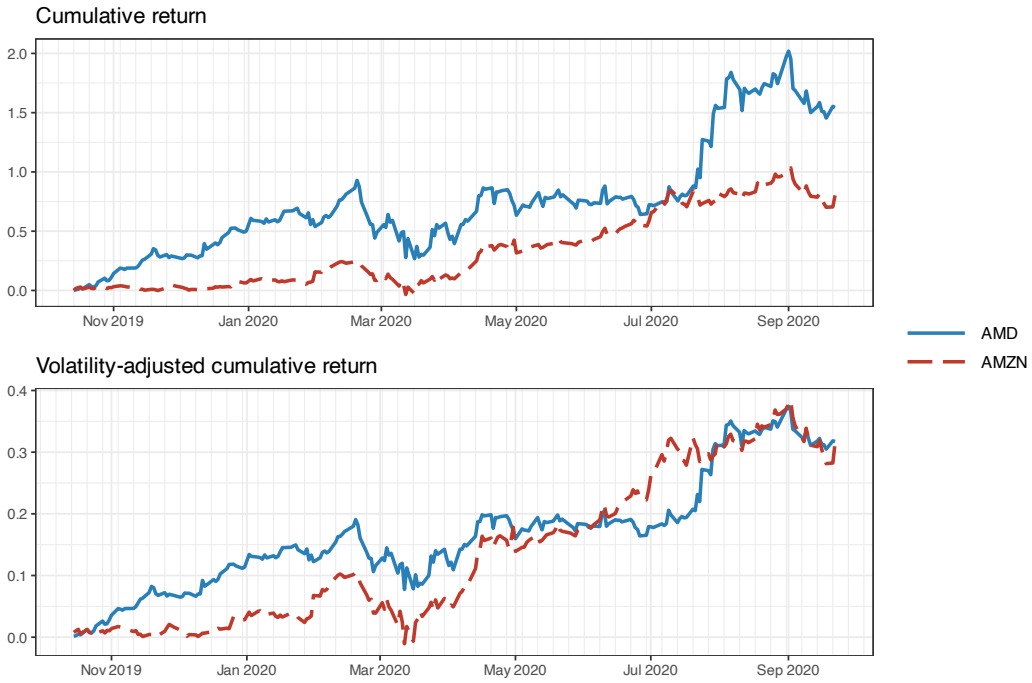


Figure 6.8 Cumulative returns and volatility-adjusted cumulative returns.

6.3.4 Sharpe Ratio (SR)

The *Sharpe ratio* (SR) is the risk-adjusted expected excess return,

$$\text{SR} = \frac{\boldsymbol{w}^\top \boldsymbol{\mu} - r_\text{f}}{\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}},$$

where r_f is the risk-free rate (e.g., the interest rate on a three-month U.S. Treasury bill). It was proposed by William Sharpe in 1966 (Sharpe, 1966); see also Sharpe (1994).

Observe that the numerator is the excess return with respect to the risk-free rate; in practice, one may assume $r_\text{f} \approx 0$ and just use the expected return. Dividing by the risk gives the risk-adjusted return, that is, the return per unit of risk.

The annualized SR is

$$\text{Annualized SR} = \sqrt{\gamma} \times \frac{\boldsymbol{w}^\top \boldsymbol{\mu} - r_\text{f}}{\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}},$$

where γ denotes the number of periods over a year as introduced in Section 6.3.1.

For long periods of time, rather than simply computing the overall SR, it may be useful to compute the *rolling SR*, which is the time series of the SR over time computed on a rolling-window basis. Figure 6.9 shows the cumulative return and rolling annualized Sharpe ratio (with a rolling window of one month) of the $1/N$ portfolio with average Sharpe ratio of 1.65 over a universe of five stocks.

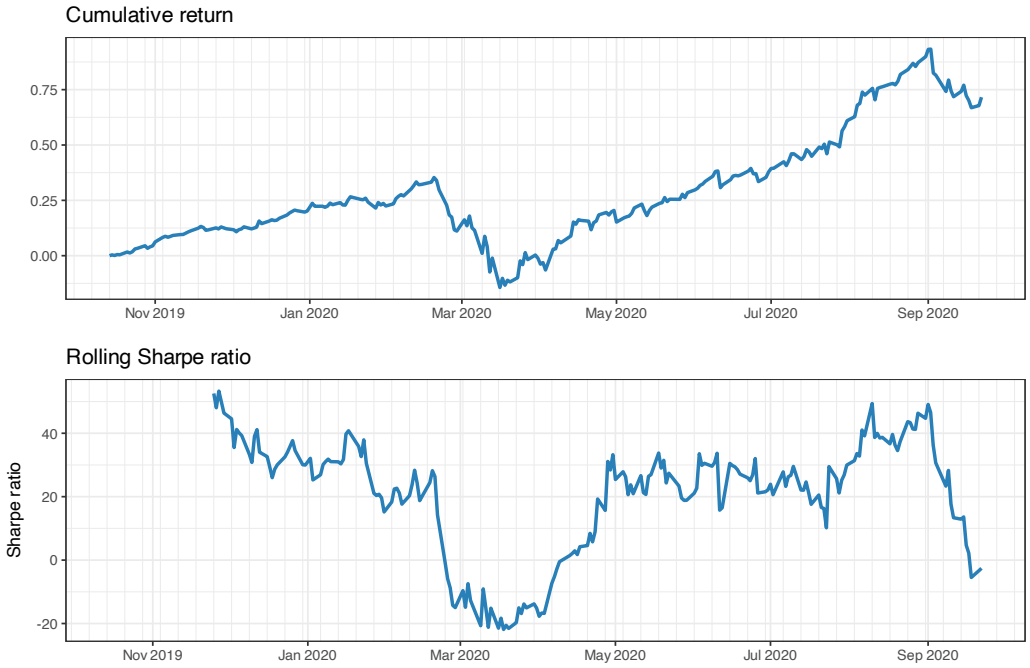


Figure 6.9 Cumulative returns and rolling annualized Sharpe ratio.

Finally, it is worth mentioning that it is common among practitioners to use the compounded

annualized return in the numerator of the Sharpe ratio (i.e., using the geometric mean as previously mentioned in Section 6.3.1), often referred to as the *geometric Sharpe ratio*.

6.3.5 Information Ratio (IR)

The Sharpe ratio uses the excess return in the numerator, that is, the portfolio return is compared or benchmarked against the return of the risk-free asset. The *information ratio* (IR) instead uses an arbitrary benchmark, for example the market index:

$$\mathrm{IR} = \frac{\mathbb{E}\left[\boldsymbol{w}^\top \boldsymbol{r}_t - r_t^{\mathrm{b}}\right]}{\mathrm{Std}\left[\boldsymbol{w}^\top \boldsymbol{r}_t - r_t^{\mathrm{b}}\right]},$$

where r_t^{b} is the return of the benchmark.

6.3.6 Downside Risk and Semi-Variance

The variance of the portfolio (the square of the volatility),

$$\mathrm{Var}\left[\boldsymbol{w}^\top \boldsymbol{r}_t\right] = \mathbb{E}\left[\left(\boldsymbol{w}^\top \boldsymbol{r}_t - \boldsymbol{w}^\top \boldsymbol{\mu}\right)^2\right],$$

is widely used as a proxy to measure the risk since it measures the deviation from the mean return. However, whereas the portfolio return underperforming the mean is clearly an undesired event, one can argue that the portfolio overperforming the mean should actually be welcomed and not counted as risk, as the variance does.

The *downside risk* precisely measures the risk of the portfolio underperforming a desired reference level. One example of a downside risk measure is the *semi-variance*, already considered by Markowitz (1959):

$$\mathrm{SemiVar}\left[\boldsymbol{w}^\top \boldsymbol{r}_t\right] = \mathbb{E}\left[\left(\left(\boldsymbol{w}^\top \boldsymbol{\mu} - \boldsymbol{w}^\top \boldsymbol{r}_t\right)^+\right)^2\right],$$

where the operator $(\cdot)^+ \triangleq \max(0, \cdot)$ precisely avoids penalizing when the return is above the mean. The square root of the semi-variance, called *semi-deviation* or *downside deviation*, plays a role equivalent to the volatility.

There are other downside risk measures, for example the more general lower partial moment (LPM),

$$\mathrm{LPM}_\alpha(\tau) = \mathbb{E}\left[\left(\left(\tau - \boldsymbol{w}^\top \boldsymbol{r}_t\right)^+\right)^\alpha\right],$$

where τ is termed the disaster level and α reflects the investor's feeling about the relative consequences of falling short of τ by various amounts (namely, $\alpha > 1$ for risk-averse investors, $\alpha = 1$ for a neutral investor, and $0 < \alpha < 1$ for risk-seeking investors). By properly choosing the parameters α and τ most downside measures used in practice can be formed (for example, setting $\alpha = 2$ and $\tau = \boldsymbol{w}^\top \boldsymbol{\mu}$ yields the semi-variance).

6.3.7 Gain–Loss Ratio (GLR)

The *gain–loss ratio* (GLR) is actually a downside risk measure that represents the relative relationship of trades with a positive return and trades with a negative return:

$$\mathrm{GLR} = \frac{\mathbb{E}\left[R_t^{\mathrm{portf}} \mid R_t^{\mathrm{portf}} > 0\right]}{\mathbb{E}\left[R_t^{\mathrm{portf}} \mid R_t^{\mathrm{portf}} < 0\right]}.$$

6.3.8 Sortino Ratio

The *Sortino ratio* is defined similarly to the Sharpe ratio but replacing the variance in the denominator by the semi-variance:

$$\mathrm{SoR} = \frac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} - r_{\mathrm{f}}}{\sqrt{\mathrm{SemiVar}\left[\boldsymbol{w}^{\mathsf{T}}\boldsymbol{r}_t\right]}}.$$

6.3.9 Value-at-Risk (VaR)

The portfolio variance (similarly the volatility) measures the deviation from the mean value as the risk. The semi-variance (similarly the semi-deviation) improves on this measure by only considering the deviations below the mean return. However, the key quantity in measuring risk is in the tail of the distribution of the random returns, simply because that is when the investor loses large amounts and can go bankrupt. Figure 6.10 illustrates the meaning of the most popular measures of risk in the context of the distribution of the portfolio returns.

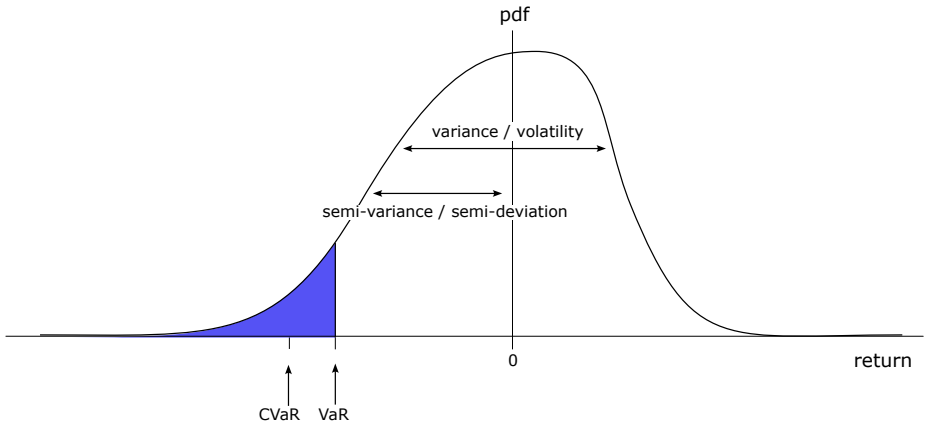


Figure 6.10 Illustration of distribution of returns and measures of risk.

To overcome the drawbacks of variance and semi-variance, another popular single side risk measurement is the *value-at-risk* (VaR). This measure emerged as a distinct concept in the late 1980s, but it was not until 1994 that its development was extensive at J. P. Morgan, which published the methodology and gave free access to estimates of the necessary underlying parameters. In words, the VaR measures the maximum loss with a specified confidence level, such as 95%; more succinctly, it can be described as the quantile of the loss.

Denoting by the random variable ξ the loss of a portfolio over a period of time (i.e., the negative return $\xi_t = -\boldsymbol{w}^\top \boldsymbol{r}_t$), the VaR is defined as

$$\mathrm{VaR}_\alpha = \inf \{\xi_0 \mid \Pr(\xi \leq \xi_0) \geq \alpha\}$$

with α the confidence level, say, $\alpha = 0.95$ for a 95% confidence level (meaning that in 95% of the cases the loss will be smaller than the VaR and only in 5% will the loss be larger; see Figure 6.10 in terms of returns rather than losses.

The drawback of this risk measure is that it does not quantify the shape of the tail. In plain words, it can tell you how often your portfolio losses will exceed $1 million, but not how much you could lose.

A second major drawback of the VaR is that the measure is not subadditive, which is critical for diversification. Subadditivity means that merging two portfolios cannot increase the risk, but VaR violates this principle.

6.3.10 Conditional Value-at-Risk (CVaR)

The *conditional value-at-risk* (CVaR), also known as *expected shortfall* (ES), is defined as the expected value of the loss tail:

$$\mathrm{CVaR}_\alpha = \mathbb{E}[\xi \mid \xi \geq \mathrm{VaR}_\alpha].$$

Thus, the CVaR improves upon the VaR by taking into account the shape of the tail (losses exceeding the VaR) via its average value, as illustrated in Figure 6.10 in terms of returns rather than losses.

In plain words, the CVaR can now answer not only how often your portfolio losses will exceed $1 million, but also how much you would lose on average.

In addition, the CVaR is subadditive, which means that the measure satisfies what is expected from diversification by merging portfolios.

6.3.11 Drawdown

The *drawdown* (DD) measures the decline from a historical peak of the cumulative profit or NAV (assuming no cash contributions/redemptions):

$$D_t = \mathrm{HWM}_t - \mathrm{NAV}_t,$$

where HWM_t is the *high-water mark* of the cumulative profit, defined as

$$\mathrm{HWM}_t = \max_{\tau \in [0,t]} \mathrm{NAV}_\tau.$$

Similarly, we can define the normalized drawdown as

$$\bar{D}_t = \frac{\mathrm{HWM}_t - \mathrm{NAV}_t}{\mathrm{HWM}_t}.$$

Figure 6.11 illustrates the drawdown corresponding to a cumulative P&L curve.

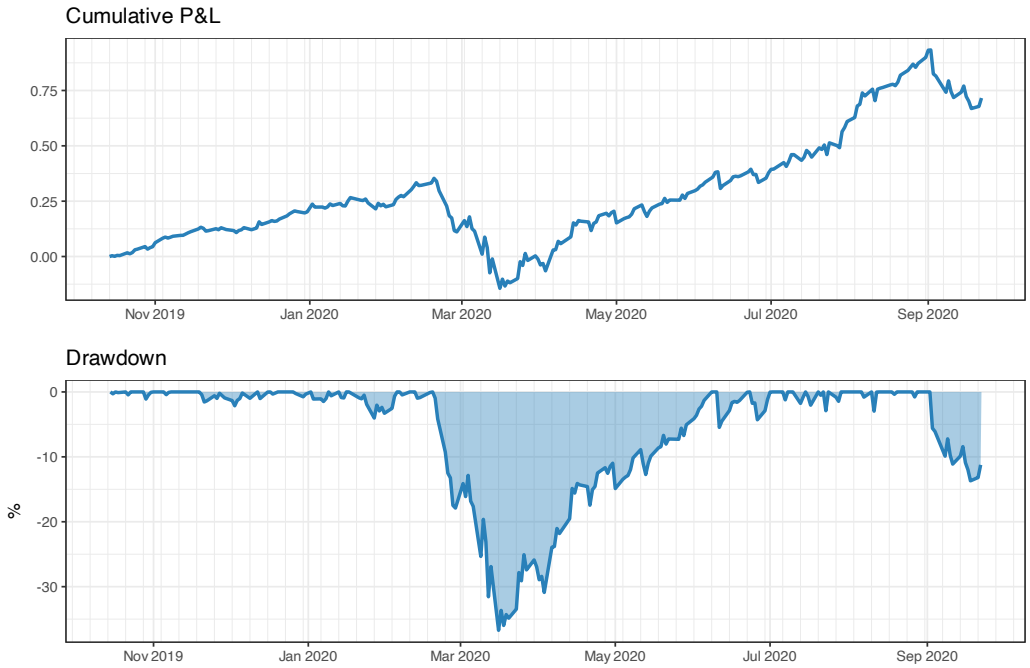


Figure 6.11 Cumulative P&L and corresponding drawdown of a portfolio.

It is interesting to notice that the drawdown is a *path-dependent measure*, as opposed to all the previously considered measures of performance. This means that it depends on the temporal order in which the returns $\boldsymbol{r}_t$ happen. This is in sharp contrast to the previously considered measures which are agnostic to the ordering of the returns.

In practice, it is useful to condense the entire drawdown curve for a given period $t = 1, \ldots, T$ into a single numerical value. This can be done in different ways, namely:

- *maximum drawdown* (Max-DD):

$$\text{Max-DD} = \max_{1 \leq t \leq T} \bar{D}_t;$$

- *average drawdown* (Ave-DD):

$$\text{Ave-DD} = \frac{1}{T} \sum_{1 \leq t \leq T} \bar{D}_t;$$

- *conditional drawdown at risk* (CDaR) (defined similarly to the CVaR):

$$\text{CDaR}_\alpha = \mathbb{E}\left[\bar{D}_t \mid \bar{D}_t \geq \text{VaR}_\alpha\right],$$

  where VaR_α is the value at risk of the drawdown $\bar{D}_t$ with confidence level α.

6.3.12 Calmar Ratio and Sterling Ratio

The *Calmar ratio* modifies the Sharpe ratio by replacing the variance in the denominator with the maximum drawdown:

$$\text{Calmar ratio} = \frac{\boldsymbol{w}^\top \boldsymbol{\mu} - r_\text{f}}{\text{Max-DD}}.$$

The *Sterling ratio* is similar to the Calmar ratio except that it adds an excess risk measure to the maximum drawdown (typically of 10%):

$$\text{Sterling ratio} = \frac{\boldsymbol{w}^\top \boldsymbol{\mu} - r_\text{f}}{\text{Max-DD} + 0.1}.$$

6.4 Heuristic Portfolios

We begin our exploration of portfolios with some *heuristic* portfolios that may not be formally derived in a mathematically sound manner, but have proven to be widely used by practitioners and can serve as benchmarks.

6.4.1 Buy and Hold Portfolio

One of the simplest investment strategies consists of selecting just one asset and allocating the whole budget to it:

$$\boldsymbol{w} = \boldsymbol{e}_i, \tag{6.10}$$

where $\boldsymbol{e}_i$ denotes an all-zero vector with a one at the ith element. This is called single-asset *buy and hold* (B&H), which lacks diversification. More generally, the B&H portfolio can be composed of several assets or ETFs, but always with a long horizon so that the portfolio is not adjusted over short horizons.

The belief behind such a strategy is that the asset will increase gradually in value over the investment period. The challenge, naturally, lies in determining which asset to invest in. One can use different methods, like fundamental analysis or technical analysis, to make the choice.

6.4.2 Global Maximum Return Portfolio (GMRP)

The B&H portfolio allocates all the budget to a single asset chosen in a discretionary (e.g., based on expertise, opinion, or technical analysis) or mathematical (e.g., based on fundamental analysis or statistical analysis) manner. Interestingly, if the choice of the top-performing asset is based on the assets' return forecast $\boldsymbol{\mu}$, then it can be formally derived as the optimal portfolio of a poorly chosen cost function, as shown next.

Mathematically, the *global maximum return portfolio* (GMRP) is formulated as

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{maximize}} \quad & \boldsymbol{w}^\top \boldsymbol{\mu} \\ \text{subject to} \quad & \mathbf{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}. \end{aligned} \tag{6.11}$$

This problem is convex (in fact, a linear program (LP)) and can be optimally solved with an

LP solver. In this case, the solution is trivial: allocate all the budget to the asset with largest return.

As a word of caution, one should not to be fooled by the sound and formal derivation of this portfolio, because it lacks diversification and it will perform poorly. Note that past performance is not a guarantee of future performance, which is related to the extremely noisy estimation of $\boldsymbol{\mu}$ in practice (Chopra & Ziemba, 1993).

6.4.3 $1/N$ Portfolio

One of the most important goals of quantitative portfolio management is to diversify the investment across different assets in a portfolio, typically summarized as "do not put all your eggs in the same basket." The simplest way to achieve diversification is by allocating the capital equally across all the N assets:

$$\boldsymbol{w} = \frac{1}{N} \times \mathbf{1}, \tag{6.12}$$

where $\mathbf{1} \in \mathbb{R}^N$ denotes the all-one vector. This strategy is called the *equally weighted portfolio* (EWP) or $1/N$ *portfolio*.

Historically, this strategy or philosophy has been called the "Talmudic rule" (Duchin & Levy, 2009) since the Babylonian Talmud recommended it approximately 1500 years ago: "A man should always place his money, one third in land, a third in merchandise, and keep a third in hand."

It has gained much interest due to its superior historical performance and the emergence of several equally weighted ETFs (DeMiguel, Garlappi, & Uppal, 2009). For example, Standard & Poor has developed many S&P 500 equally weighted indices. In addition, the $1/N$ portfolio has been claimed to outperform Markowitz's mean–variance portfolio considered later in Section 7.1 (DeMiguel, Garlappi, & Uppal, 2009), although other studies disagree with these empirical results (Kritzman et al., 2010).

$1/N$ Portfolio as an Optimal Robust Portfolio

The $1/N$ portfolio has been introduced as a common-sense heuristic portfolio for diversification. Interestingly, it was recently shown to correspond to a formally derived optimal portfolio robust to estimation errors in $\boldsymbol{\mu}$ (Zhou & Palomar, 2020). The reader is referred to the robust formulation in (6.14) for more details and to Chapter 14 for an extensive treatment of robust portfolios.

6.4.4 Quintile Portfolio

The *quintile portfolio* is widely used by practitioners. It is similar to the $1/N$ portfolio in that it allocates equally to the invested assets; however, it only uses the top-performing assets out of the universe of N assets. In particular, the quintile portfolio selects the top fifth (or 20%) of the assets; however, one could select a different fraction, such as in the quartile portfolio (one fourth or 25% of the assets) or the decile portfolio (one tenth or 10% of the assets). If

shorting is desired, then the bottom part of the assets can be short-sold, termed a long–short quintile portfolio.

Suppose the assets have been properly reordered from top to bottom performers (and that N is divisible by five for the sake of notation). Then, the quintile portfolio can be expressed as

$$\boldsymbol{w} = \frac{1}{N/5} \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} 20\% \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} 80\% \end{matrix} . \tag{6.13}$$

The crux of the quintile portfolio lies in the ranking of the assets. One can rank the stocks in a multitude of ways, typically based on expensive factors that investment funds purchase at a premium price. It is also called *factor investing* as it mimics the returns of some common factors (Jurczenko, 2017).

Factor models, such as the capital asset pricing model (CAPM) (Sharpe, 1964) and Fama–French model (Fama & French, 1993), were initially introduced to explain stock returns. Some of the factors used in these models are calculated as the excess returns of assets with attractive characteristics. For example, five well-known factors, namely, value, low size, low volatility, momentum, and quality, may be obtained by ranking the assets according to book-to-price ratio, low market capitalization, low standard deviation, return, and return on equity, respectively (Bender et al., 2013). Empirical studies show that these factors have exhibited excess returns above the market. The quintile portfolio based on the momentum measured by the estimated return in the past few months is one of the most famous ones. A study found that about 77% of 155 mutual funds were actually using such a portfolio over the 1975–1984 period (Grinblatt et al., 1995). Interestingly, some investors might prefer the quintile portfolio based on the opposite of short-term return because they believe in short-term reversals. It has been a widely debated mystery how these simple portfolios defeat the more theoretical-based portfolios.

In the simplest case, if one only has access to publicly available price data, then some common possible ways to rank the assets are according to the following readily available scores:

- expected performance:

$$\text{score} = \boldsymbol{\mu};$$

- Sharpe ratios:

$$\text{score} = \frac{\boldsymbol{\mu}}{\sqrt{\text{diag}\,(\boldsymbol{\Sigma})}};$$

- mean-variance ratios:

$$\text{score} = \frac{\boldsymbol{\mu}}{\text{diag}\,(\boldsymbol{\Sigma})}.$$

Quintile Portfolio as an Optimal Robust Portfolio

The quintile portfolio has been introduced as a common-sense heuristic portfolio. However, it was recently shown to correspond to a formally derived optimal portfolio robust to estimation errors in $\boldsymbol{\mu}$ (Zhou & Palomar, 2020), as briefly described. The reader is referred to Chapter 14 for an extensive treatment of robust portfolios.

Consider a robust formulation of the GMRP in (6.11):

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \min\limits_{\boldsymbol{\mu}\in\mathcal{S}} \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} \\ \text{subject to} & \mathbf{1}^\mathsf{T}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array} \tag{6.14}$$

where the objective function is the worst-case return among the possible return vectors $\boldsymbol{\mu} \in \mathcal{S}$, and the uncertainty set $\mathcal{S}$ is defined as an ℓ_1-norm ball around the estimated value of the returns $\hat{\boldsymbol{\mu}}$,

$$\mathcal{S} = \{\hat{\boldsymbol{\mu}} + \boldsymbol{e} \mid \|\boldsymbol{e}\|_1 \leq \epsilon\},$$

with ϵ being the size of the uncertainty region.

It turns out that for an appropriate choice of ϵ, the solution to the robust formulation in (6.14) is precisely the quintile portfolio in (6.13). Furthermore, the $1/N$ portfolio in (6.12) can also be obtained as the solution to the robust formulation in (6.14) for a sufficiently large value of ϵ. This theoretical result helps explain the good performance exhibited by the $1/N$ portfolio and the quintile portfolio in practice, as they are indeed robust portfolios.

6.4.5 Numerical Experiments

We now compare the heuristic GMRP, $1/N$ portfolio, and quintile portfolio based on different ranking of the stocks, namely, μ, μ/σ, and μ/σ^2. We consider eight stocks from the S&P 500 during the period 2019–2020, and perform a simple backtest using 70% of the observations as training data while keeping 30% as test data (i.e., out of sample). More realistically, one should do multiple rolling-window backtests (see Chapter 8 for an extensive treatment of backtesting).

Figure 6.12 shows the normalized dollar allocation of the portfolios, where one can observe the difference in diversification. Figure 6.13 shows the out-of-sample backtest results in terms of cumulative P&L and drawdown. While it may initially look from the P&L that the GMRP performs the best, this is deceiving due to the increase in volatility, as can be observed clearly in the drawdown. The $1/N$ portfolio is the most diversified one and the best in terms of drawdown in this particularly backtest.

6.5 Risk-Based Portfolios

We continue our exploration of portfolios with several *risk-based portfolios* that are solely based on risk considerations, characterized by the covariance matrix $\boldsymbol{\Sigma}$ (Ardia et al., 2017), while totally ignoring the return forecast $\boldsymbol{\mu}$, due to the extremely noisy estimation of $\boldsymbol{\mu}$ in practice (Chopra & Ziemba, 1993).

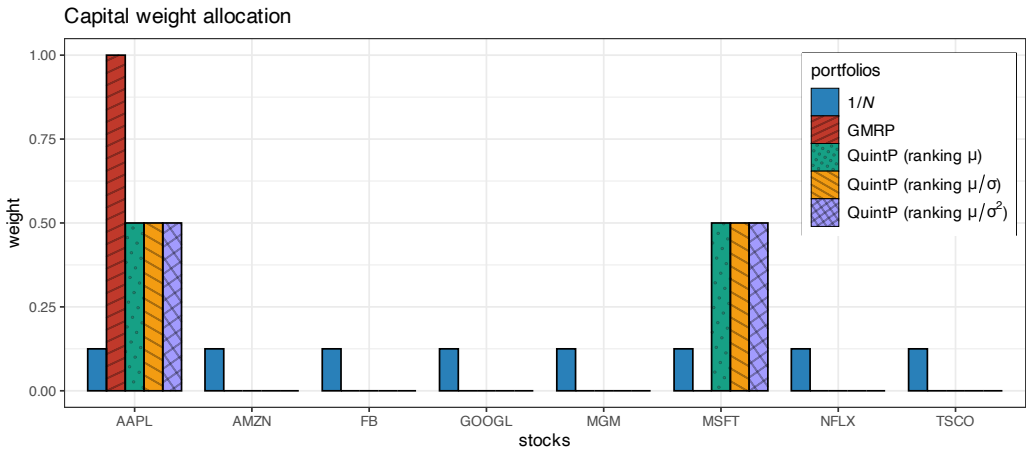


Figure 6.12 Portfolio allocation of some heuristic portfolios.

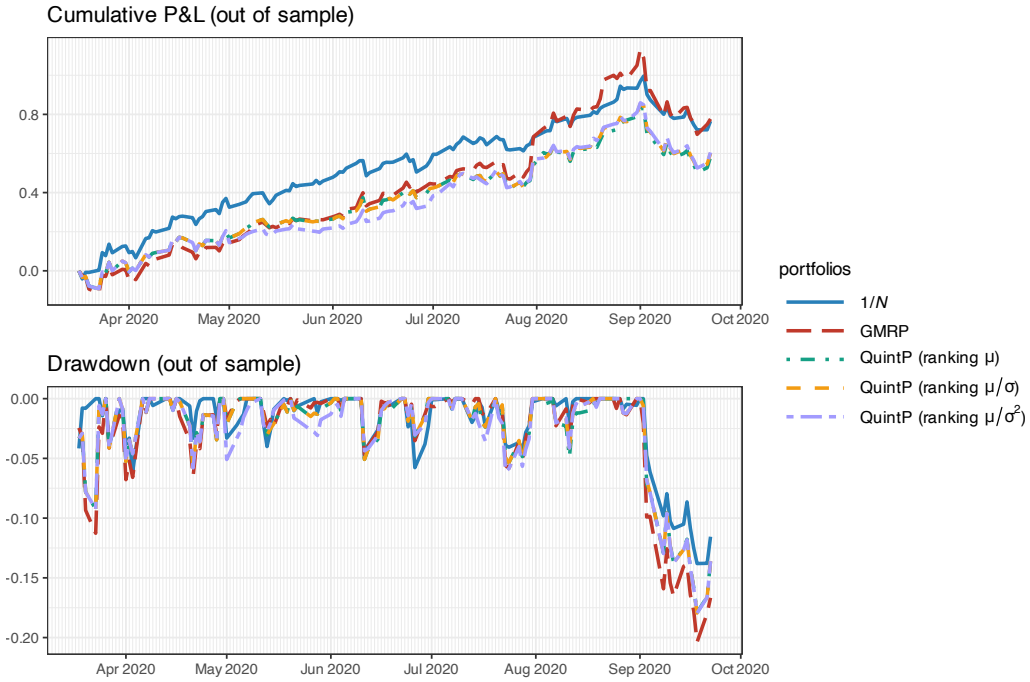


Figure 6.13 Backtest cumulative P&L and drawdown of some heuristic portfolios.

6.5.1 Global Minimum Variance Portfolio (GMVP)

In Section 6.3, we saw that the portfolio volatility $\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}$ or, similarly, the variance $\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}$, can be used as proxies for portfolio risk. The *global minimum variance portfolio* (GMVP) is

formulated as

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w} \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}. \end{array} \tag{6.15}$$

This problem is a convex quadratic program (QP) that can be easily solved with a QP solver.

If we ignore the no-shorting constraint $\boldsymbol{w} \geq \mathbf{0}$, then the problem admits the simple closed-form solution

$$\boldsymbol{w} = \frac{1}{\mathbf{1}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\mathbf{1}}\boldsymbol{\Sigma}^{-1}\mathbf{1}. \tag{6.16}$$

This portfolio is widely used in academic papers for simplicity of evaluation, as well as for comparison purposes when evaluating different estimators of the covariance matrix $\boldsymbol{\Sigma}$ (while ignoring the effect of the estimation of $\boldsymbol{\mu}$).

6.5.2 Inverse Volatility Portfolio (IVolP)

The aim of the *inverse volatility portfolio* (IVolP) is to control the portfolio risk in a simple way. It is defined as

$$\boldsymbol{w} = \frac{\boldsymbol{\sigma}^{-1}}{\mathbf{1}^{\mathsf{T}}\boldsymbol{\sigma}^{-1}}, \tag{6.17}$$

where $\boldsymbol{\sigma}^2 = \text{diag}\,(\boldsymbol{\Sigma})$ denotes the assets' variances and $\boldsymbol{\sigma}$ are the assets' volatilities.

This portfolio allocates smaller weights to high-volatility assets and larger weights to low-volatility assets. It is also called *equal volatility portfolio* because the weighted constituent assets have equal volatility:

$$\text{Std}(w_i r_i) = w_i \sigma_i \propto 1/N.$$

It is worth comparing the IVolP with the GMVP in (6.16) when the covariance matrix is assumed diagonal; in this case, the GVMP simplifies to

$$\boldsymbol{w} = \frac{\boldsymbol{\sigma}^{-2}}{\mathbf{1}^{\mathsf{T}}\boldsymbol{\sigma}^{-2}},$$

which, similary to (6.17), allocates smaller weights to high-volatility assets and vice versa.

6.5.3 Risk Parity Portfolio (RPP)

The *risk parity portfolio* (RPP), also termed *equal risk portfolio* (ERP), aims at equalizing the risk contribution from the invested assets to the global portfolio risk (Qian, 2005, 2016; Roncalli, 2013). This portfolio design is extensively treated in Chapter 11 and here we provide a very concise summary.

The IVolP in (6.17) attempts to control the risk of each of the assets in a simple way, but it totally ignores the assets' correlations, that is, it implicitly assumes that the covariance matrix $\boldsymbol{\Sigma}$ is diagonal. The RPP can be understood as a more sound formulation of the IVolP by properly taking the assets' correlations into account.

The RPP attemps to equalize the risk contributions from the assets, reminiscent of the way the $1/N$ portfolio equalizes the capital allocated to the assets, as shown in Figure 6.14.

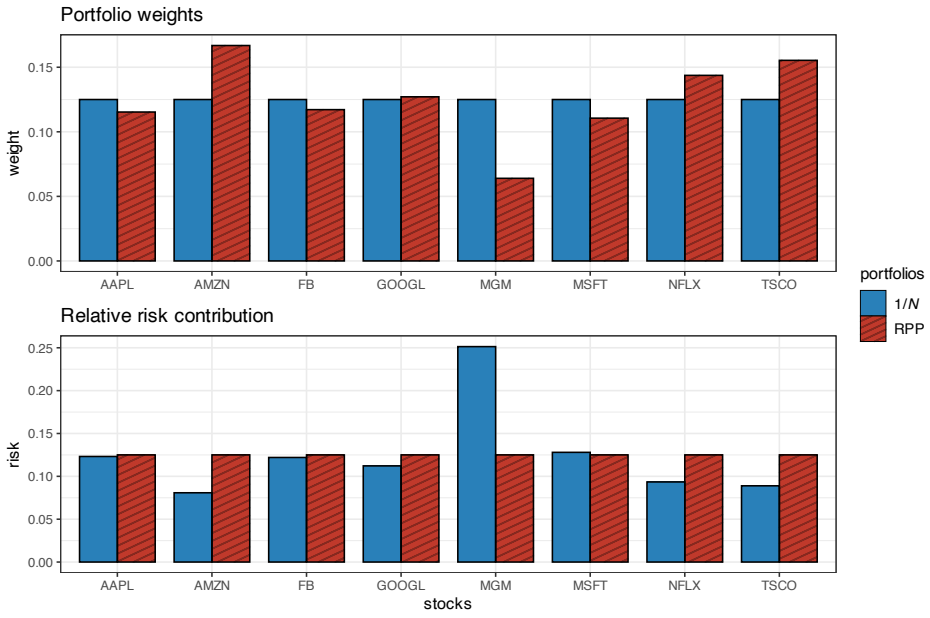


Figure 6.14 From dollar diversification ($1/N$ portfolio) to risk diversification (RPP).

6.5.4 Most Diversified Portfolio (MDivP)

If markets are risk-efficient, it can be argued that assets will produce returns in proportion to their risk (Choueifaty & Coignard, 2008). This suggests that one might use a proxy for the assets' risk, such as the volatilities, $\boldsymbol{\sigma}$, in lieu of the forecast $\boldsymbol{\mu}$.

Specifically, the *diversification ratio* (DR) is defined similarly to the Sharpe ratio (Section 6.3) but using $\boldsymbol{\sigma}$ in lieu of $\boldsymbol{\mu}$:

$$\text{DR} = \frac{\boldsymbol{w}^\top \boldsymbol{\sigma}}{\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}}.$$

For long-only portfolios, it can be shown that DR $\geq$ 1 (for a single stock, it reduces to DR = 1).

The *most diversified portfolio* (MDivP) is formulated as the maximization of the DR (akin to the maximization of the Sharpe ratio considered in Section 7.2 from Chapter 7):

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{\boldsymbol{w}^\top \boldsymbol{\sigma}}{\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}} \\ \text{subject to} \quad & \mathbf{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}. \end{aligned} \tag{6.18}$$

6.5.5 Maximum Decorrelation Portfolio (MDecP)

The *maximum decorrelation portfolio* (MDecP) is defined similarly to the GMVP but using the *correlation matrix* $\boldsymbol{C}$ in lieu of the covariance matrix $\boldsymbol{\Sigma}$ (Christoffersen et al., 2012):

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \boldsymbol{w}^\top \boldsymbol{C} \boldsymbol{w} \\ \text{subject to} \quad & \mathbf{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}. \end{aligned} \tag{6.19}$$

The correlation matrix $\boldsymbol{C}$ can be interpreted as the covariance matrix of the normalized assets such that they have unit volatility, that is,

$$\boldsymbol{C} = \text{Diag}\,(\boldsymbol{\Sigma})^{-1/2}\, \boldsymbol{\Sigma} \text{Diag}\,(\boldsymbol{\Sigma})^{-1/2},$$

where $\text{Diag}(\cdot)$ keeps the diagonal elements of the matrix argument and sets the off-diagonal elements to zero.

The MDecP is not only related to the GMVP but also to the MDivP: when the MDecP weights are divided by their respective assets' volatilities and renormalized so that they sum to 1, the MDivP weights are obtained.

6.5.6 Numerical Experiments

We now compare the following risk-based portfolios: GMVP, IVolP, RPP, MDivP, and MDecP. We consider eight stocks from the S&P 500 during the period 2019–2020, under the same setting as the experiments in Section 6.4.5.

Figure 6.15 shows the normalized dollar allocation of the risk-based portfolios, whereas Figure 6.16 shows the out-of-sample backtest results in terms of cumulative P&L and drawdown. Since all these portfolios are precisely obtained by somehow minizing the risk, the resulting drawdowns are small (compare with the drawdowns of the heuristic portfolios in Figure 6.13).

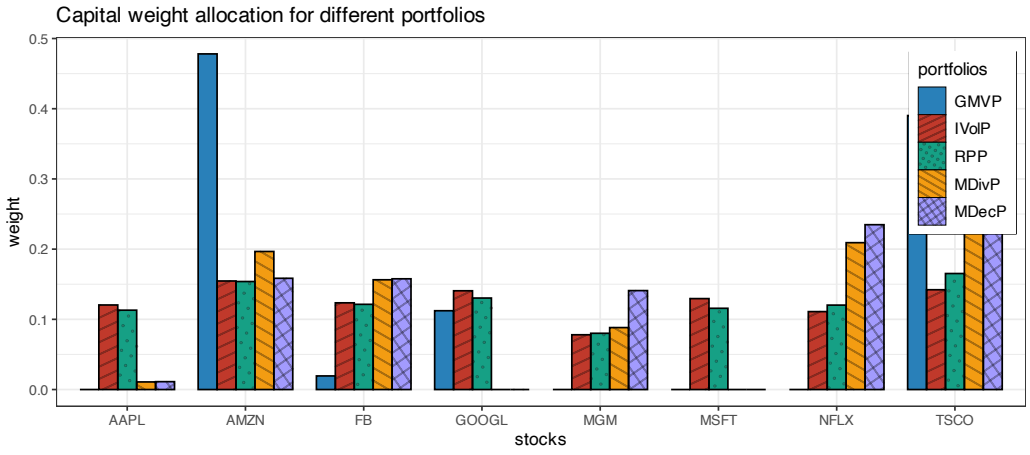


Figure 6.15 Portfolio allocation of risk-based portfolios.

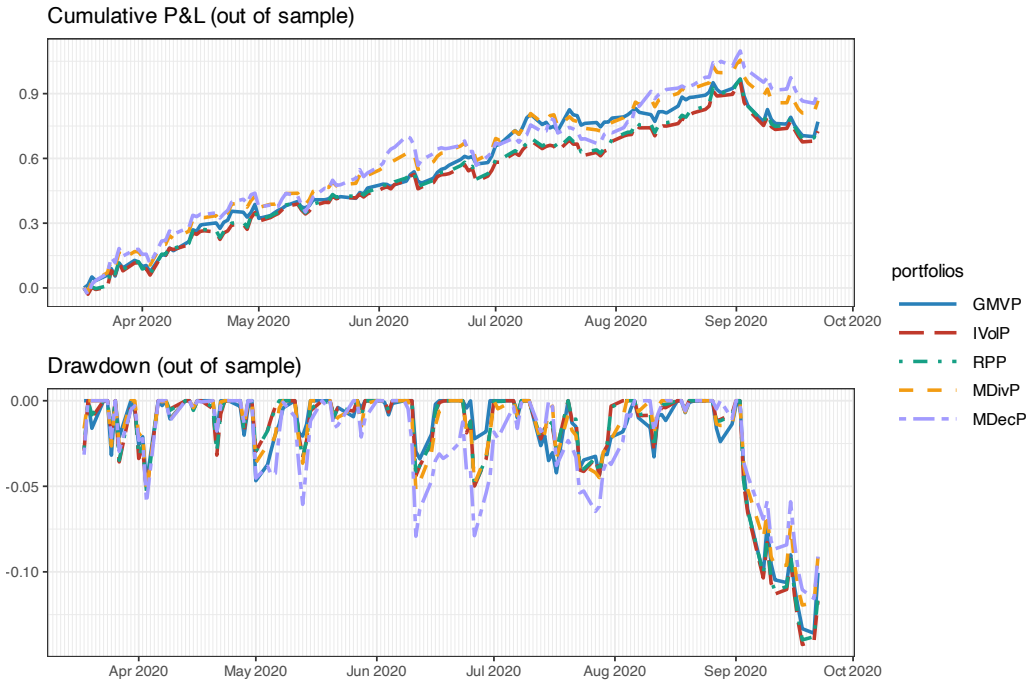


Figure 6.16 Backtest cumulative P&L and drawdown of risk-based portfolios.

6.6 Summary

- A portfolio over N assets is conveniently represented by the N-dimensional vector $\boldsymbol{w}$. The portfolio return is given by $\boldsymbol{w}^\top \boldsymbol{r}$, where $\boldsymbol{r}$ contains the returns of the assets.
- In practice, portfolios have to be periodically rebalanced, incurring transaction costs which erode the potential return.
- Portfolios typically need to satisfy many constraints, some imposed by the regulators or brokers (such as shorting, leverage, and margin requirements), while others depend on the investor's views (such as market neutrality, portfolio diversity, sparsity level, and turnover control). Most of these constraints are expressed as convex functions that can be easily handled in the optimization process (a notable exception is sparsity control).
- The performance of portfolios can be measured and monitored in a multitude of ways (such as Sharpe ratio, downside risk measures, drawdown, and CVaR). Most interesting is when such performance measures are included in the optimization process, leading to challenging formulations.
- Some heuristic portfolios have endured the test of time and can be easily calculated without the need for complicated optimization methods; notable examples are the $1/N$ portfolio and the family of quintile portfolios.
- Risk-based portfolios are focused on reducing the variability in the returns without

attempting to ride the market trend; they are typically easy to compute and exhibit good results in practice.

Exercises

6.1 (Effect of rebalancing)

a. Download market data corresponding to N assets (e.g., stocks or cryptocurrencies) during a period with T observations, $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T \in \mathbb{R}^N$.
b. Start with the $1/N$ portfolio at time $t = 1$ and let the portfolio weights naturally evolve as the assets' prices change over time. Plot the portfolio weights and the NAV over time (assuming transaction costs of 90 bps).
c. Repeat using a regular calendar-based rebalancing scheme.
d. Repeat using an adaptive rebalancing scheme when the difference exceeds a threshold.

6.2 (Portfolio constraints) Consider a universe of $N = 2$ assets and draw the set of feasible portfolios under the following constraints:

a. Budget and no-shorting constraints:

$$\mathbf{1}^\mathsf{T}\boldsymbol{w} \leq 1, \quad \boldsymbol{w} \geq \mathbf{0}.$$

b. Budget fully invested and no-shorting constraints:

$$\mathbf{1}^\mathsf{T}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}.$$

c. Budget, no-shorting, and holding constraints:

$$\mathbf{1}^\mathsf{T}\boldsymbol{w} \leq 1, \quad \boldsymbol{w} \geq \mathbf{0}, \quad \boldsymbol{w} \leq 0.6 \times \mathbf{1}.$$

d. Budget and turnover constraints:

$$\mathbf{1}^\mathsf{T}\boldsymbol{w} \leq 1, \quad \|\boldsymbol{w} - \boldsymbol{w}_0\|_1 \leq 0.5,$$

with $\boldsymbol{w}_0$ denoting the $1/N$ portfolio.
e. Leverage constraint:

$$\|\boldsymbol{w}\|_1 \leq 1.$$

6.3 (Performance measures)

a. Download market data corresponding to the S&P 500 index.
b. Plot the returns and cumulative returns over time.
c. Calculate the annualized expected return with arithmetic and geometric compounding.
d. Calculate the annualized volatility.
e. Plot the volatility-adjusted returns and cumulative returns over time.
f. Calculate the annualized Sharpe ratio with arithmetic and geometric compounding.
g. Calculate the annualized semi-deviation and Sortino ratio.
h. Calculate the VaR and CVaR.
i. Plot the drawdown over time.

6.4 (Heuristic portfolios)

a. Download market data corresponding to N assets during a period with T observations.
b. Using 70% of the data, compute the $1/N$ portfolio and quintile portfolios using different ranking mechanisms.
c. Plot and compare the different portfolio allocations.
d. Using the remaining 30% of the data, assess the portfolios in terms of cumulative returns, volatility-adjusted cumulative returns, Sharpe ratio, and drawdown.

6.5 (Risk-based portfolios) Repeat Exercise 6.4 with the following risk-based portfolios:

a. GMVP
b. IVolP
c. MDivP
d. MDecP

References

Ardia, D., Bolliger, G., Boudt, K., & Gagnon-Fleury, J.-P. (2017). The impact of covariance misspecification in risk-based portfolios. *Annals of Operations Research*, *254*(1–2), 2–16.

Bacon, C. (2008). *Practical Portfolio Performance Measurement and Attribution* (2nd ed.). John Wiley & Sons.

Bender, J., Briand, R., Melas, D., & Subramanian, R. A. (2013). Foundations of factor investing. *MSCI Research Insight*.

Chopra, V., & Ziemba, W. (1993). The effect of errors in means, variances and covariances on optimal portfolio choice. *Journal of Portfolio Management*, *19*(2), 6–11.

Choueifaty, Y., & Coignard, Y. (2008). Toward maximum diversification. *Journal of Portfolio Management*, *35*, 40–51.

Christoffersen, P., Errunza, V., Jacobs, K., & Langlois, H. (2012). Is the potential for international diversification disappearing? A dynamic copula approach. *The Review of Financial Studies*, *25*, 3711–3751.

DeMiguel, V., Garlappi, L., Nogales, F. J., & Uppal, R. (2009). A generalized approach to portfolio optimization: Improving performance by constraining portfolio norms. *Management Science*, *55*(5), 798–812.

DeMiguel, V., Garlappi, L., & Uppal, R. (2009). Optimal versus naive diversification: How inefficient is the 1/N portfolio strategy? *The Review of Financial Studies*, *22*(5), 1915–1953.

Duchin, R., & Levy, H. (2009). Markowitz versus the Talmudic portfolio diversification strategies. *Journal of Portfolio Management*, *35*(2), 71.

Fama, E. F., & French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, *33*(1), 3–56.

Goetzmann, W. N., & Kumar, A. (2008). Equity portfolio diversification. *Review of Finance*, *12*(3), 433–463.

Grinblatt, M., Titman, S., & Wermers, R. (1995). Momentum investment strategies, portfolio performance, and herding: A study of mutual fund behavior. *American Economic Review*, *85*, 1088–1105.

Jacobs, B. I., Levy, K. N., & Markowitz, H. M. (2005). Portfolio optimization with factors, scenarios, and realistic short positions. *Operations Research*, *53*(4), 586–599.

Jurczenko, E. (2017). *Factor Investing: From Traditional to Alternative Risk Premia*. Elsevier.

Kritzman, M., Page, S., & Turkington, D. (2010). In defense of optimization: The fallacy of 1/N. *Financial Analysts Journal*, *66*(2), 31–39.

Markowitz, H. M. (1952). Portfolio selection. *The Journal of Finance*, *7*(1), 77–91.

Markowitz, H. M. (1959). *Portfolio Selection: Efficient Diversification of Investments*. John Wiley & Sons.

Michaud, R. O., & Michaud, R. O. (2008). *Efficient Asset Management: A Practical Guide to Stock Portfolio Optimization and Asset Allocation* (2nd ed.). Oxford University Press.

Qian, E. (2005). Risk parity portfolios: Efficient portfolios through true diversification. *PanAgora Asset Management*.

Qian, E. (2016). *Risk Parity Fundamentals*. CRC Press.

Roncalli, T. (2013). *Introduction to Risk Parity and Budgeting*. Chapman & Hall/CRC.

Scherer, B. (2002). Portfolio resampling: Review and critique. *Financial Analysts Journal*, *58*(6), 98–109.

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, *19*(3), 425–442.

Sharpe, W. F. (1966). Mutual fund performance. *The Journal of Business*, *39*(1), 119–138.

Sharpe, W. F. (1994). The Sharpe ratio. *Journal of Portfolio Management*, *21*, 49–58.

Zhou, R., & Palomar, D. P. (2020). Understanding the quintile portfolio. *IEEE Transactions on Signal Processing*, *68*, 4030–4040.

7

Modern Portfolio Theory

You must read, you must persevere, you must sit up nights, you must inquire, and exert the utmost power of your mind. If one way does not lead to the desired meaning, take another; if obstacles arise, then still another; until, if your strength holds out, you will find that clear which at first looked dark.

— Giovanni Boccaccio, *Genealogia deorum gentilium*

Modern portfolio theory (MPT) started with Harry Markowitz's 1952 seminal paper "Portfolio Selection" (Markowitz, 1952), for which he would receive the Nobel prize in 1990. He put forth the idea that risk-adverse investors should optimize their portfolio based on a combination of two objectives: expected return and risk. That idea has remained central to portfolio optimization. In practice, however, the vanilla Markowitz portfolio formulation has some issues and drawbacks; as a consequence most practitioners tend to combine it with several heuristics or avoid it altogether. In this chapter, we explore the mean–variance Markowitz portfolio in its many facets.

7.1 Mean–Variance Portfolio (MVP)

This section considers the *mean–variance portfolio* (MVP) proposed by Markowitz in his 1952 seminar paper (Markowitz, 1952); see also the monographs Rubinstein (2002) and Kolm et al. (2014) with a retrospective view.

7.1.1 Return–Risk Trade-Off

While the expected return of the portfolio $\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}$ is a relevant quantity that measures the average or expected benefit (see Section 6.3), it leaves one key element out: the risk. An investor needs to control the probability of going bankrupt. A *risk measure* precisely quantifies how risky an investment strategy is. The most basic risk measure is the volatility $\sqrt{\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}}$ or, similarly, the variance $\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}$: a higher variance means that there are large peaks in the distribution of the returns, which may cause big losses. The volatility/variance has its limitations and a number of more sophisticated risk measures have been proposed in the literature such as downside risk measures (e.g., semi-variance), VaR, and CVaR (see Section 6.3).

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

Precisely, Markowitz put forth the idea that risk-adverse investors should optimize their portfolio based on a combination of two objectives: expected return and risk (Markowitz, 1952). However, there is a trade-off between these two objectives: the higher the expected return, the higher the risk; the lower the risk, the lower the expected return. In other words, we are dealing with a multi-objective optimization problem (see Section A.7 in Appendix A) with its corresponding optimal trade-off curve (Pareto-optimal points), which is called the *efficient frontier* in this portfolio context. Basically, the efficient frontier is a curve representing the best possible pair values of expected return and volatility that can be achieved by any feasible portfolio. The choice of a specific point on this trade-off curve depends on how aggressive or risk-averse the investor is. Figure 7.1 shows the trade-off between expected return and volatility.

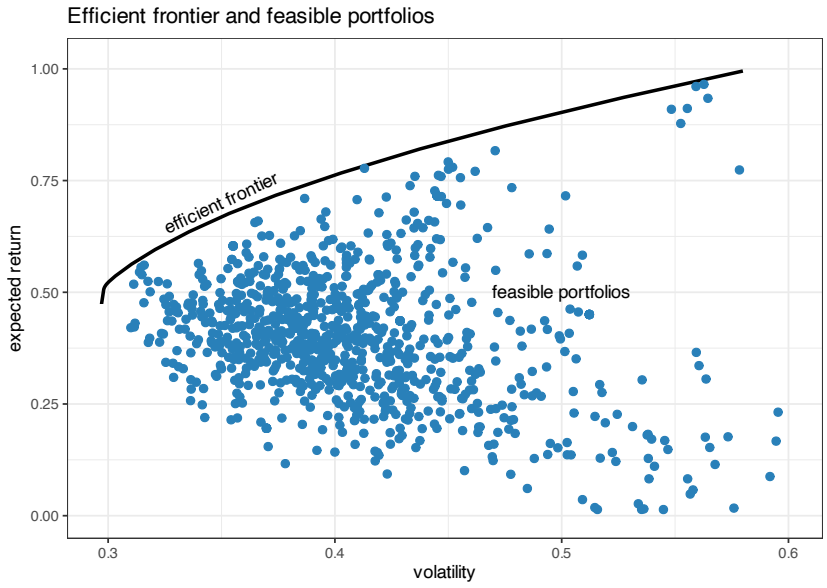


Figure 7.1 Trade-off between expected return and volatility: efficient frontier and 1,000 random feasible portfolios.

7.1.2 MVP Formulation

Markowitz's portfolio formulation is a bi-objective optimization problem with the two objectives being the expected return $\boldsymbol{w}^\top \boldsymbol{\mu}$ and the risk measured by the volatility $\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}$ or, similarly, the variance $\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}$. From an algorithmic perspective, it is more computationally efficient to use the variance than the volatility, since the former involves quadratic programming whereas the latter results in second-order cone programming (see Appendix A and, more specifically, Section B.1 in Appendix B for algorithmic aspects of solvers).

There are several ways to formulate a bi-objective optimization problem, as discussed in Section A.7. The most convenient way is via scalarization of the two objectives into a single

objective with a weighted sum (Markowitz, 1952):

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\top \boldsymbol{\mu} - \frac{\lambda}{2} \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w} \\ \text{subject to} & \mathbf{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array} \tag{7.1}$$

where λ is a hyper-parameter that controls how risk-averse the investor is. The two constraints included are just for illustration purposes; in practice, one can include many of the other constraints listed in Section 6.2, as elaborated in Section 7.1.4.

The choice of λ in the MVP in (7.1) will produce different portfolios that lie along the efficient frontier shown in Figure 7.2. In fact, by letting λ vary from 0 to ∞, one can recover the whole efficient frontier. In particular, for $\lambda = 0$ we recover the global maximum return portfolio (GMRP) described in Section 6.4.2 (i.e., only the expected return is considered while the variance is ignored), whereas for $\lambda \to \infty$ we recover the global minimum variance portfolio (GMVP) considered in Section 6.5.1 (i.e., the expected return is not used at all). Figure 7.2 shows the efficient frontier together with these portfolios.

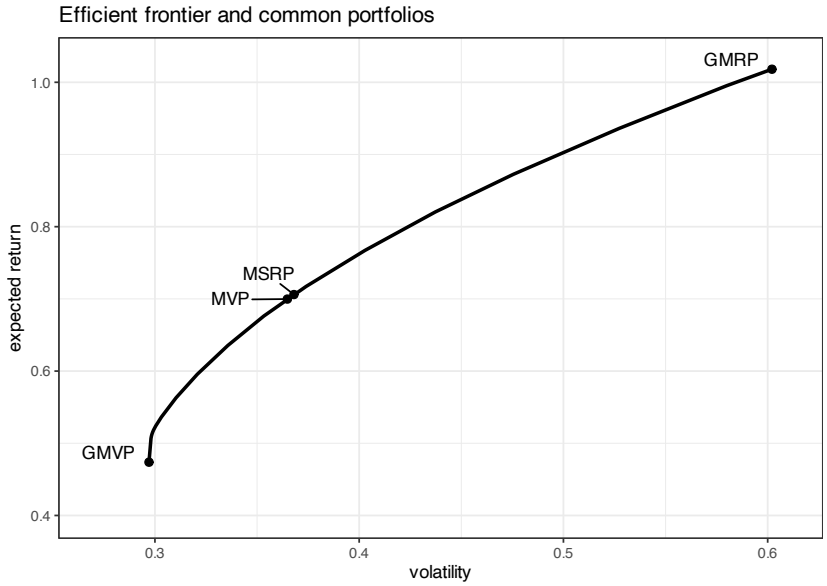


Figure 7.2 Efficient frontier and common portfolios.

Problem (7.1) is a quadratic problem (QP) that can be easily solved with a QP solver (see Appendix B). If we ignore the no-shorting constraint $\boldsymbol{w} \geq \mathbf{0}$, then the problem admits the simple closed-form solution

$$\boldsymbol{w} = \frac{1}{\lambda} \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{\mu} + \nu \mathbf{1} \right),$$

where ν is the optimal dual variable $\nu = \frac{\lambda - \mathbf{1}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}}{\mathbf{1}^\top \boldsymbol{\Sigma}^{-1} \mathbf{1}}$ chosen to satisfy the normalization constraint $\mathbf{1}^\top \boldsymbol{w} = 1$.

Example 7.1 (Optimum investment sizing) Suppose we have a single asset or a fully invested portfolio ($\mathbf{1}^\top \boldsymbol{w} = 1$) with some returns over time $r_1, r_2, \ldots$ The problem of investment sizing

refers to determining how much of the budget should be allocated to this risky asset and how much should be kept in cash. The optimal sizing can be obtained from the MVP formulation in (7.1) particularized to $N = 1$ as

$$\begin{array}{ll} \underset{w}{\text{maximize}} & w\mu - \frac{\lambda}{2} w^2 \sigma^2 \\ \text{subject to} & 0 \leq w \leq 1, \end{array}$$

with solution

$$w = \left[\frac{1}{\lambda} \frac{\mu}{\sigma^2} \right]_0^1,$$

where $[\cdot]_0^1$ denotes projection on the interval $[0, 1]$. In particular, the growth rate is maximized when $\lambda = 1$ (see Section 7.3.1 for details) and then the optimal sizing turns out to be the (projected) mean-to-variance ratio: $w = [\mu/\sigma^2]_0^1$.

There are two other widely used reformulations of Markowitz's portfolio. One formulation has the variance term as a constraint:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T} \boldsymbol{\mu} \\ \text{subject to} & \boldsymbol{w}^\mathsf{T} \boldsymbol{\Sigma} \boldsymbol{w} \leq \alpha, \\ & \mathbf{1}^\mathsf{T} \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array} \tag{7.2}$$

where α is a hyper-parameter that controls the maximum level of variance accepted. The other formulation, instead, has the expected return as a constraint:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \boldsymbol{w}^\mathsf{T} \boldsymbol{\Sigma} \boldsymbol{w} \\ \text{subject to} & \boldsymbol{w}^\mathsf{T} \boldsymbol{\mu} \geq \beta, \\ & \mathbf{1}^\mathsf{T} \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array} \tag{7.3}$$

where β is a hyper-parameter that controls the minimum level of expected return accepted.

Similarly to what happens with formulation (7.1) as the hyper-parameter λ is varied, formulations (7.2) and (7.3) can also recover the whole efficient frontier by changing the hyper-parameters α and β, respectively.

The hyper-parameters in formulations (7.2) and (7.3) have a more intuitive interpretation than that in formulation (7.1), that is, maximum accepted variance and minimum accepted expected return. On the other hand, formulations (7.2) and (7.3) may be infeasible if the hyper-parameters are not properly chosen, whereas formulation (7.1) is always feasible regardless of the hyper-parameter λ. To avoid running into infeasibility issues, it is customary to choose the hyper-parameters based on benchmark portfolios such as the $1/N$ portfolio, i.e., $\alpha = \frac{1}{N^2} \mathbf{1}^\mathsf{T} \boldsymbol{\Sigma} \mathbf{1}$ or $\beta = \frac{1}{N} \mathbf{1}^\mathsf{T} \boldsymbol{\mu}$.

Problem (7.3) is still a QP that can be solved efficiently with a QP solver; however, problem (7.2) is a quadratically-constrained QP (QCQP) which typically requires a higher complexity either by using a QCQP solver or an SOCP solver (see Section B.1 for details).

Commonly used programming languages in finance offer packages specifically designed to optimize portfolios under a wide variety of formulations and constraints, such as the popular R package `fPortfolio` (Wuertz et al., 2023) and the Python libraries `skfolio` (Delatte

et al., 2026; Nicolini et al., 2025), `Riskfolio-Lib` (Cajas, 2026), and `PyPortfolioOpt` (Martin, 2021).

Figure 7.3 shows the portfolio allocation of the MVP in (7.1) for different values of λ, from which we can see the critical effect of the hyper-parameter in the final allocation. Figure 7.4 and Table 7.1 show the corresponding backtest results, indicating that smaller values of λ suffer from a worse Sharpe ratio and a much severe drawdown than the largest ones.

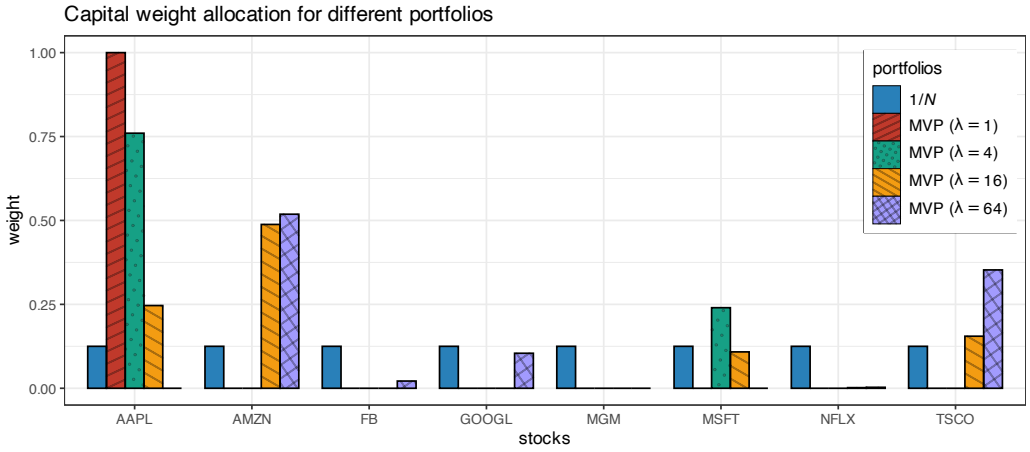


Figure 7.3 Portfolio allocation of MVP with different values of hyper-parameter λ.

Table 7.1 Backtest performance of MVP with different values of hyper-parameter λ.

Portfolio	Sharpe ratio	Annual return	Annual volatility	Max drawdown
$1/N$	3.34	115%	35%	14%
MVP ($\lambda = 1$)	2.60	112%	43%	20%
MVP ($\lambda = 4$)	2.57	106%	41%	19%
MVP ($\lambda = 16$)	3.37	113%	33%	15%
MVP ($\lambda = 64$)	3.65	116%	32%	14%

7.1.3 MVP as a Regression

Interestingly, the MVP formulation can be interpreted as a regression problem. The key observation is that the variance of the portfolio can be seen as an ℓ_2-norm error term. First, we rewrite the variance as

$$\begin{aligned} \boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w} &= \boldsymbol{w}^\mathsf{T}\mathbb{E}\left[(\boldsymbol{r}_t - \boldsymbol{\mu})(\boldsymbol{r}_t - \boldsymbol{\mu})^\mathsf{T}\right]\boldsymbol{w} \\ &= \mathbb{E}\left[(\boldsymbol{w}^\mathsf{T}(\boldsymbol{r}_t - \boldsymbol{\mu}))^2\right] \\ &= \mathbb{E}\left[(\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t - \rho)^2\right], \end{aligned}$$

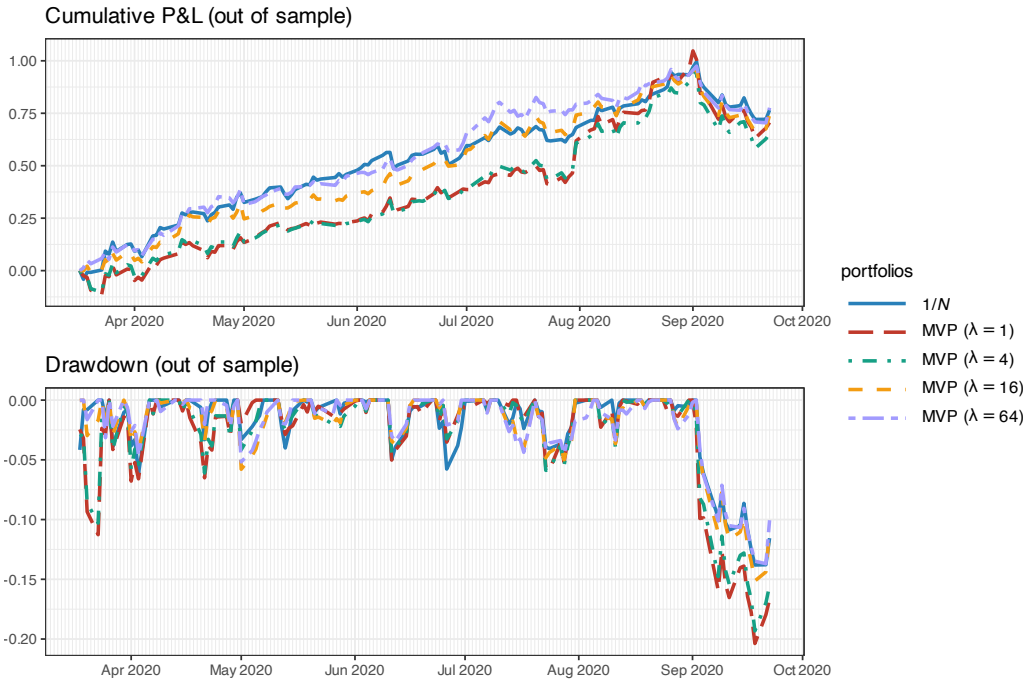


Figure 7.4 Backtest performance of MVP with different values of hyper-parameter λ.

where $\rho = \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}$. Then, we use the sample approximation for the expected value,

$$\mathbb{E}\left[(\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t - \rho)^2\right] \approx \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t - \rho)^2 = \frac{1}{T}\|\boldsymbol{R}\boldsymbol{w} - \rho\mathbf{1}\|_2^2,$$

where $\boldsymbol{R} \triangleq [\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T]^\mathsf{T}$.

Now we can continue by rewriting the MVP formulation (7.1) as the minimization

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w} - \frac{2}{\lambda}\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} \\ \text{subject to} & \mathbf{1}^\mathsf{T}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0} \end{array}$$

and finally substitute the variance with the ℓ_2-norm expression:

$$\begin{array}{ll} \underset{\boldsymbol{w},\rho}{\text{minimize}} & \frac{1}{T}\|\boldsymbol{R}\boldsymbol{w} - \rho\mathbf{1}\|_2^2 - \frac{2}{\lambda}\rho \\ \text{subject to} & \rho = \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}, \\ & \mathbf{1}^\mathsf{T}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}. \end{array} \tag{7.4}$$

Note that the expected return, denoted by ρ, is also an optimization variable. Alternatively, one could fix ρ to some predetermined value.

From the MVP formulation as a regression in (7.4), we can interpret the portfolio $\boldsymbol{w}$ as trying to obtain returns over time as constant as possible and equal to ρ. In other words, it is trying to achieve or track the expected return ρ with minimum variance as measured by the ℓ_2-norm. Interestingly, this interpretation is in fact related to the topic of index tracking in Chapter 13.

7.1.4 MVP with Practical Constraints

The previous MVP formulations in (7.1), (7.2), and (7.3) have included the two simple constraints $\mathbf{1}^T\boldsymbol{w} = 1$ and $\boldsymbol{w} \geq \mathbf{0}$ for simplicitly of exposition. In practice, however, there is a multitude of other constraints that an investor may want to use, as listed in Section 6.2; see also Kolm et al. (2014).

For example, we can start with the basic MVP formulation in (7.1) that only contains the budget and no-shorting constraints,

$$\begin{array}{lll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^T\boldsymbol{\mu} - \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w} & \\ \text{subject to} & \mathbf{1}^T\boldsymbol{w} = 1 & \text{budget,} \\ & \boldsymbol{w} \geq \mathbf{0} & \text{no-shorting,} \end{array}$$

and change the constraints to reflect a more realistic trading situation:

$$\begin{array}{lll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^T\boldsymbol{\mu} - \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w} & \\ \text{subject to} & \|\boldsymbol{w}\|_1 \leq \gamma & \text{leverage,} \\ & \|\boldsymbol{w} - \boldsymbol{w}_0\|_1 \leq \tau & \text{turnover,} \\ & |\boldsymbol{w}| \leq \boldsymbol{u} & \text{max positions,} \\ & \boldsymbol{\beta}^T\boldsymbol{w} = 0 & \text{market neutral,} \\ & \|\boldsymbol{w}\|_0 \leq K & \text{sparsity,} \end{array} \tag{7.5}$$

where $\gamma \geq 1$ controls the amount of shorting and leverage, $\tau > 0$ controls the turnover (to limit the transaction costs in the rebalancing), $\boldsymbol{u}$ limits the position in each stock, $\boldsymbol{\beta}$ denotes the beta of the stocks, and K controls the cardinality of the portfolio (to select a small set of stocks from the universe).

More generally, we can write the MVP formulation with a general set of constraints, denoted by $\mathcal{W}$, as

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^T\boldsymbol{\mu} - \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w} \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}. \end{array} \tag{7.6}$$

From the optimization perspective, as long as the constraints in $\mathcal{W}$ are convex the problem will remain convex and will still be easy to solve. The only constraint that is nonconvex in (7.5) is the cardinality constraint $\|\boldsymbol{w}\|_0 \leq K$.

7.1.5 Improving the MVP with Heuristics

One of the main problems of the MVP is the lack of diversification, which goes against common practice. To address this issue, several heuristics have been proposed with good practical results such as no-shorting constraints, upper bound constraints, and ℓ_2-norm constraints; see Kolm et al. (2014) for more details.

Imposing no-shorting constraints, $\boldsymbol{w} \geq \mathbf{0}$, seems to have significant practical benefits in reducing the amplification of the noise inherent in the estimated covariance matrix, even when the constraints are wrong (Jagannathan & Ma, 2003). Surprisingly, with no-shorting constraints, the sample covariance matrix performs as well as more sophisticated covariance

matrix estimators based on factor models, shrinkage estimators, or higher-frequency returns. Including upper bound constraints also has a regularization effect on the covariance matrix.

For example, if we consider the GMVP from Section 6.5.1 with constraints $\mathbf{1}^\mathsf{T}\boldsymbol{w} = 1$ and $\mathbf{0} \leq \boldsymbol{w} \leq \boldsymbol{u}$, the optimal solution (6.16) is still valid using instead the regularized covariance matrix

$$\tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma} + \boldsymbol{\lambda}_0\mathbf{1}^\mathsf{T} + \mathbf{1}\boldsymbol{\lambda}_0^\mathsf{T} + \boldsymbol{\lambda}_\mathrm{u}\mathbf{1}^\mathsf{T} + \mathbf{1}\boldsymbol{\lambda}_\mathrm{u}^\mathsf{T},$$

where $\boldsymbol{\lambda}_0$ and $\boldsymbol{\lambda}_\mathrm{u}$ are the Lagrange multipliers corresponding to the no-shorting and upper bound constraints, respectively (Jagannathan & Ma, 2003). This regularized covariance matrix $\tilde{\boldsymbol{\Sigma}}$ can be interpreted as a shrunk version of $\boldsymbol{\Sigma}$ with reduced sampling error. Interestingly, the same result can be obtained by instead including an ℓ_1-norm constraint $\|\boldsymbol{w}\|_1 \leq \delta$ (DeMiguel et al., 2009).

One way to enforce diversity is via the diversification constraint $\|\boldsymbol{w}\|_2^2 \leq D$ (see Section 6.2). The maximum diversity level D is lower bounded by $1/N$ (achieved by the $1/N$ portfolio) and could be chosen as the diversity achieved by some benchmark portfolio, for example, the GMVP gives $D = \mathbf{1}^\mathsf{T}\boldsymbol{\Sigma}^{-2}\mathbf{1}/(\mathbf{1}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\mathbf{1})^2$.

For example, if we consider the GMVP with constraints $\mathbf{1}^\mathsf{T}\boldsymbol{w} = 1$ and $\|\boldsymbol{w}\|_2^2 \leq D$, the optimal solution (6.16) is still valid using instead the regularized covariance matrix

$$\tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma} + \gamma\boldsymbol{I},$$

leading to the portfolio

$$\boldsymbol{w} = \frac{1}{\mathbf{1}^\mathsf{T}\left(\boldsymbol{\Sigma} + \gamma\boldsymbol{I}\right)^{-1}\mathbf{1}}\left(\boldsymbol{\Sigma} + \gamma\boldsymbol{I}\right)^{-1}\mathbf{1},$$

where $\gamma \geq 0$ is the Lagrange multiplier corresponding to the diversification constraint (DeMiguel et al., 2009). For $\gamma = 0$ we recover the solution (6.16), whereas for $\gamma \to \infty$ we obtain the $1/N$ portfolio. Interestingly, this solution is like shrinking the covariance matrix to the identity matrix with γ being the shrinkage intensity (see Chapter 3).

Figure 7.5 illustrates the effect of two diversification heuristics on the portfolio allocation (namely, with upper bound $\|\boldsymbol{w}\|_\infty \leq 0.25$ and diversification constraint $\|\boldsymbol{w}\|_2^2 \leq 0.25$), which indeed improve the diversification. Figure 7.6 and Table 7.2 show the corresponding backtest results, indicating that the more diversified MVPs have a better Sharpe ratio and drawdown.

Table 7.2 Backtest performance of MVP under two diversification heuristics (upper bound and diversification constraint).

Portfolio	Sharpe ratio	Annual return	Annual volatility	Max drawdown
$1/N$	3.34	115%	35%	14%
GMVP	3.67	115%	31%	14%
MVP	2.44	99%	41%	19%
MVP with upper bound	2.98	96%	32%	14%
MVP with diversific. const.	2.79	97%	35%	16%

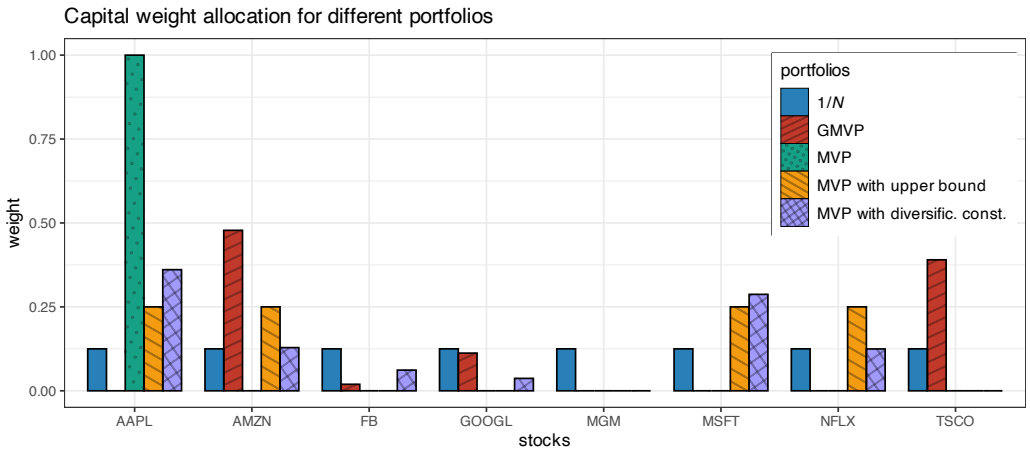


Figure 7.5 Portfolio allocation of MVP under two diversification heuristics (upper bound and diversification constraint).

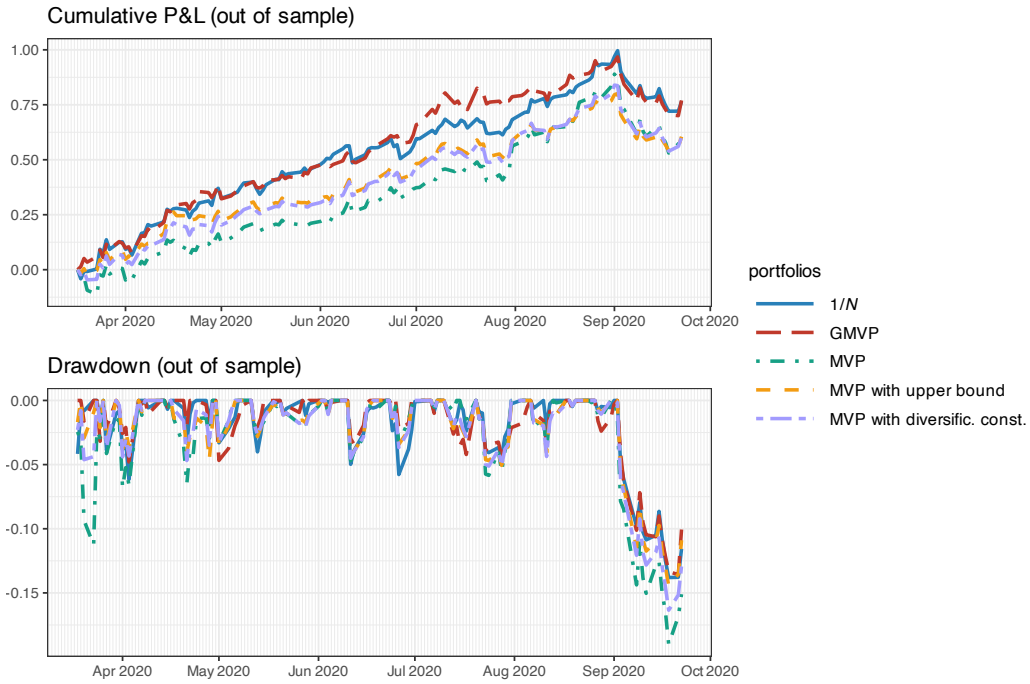


Figure 7.6 Backtest performance of MVP under two diversification heuristics (upper bound and diversification constraint).

7.2 Maximum Sharpe Ratio Portfolio

Markowitz's mean–variance framework provides portfolios along the efficient frontier, that is, formulations (7.1), (7.2), and (7.3), by varying the hyper-parameters λ, α, and β, respectively. The specific choice of a point on the efficient frontier depends on the risk aversion of the

investor. Nevertheless, the most widely used performance measure is the Sharpe ratio and there is only one portfolio on the efficient frontier that achieves the maximum value, as indicated in Figure 7.2 under MSRP.

Precisely, in 1966, Sharpe proposed the *maximum Sharpe ratio portfolio* (MSRP) formulation (Sharpe, 1966) as

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \dfrac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} - r_{\mathrm{f}}}{\sqrt{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}}} \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array} \tag{7.7}$$

where r_{f} is the return of the risk-free asset.

This problem is not convex, but it belongs to the class of *fractional programs* (FPs) for which many solving methods are available, namely, the bisection, Dinkelbach, and Schaible transform methods as described next (see Section A.5 in Appendix A for a description of FPs and Section B.5 in Appendix B for more details on the algorithms).

More specifically, the numerator $\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} - r_{\mathrm{f}}$ is affine and hence concave, while the denominator $\sqrt{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}} = \|\boldsymbol{\Sigma}^{1/2}\boldsymbol{w}\|_2$ is a norm and hence convex (and positive on the feasible set). Formulation (7.7) is therefore a *concave–convex FP*.

7.2.1 Bisection Method

Concave–convex FP can be conveniently solved via a sequence of convex feasibility problems, termed the bisection method (see Section A.4 for details). For problem (7.7), the sequence of convex feasibility problems are of the form

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{find}} & \boldsymbol{w} \\ \text{subject to} & t\sqrt{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}} \leq \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} - r_{\mathrm{f}}, \\ & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array} \tag{7.8}$$

where $t > 0$ is a fixed parameter (not an optimization variable). This problem is convex and, in fact, a second-order cone program (SOCP) since the volatility can be written as an ℓ_2-norm, $\sqrt{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}} = \|\boldsymbol{\Sigma}^{1/2}\boldsymbol{w}\|_2$ (see Section A.5). Note that this convex feasibility reformulation can be infeasible in practice (e.g., if all the elements of $\boldsymbol{\mu}$ are negative), so care has to be taken for such a case. The method is summarized in Algorithm 7.1.

7.2.2 Dinkelbach Method

Concave–convex FPs can be solved via the Dinkelbach method (Dinkelbach, 1967) by solving a sequence of simpler convex problems (see Section B.5.2 for details). For problem (7.7), the sequence of convex problems is, in fact, a sequence of SOCPs of the form:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} - r_{\mathrm{f}} - y^k\sqrt{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}} \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array} \tag{7.9}$$

Algorithm 7.1: Bisection method to solve the MSRP in (7.7).

Choose interval $[l, u]$ (with $l > 0$) that contains the optimal Sharpe ratio, tolerance $\epsilon > 0$;
repeat
  $t \leftarrow (l + u)/2$;
  Solve the convex feasibility problem (7.8);
  if feasible **then**
    $l \leftarrow t$ and keep solution $\boldsymbol{w}$;
  else
    $u \leftarrow t$;
  end if
until $u - l \leq \epsilon$;

where the parameter y^k is sequentially updated as

$$y^k = \frac{(\boldsymbol{w}^k)^\mathsf{T} \boldsymbol{\mu} - r_\mathrm{f}}{\sqrt{(\boldsymbol{w}^k)^\mathsf{T} \boldsymbol{\Sigma} \boldsymbol{w}^k}} \tag{7.10}$$

with k the iteration index. This is summarized in Algorithm 7.2.

Algorithm 7.2: Dinkelbach method to solve the MSRP in (7.7).

Choose initial point $\boldsymbol{w}^0$;
Set $k \leftarrow 0$;
repeat
  Set y^k as in (7.10);
  Solve the convex problem (7.9) and keep current solution as $\boldsymbol{w}^{k+1}$;
  $k \leftarrow k + 1$;
until convergence;

7.2.3 Schaible Transform Method

Concave–convex FPs can be more efficiently solved via the Schaible transform (Schaible, 1974) without the need to resort to iterative schemes (see Section B.5.3 for details). It turns out that problem (7.7) can be rewritten as

$$\begin{aligned} \underset{\boldsymbol{y}, t}{\text{maximize}} \quad & \boldsymbol{y}^\mathsf{T} (\boldsymbol{\mu} - r_\mathrm{f} \mathbf{1}) \\ \text{subject to} \quad & \sqrt{\boldsymbol{y}^\mathsf{T} \boldsymbol{\Sigma} \boldsymbol{y}} \leq 1, \\ & t > 0, \\ & \mathbf{1}^\mathsf{T} \boldsymbol{y} = t, \quad \boldsymbol{y} \geq \mathbf{0}, \end{aligned}$$

which can be further simplified (eliminating variable t) to

$$\begin{array}{ll} \underset{\boldsymbol{y}}{\text{maximize}} & \boldsymbol{y}^{\mathsf{T}}\left(\boldsymbol{\mu}-r_{\mathrm{f}}\mathbf{1}\right) \\ \text{subject to} & \boldsymbol{y}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{y}\leq 1, \\ & \mathbf{1}^{\mathsf{T}}\boldsymbol{y}>0, \quad \boldsymbol{y}\geq\mathbf{0}, \end{array} \tag{7.11}$$

from which the original variable $\boldsymbol{w}$ can be easily recovered from $\boldsymbol{y}$, and $t = \mathbf{1}^{\mathsf{T}}\boldsymbol{y}$ as $\boldsymbol{w} = \boldsymbol{y}/\left(\mathbf{1}^{\mathsf{T}}\boldsymbol{y}\right)$. Note that, since $\boldsymbol{y} \geq \mathbf{0}$, the constraint $\mathbf{1}^{\mathsf{T}}\boldsymbol{y} > 0$ can be safely ignored when solving the problem with an interior-point method (see Section B.4 for details).

Interestingly, if we reformulate problem (7.7) as the minimization of $\sqrt{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}}/\left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu}-r_{\mathrm{f}}\right)$, the Schaible transform leads to

$$\begin{array}{ll} \underset{\boldsymbol{y}}{\text{minimize}} & \boldsymbol{y}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{y} \\ \text{subject to} & \boldsymbol{y}^{\mathsf{T}}\left(\boldsymbol{\mu}-r_{\mathrm{f}}\mathbf{1}\right)\geq 1, \\ & \mathbf{1}^{\mathsf{T}}\boldsymbol{y}>0, \quad \boldsymbol{y}\geq\mathbf{0}, \end{array} \tag{7.12}$$

where the inequality $\boldsymbol{y}^{\mathsf{T}}\left(\boldsymbol{\mu}-r_{\mathrm{f}}\mathbf{1}\right) \geq 1$ can be alternatively written as equality. Observe that the Schaible transform requires the denominator to be nonnegative, which in this case means $\boldsymbol{y}^{\mathsf{T}}\left(\boldsymbol{\mu}-r_{\mathrm{f}}\mathbf{1}\right) > 0$ or $\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} - r_{\mathrm{f}} > 0$; in other words, this alternative reformulation may or may not be feasible and care has to be taken for this case.

Problem (7.11) is a (convex) QCQP, which can be easily solved with a QCQP solver. However, the alternative problem reformulation in (7.12) is a simpler QP, which is preferred since it can be solved with more efficient QP solvers (see Section B.1 for details).

Example 7.2 (MSRP with return and upper bound constraints) Consider the MSRP formulation with minimum return $\beta > 0$ and upper bound $\boldsymbol{u}$ constraints:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \dfrac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu}}{\sqrt{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}}} \\ \text{subject to} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu}\geq\beta, \\ & \mathbf{1}^{\mathsf{T}}\boldsymbol{w}=1, \quad \mathbf{0}\leq\boldsymbol{w}\leq\boldsymbol{u}. \end{array}$$

After applying the Schaible transform, the problem simplifies to the QP

$$\begin{array}{ll} \underset{\boldsymbol{y}}{\text{minimize}} & \boldsymbol{y}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{y} \\ \text{subject to} & \boldsymbol{y}^{\mathsf{T}}\boldsymbol{\mu}\geq 1, \\ & 0<\mathbf{1}^{\mathsf{T}}\boldsymbol{y}\leq\beta^{-1}, \quad \mathbf{0}\leq\boldsymbol{y}\leq\boldsymbol{u}\cdot\left(\mathbf{1}^{\mathsf{T}}\boldsymbol{y}\right), \end{array}$$

from which the portfolio is obtained as $\boldsymbol{w} = \boldsymbol{y}/\left(\mathbf{1}^{\mathsf{T}}\boldsymbol{y}\right)$.

Example 7.3 (MSRP with shorting and return constraint) Consider the MSRP formulation with minimum return $\beta > 0$ and with shorting allowed:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \dfrac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu}}{\sqrt{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}}} \\ \text{subject to} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu}\geq\beta \\ & \|\boldsymbol{w}\|_1=1. \end{array}$$

After applying the Schaible transform, the problem simplifies to the QP

$$\begin{array}{ll} \underset{\boldsymbol{y}}{\text{minimize}} & \boldsymbol{y}^\top \boldsymbol{\Sigma} \boldsymbol{y} \\ \text{subject to} & \boldsymbol{y}^\top \boldsymbol{\mu} \geq 1 \\ & 0 < \|\boldsymbol{y}\|_1 \leq \beta^{-1}, \end{array}$$

from which the portfolio is obtained as $\boldsymbol{w} = \boldsymbol{y}/\|\boldsymbol{y}\|_1$.

7.3 Utility-Based Portfolios

All the previous porfolio formulations in Sections 7.1 and 7.2 are based on some judicious combination of the mean and variance of the returns $R_t^{\text{portf}} = \boldsymbol{w}^\top \boldsymbol{r}_t$ (here, $\boldsymbol{r}_t$ denotes linear returns). However, it is possible to express the interest of the investor in a more general way via *utility functions*.

7.3.1 Kelly Criterion Portfolio

In 1956, a scientist working for Bell Labs, John Larry Kelly, Jr., brought together game theory and information theory (Kelly, Jr., 1956). He showed that in order to achieve maximum growth of wealth, a gambler should place bets that maximize the expected value of the logarithm of the capital, usually referred to as the *Kelly criterion*. The Kelly criterion was applied to portfolio design in Markowitz (1959); see also Thorp (1971) and Thorp (1997).[1]

Recall from (6.9) that, for a given fixed portfolio $\boldsymbol{w}$, the returns are $R_t^{\text{portf}} = \boldsymbol{w}^\top \boldsymbol{r}_t$. From the geometric compounding of the portfolio wealth of NAV in (6.7), we can write the wealth accumulated during the periods $t = 1, \ldots, T$ as

$$W_T = W_0 \prod_{t=1}^{T} \frac{W_t}{W_{t-1}} = W_0 \prod_{t=1}^{T} \left(1 + \boldsymbol{w}^\top \boldsymbol{r}_t\right),$$

where W_0 is the initial wealth and W_t the wealth at time period t.

It turns out that the wealth grows exponentially as $W_t \sim e^{t \times G}$ (Cover & Thomas, 1991), where the exponent G is the exponential rate of growth or, simply, *growth rate*:

$$G = \lim_{T \to \infty} \log \left(\frac{W_T}{W_0}\right)^{1/T}.$$

The growth rate can be estimated asymptotically by the law of large numbers as

$$G = \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \log \left(1 + \boldsymbol{w}^\top \boldsymbol{r}_t\right) = \mathbb{E}\left[\log \left(1 + \boldsymbol{w}^\top \boldsymbol{r}\right)\right],$$

where $\boldsymbol{r}$ is a random variable with the same distribution as each $\boldsymbol{r}_t$.

[1] Edward O. Thorp was an American math professor, author, and blackjack player who wrote *Beat the Dealer* (Thorp, 1962), which became a classic and was the first book to prove mathematically that the house advantage in blackjack could be overcome by card counting.

Maximizing the growth rate effectively maximizes the long-term wealth and it is a very compelling choice for portfolio design. We call this formulation the *Kelly criterion portfolio*:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \mathbb{E}\left[\log\left(1+\boldsymbol{w}^\mathsf{T}\boldsymbol{r}\right)\right] \\ \text{subject to} & \mathbf{1}^\mathsf{T}\boldsymbol{w}=1, \quad \boldsymbol{w}\geq\mathbf{0}. \end{array} \tag{7.13}$$

Good and bad properties of the Kelly criterion are discussed in MacLean et al. (2010).

Problem (7.13) is convex since the log is a concave function and it is a maximization problem. In practice, however, we need to find an appropriate way to deal with the expected value in the objective function as we discuss next via sample averages, the exponential cone, and other approximations.

Solving the Kelly Criterion Portfolio Directly via Sample Average

In practice, the expectation in problem (7.13) can be approximated by the sample mean:

$$\mathbb{E}\left[\log\left(1+\boldsymbol{w}^\mathsf{T}\boldsymbol{r}\right)\right] \approx \frac{1}{T}\sum_{t=1}^{T}\log\left(1+\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t\right).$$

However, finding a solver that can deal directly with the log function may be challenging.

Solving the Kelly Criterion Portfolio via Exponential Cone Programming

Interestingly, this problem can be reformulated in terms of the exponential cone (Cajas, 2021), leading to *exponential cone programming* for which solvers can be found.

Consider problem (7.13) with the sample average approximation and with the additional slack variables q_t:

$$\begin{array}{ll} \underset{\boldsymbol{w},\{q_t\}}{\text{maximize}} & \frac{1}{T}\sum_{t=1}^{T} q_t \\ \text{subject to} & q_t \leq \log\left(1+\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t\right), \quad t=1,\ldots,T, \\ & \mathbf{1}^\mathsf{T}\boldsymbol{w}=1, \quad \boldsymbol{w}\geq\mathbf{0}. \end{array}$$

The constraints $q_t \leq \log\left(1+\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t\right)$ can be equivalently written as $\exp(q_t) \leq 1+\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t$ and the Kelly portfolio can be finally written as the exponential cone program

$$\begin{array}{ll} \underset{\boldsymbol{w},\{q_t\}}{\text{maximize}} & \frac{1}{T}\sum_{t=1}^{T} q_t \\ \text{subject to} & (q_t, 1, 1+\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t) \in \mathcal{K}_{\text{exp}}, \quad t=1,\ldots,T, \\ & \mathbf{1}^\mathsf{T}\boldsymbol{w}=1, \quad \boldsymbol{w}\geq\mathbf{0}, \end{array} \tag{7.14}$$

where $\mathcal{K}_{\text{exp}}$ is the exponential cone (Chares, 2007) defined as

$$\mathcal{K}_{\text{exp}} \triangleq \left\{(a,b,c) \mid c \geq b\, e^{a/b}, b>0\right\} \cup \left\{(a,b,c) \mid a\leq 0, b=0, c\geq 0\right\}.$$

Solving the Kelly Criterion Portfolio via Mean–Variance Approximations

The most common way to deal with the expectation in the objective of problem (7.13) is via a first-order Taylor approximation[2] around the point $\boldsymbol{r} = \mathbf{0}$ (Markowitz, 1959):

$$\mathbb{E}\left[\log\left(1 + \boldsymbol{w}^\mathsf{T}\boldsymbol{r}\right)\right] \approx \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \frac{1}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}, \tag{7.15}$$

which is a surprising and beautiful justification for the mean–variance formulation in (7.1) (with $\lambda = 1$), that is,

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \frac{1}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w} \\ \text{subject to} & \mathbf{1}^\mathsf{T}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}. \end{array}$$

It is possible to find better Taylor approximations than (7.15) around the point $\boldsymbol{r} = \boldsymbol{\mu}$, such as (Markowitz, 1959)

$$\mathbb{E}\left[\log\left(1 + \boldsymbol{w}^\mathsf{T}\boldsymbol{r}\right)\right] \approx \log\left(1 + \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}\right) - \frac{1}{2}\frac{\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}}{\left(1 + \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}\right)^2} \tag{7.16}$$

or, further approximated,

$$\mathbb{E}\left[\log\left(1 + \boldsymbol{w}^\mathsf{T}\boldsymbol{r}\right)\right] \approx \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \frac{1}{2}(\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu})^2 - \frac{1}{2}\frac{\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}}{1 + 2\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}}. \tag{7.17}$$

One can also try an approximation over an interval (as opposed to the Taylor approximations, which focus on a single point) such as the Levy-Markowitz approximation (Levy & Markowitz, 1979):

$$\begin{aligned} \mathbb{E}\left[\log\left(1 + \boldsymbol{w}^\mathsf{T}\boldsymbol{r}\right)\right] \approx \frac{1}{2\kappa^2}\log\left(\left(1 + \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}\right)^2 - \kappa^2\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}\right) \\ + \left(1 - \frac{1}{\kappa^2}\right)\log\left(1 + \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}\right), \end{aligned} \tag{7.18}$$

where κ measures the width of the approximating interval in standard deviations.

However, these other approximations may not bring any benefit in practice (Pulley, 1983) due to the nonconvexity and the fact that the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ contain estimation errors an order of magnitude larger than the potential benefit of these refined approximations. See Markowitz (2014) for a historical perspective on mean–variance approximations. Chapter 9 explores higher-order moments, that is, skewness and kurtosis, to obtain better approximations for portfolio design.

7.3.2 Expected Utility Theory

The model of rational decision-making in most of economics and statistics is *expected utility theory*, which was axiomatized by von Neumann and Morgenstern (1944) and Savage (1954).

[2] The Taylor expansion of the log function around the point $x = x_0$ is

$$\log(1 + x) = \log(1 + x_0) + \frac{1}{1 + x_0}(x - x_0) + \frac{1}{2}\frac{-1}{(1 + x_0)^2}(x - x_0)^2 + \cdots$$

In the context of portfolio design, the utility $U(\cdot)$ is a function of the random portfolio return $\boldsymbol{w}^\mathsf{T}\boldsymbol{r}$ and the objective is the maximization of the expected utility:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \mathbb{E}\left[U(\boldsymbol{w}^\mathsf{T}\boldsymbol{r})\right] \\ \text{subject to} & \mathbf{1}^\mathsf{T}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}. \end{array} \tag{7.19}$$

The Kelly criterion portfolio in (7.13) is a particular case of the expected utility maximization in (7.19) by choosing the log utility $U(x) = \log(1+x)$. Some (concave) utilities of interest include:

- $U(x) = \log(1+x)$
- $U(x) = \sqrt{1+x}$
- $U(x) = -\dfrac{1}{x}$
- $U(x) = -p\dfrac{1}{x^p}$ for $p > 0$ (as $p \to 0$ it converges to the log utility)
- $U(x) = -\dfrac{1}{\sqrt{1+x}}$
- $U(x) = 1 - \exp(-\lambda x)$, with $\lambda > 0$ being the risk aversion parameter.

However, this general expected utility theory, while useful in theory, is an elusive concept that may be of little practical help when faced with investment decisions, as opposed to the statistically sound Kelly criterion. As stated in Roy (1952):

> In calling in a utility function to our aid, an appearance of generality is achieved at the cost of a loss of practical significance, and applicability in our results. A man who seeks advice about his actions will not be grateful for the suggestion that he maximise expected utility.

Problem (7.19) is convex as long as the utility $U(\cdot)$ is a concave function. Similarly to the Kelly criterion portfolio, the expected utility portfolio formulation can be numerically solved in a direct way or via mean–variance approximations as described next.

Solving the Expected Utility Portfolio Directly via Sample Average

In practice, problem (7.19) can be approximated by directly replacing the expectation in the objective function by the sample mean:

$$\mathbb{E}\left[U(\boldsymbol{w}^\mathsf{T}\boldsymbol{r})\right] \approx \frac{1}{T}\sum_{t=1}^{T} U\left(\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t\right).$$

However, finding a solver that can deal directly with the utility function $U(\cdot)$ may be challenging, even if it is a concave function.

Solving the Expected Utility Portfolio via Mean–Variance Approximations

Similarly to the Kelly criterion portfolio, the most common way to deal with the expectation in the objective of problem (7.19) is via a mean–variance approximations similar to that in (7.15); see Markowitz (2014).

We now consider some Taylor approximations around specific points[3] (Markowitz, 1959) and the Levy–Markowitz approximation along an interval (more specifically on three points of the interval) (Levy & Markowitz, 1979).

The second-order Taylor approximation around the point $\boldsymbol{r} = \mathbf{0}$ is

$$\begin{aligned}\mathbb{E}\left[U(\boldsymbol{w}^\mathsf{T}\boldsymbol{r})\right] &\approx U(0) + U'(0)\ \mathbb{E}\left[\boldsymbol{w}^\mathsf{T}\boldsymbol{r}\right] + \frac{1}{2}U''(0)\ \mathbb{E}\left[(\boldsymbol{w}^\mathsf{T}\boldsymbol{r})^2\right] \\ &= U(0) + U'(0)\ \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} + \frac{1}{2}U''(0)(\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w} + (\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu})^2),\end{aligned} \tag{7.20}$$

whereas the approximation around the point $\boldsymbol{r} = \boldsymbol{\mu}$ reads

$$\begin{aligned}\mathbb{E}\left[U(\boldsymbol{w}^\mathsf{T}\boldsymbol{r})\right] &\approx U(\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}) + U'(\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu})\mathbb{E}\left[\boldsymbol{w}^\mathsf{T}(\boldsymbol{r}-\boldsymbol{\mu})\right] + \frac{1}{2}U''(\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu})\mathbb{E}\left[(\boldsymbol{w}^\mathsf{T}(\boldsymbol{r}-\boldsymbol{\mu})^2\right] \\ &= U(\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}) + \frac{1}{2}U''(\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu})\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}.\end{aligned} \tag{7.21}$$

An approximation that fits the utility $U(\cdot)$ simultaneously on three points, namely, the mean $\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}$ (like the previous Taylor approximation) and the two points $\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} \pm \kappa\sqrt{\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}}$ is Levy and Markowitz (1979) [4]

$$\begin{aligned}\mathbb{E}\left[U(\boldsymbol{w}^\mathsf{T}\boldsymbol{r})\right] &\approx U(\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}) \\ &\quad + \frac{U\left(\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} + \kappa\sqrt{\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}}\right) + U\left(\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \kappa\sqrt{\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}}\right) - 2U\left(\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}\right)}{2\kappa^2}.\end{aligned} \tag{7.22}$$

Many empirical analyses have concluded that these mean–variance approximations perform well in practice for real data, and their difference is negligible (Levy & Markowitz, 1979; Markowitz, 1959, 2014; Pulley, 1983). Chapter 9 explores higher-order moments, that is, skewness and kurtosis, in these approximations for portfolio design.

7.4 Universal Algorithm

This chapter has explored a large variety of portfolio formulations with one common theme: they are all based on different combinations of the mean and variance. Even the Kelly criterion-based or utility-based portfolios can be well approximated in terms of the mean and variance. Nevertheless, each of the formulations results in a different type of optimization problem (see the taxonomy of problems in Section A.5). This implies that each formulation

[3] The Taylor expansion of a utility function $U(\cdot)$ around the point $x = x_0$ is

$$U(x) \approx U(x_0) + U'(x_0)(x - x_0) + \frac{1}{2}U''(x_0)(x - x_0)^2 + \cdots$$

[4] The Levy approximation of an expected utility around an interval of width κ standard deviations centered at the mean is

$$\mathbb{E}\left[U(X)\right] \approx U(\mu) + \frac{1}{2\kappa^2}\left[U(\mu + \kappa\sigma) + U(\mu - \kappa\sigma) - 2U(\mu)\right],$$

where X denotes a random variable (with mean μ and standard deviation σ) and $U(\cdot)$ is the utility function.

requires a different numerical method or solver (see Appendix B for details on algorithms) as listed next:

- scalarized MVP in (7.1): requires a QP solver;
- mean-constrained MVP in (7.3): also needs a QP solver;
- variance-constrained MVP in (7.2): requires a QCQP solver (with a higher computational complexity than a QP solver);
- MSRP in (7.7): this is an FP and can be solved via bisection of SOCPs, via the Dinkelbach sequence of SOCPs, or via the one-shot Schaible transformed QP;
- Kelly portfolio in (7.13): after the approximation in (7.15) this can be solved with a QP solver; and
- utility-based portfolios as in (7.19): also with a QP solver after some mean–variance approximation like in (7.20) or (7.21).

Interestingly, since all such different formulations can be expressed as trade-offs of the mean and variance, the portfolios will naturally lie on the efficient frontier. This means that rather than dealing with each of the formulations separately, an alternative is to solve the basic mean–variance formulation with a properly chosen value of the hyper-parameter λ:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \dfrac{\lambda}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w} \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array} \tag{7.23}$$

where $\mathcal{W}$ denotes a general set of constraints for the portfolio (assumed for convenience to contain only linear and quadratic terms, which includes ℓ_1-norms, ℓ_∞-norms, and ℓ_2-norms). The challenge naturally lies in determining the appropriate value of the hyper-parameter λ (Xiu et al., 2023).

Thus, a general framework can be formulated that embraces all such mean–variance problems as

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & f\left(\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}, \boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}\right) \\ \text{subject to} & g\left(\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}, \boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}\right) \leq 0, \\ & \boldsymbol{w} \in \mathcal{W}, \end{array} \tag{7.24}$$

where $f(x, y)$ and $g(x, y)$ are two functions that consider a trade-off between the two arguments x and y, which represent the mean and the variance of the portfolio. Table 7.3 shows how $f(x, y)$ and $g(x, y)$ particularize to the many mean–variance formulations considered in this chapter.

Table 7.3 Portfolio formulations with the corresponding functions f and g in the general mean–variance formulation (7.24).

Portfolio	$f(x, y)$	$g(x, y)$
MVP	$-x + \frac{\lambda}{2} y$	—
Mean–volatility portfolio	$-x + \kappa\sqrt{y}$	—
Mean-constrained MVP	y	$\beta - x$
Variance-constrained MVP	$-x$	$y - \alpha$
MSRP	$-\dfrac{x - r_{\mathrm{f}}}{\sqrt{y}}$	—
Kelly portfolio	$-x + \frac{1}{2} y$	—
Kelly portfolio	$-\log(1 + x) + \frac{1}{2} \dfrac{y}{(1 + x)^2}$	—
Kelly portfolio	$-x + \frac{1}{2} x^2 + \frac{1}{2} \dfrac{y}{1 + 2x}$	—
Kelly portfolio	$-\left(1 - \dfrac{1}{\kappa^2}\right) \log(1 + x) - \dfrac{1}{2\kappa^2} \log\left((1 + x)^2 - \kappa^2 y\right)$	—
Expected utility portfolio	$-U(0) - U'(0)x - \frac{1}{2} U''(0)(y + x^2)$	—
Expected utility portfolio	$-U(x) - \frac{1}{2} U''(x) y$	—
Expected utility portfolio	$-\left(1 - \dfrac{1}{\kappa^2}\right) U(x) - \dfrac{U(x + \kappa\sqrt{y}) + U(x - \kappa\sqrt{y})}{2\kappa^2}$	—

Notably, it is possible to develop a universal algorithm for mean–variance formulations as in (7.24) (Xiu et al., 2023) with a computational cost orders of magnitude below off-the-shelf general-purpose QP, QCQP, or SOCP solvers. This unified formulation and algorithm provides several practical advantages:

1. Computational efficiency: It is considerably more efficient to solve a QP than some other more complicated class of problems such as QCQP or SOCP (see Section B.1 for details).

2. Code reusability: One can devote time and energy to implementing the proper solver function call for problem (7.23), with the advantage that the same code can then be reused for many different formulations as long as they are based on the mean and variance (simply using a different choice of λ).

3. Code specialization: Rather than using a general-purpose QP solver, an advanced user can develop a tailored numerical algorithm that takes advantage of the particularities of the specific portfolio formulation at hand (this can include the natural sparsity of the long-only portfolio via the active set method or any other numerical trick).

The derivation of the universal algorithm to solve (7.24) is based on the *successive convex approximation* (SCA) method (Scutari et al., 2014); see Section B.8 in Appendix B for details. In particular, we will successively approximate the problem by a QP, hence the method is called *sequential QP* (SQP). For the sake of exposition, we will only consider the formulation

(7.24) without the constraint function $g(x, y)$, that is,

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & f\left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu}, \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}\right) \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}. \end{array} \tag{7.25}$$

For the more general case (7.24), the reader is referred to Xiu et al. (2023).

The idea of SCA (or SQP in this case) is to solve (7.25) by instead solving a sequence of simpler surrogate problems:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \tilde{f}\left(\boldsymbol{w}; \boldsymbol{w}^k\right) + \frac{\tau^k}{2}\|\boldsymbol{w} - \boldsymbol{w}^k\|_2^2 \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array} \tag{7.26}$$

where k denotes the iteration index, the surrogate function $\tilde{f}\left(\boldsymbol{w}; \boldsymbol{w}^k\right)$ is a quadratic approximation of f around the previous point $\boldsymbol{w}^k$, and the last term is a quadratic proximal term added to make sure the objective function is strongly convex for convergence reasons (Scutari et al., 2014), which we will take as $\tau^k = 0$. Solving (7.26) sequentially will produce the iterates $\boldsymbol{w}^0, \boldsymbol{w}^1, \boldsymbol{w}^2, \ldots$ until some convergence criterion is satisfied.

The surrogate quadratic function $\tilde{f}$ can be easily obtained by linearizing the original function f in terms of $x = \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu}$ and $y = \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}$, leading to (ignoring the irrelevant constant terms):

$$\tilde{f}\left(\boldsymbol{w}; \boldsymbol{w}^k\right) = -\alpha^k \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} + \frac{\beta^k}{2}\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}, \tag{7.27}$$

where

$$\begin{aligned} \alpha^k &= -\frac{\partial f}{\partial x}\left(x^k = \left(\boldsymbol{w}^k\right)^{\mathsf{T}}\boldsymbol{\mu}, y^k = \left(\boldsymbol{w}^k\right)^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}^k\right), \\ \beta^k &= 2\frac{\partial f}{\partial y}\left(x^k = \left(\boldsymbol{w}^k\right)^{\mathsf{T}}\boldsymbol{\mu}, y^k = \left(\boldsymbol{w}^k\right)^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}^k\right). \end{aligned} \tag{7.28}$$

Table 7.4 shows the expressions of α^k and β^k for some of the portfolio formulations.

Table 7.4 Portfolio formulations with the corresponding expressions for α^k and β^k.

Portfolio	$f(x, y)$	$\frac{\partial f}{\partial x}$	$\frac{\partial f}{\partial y}$	α^k	β^k
MVP	$-x + \frac{\lambda}{2}y$	-1	$\lambda/2$	1	λ
Mean–volatility portfolio	$-x + \kappa\sqrt{y}$	-1	$\frac{\kappa}{2\sqrt{y}}$	1	$\frac{\kappa}{\sqrt{(\boldsymbol{w}^k)^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}^k}}$
MSRP	$-\frac{x - r_{\mathrm{f}}}{\sqrt{y}}$	$-\frac{1}{\sqrt{y}}$	$\frac{x - r_{\mathrm{f}}}{2y^{3/2}}$	$\frac{1}{\sqrt{(\boldsymbol{w}^k)^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}^k}}$	$\frac{(\boldsymbol{w}^k)^{\mathsf{T}}\boldsymbol{\mu} - r_{\mathrm{f}}}{\left((\boldsymbol{w}^k)^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}^k\right)^{3/2}}$
Kelly portfolio	$-x + \frac{1}{2}y$	-1	$1/2$	1	1

If we now plug (7.27) into (7.26), after rearranging terms we get

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & -\boldsymbol{w}^{\mathsf{T}}\left(\alpha^k\boldsymbol{\mu} + \tau^k\boldsymbol{w}^k\right) + \frac{\beta^k}{2}\boldsymbol{w}^{\mathsf{T}}\left(\boldsymbol{\Sigma} + \frac{\tau^k}{\beta^k}\boldsymbol{I}\right)\boldsymbol{w} \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array}$$

which is clearly in the form of the basic mean–variance formulation in (7.23).

Summarizing, we have essentially accomplished our original goal of solving any mean–variance formulation in the form of (7.25) (or, more generally, (7.24)) by solving a sequence of problems like (7.23):

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu}^k - \dfrac{\lambda^k}{2}\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}^k\boldsymbol{w} \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array} \tag{7.29}$$

with

$$\begin{aligned} \boldsymbol{\mu}^k &= \boldsymbol{\mu} + \frac{\tau^k}{\alpha^k}\boldsymbol{w}^k, \\ \boldsymbol{\Sigma}^k &= \boldsymbol{\Sigma} + \frac{\tau^k}{\beta^k}\boldsymbol{I}, \\ \lambda^k &= \frac{\beta^k}{\alpha^k}. \end{aligned} \tag{7.30}$$

Algorithm 7.3 summarizes this SQP procedure for mean–variance formulations.

Algorithm 7.3: Universal SQP-MVP method to solve (7.25).

1: Choose initial point $\boldsymbol{w}^0 \in \mathcal{W}$, sequences $\{\tau^k\}$ and $\{\gamma^k\}$;
2: Set $k \leftarrow 0$;
3: **repeat**
4: Compute α^k and β^k as in (7.28);
5: Compute $\boldsymbol{\mu}^k$, $\boldsymbol{\Sigma}^k$, and λ^k as in (7.30);
6: Solve the QP in (7.29) and keep solution as $\boldsymbol{w}^{k+1/2}$;
7: Set $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \gamma_t\left(\boldsymbol{w}^{k+1/2} - \boldsymbol{w}^k\right)$;
8: $k \leftarrow k + 1$;
9: **until** convergence;

Example 7.4 (MSRP) Consider the MSRP formulation

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \dfrac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} - r_{\mathrm{f}}}{\sqrt{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}}} \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}. \end{array}$$

This problem is an FP and can be solved in different ways as covered in Section 7.2. Alternatively, it can be solved via a sequence of simple QPs with Algorithm 7.3 (using $\tau^k = 0$ and $\gamma^k = 1$) as follows. From Table 7.4, we read the expressions for α^k and β^k, leading to the surrogate mean–variance problem

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} - \dfrac{\lambda^k}{2}\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w} \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array}$$

where $\lambda^k = \beta^k/\alpha^k = \dfrac{(\boldsymbol{w}^k)^{\mathsf{T}}\boldsymbol{\mu} - r_{\mathrm{f}}}{(\boldsymbol{w}^k)^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}^k}$. Figure 7.7 shows the convergence of this numerical

method compared with the solution via the Schaible transform (see Section 7.2), from which we can see that it converges in one to two iterations.

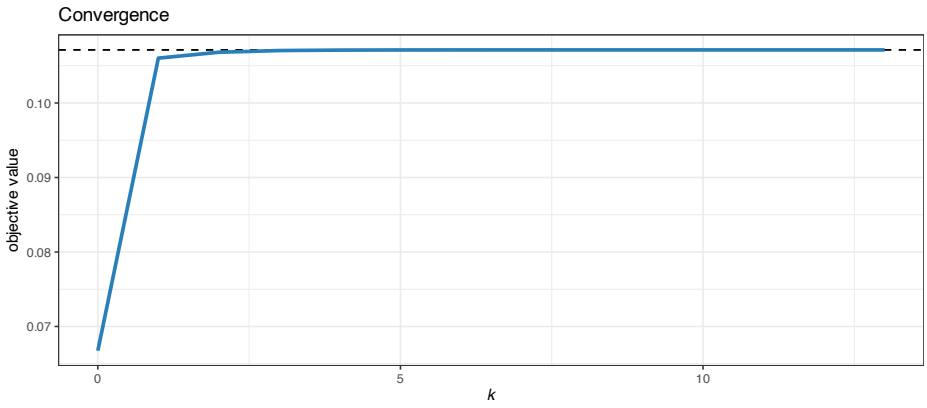


Figure 7.7 Convergence of the SQP-MVP algorithm for the MSRP formulation.

Example 7.5 (Mean–volatility portfolio) Consider the mean–volatility formulation

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} - \kappa\sqrt{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}} \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array}$$

where κ is a given positive number to control the risk aversion. This problem is a convex SOCP and can be solved with an SOCP solver. Alternatively, it can be solved via a sequence of simple QPs with Algorithm 7.3 (using $\tau^k = 0$ and $\gamma^k = 1$) as follows. From Table 7.4, we obtain $\alpha^k = 1$ and $\beta^k = \kappa/\sqrt{(\boldsymbol{w}^k)^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}^k}$, which leads to the surrogate mean–variance problem

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} - \dfrac{\lambda^k}{2}\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w} \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array}$$

where $\lambda^k = \beta^k/\alpha^k = \kappa/\sqrt{(\boldsymbol{w}^k)^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}^k}$. Figure 7.8 shows the convergence of this numerical method compared with the solution via an SOCP solver; basically it converges in one iteration.

7.5 Drawbacks

Markowitz's mean–variance portfolio, while great in theory, is dangerous in practice, so practitioners use it with caution. It has even been referred to as the "Markowitz optimization enigma" and sometimes the portfolio optimization is referred to as "error maximizer" (Michaud, 1989). There are multiple reasons for this, and academics and practitioners have proposed improvements and alternatives since Markowitz's seminal paper (Markowitz, 1952). As stated in Michaud (1989):

> . . . it remains one of the outstanding puzzles of modern finance that MV optimization has yet to meet with widespread acceptance by the investment community. . .

The extensions and improvements to the basic MVP are endless. To name a few: improved parameter estimation via shrinkage estimators or Black–Litterman-style approaches, robust

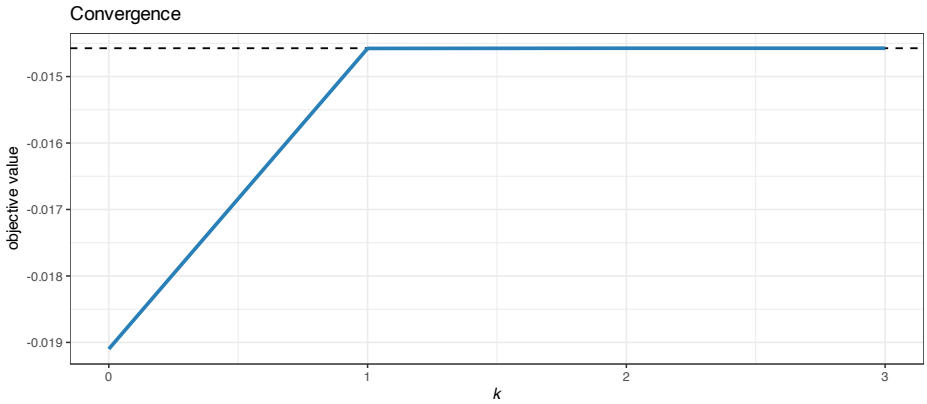


Figure 7.8 Convergence of the SQP-MVP algorithm for the mean–volatility formulation.

portfolio optimization, alternative measures of risk, modeling of transaction costs, and multi-period portfolio optimization (see Kolm et al. (2014)). We next elaborate on some of these points.

7.5.1 Noisy Estimation of the Expected Returns

It has long been recognized that mean–variance efficient portfolios constructed using sample means and sample covariance matrices perform poorly out of sample. The primary reason is that the sample mean is an extremely imprecise estimator of the population mean $\boldsymbol{\mu}$ (Chopra & Ziemba, 1993). The estimation of the covariance matrix $\boldsymbol{\Sigma}$ also contains errors but the magnitude of such errors (and their effect in the portfolio) is not comparable to that of the errors in the estimation of $\boldsymbol{\mu}$. For example, in Jagannathan and Ma (2003) it is stated:

> The estimation error in the sample mean is so large nothing much is lost in ignoring the mean altogether when no further information about the population mean is available. For example, the global minimum variance portfolio has as large an out-of-sample Sharpe ratio as other efficient portfolios when past historical average returns are used as proxies for expected returns.

Figure 7.9 illustrates how unstable the MVP is by showing several realizations of the portfolio under different resamplings of the returns used to estimate the expected returns.

This motivates the pragmatic employment of the risk-based portfolios considered in Section 6.5, which do not make use of $\boldsymbol{\mu}$ at all, or the use of the $1/N$ portfolio introduced in Section 6.4.3. Alternatively, Section 7.1.5 explores several heuristic constraints that can help improve the practical performance of the otherwise unstable MVP. Indeed, as stated in Michaud (1989):

> The major problem with MV optimization is its tendency to maximize the effects of errors in the input assumptions. Unconstrained MV optimization can yield results that are inferior to those of simple equal-weighting schemes.

There are at least two nonexclusive ways to attempt to fix this problem of the noise in the estimated parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and its effect in the portfolio: improved estimators and robust optimization. One way is by improving the estimation process to reduce the noise as

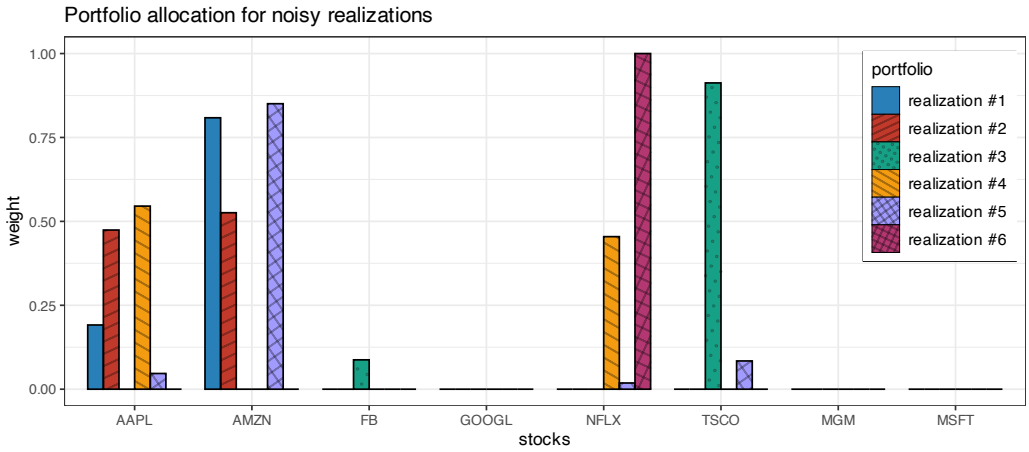


Figure 7.9 Effect of parameter estimation noise in the MVP allocation.

explored in Chapter 3. This includes the use of prior information in the Black–Litterman framework or via shrinkage, as well as employing better statistical models for the data such as heavy-tailed distributions (as opposed to the Gaussian or normal distribution). Another way is to embrace the fact that the parameters contain noise and employ statistical techniques such as bootstrapping or resampling, as well as the more sophisticated robust optimization techniques explored in Chapter 14.

7.5.2 Variance or Volatility as Measure of Risk

In Markowtiz's mean–variance portfolio, the risk is measured by the variance or, equivalently, by the volatility (Markowitz, 1952). The rationale is that a higher variance means that there are large peaks in the return distribution which may cause big losses. However, Markowitz himself already recognized and stressed the limitations of mean–variance analysis (Markowitz, 1959).

To start with, the variance is not a good measure of risk in practice since it penalizes both the unwanted losses and the desired negative losses (i.e., positive gains). Indeed, the mean–variance portfolio framework penalizes up-side and down-side risk equally, whereas most investors do not mind up-side risk. In addition, the variance only measures the width of the main mode of the probability distribution, whereas the most critical part lies in the tail of big losses (see Figure 6.10). The solution is to use alternative measures for risk such as downside risk, semi-variance, VaR, CVaR, and drawdown (McNeil et al., 2015); see Section 6.3.

Chapter 10 explores portfolio formulations under alternative risk measures such as downside risk, semi-variance, VaR, CVaR, and drawdown.

7.5.3 Single-Number Measure of Risk

Regardless of the choice of risk measure (i.e., variance, volatility, downside risk, semi-variance, VaR, CVaR, or worst drawdown), the risk of the portfolio is measured by a single number. This

risk characterization may not be enough to properly understand the risk contribution from the different assets, which leads us to the concept of *risk diversification* to avoid concentration of risk into a few assets (e.g., as was observed during the 2008 financial crisis).

Particularly, the *risk parity portfolio*, considered in Chapter 11, delves into the decomposition of the overall risk into contributions from each of the assets (Qian, 2005) and allows proper risk diversification.

7.6 Summary

- In 1952, Markowitz published a seminal paper that initiated the era referred to as modern portfolio theory, which postulates the design of a portfolio in terms of expected return and variance as a measure of risk.
- This mean–variance formulation is in the form of a convex problem that can be efficiently solved and has remained central in portfolio optimization. Rather than a unique solution, it produces an efficient frontier of portfolios with different risk profiles.
- Unfortunately, it performs poorly in practice due to a multitude of reasons, such as the sensitivity to errors in the parameters that characterize the market (i.e., the expected return vector and covariance matrix) or the simplistic characterization of risk via the variance or volatility.
- To overcome these drawbacks, practitioners have come up with a variety of tricks and improvements, namely, adding heuristic constraints to control the solution, improving the estimators of the market parameters (such as shrinkage estimators or robust estimators), using alternative measures of risk, characterizing the risk with a more refined risk-profile vector, and so on.
- One particular solution of interest that lies on the efficient frontier is the portfolio that maximizes the Sharpe ratio. Its formulation leads to a nonconvex problem with potentially difficult resolution. Fortunately, a number of practical numerical methods exist that produce the optimal solution.
- The Kelly criterion portfolio and, more generally, expected utility portfolios are generalizations of the formulation of the trade-off between expected return and risk. In practice, however, they are closely approximated by the mean–variance framework and efficient numerical algorithms are available.

Exercises

7.1 (Efficient frontier)

a. Download market data corresponding to N assets (e.g., stocks or cryptocurrencies) during a period with T observations, $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T \in \mathbb{R}^N$.
b. Estimate the expected return vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.
c. Plot the mean–volatility efficient frontier computed by solving different mean–variance formulations, namely:

- the scalarized form:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\top \boldsymbol{\mu} - \frac{\lambda}{2} \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w} \\ \text{subject to} & \mathbf{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}; \end{array}$$

- the variance-constrained form:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\top \boldsymbol{\mu} \\ \text{subject to} & \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w} \leq \alpha, \\ & \mathbf{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}; \end{array}$$

- the expected return-constrained form:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w} \\ \text{subject to} & \boldsymbol{w}^\top \boldsymbol{\mu} \geq \beta, \\ & \mathbf{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}. \end{array}$$

d. Discuss the benefits and drawbacks of the three methods for calculating the efficient frontier.

7.2 (Efficient frontier with practical constraints) Repeat Exercise 7.1 including different realistic constraints and discuss the differences. In particular:

- leverage constraint: $\|\boldsymbol{w}\|_1 \leq \gamma$
- turnover constraint: $\|\boldsymbol{w} - \boldsymbol{w}_0\|_1 \leq \tau$
- max position constraint: $|\boldsymbol{w}| \leq \boldsymbol{u}$
- market neutral constraint: $\boldsymbol{\beta}^\top \boldsymbol{w} = 0$
- sparsity constraint: $\|\boldsymbol{w}\|_0 \leq K$.

7.3 (Efficient frontier out of sample)

a. Download market data corresponding to N assets during a period with T observations.
b. Using 70% of the data:
   - estimate the expected return vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$;
   - plot the mean–volatility efficient frontier by solving mean–variance formulations; and
   - plot some randomly generated feasible portfolios.
c. Using the remaining 30% of the data (out of sample):
   - estimate the expected return vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$;
   - plot the new mean–volatility efficient frontier; and
   - re-evaluate and plot the mean and volatility of the previously computed portfolios (the ones defining the efficient frontier and the random ones).
d. Discuss the difference between the two efficient frontiers, as well as how the portfolios shift from in-sample to out-of-sample performance.

7.4 (Improving the mean–variance portfolio with heuristics) Repeat Exercise 7.3 including the following heuristic constraints to regularize the mean–variance portfolios:

- upper bound constraint: $\|\boldsymbol{w}\|_\infty \leq 0.25$
- diversification constraint: $\|\boldsymbol{w}\|_2^2 \leq 0.25$.

7.5 (Computation of the MSRP)

a. Download market data corresponding to N assets during a period with T observations.
b. Estimate the expected return vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.
c. Compute the maximum Sharpe ratio portfolio

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \dfrac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} - r_{\mathrm{f}}}{\sqrt{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}}} \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array}$$

with the following methods:
- bisection method
- Dinkelbach method
- Schaible transform method.

7.6 (Kelly portfolio)

a. Download market data corresponding to N assets during a period with T observations.
b. Compute the Kelly portfolio

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \mathbb{E}\left[\log\left(1 + \boldsymbol{w}^{\mathsf{T}}\boldsymbol{r}\right)\right] \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array}$$

with the following methods:
- sample average approximation
- mean–variance approximation
- Levy–Markowitz approximation.

7.7 (Expected utility portfolio)

a. Download market data corresponding to N assets during a period with T observations.
b. Compute the expected utility portfolio

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \mathbb{E}\left[U(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{r})\right] \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array}$$

with different utilities such as
- $U(x) = \log(1 + x)$
- $U(x) = \sqrt{1 + x}$
- $U(x) = -1/x$
- $U(x) = -p/x^p$ with $p > 0$
- $U(x) = -1/\sqrt{1 + x}$
- $U(x) = 1 - \exp(-\lambda x)$ with $\lambda > 0$.

7.8 (Universal successive mean–variance approximation method)

a. Consider the maximum Sharpe ratio portfolio,

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \dfrac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} - r_{\mathrm{f}}}{\sqrt{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}}} \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array}$$

and the mean–volatility portfolio,

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} - \kappa\sqrt{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}} \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array}$$

both of which lie on the efficient frontier.

b. Solve them with some appropriate method.
c. Solve them via the universal successive mean–variance approximation method, which at each iteration k, solves the mean–variance problem

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} - \dfrac{\lambda^k}{2}\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w} \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array}$$

with a properly chosen λ^k.
d. Compare the obtained solutions and the computational cost.

References

Cajas, D. (2021). Kelly portfolio optimization: A disciplined convex programming framework. *SSRN Electronic Journal*. https://dx.doi.org/10.2139/ssrn.3833617

Cajas, D. (2026). Riskfolio-Lib [Python library]. https://github.com/dcajasn/Riskfolio-Lib

Chares, R. (2007). *Cones and Interior-point Algorithms for Structured Convex Optimization Involving Powers and Exponentials* [Doctoral dissertation, Université Catholique de Louvain and École Polytechnique de Louvain].

Chopra, V., & Ziemba, W. (1993). The effect of errors in means, variances and covariances on optimal portfolio choice. *Journal of Portfolio Management*, *19*(2), 6–11.

Cover, T., & Thomas, J. (1991). *Elements of Information Theory*. John Wiley & Sons.

Delatte, H., Nicolini, C., & Manzi, M. (2026). skfolio [Computer software]. https://doi.org/10.5281/zenodo.16148630

DeMiguel, V., Garlappi, L., Nogales, F. J., & Uppal, R. (2009). A generalized approach to portfolio optimization: Improving performance by constraining portfolio norms. *Management Science*, *55*(5), 798–812.

Dinkelbach, W. (1967). On nonlinear fractional programming. *Management Science*, *133*(7), 492–498.

Jagannathan, R., & Ma, T. (2003). Risk reduction in large portfolios: Why imposing the wrong constraints helps. *The Journal of Finance*, *58*(4), 1651–1684.

Kelly, Jr., J. L. (1956). A new interpretation of information rate. *The Bell System Technical Journal*, *35*(4), 917–926.

Kolm, P. N., Tütüncü, R., & Fabozzi, F. J. (2014). 60 years of portfolio optimization: Practical challenges and current trends. *European Journal of Operational Research*, *234*(2), 356–371.

Levy, H., & Markowitz, H. M. (1979). Approximating expected utility by a function of mean and variance. *The American Economic Review*, *69*(3), 308–317.

MacLean, L., Thorp, E., & Ziemba, W. (2010). Long-term capital growth: The good and bad properties of the Kelly and fractional Kelly capital growth criteria. *Quantitative Finance*, *10*(7), 681–687.

Markowitz, H. M. (1952). Portfolio selection. *The Journal of Finance*, *7*(1), 77–91.

Markowitz, H. M. (1959). *Portfolio Selection: Efficient Diversification of Investments*. John Wiley & Sons.

Markowitz, H. M. (2014). Mean–variance approximations to expected utility. *European Journal of Operational Research*, *234*(2), 346–355.

Martin, R. A. (2021). PyPortfolioOpt: Portfolio optimization in Python. *Journal of Open Source Software*, *6*(61), 1–5.

McNeil, A. J., Frey, R., & Embrechts, P. (2015). *Quantitative Risk Management*. Princeton University Press.

Michaud, R. O. (1989). The Markowitz optimization enigma: Is "optimized" optimal? *Financial Analysts Journal*, *45*(1), 31–42.

Nicolini, C., Manzi, M., & Delatte, H. (2025). skfolio: Portfolio optimization in Python. *Proceedings of the International Research and Industry Symposium on AI (IRIS-AI 25)*, 35–41. https://doi.org/10.65236/iris-ai/proceedings/2025/article/3

Pulley, L. B. (1983). Mean–variance approximations to expected logarithmic utility. *Operations Research*, *31*(4), 685–696.

Qian, E. (2005). Risk parity portfolios: Efficient portfolios through true diversification. *PanAgora Asset Management*.

Roy, A. (1952). Safety first and the holding of assets. *Econometrica*, *20*(3), 431–449.

Rubinstein, M. (2002). Markowitz's "portfolio selection": A fifty-year retrospective. *The Journal of Finance*, *57*(3), 1041–1045.

Savage, L. J. (1954). *The Foundations of Statistics*. John Wiley & Sons.

Schaible, S. (1974). Parameter-free convex equivalent and dual programs of fractional programming problems. *Zeitschrift fur Operations Research*, *18*(5), 187–196.

Scutari, G., Facchinei, F., Song, P., Palomar, D. P., & Pang, J.-S. (2014). Decomposition by partial linearization: Parallel optimization of multi-agent systems. *IEEE Transactions on Signal Processing*, *62*(3), 641–656.

Sharpe, W. F. (1966). Mutual fund performance. *The Journal of Business*, *39*(1), 119–138.

Thorp, E. O. (1962). *Beat the Dealer: A Winning Strategy for the Game of Twenty-One*. Blaisdell Publishing.

Thorp, E. O. (1971). Portfolio choice and the Kelly criterion [Reprinted in Stochastic Optimization Models in Finance, W. Ziemba and R. Vickson (Eds.), Academic Press (1975)]. *Business and Economics Statistics Section, Proceedings of the American Statistical Association*, 215–224.

Thorp, E. O. (1997). The Kelly criterion in Blackjack, sports betting, and the stock market. *Proceedings of the 10th International Conference on Gambling and Risk Taking*.

von Neumann, J., & Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press.

Wuertz, D., Setz, T., Chalabi, Y., Chen, W., & Theussl, S. (2023). *fPortfolio: Rmetrics – Portfolio Selection and Optimization* [R package]. https://cran.r-project.org/package=fPortfolio

Xiu, S., Wang, X., & Palomar, D. P. (2023). A fast successive QP algorithm for general mean–variance portfolio optimization. *IEEE Transactions on Signal Processing*, *71*, 2713–2727.

8

Portfolio Backtesting

"I am a dreamer. I know so little of real life that I just can't help re-living such moments as these in my dreams, for such moments are something I have very rarely experienced. I am going to dream about you the whole night, the whole week, the whole year."

— Fyodor Dostoyevsky, *White Nights*

A backtest is a historical simulation of how a strategy would have performed had it been run over a past period of time. It is an essential step prior to actual live trading with real money. Nevertheless, backtesting is one of the least understood techniques in the quant's toolbox.[1] The reality is that backtesting is full of dangers and virtually impossible to execute properly. This chapter will explore portfolio backtesting in detail, so that we become aware of all the potential pitfalls.

8.1 A Typical Backtest

A backtest is a historical simulation of a strategy in some past period of time. We can see backtest results in academic publications, fund brochures, practitioner blogs, and so on.

Since strategies typically require the estimation of some parameters, such as the assets' expected return vector $\boldsymbol{\mu}$ or covariance matrix $\boldsymbol{\Sigma}$, the data is commonly split into an *in-sample* dataset, which acts as historical data that can be used to estimate parameters, and an *out-of-sample* dataset, which serves as "future" data that is used to assess the performance.

As an illustrative example, suppose we want to evaluate three portfolios: the $1/N$ portfolio (see Section 6.4.3), the inverse volatility portfolio (IVolP) (see Section 6.5.2), and the global minimum variance portfolio (GMVP) (see Section 6.5.1). Figure 8.1 shows the cumulative P&L and drawdown of these portfolios. This gives an assessment of the behavior of the portfolios over time. In addition, Table 8.1 provides more concrete numerical values of different performance measures over the whole period.

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

[1] A quant is a financial professional who uses complex mathematical models, computer algorithms, and statistical analysis to analyze markets, price securities, and identify trading opportunities.

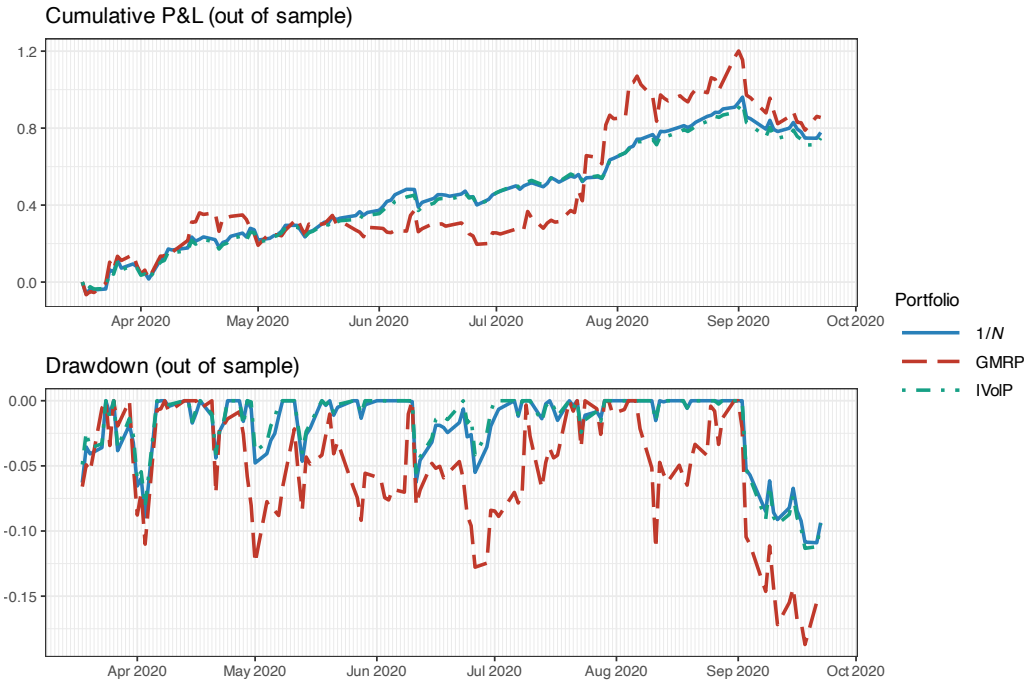


Figure 8.1 Example of a backtest result in the form of cumulative P&L and drawdown plots.

Table 8.1 Example of a backtest result in the form of performance measures.

Portfolio	Sharpe ratio	Annual return	Annual volatility	Sortino ratio	Max drawdown	CVaR (0.95)
1/*N*	3.23	117%	36%	5.40	11%	5%
GMRP	2.19	138%	63%	4.09	19%	7%
IVolP	3.35	113%	34%	5.61	11%	4%

Of course, more detailed results could be provided, such as a rolling Sharpe ratio plot over time (see Section 6.3.4) or a table with performance measures on a monthly basis instead of the overall annualized values. The reader is referred to Section 6.3 for a list of common performance measures.

The Global Investment Performance Standard (GIPS)[2] is a set of standardized, industry-wide ethical principles that apply to the way investment performance is presented to potential and existing clients of asset managers, regulators, pension funds, financial advisers, and financial companies from around the globe. These standards guide investment firms on how to calculate and present their investment results to prospective clients. GIPS are standards, not laws. Firms do not have to be GIPS compliant; however, the standards provide discipline and claiming

[2] www.gipsstandards.org

compliance with them demonstrates a firm-wide commitment to ethical best practices and that the firm employs strong internal control processes.

Nevertheless, the fact of the matter is that all these backtest results only provide very limited information on the real performance of the portfolios and, even worse, most likely the results are faulty and misleading. Indeed, this is clearly stated by notable authors, such as Harvey et al. (2016): "Most claimed research findings in financial economics are likely false," and López de Prado (2018), "Most backtests published in journals are flawed, as the result of selection bias on multiple tests."

Sections 8.2 and 8.3 explore the many ways in which backtests are misleading and can provide wrong results. Then, Sections 8.4 and 8.5 go over details on how to execute backtests as safe as possible. A brief summary is then given in Section 8.6.

8.2 The Seven Sins of Quantitative Investing

In 2005, a practitioner report compiled the "Seven Sins of Fund Management" (Montier, 2005) (of which sins #1 and #5 are the most directly related to backtesting):

- Sin #1: Forecasting (Pride)
- Sin #2: The illusion of knowledge (Gluttony)
- Sin #3: Meeting companies (Lust)
- Sin #4: Thinking you can out-smart everyone else (Envy)
- Sin #5: Short time horizons and overtrading (Avarice)
- Sin #6: Believing everything you read (Sloth)
- Sin #7: Group-based decisions (Wrath).

In 2014, a team of quants at Deutsche Bank published a study under the suggestive title "Seven Sins of Quantitative Investing" (Luo et al., 2014). These seven sins are a few basic backtesting errors that most journal publications make routinely:

- Sin #1: Survivorship bias
- Sin #2: Look-ahead bias
- Sin #3: Storytelling bias
- Sin #4: Overfitting and data snooping bias
- Sin #5: Turnover and transaction cost
- Sin #6: Outliers
- Sin #7: Asymmetric pattern and shorting cost.

In the following we will go over these seven sins of quantitative investing with illustrative examples.

8.2.1 Sin #1: Survivorship Bias

Survivorship bias is one of the common mistakes investors tend to make. Most people are aware of this bias, but few understand its significance.

Practitioners tend to backtest investment strategies using only those companies that are

currently in business and still performing well, most likely listed in some index, such as the S&P 500 stock index. By doing that, they are ignoring stocks that have left the investment universe due to bankruptcy, delisting, being acquired, or simply underperforming the index.

In simple words, survivorship bias happens when you do not take into account stocks that you know in advance will not perform well in the future of the backtest period. Similarly, simply removing stocks from the universe because they have missing values in their data may have a misleading effect.

In fact, most available databases suffer from survivorship bias. In this context, the Center for Research in Security Prices (CRSP)[3] maintains a comprehensive database of historical stock market data that is considered highly reliable and accurate, making it a valuable resource for those studying finance and investment. CRSP is widely used by academic researchers and financial professionals for conducting empirical research, analyzing historical stock market trends, and developing investment strategies. Figure 8.2 shows the effect of survivorship bias on the $1/N$ portfolio on the S&P 500 stocks over several years.

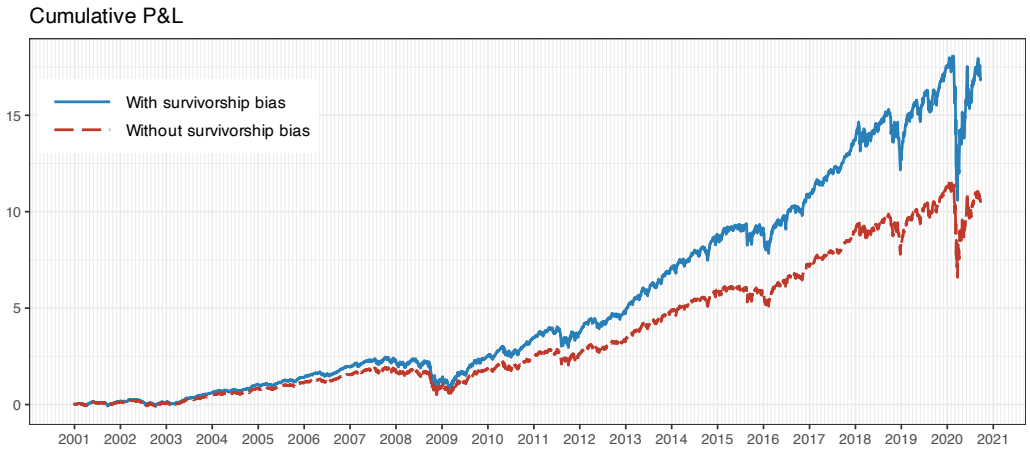


Figure 8.2 Effect of survivorship bias on the S&P 500 stocks.

Interestingly, the concept of survivorship bias does not happen uniquely in financial investment; it is rather a persistent phenomenon in many other areas. An illustrative example is that of modern-day billionaires who dropped out of college and went on to become highly successful (e.g., Bill Gates and Mark Zuckerberg). These few success stories distort people's perceptions because they ignore the majority of college dropouts who are not billionaires.

8.2.2 Sin #2: Look-Ahead Bias

Look-ahead bias is the bias created by using information or data that were unknown or unavailable at the time when the backtesting was conducted. It is a very common bias in backtesting.

[3] www.crsp.org

An obvious example of look-ahead bias lies in companies' financial statement data. One has to be certain about the timestamp for each data point and take into account release dates, distribution delays, and backfill corrections.

A less conspicuous example of look-ahead bias comes from coding errors. This may happen, for instance, when training the parameters of a system using future information or simply when data is pre-processed with statistics collected from the whole block of data. Another common error comes from time alignment errors in the backtesting code. In more detail, computing the portfolio return as $R_t^{\text{portf}} = \boldsymbol{w}_t^{\mathrm{T}} \boldsymbol{r}_t$ would be totally incorrect (see (6.5) for the correct expression) since the design of the portfolio $\boldsymbol{w}_t$ used information up to time t and the returns $\boldsymbol{r}_t = (\boldsymbol{p}_t - \boldsymbol{p}_{t-1}) \oslash \boldsymbol{p}_{t-1}$ implicitly assume that the position was executed at time $t-1$ (of course this argument becomes invalid if the portfolio $\boldsymbol{w}_t$ is assumed to use information only up to $t-1$, or if the returns are defined with a time lag as $\boldsymbol{r}_t = (\boldsymbol{p}_{t+1} - \boldsymbol{p}_t) \oslash \boldsymbol{p}_t$). An illustration of this issue can be found in Glabadanidis (2015) as explained in Zakamulin (2018), where the seemingly amazing performance of a strategy based on moving-average indicators vanishes completely under a proper backtest.

Figure 8.3 illustrates the effect of look-ahead bias (from a time alignment mistake when computing the returns) when trading a single stock with a simple strategy based on a moving average (to be exact, buying when the price is above the moving average of the past 10 values).

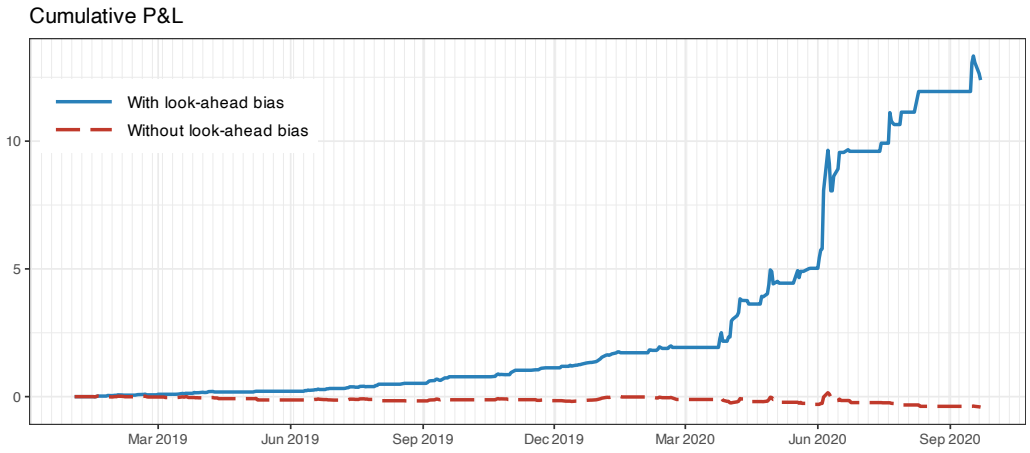


Figure 8.3 Effect of look-ahead bias (from a time alignment mistake) trading a single stock.

8.2.3 Sin #3: Storytelling Bias

We all love stories. It is believed that storytelling played a key role in the evolution of the human species. In fact, most bestselling popular science books are based on anecdotal stories, which may or may not have a corresponding solid statistical foundation. This is simply because they appeal to the general public, while statistics do not. Indeed, one of the most important ways to leave a deep impression with your audience is to tell stories rather than simply repeating facts and numbers.

Storytelling bias in financial data (or any other type of data, for that matter) happens when we make up a story ex post (i.e., after the observation of a particular event) to justify some random pattern. This is related to *confirmation bias*, which consists of favoring information that supports one's pre-existing beliefs and ignoring contradicting evidence. Storytelling is pervasive in financial news, where supposed "experts" can justify any random pattern after the fact.

The antidote to storytelling bias is the collection of more historical data to see if the story passes a statistical test or the test of time. Unfortunately, in contrast to fields like physics, economics and finance have a limited number of observations, which hinders the resolution of storytelling bias.

Figure 8.4 illustrates the effect of story-telling bias by trading a single stock with a position indicated by a random binary sequence (which might have been generated by "Paul the Octopus"[4]). Before August 2018 one could be inclined to believe the story that the random sequence was a good predictor of the stock trend; however, eventually, when more data is collected, one has to come to the inevitable conclusion that it was a fluke.

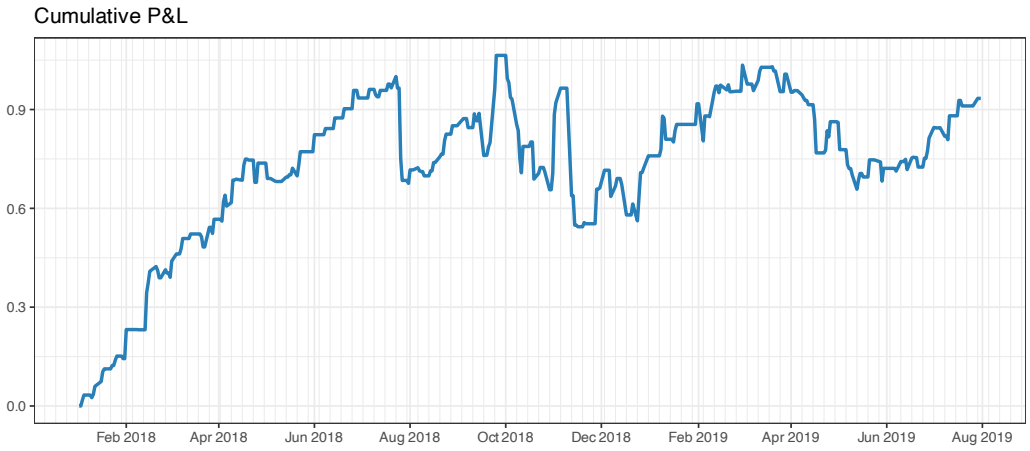


Figure 8.4 Effect of story-telling bias in the form of a random strategy that performs amazingly well until August 2018, but not afterwards.

8.2.4 Sin #4: Overfitting and Data Snooping Bias

In the fields of computer science and statistics, *data mining* refers to the computational process of discovering patterns in large data sets, often involving sophisticated statistical techniques, computation algorithms, and large-scale database systems. In principle, there is nothing negative about data mining. In finance, however, it often means manipulating data or models to find the desired pattern that an analyst wants to show.

[4] Paul the Octopus was a common octopus used to predict the results of association football matches. Accurate predictions in the 2010 World Cup brought him worldwide attention as an animal oracle. https://en.wikipedia.org/wiki/Paul_the_Octopus

Data snooping bias in finance (also loosely referred to as data mining bias) refers to the behavior of extensively searching for patterns or rules so that a model fits the data perfectly. Analysts often fine tune the parameters of their models and choose the ones that perform well in the backtesting. This is also referred to as *overfitting* in the machine learning literature.

With enough data manipulation, one can almost always find a model that performs very well on that data. This is why it is standard practice to split the data into in-sample and out-of-sample data: the in-sample data is used to estimate or train the model and design the portfolio, whereas the out-of-sample data is used to evaluate and test the portfolio. They are also referred to as *training data* and *test data*, respectively.

With the separation of data into training data and test data, it seems we should be safe; unfortunately, this is not the case. It is almost inevitable for the person or team researching a strategy to iterate the process by which the strategy under design is evaluated with the test data and then some adjustments are performed on the strategy. This is a vicious cycle that leads to catastrophic results. By doing that, the test data has inadvertently been used too many times and it has effectively become part of the training data. Unfortunately, this happens in almost all the publications to the point that one cannot trust published results: "Most backtests published in journals are flawed, as the result of selection bias on multiple tests" (López de Prado, 2018).

In other words, looking long and hard enough at a given dataset will often reveal one or more models that seems promising but are in fact spurious (White, 2000).

One piece of advice that may help avoid overfitting is to avoid fine tuning the parameters (Chan, 2008). Even better, one could perform a sensitivity analysis on the parameters. That is, if a small change in some parameter affects the performance drastically, it is an indication that the strategy is too sensitive and lacks robustness. A sensitive strategy is dangerous because one cannot assess the future behavior with new data with any degree of confidence.

An illustrative example, provided in Arnott et al. (2019), shows a strategy with an impressive backtest result, only to reveal that the strategy is a preposterous long–short quintile portfolio that goes long on the stocks with an "S" on the third letter of the ticker symbol and goes short on the ones with the letter "U".

Data snooping bias or overfitting is probably the most difficult bias to deal with. The ultimate test after a strategy or portfolio has been designed is to trade it with new data. This can be done in three different levels of accuracy: (i) simply wait to collect new data and then perform a proper backtest; (ii) use *paper trading* offered by most brokers, which consists of a realistic trading simulation without real money; and (iii) trade with real money, that is, *live trading*, albeit typically starting with a small budget.

Figure 8.5 illustrates the effect of data snooping or overfitting by trading a single stock with a strategy based on machine learning. In particular, a linear return forecast with a lookback window of 10 values is trained in two scenarios: using only the training data (as it should be) and using the training + test data (obviously, this produces overfitting). The overfitted backtest seems to indicate prediction power in the out of sample, whereas the reality could not be further from the truth.

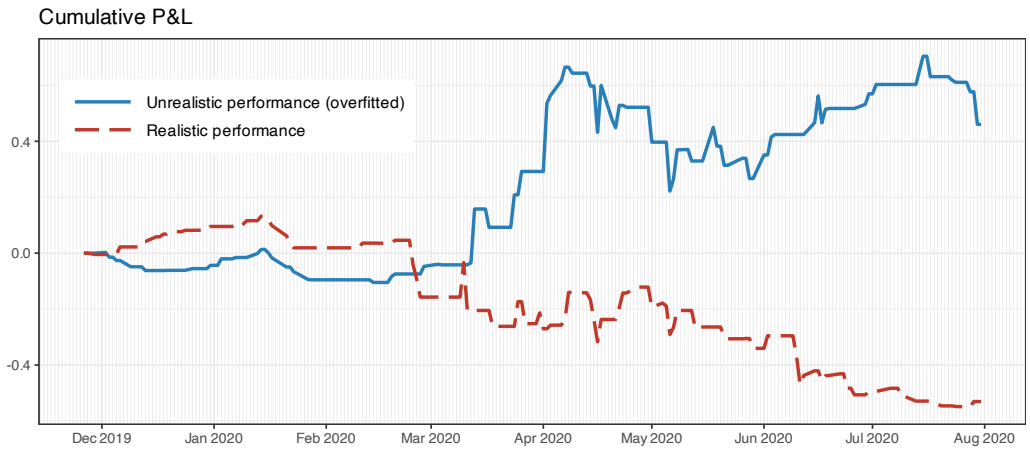


Figure 8.5 Effect of data snooping or overfitting on a backtest after tweaking the strategy too many times.

8.2.5 Sin #5: Turnover and Transaction Cost

Backtesting is often conducted in an ideal world: no transaction cost, no turnover constraint, unlimited long and short availability, and perfect liquidity. In reality, all investors are limited by some constraints. We now focus on the turnover and the associated transaction cost.

Turnover refers to the overall amount of orders to be executed when rebalancing the portfolio from $\boldsymbol{w}_t$ to $\boldsymbol{w}_t^{\text{reb}}$, and is calculated as $\|\boldsymbol{w}_t^{\text{reb}} - \boldsymbol{w}_t\|_1$. As a first approximation, the transaction costs can be modeled as proportional to the turnover (see Section 6.1.4 for details). However, if the liquidity is not enough compared to the size of the turnover, then slippage may have a significant effect. This becomes more relevant as the rebalancing frequency increases. In the limit, simulating the transaction cost at the level of the limit order book can be extremely challenging and the only way to be certain about the cost incurred is to actually trade.

Figure 8.6 illustrates the detrimental effect of transaction costs on the daily-rebalanced inverse volatility portfolio on the S&P 500 stocks with fees of 60 bps. The effect of transaction costs slowly accumulates over time.

The overall transaction cost depends on the turnover per rebalancing and the rebalancing frequency. Portfolios with a high rebalancing frequency are more prone to have an overall large turnover, which translates into high transaction costs. To be on the safe side, either the rebalancing frequency should be kept to a minimum or the turnover per rebalancing should be controlled (see Section 6.1.5). Nevertheless, having too slow a rebalancing frequency may lead to a portfolio that fails to adapt to the changing signal. Thus, deciding the rebalancing frequency of a strategy is a critical step in practice (see Section 6.1.5 in Chapter 6 for details).

8.2.6 Sin #6: Outliers

Outliers are events that do not fit the normal and expected behavior. They are not too uncommon in financial data and they can happen due to different reasons. Some outliers



Figure 8.6 Effect of transaction costs on a portfolio (with daily rebalancing and fees of 60 bps).

reflect a reality that actually happened due to some historical event. Others may be an artifact of the data itself, perhaps due to a momentaneous lack of liquidity of the market, or an abnormally large execution order, or even some error in the data.

In principle, outliers cannot be predicted and one can only try to be robust to them; this is why robust estimation methods (as in Chapter 3) and robust portfolio techniques (as in Chapter 14) are important in practice.

The danger when it comes to backtesting is to accidentally benefit from a few outliers, because that would distort the realistic assessment of a portfolio. More often than not, outliers are caused by data errors or specific events that are unlikely to be repeated in the future. Thus, one should not base the success of a portfolio on a few historical outliers, as the future performance will then rely on the realization of similar outliers. How should we then treat outliers in the historical data?

One way to deal with outliers is to control them so that they do not distort the backtest results. Traditional outlier control techniques include: winsorization (capping data at certain percentiles) and truncation/trimming/censoring (removing outliers from data sample). The data normalization process is closely related to outlier control.

Another way to deal with outliers is to keep the outliers, but making sure the potential success of the backtested strategies does not rely on them (unless one is actually trying to design a strategy solely based on outliers).

Figure 8.7 illustrates the effect of outlier control in the design phase of a quintile portfolio with hourly cryptocurrency data. In this case, outliers are removed if they are larger than 5% (recall these are hourly returns).

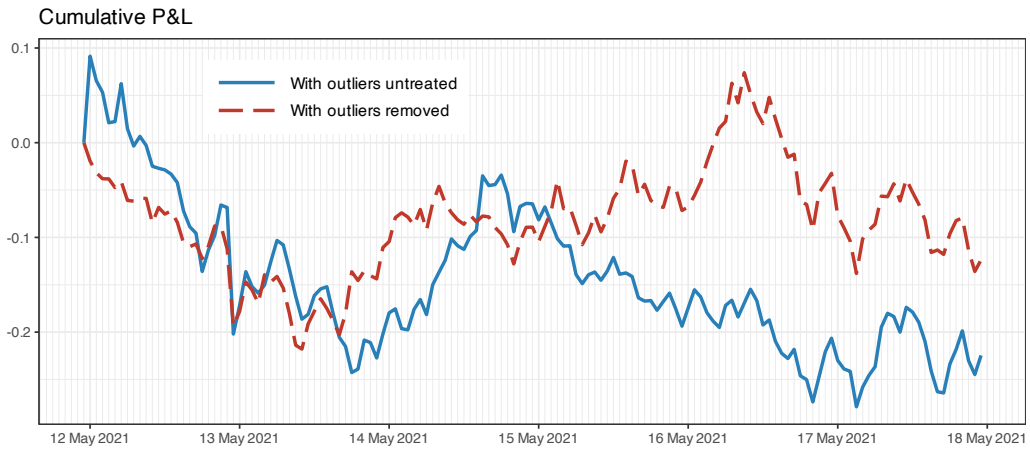


Figure 8.7 Effect of outliers on a backtest with hourly cryptocurrency data.

8.2.7 Sin #7: Asymmetric Pattern and Shorting Cost

In typical backtesting, analysts generally assume they can short any stocks at no cost or at the same level of cost. However, we need to be aware that borrowing cost can be prohibitively high for some stocks, while on other occasions, it could be impossible to locate the borrowing. For certain stocks, industries, or countries, there could also be government or exchange imposed rules that prohibit any shorting at all. Indeed, some countries do not allow short selling at all, while others limit its extent.[5]

For example, in the US market, during the 2008 global financial crisis period borrowing costs sky-rocketed – some financial stocks were even banned from being shorted, reflected by higher percentages of expensive-to-borrow stocks during this episode.

Interestingly, the effect of short availability not only affects backtests of portfolios that actually short sell but long positions may also suffer from the so-called "limited arbitrage" argument, by which arbitrageurs are prevented from immediately forcing prices to fair values.

How much difference would it make if we cannot short those hard-to-borrow stocks? Figure 8.8 shows the performance of two long–short quintile portfolios: an unrealistic portfolio that can perfectly long and short the top 20% and bottom 20% of the stocks, respectively, and a more realistic portfolio where only some easy-to-borrow stocks can actually be shorted (the definition of easy-to-borrow stocks used herein is a bit loose for illustration purposes).

8.3 The Dangers of Backtesting

We have learned from the "seven sins of quantitative investing" in Section 8.2 that backtesting is a dangerous process, fraught with many potential pitfalls. In fact, there are more than these

[5] In China, as of 2021, regulators only allow investors to short a portion of stocks traded on the Shanghai and Shenzhen stock exchanges. The list of stocks changes regularly and typically only includes companies with good fundamentals.

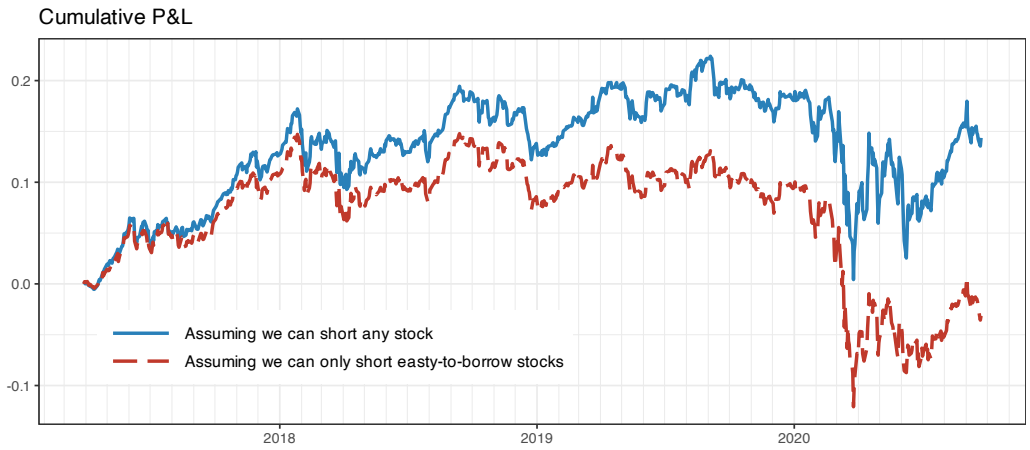


Figure 8.8 Effect of shorting availability in a long–short quintile portfolio.

seven types of errors one can make: "A full book could be written listing all the different errors people make while backtesting" (López de Prado, 2018).

Arguably, the most common mistake in backtesting involves overfitting or data snooping. The following quote by John von Neumann about the general concept of overfitting is quite amusing and illustrative: "With four parameters I can fit an elephant, and with five I can make him wiggle his trunk" (Mayer et al., 2010).

8.3.1 Backtest Overfitting

Overfitting is a concept borrowed from machine learning and refers to the situation when a model targets particular (noisy) observations rather than a general and persistent structure in the data. In the context of investment strategies, it takes place when a strategy is developed to perform well on a backtest, by monetizing random historical patterns. Because those random patterns are unlikely to occur again in the future, the strategy so developed will fail.

This is why the performance of a strategy in the training data (in-sample data) can be totally misleading and we need to resort to test data (out-of-sample data). However, even the performance in the test data can also be totally misleading (Bailey, Borwein, & López de Prado, 2016; Bailey et al., 2014). The reason is that, when a researcher is backtesting an investment strategy, it is only natural to try to adjust some of the strategy's parameters to see the effect in the backtest performance. By doing this, typically over and over again, it is inevitable that the test data has indirectly become part of the training data (or cross-validation data) and it is not really test data anymore. This leads to the unfounded belief that a portfolio is going to perform well only to find out when trading live that it does not live up to expectations. This also leads to publications with backtest results that are not representative of reality.

The reality is that it takes a relatively small number of trials to identify an investment strategy with a spuriously high backtested performance, especially for complex strategies. Bailey et al.

(2014) argued that "Not reporting the number of trials involved in identifying a successful backtest is a similar kind of fraud."

Indeed, what makes backtest overfitting so hard to assess is that the probability of false positives changes with every new test conducted on the same dataset. That information is either unknown by the researcher or not shared with investors or referees. The only backtests that most people share are those that portray supposedly winning investment strategies.

8.3.2 p-Hacking

In fact, backtest overfitting due to multiple testing is related to a more general phenomenon in statistics called "p-hacking," also known as cherry-picking, data dredging, significance chasing, significance questing, and selective inference. The term p-hacking refers to the misuse of data analysis to find patterns in data that can be presented as statistically significant, thus dramatically increasing and understating the risk of false positives. This is done by performing many statistical tests on the data and only reporting those that come back with significant results. We next elaborate on the concepts of p-value and p-hacking.

In hypothesis testing, one wants to determine whether the data really come from some candidate distribution, the so-called *null hypothesis*. This can be formally assessed via the p-value, which is the probability of obtaining the observed results under the assumption that the null hypothesis is correct. A small p-value means that there is strong evidence to reject the null hypothesis and accept the alternative hypothesis. Typical thresholds for determining whether a p-value is small enough are in the range 0.01–0.05. Thus, if the p-value is smaller than the threshold, then we can reject the null hypothesis that the data came from that distribution. The p-value is routinely used in all scientific areas, such as physics (to determine whether the data supports or rejects a hypothesis) or medicine (to determine the effectiveness of new drugs).

The term "p-hacking" refers to the dangerous practice of testing multiple hypotheses and only reporting (cherry-picking) the one that produces a small p-value. For example, a researcher may report a portfolio showing excellent results during the period 1970–2017, but does not reveal that the same result is weaker for the period 1960–2017. Similarly, a portfolio may look profitable using some specific universe of stocks, but it is not reported that a variation of the universe produced a degraded performance. The problem with this practice is that, when reporting the results, typically the number of experiments conducted is omitted. The reader then may wrongly infer that it was a single trial.

Indeed, most of the claimed research findings in financial economics are likely false due to p-hacking (Harvey, 2017; Harvey et al., 2016). For example, some observations from Harvey (2017) include: "Empirical research in financial economics relies too much on p-values, which are poorly understood in the first place" and "Journals want to publish papers with positive results and this incentivizes researchers to engage in data mining and p-hacking."

8.3.3 Backtests Are Not Experiments

Experiments, for example in physics, are conducted in a lab and can be repeated multiple times to control for different variables. In contrast, a backtest is a historical simulation of how a strategy would have performed in the past. Thus, a backtest is not an experiment, and it does not prove anything.

In fact, a backtest guarantees nothing, not even achieving that Sharpe ratio if we could travel back in time, simply because random draws would have been different and the past would not repeat itself (López de Prado, 2018).

8.3.4 The Paradox of Flawless Backtests

The irony of a backtest is that, even if it is flawless, it is probably wrong (López de Prado, 2018). Indeed, suppose you have implemented a flawless backtest (i.e., everyone can reproduce your results, you have considered more than the necessary slippage and transaction costs, etc.) and it still shows good performance. Unfortunately, this flawless backtest is still probably wrong. Why?

First of all, only an expert can produce a flawless backtest. This expert must have run a myriad of backtests over the years. So we need to account for the possibility that this is a false discovery, a statistical fluke that inevitably comes up after running multiple tests on the same dataset (i.e., overfitting).

The maddening thing about backtesting is that the better you become at it, the more likely false discoveries will pop up (López de Prado, 2018).

8.3.5 Limitations of Backtesting Insights

Backtesting provides us with very little insight into the reason why a particular strategy would have made money (López de Prado, 2018). Just as a lottery winner may feel he has done something to deserve his luck, there is always some ex post story.

Regarding financial data, many authors claim to have found hundreds of "alphas" and "factors," and there is always some convoluted explanation for them. Instead, what they have found are the lottery tickets that won the last game. Those authors never tell us about all the tickets that were sold, that is, the millions of simulations it took to find these "lucky" alphas.

8.3.6 What is the Point of Backtesting Then?

While backtesting cannot guarantee the future good performance of a strategy, it can serve the opposite purpose, that is, to identify strategies that underperform so that we can eliminate them.

In addition, a backtest can provide a sanity check on a number of variables, including bet sizing, turnover, resilience to costs, and behavior under a given scenario.

Thus, the purpose of a backtest is to discard bad models, not to improve them. It may sound

counter-intuitive, but one should not adjust the model based on the backtest results since it is a waste of time and it is dangerous due to overfitting.

One should invest time and effort developing a sound strategy. However, by the time backtests are performed, it is too late to modify the strategy. So never backtest until your model has been fully specified (López de Prado, 2018).

Summarizing, a good backtest can still be extremely helpful, but backtesting well is extremely hard to execute.

8.3.7 Recommendations to Avoid Overfitting

How to address backtest overfitting is arguably the most fundamental question in quantitative finance. While there is no easy way to prevent backtest overfitting, a number of recommendations were compiled in López de Prado (2018) and some are listed here for convenience:

- Develop models for entire asset classes or investment universes, rather than for specific securities, to reduce the probability of false discoveries.
- Apply model averaging (see Chapter 14 for details) as a means to both prevent overfitting and reduce the variance of the forecasting error.
- Do not backtest until all your research is complete (i.e., do not fall into the vicious cycle of keeping tweaking parameters and running the backtest over and over again).
- Keep track of the number of backtests conducted on a dataset so that the probability of backtest overfitting may be estimated and the Sharpe ratio may be properly deflated (Bailey & López de Prado, 2014).
- Apart from backtesting on historical data, consider simulating scenarios rather than history, such as stress tests (see Section 8.5 for details). Your strategy should be profitable under a wide range of scenarios, not just the anecdotal historical path.

A list of best research practices for backtesting were proposed in Arnott et al. (2019), such as:

- Establish an ex ante economic foundation: following the scientific method, a hypothesis is developed and the empirical tests attempt to find evidence inconsistent with the hypothesis.
- Beware an ex post economic foundation: it is also almost always a mistake to create an economic story – a rationale to justify the findings – after the data mining has occurred.
- Keep track of the multiple tests tried: this is needed to assess the statistical significance of the results; too many trials and any spurious result can be obtained.
- Define the test sample ex ante: the sample data and data transformations (such as volatility scaling or standardization) should never change after the research begins.
- Acknowledge that out-of-sample data is not really out of sample: simply because researchers have lived through the hold-out sample and thus understand the history, are knowledgeable about when markets rose and fell, and associate leading variables with past experience.

As such, no true out-of-sample data exists; the only true out of sample is the live trading experience.

- Understand iterated out of sample is not out of sample.
- Do not ignore trading costs and fees.
- Refrain from tweaking the model.
- Beware of model complexity: pursue simplicity and regularization.

8.3.8 Mathematical Tools to Combat Overfitting

A number of mathematical techniques have been proposed in the past decade to combat backtest overfitting; to name a few:

- A general framework to assess the probability of backtest overfitting was proposed in Bailey et al. (2017).
- For the single testing case, the minimum backtest length metric was proposed in Bailey et al. (2014) to avoid selecting a strategy with a high Sharpe ratio on in-sample data, but zero or less on out-of-sample data. A *probabilistic Sharpe ratio* was proposed in Bailey and López de Prado (2012) to calculate the probability of an estimated SR being greater than a benchmark Sharpe ratio.
- For the multiple testing case, the *deflated Sharpe ratio* was developed in Bailey and López de Prado (2014) to provide a more robust performance statistic, in particular, when the returns follow a nonnormal distribution.
- Online tools are presented in Bailey, Borwein, López de Prado, et al. (2016) to demonstrate how easy it is to overfit an investment strategy, and how this overfitting may affect the financial bottom-line performance.
- Section 8.4.4 describes a way to execute multiple randomized backtests that helps the prevention of overfitting.

8.4 Backtesting with Historical Market Data

As we have already discussed, a backtest evaluates the out-of-sample performance of an investment strategy using past observations. These observations can be used in a multitude of ways from the simplest method to more sophisticated versions.

One first classification is according to whether the past observations are used (i) directly to assess the historical performance as if the strategy had been run in the past (considered in detail in this section), or (ii) indirectly to simulate scenarios that did not happen in the past, such as stress tests (treated in the next section). Each approach has its pros and cons; in fact, both are useful and complement each other.

Assuming the historical data is used directly to assess the performance, we can further differentiate four types of backtest methods:

- vanilla (one-shot) backtest
- walk-forward backtest
- k-fold cross-validation backtest
- multiple randomized backtest.

The walk-forward backtest technique is so prevalent that, in fact, the term "backtest" has become a de facto synonym for walk-forward "historical simulation." We now elaborate on these different approaches.

8.4.1 Vanilla Backtest

The simplest possible backtest, which we call a *vanilla backtest*, involves dividing the available data into *in-sample data* and *out-of-sample data*. The in-sample data is used to design the strategy and the out-of-sample to evaluate it. The reason we need two sets of data should be clear by now: if the same data is used to design the strategy and to evaluate it, we would obtain spectacular but unrealistic performance results that would not be representative of the future performance with new data (this is because the strategy would overfit the data).

The in-sample data is typically further divided into *training data* and *cross-validation (CV) data*, whereas the out-of-sample data is also called *test data*. The training data is used to fit the model, that is, to choose the parameters of the model, whereas the CV data is employed to choose the so-called hyper-parameters. Figure 8.9 illustrates the data split for this type of vanilla backtest. Additionally, one may leave some small gap in between the in-sample and out-of-sample data (to model the fact that one may not be able to execute the designed portfolio immediately).

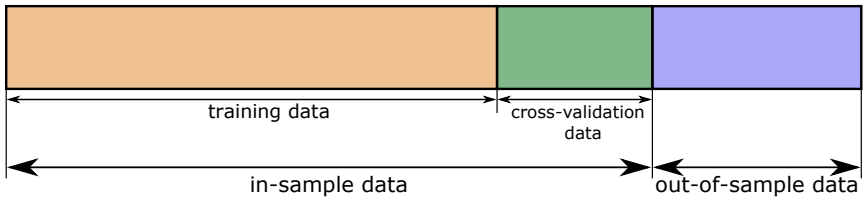


Figure 8.9 Data split in a vanilla backtest.

As an illustrative example, one may divide the available data into 70% in-sample and 30% out-of-sample for testing. The in-sample data may be further divided into 70% for training and 30% for cross validation. The training data may be used, for instance, to estimate the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, whereas the CV data may be used to choose the hyper-parameter λ in the mean–variance portfolio formulation (see Section 7.1). One can simply try different values of λ, say, 0.1, 0.5, 1.0, and 2.0, then evaluate the performance of each design with the CV data to choose the best-performing value of λ. At this point, one can fit the model again (i.e., estimate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$) using all the in-sample data (i.e., training plus cross-validation data). Finally, one can use the test data to assess the performance of the designed portfolio.

The result of a vanilla backtest can be seen in Figure 8.10 in the form of cumulative P&L and drawdown of the portfolios, as well as in Table 8.2 in the form of numerical values of different performance measures over the whole period.

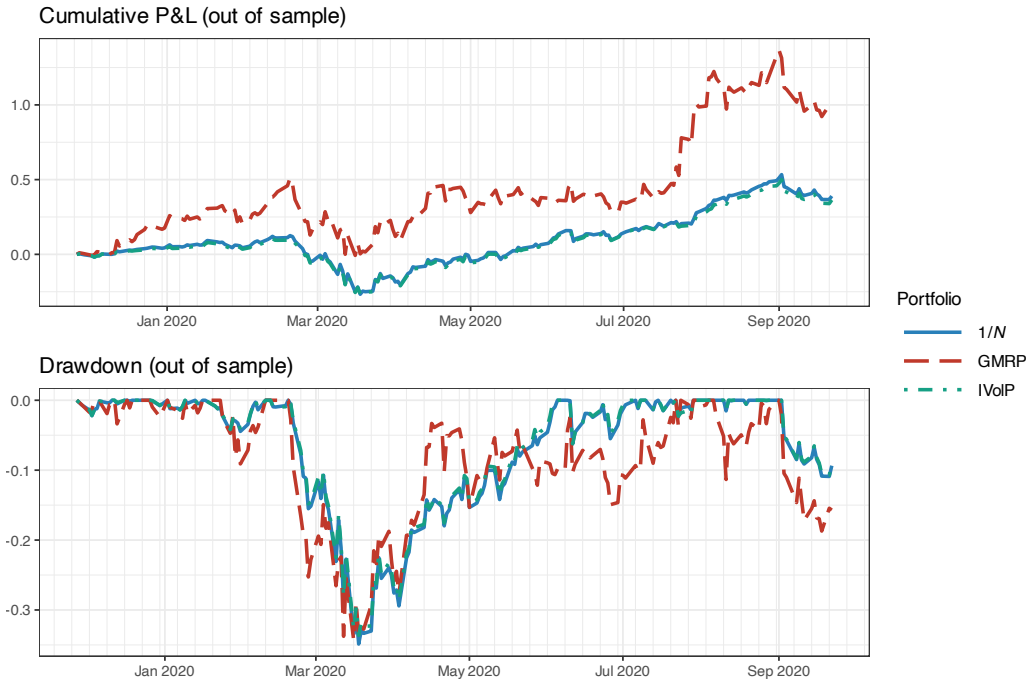


Figure 8.10 Vanilla backtest: cumulative P&L and drawdown.

Table 8.2 Vanilla backtest: performance measures.

Portfolio	Sharpe ratio	Annual return	Annual volatility	Sortino ratio	Max drawdown	CVaR (0.95)
1/*N*	1.18	49%	42%	1.63	35%	7%
GMRP	1.62	105%	65%	2.58	34%	9%
IVolP	1.14	46%	41%	1.58	34%	6%

The vanilla backtest is widely used in academic publications, blogs, fund brochures, and so on. However, it has two main problems:

1. A single backtest is performed, in the sense that a single historical path is evaluated.
2. The execution of the backtest is not representative of the way it would have been conducted in real life, that is, the portfolio is designed once and kept fixed for the whole test period, whereas in real life as new data becomes available the portfolio is updated (this is precisely addressed by the walk-forward backtest).

8.4.2 Walk-Forward Backtest

The *walk-forward backtest* improves the vanilla backtest by redesigning the portfolio as new data becomes available, effectively mimicking the way it would be done in live trading. It is therefore a historical simulation of how the strategy would have performed in the past (its performance can be reconciled with paper trading). This is the most common backtest method in the financial literature (Pardo, 2008).

In signal processing, this is commonly referred to as implementation on a *rolling-window* or *sliding-window* basis. This means that at time t one will use a window of the past k samples $t-k,\ldots,t-1$, called the *lookback window*, as training data or in-sample data. As with the vanilla backtest, one can leave a gap between the last-used observation and the time in which the portfolio is executed (this is to be on the safe side and avoid potential issues, such as look-ahead bias). A variation of the rolling window is the so-called *expanding window* (also called anchored walk-forward backtest), by which at time t all the previous data is used, that is, all the samples $1,2,\ldots,t-1$ (the window is expanding with t, hence the name).

Figure 8.11 illustrates the rolling-window split of data into in-sample and out-of-sample corresponding to a walk-forward backtest.

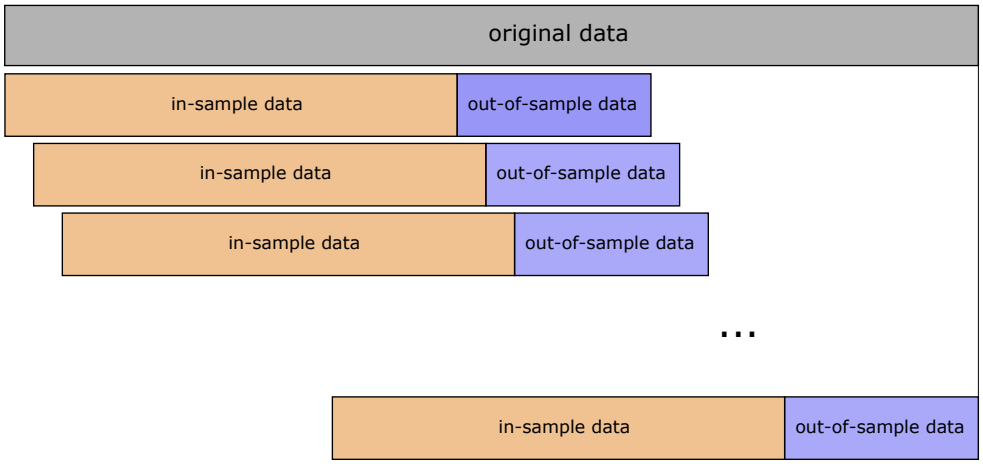


Figure 8.11 Data splitting in a rolling-window or walk-forward backtest.

The result of a walk-forward backtest with daily reoptimization can be seen in Figure 8.12 in the form of cumulative P&L and drawdown of the portfolios, as well as in Table 8.3 in the form of numerical values of different performance measures over the whole period.

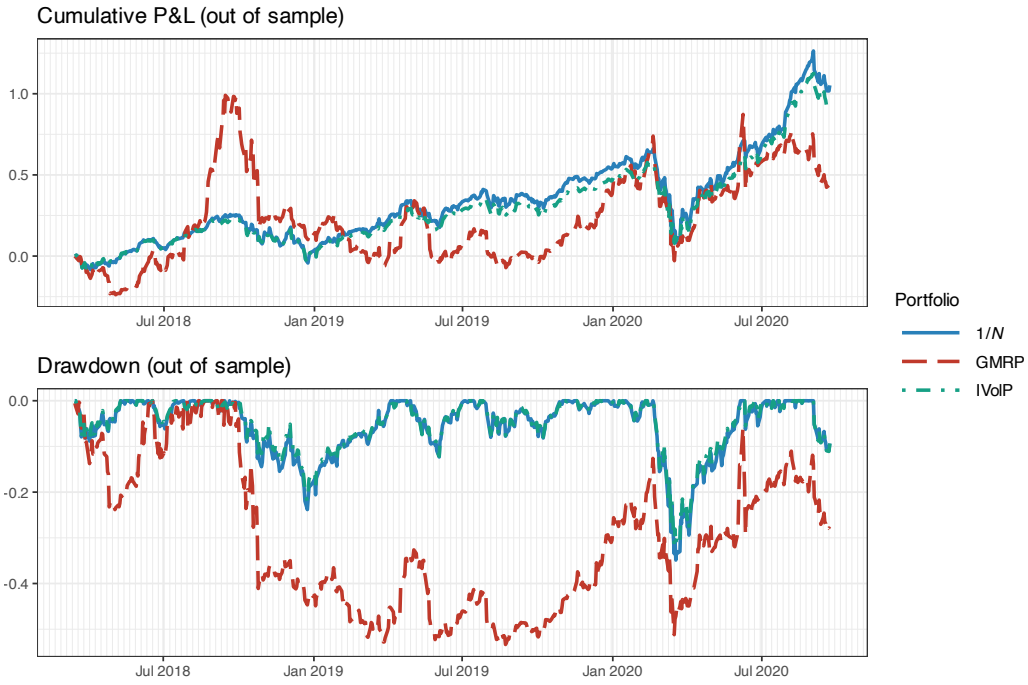


Figure 8.12 Walk-forward backtest: cumulative P&L and drawdown.

Table 8.3 Walk-forward backtest: performance measures.

Portfolio	Sharpe ratio	Annual return	Annual volatility	Sortino ratio	Max drawdown	CVaR (0.95)
$1/N$	1.10	33%	30%	1.54	35%	5%
GMRP	0.54	27%	50%	0.75	53%	8%
IVolP	1.09	31%	28%	1.52	32%	4%

Thus, the walk-forward backtest mimics the way live trading would be implemented and is the most common backtest method in finance. However, it still suffers from two critical issues:

1. It still represents a single backtest, in the sense that a single historical path is evaluated (hence, the danger of overfitting).
2. It is not uncommon to make some mistake with the time alignment and generate leakage (i.e., a form of look-ahead bias where future information leaks and is incorrectly used).

8.4.3 k-Fold Cross-Validation Backtest

The main drawback of the vanilla backtest and the walk-forward backtest is that a single historical path is evaluated. That is, in both cases a single backtest is performed. The idea

in the *k-fold cross-validation backtest* is to test k alternative scenarios (of which only one corresponds to the historical sequence).

In fact, k-fold cross-validation is a common approach in machine learning (ML) applications (James et al., 2013), composed of the following steps:

1. Partition the dataset into k subsets.
2. For each subset $i = 1, \ldots, k$:
   a. train the ML algorithm on all subsets excluding i; and
   b. test the fitted ML algorithm on the subset i.

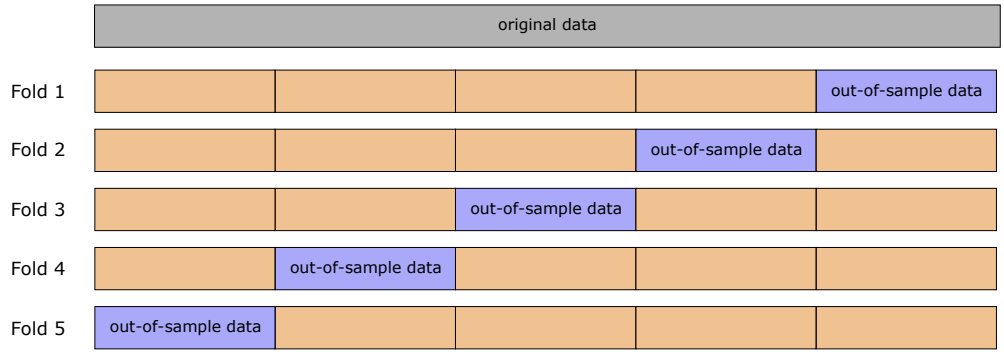


Figure 8.13 Data splitting in a k-fold cross-validation backtest (with $k = 5$).

Figure 8.13 illustrates the k-fold cross-validation split of data for $k = 5$. An implicit assumption in k-fold cross-validation is that the order of the blocks is irrelevant. This is true in many ML applications where the data is i.i.d. With financial data, however, this does not hold. While one may model the returns as uncorrelated, they are clearly not independent (e.g., the absolute values of the returns are highly correlated, as observed in Chapter 2, Figures 2.19–2.22).

In fact, leakage refers to situations when the training set contains information that also appears in the test set (it is a form of look-ahead bias). Some techniques can be employed to reduce the likelihood of leakage (López de Prado, 2018). Nevertheless, due to the temporal structure in financial data, it is the author's opinion that k-fold cross-validation backtests can be very dangerous in practice and are better avoided.

Summarizing, some of the issues with the k-fold cross-validation backtest include:

1. It is still using a single path of data.
2. It does not have a clear historical interpretation.
3. Most importantly, leakage is possible (and likely) because the training data does not trail the test data.

8.4.4 Multiple Randomized Backtests

The main drawback of the vanilla and walk-forward backtests is that a single historical path is evaluated. The k-fold cross-validation backtest attempts to address this issue by testing k

alternative scenarios; however, there is a significant problem with leakage and, in addition, only one of them corresponds to the historical sequence. With the *multiple randomized backtest* we can effectively deal with those issues in a satisfactory manner by generating a number of different backtests (each of them representing a different historical path) while respecting the order of training data followed by test data (to avoid leakage).

The basic idea of multiple randomized backtests is very simple:

1. Start with a large amount of historical data (preferably large in time and in assets).
2. Repeat k times:
   a. resample dataset: choose randomly a subset of the N available assets and a (contiguous) subset of the total period of time; and
   b. perform a walk-forward or rolling-window backtest of this resampled dataset.
3. Collect statistics on the results of the k backtests.

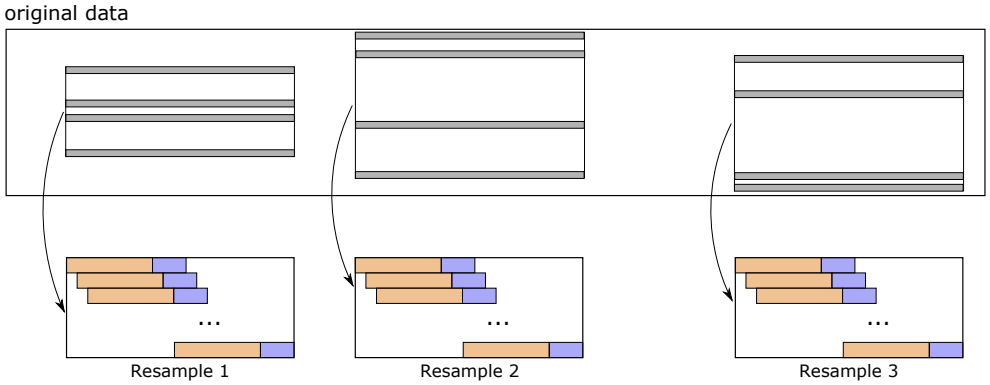


Figure 8.14 Data splitting in multiple randomized backtests.

Figure 8.14 illustrates the data split in multiple randomized backtests. For example, if the data contains 500 stocks over a period of 10 years, one can resample the data by choosing, say, 200 randomly chosen stocks over random periods of 2 contiguous years. This will introduce some randomness in each individual dataset and it will span different market regimes encountered over the 10 years.

To illustrate multiple randomized backtests, we take a dataset of $N = 10$ stocks over the period 2017–2020 and generate 200 resamples each with $N = 8$ randomly selected stocks and a random period of two years. Then we perform a walk-forward backtest with a lookback window of one year, reoptimizing the portfolio every month.

Table 8.4 shows the backtest results in table form with different performance measures over the whole period. Figure 8.15 plots the barplots of the median values of maximum drawdown and annualized volatility (over the 200 individual backtests), whereas Figure 8.16 shows the boxplots of the Sharpe ratio and maximum drawdown. Clearly, the statistics on the multiple individual backtests provide a more accurate representation of the true capabilities

of each portfolio. Nevertheless, it is still valid to say that this does not guarantee any future performance.

Table 8.4 Multiple randomized backtest: performance measures.

Portfolio	Sharpe ratio	Annual return	Annual volatility	Sortino ratio	Max drawdown	CVaR (0.95)
1/*N*	1.01	28%	25%	1.42	25%	4%
GMVP	0.72	16%	22%	1.01	24%	3%
IVolP	0.94	24%	23%	1.33	23%	3%

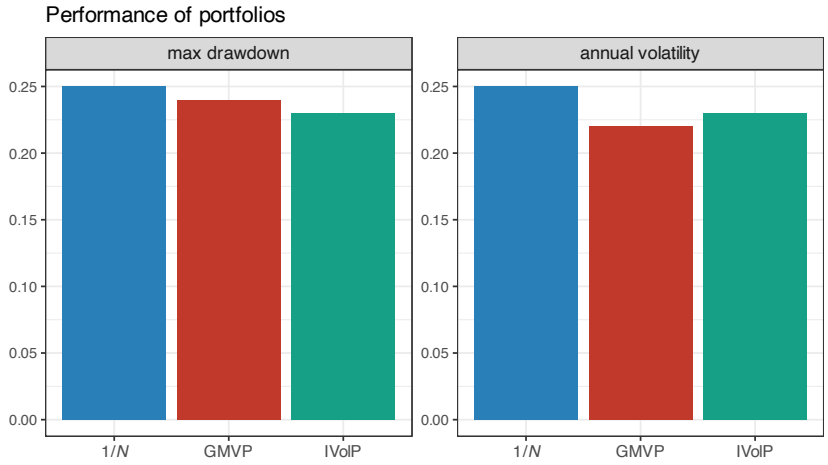


Figure 8.15 Multiple randomized backtests: barplots of maximum drawdown and annualized volatility.

Multiple randomized backtests, while not perfect, seem to address the main drawbacks of the other types of backtests covered here. Therefore, in the author's opinion, it is the preferred method for backtesting.

8.5 Backtesting with Synthetic Data

As we have already discussed, a backtest evaluates the out-of-sample performance of an investment strategy using past observations. Section 8.4 has explored different ways to directly use the historical data to assess the performance as if the strategy had been run in the past. This section considers a more indirect way to employ historical data by generating synthetic – yet realistic – data to simulate scenarios that did not happen in the past, such as *stress tests* where different market scenarios are recreated to test the strategy.

Monte Carlo simulations allow us to create synthetic data that resembles a given set of historical data. They can be divided into three categories:

- *Parametric methods*: These postulate and fit a model to the data. Then, they generate as much synthetic data as necessary from the model.

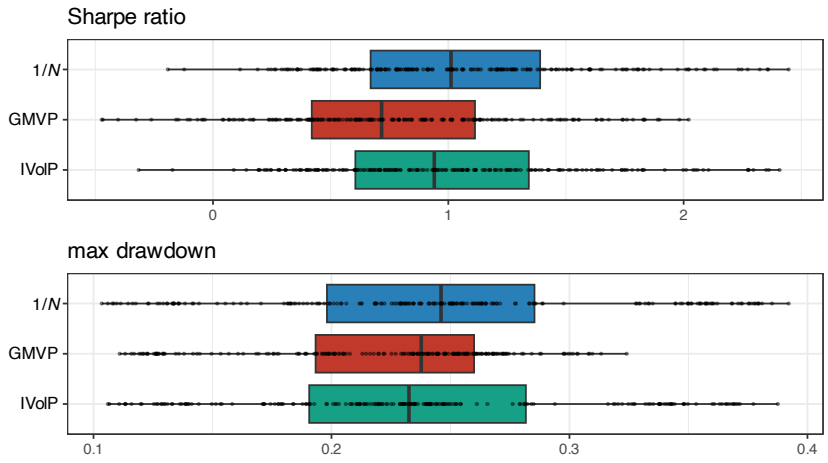


Figure 8.16 Multiple randomized backtests: boxplots of Sharpe ratio and maximum drawdown.

- *Nonparametric methods*: These directly resample the historical data without any modeling.
- *Hybrid methods*: These combine the modeling approach with resampling for the model residual (which can follow a parametric or nonparametric method).

Generating such realistic synthetic data will allow us to backtest a strategy on a large number of unseen, synthetic testing sets, hence reducing the likelihood that the strategy has been fitted to a particular dataset. The quality of the data generated under parametric methods depends on the model assumption: if the model is wrong, the data generated will not be realistic. Nonparametric methods are more robust, but they can also potentially destroy some temporal structure in the data. Hybrid methods are more appealing as they can model as much structure as possible and then the residual data is that generated following either the parametric or nonparametric approach. These methods are compared and illustrated next.

8.5.1 I.I.D. Assumption

The simplest possible example of a parametric method is based on modeling the returns as i.i.d. and fitting some distribution function, such as Gaussian or preferably a heavy-tailed distribution. Once the parameters of the distribution have been estimated (i.e., the model has been fitted to the data), then synthetic data can be generated, which will resemble (statistically) the original data.

The simplest example of a nonparametric method, also based on assuming the returns are i.i.d., is even simpler to implement: just resample the original returns with replacement.

It is important to emphasize that these two examples of parametric and nonparametric methods are based on the i.i.d. assumption, which obviously does not hold for financial data (e.g., the absolute values of the returns are highly correlated as observed in Chapter 2, Figures 2.19–2.22).

Figure 8.17 illustrates the synthetic generation of return data under the i.i.d. parametric

(assuming a Gaussian distribution) and nonparametric methods. Since both methods assume i.i.d. data, they totally destroy the volatility clustering structure present in the original data. In addition, the parametric method is assuming (wrongly) a Gaussian distribution and therefore the generated data does not have the same deep spikes typical of a heavy-tailed distribution. The nonparametric method, on the other hand, includes deep spikes but dispersed over time rather than clustered. We next consider a more realistic case by incorporating the temporal structure.

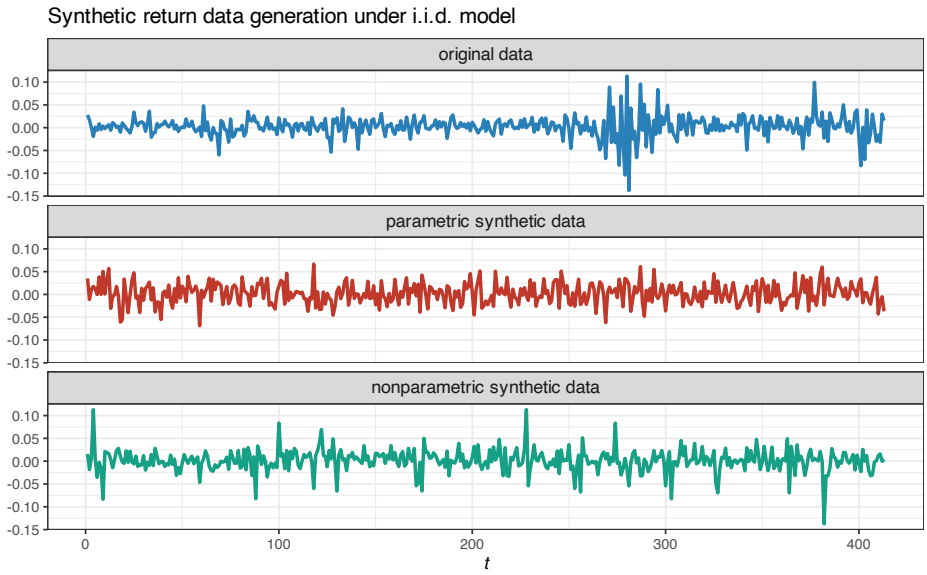


Figure 8.17 Example of an original sequence and two synthetic sequences generated with i.i.d. parametric and nonparametric methods.

8.5.2 Temporal Structure

To illustrate the hybrid method, suppose now that a more sophisticated model is used to model the expected returns based on the past values of the returns, denoted by $\boldsymbol{\mu}_t = \boldsymbol{f}(\boldsymbol{r}_{t-k}, \ldots, \boldsymbol{r}_{t-1})$ (see Chapter 4 for time-series mean models). This means that the returns can be written as the forecast plus some residual error,

$$\boldsymbol{r}_t = \boldsymbol{\mu}_t + \boldsymbol{u}_t,$$

where $\boldsymbol{u}_t$ is the zero-mean residual error at time t with covariance matrix $\boldsymbol{\Sigma}$.

An even more sophisticated model can combine the mean model for $\boldsymbol{\mu}_t$ with a covariance model for $\boldsymbol{\Sigma}_t$:

$$\boldsymbol{r}_t = \boldsymbol{\mu}_t + \boldsymbol{\Sigma}_t^{1/2}\boldsymbol{\epsilon}_t,$$

where $\boldsymbol{\epsilon}_t$ is a standardized zero-mean identity-covariance residual error at time t. The reader is referred to Chapter 4 for time-series mean and covariance models.

In these more sophisticated modeling cases, we can now take two different approaches to generate the sequence of residuals and, implicitly, the sequence of synthetic returns:

- following the parametric paradigm: we can model the residuals with an i.i.d. model and generate new synthetic residuals; and
- following the nonparametric paradigm: we can just resample the residuals from the historical data.

Figure 8.18 illustrates the synthetic generation of return data properly modeling the volatility clustering and then using both the parametric (assuming a Gaussian distribution) and nonparametric methods. As expected, the volatility clustering is preserved as it appears in the original data. The parametric method is assuming (wrongly) a Gaussian distribution for the residuals and it could be further improved by employing a heavy-tailed distribution. The nonparametric method, on the other hand, is more robust to modeling errors since it is directly resampling the original residuals.

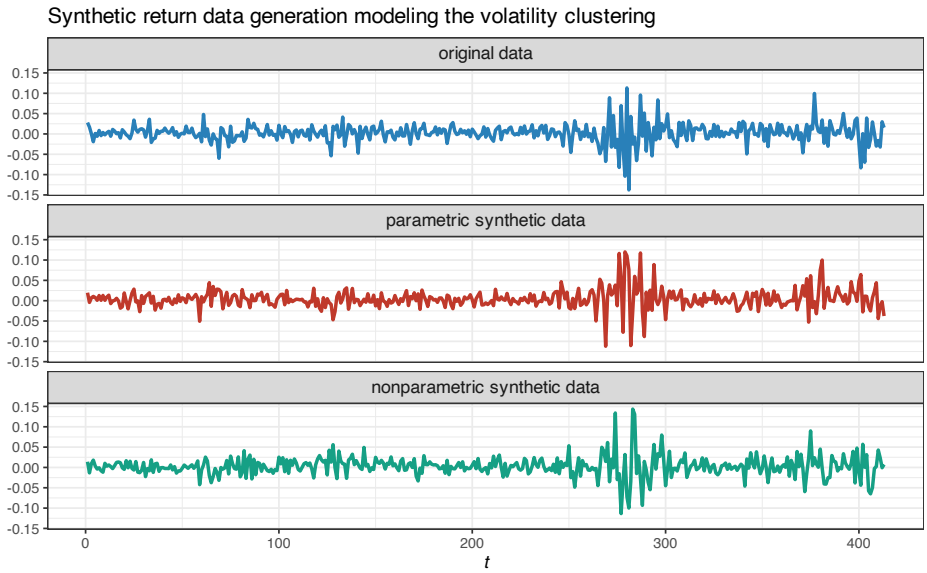


Figure 8.18 Example of an original sequence and two synthetic sequences generated by modeling the volatility clustering and the residuals with parametric and nonparametric methods.

8.5.3 Stress Tests

Stress tests are yet another set of tools in the backtest toolkit. They also fall into the category of synthetic generated data but they are more like an "à la carte menu." The idea is to be able to generate realistic synthetic data recreating different market scenarios, such as the choice of a strong bull market, a weak bull market, a side market, a weak bear market, and a strong bear market.

Other examples of stress tests could consider specific periods of crises such as the stock

market crash of October 1987, the Asian crisis of 1997, and the tech bubble that burst in 1999–2000.

In other words, stress testing tests the resilience of investment portfolios against possible future financial situations.

Figure 8.19 illustrates the generation of synthetic data for stress testing corresponding to a bull market (the reference bull market period is April–August, 2020). On the other hand, Figure 8.20 illustrates the generation of synthetic data for stress testing corresponding to a bear market (the reference bear market period is September–December, 2018).

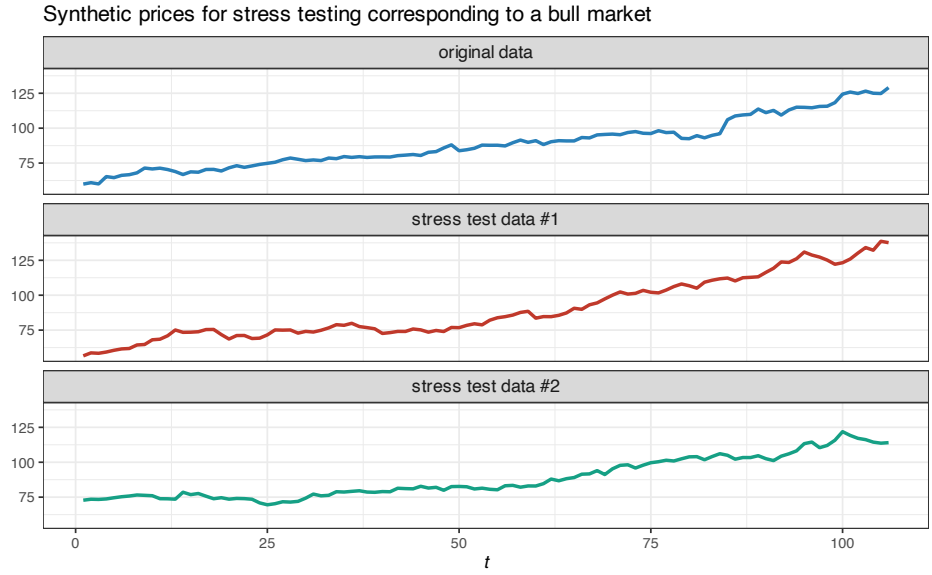


Figure 8.19 Example of original data corresponding to a bull market and two synthetic generations of bull markets for stress testing.

8.6 Summary

Backtesting of portfolios is an essential part in the process of strategy development and evaluation. Nevertheless, it remains widely misunderstood and the dangers are routinely underestimated. Some of the key points to keep in mind include the following:

- There are multiple reasons why backtest results have to be taken with "a grain of salt," namely, survivorship bias, look-ahead bias, storytelling bias, overfitting or data-snooping bias, turnover and transaction cost, outliers, and asymmetric pattern and shorting cost, among others.
- Arguably, the single main reason why any backtest may be faulty and misleading is overfitting.
- Due to these potential pitfalls, one cannot trust any backtest results provided in academic publications, blogs, investment fund brochures, and so on.

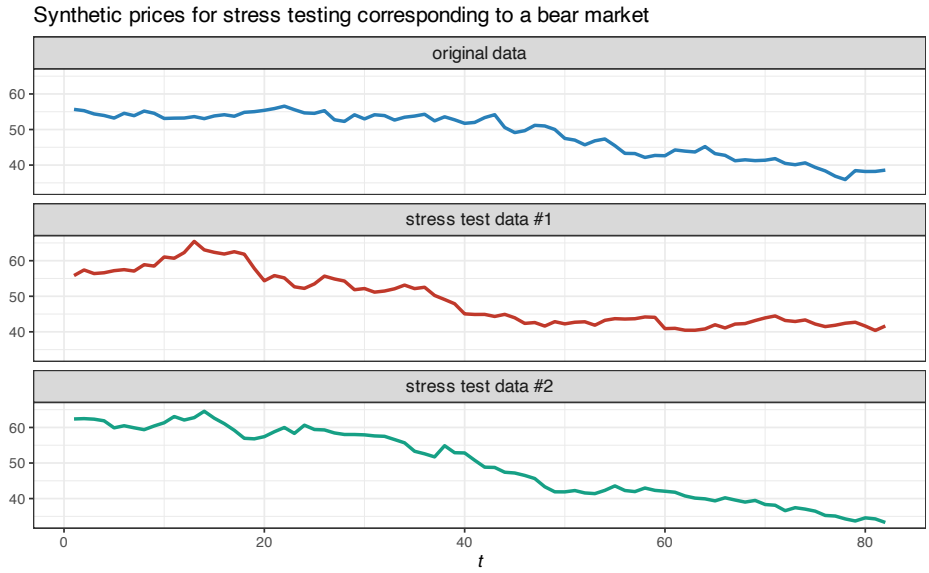


Figure 8.20 Example of original data corresponding to a bear market and two synthetic generations of bear markets for stress testing.

- Armed with this knowledge, it is recommended to conduct multiple randomized backtests, perhaps combined with stress tests under different scenarios.

The reader has been warned and provided with tools, and hopefully this will serve as a guide for the future.

Exercises

8.1 (Survivorship bias) Download stock price data corresponding to some index over several years and find an example where survivorship bias makes a big difference to the portfolio performance.

8.2 (Look-ahead bias) Download stock price data and find an example where look-ahead bias makes a big difference to the portfolio performance.

8.3 (Storytelling or confirmation bias) Download stock price data and find an example where storytelling or confirmation bias makes a big difference to the portfolio performance.

8.4 (Overfitting) Download stock price data and design a portfolio with overfitting in a way that makes a big difference to the portfolio performance.

8.5 (Turnover and transaction cost) Download stock price data and design a portfolio with high turnover so that the performance ignoring and including the transaction cost makes a big difference.

8.6 (Outliers) Download stock price data and find a portfolio example that accidentally benefits from some random outlier during the training phase, but with bad performance during the test phase.

8.7 (Single vs. multiple backtests) Download stock price data and choose several portfolio designs. Then evaluate them first with a single backtest and then with multiple randomized backtests. Find some example where the single backtest is totally misleading compared to the performance statistics obtained from the multiple backtests.

8.8 (Stress tests) Download stock price data and experiment with the generation of synthetic data corresponding to different market regimes of the market data. Then, repeat the experiment with multiple stocks, including the correlation among the stocks.

References

Arnott, R., Harvey, C. R., & Markowitz, H. M. (2019). A backtesting protocol in the era of machine learning. *The Journal of Financial Data Science*, *1*(1), 64–74.

Bailey, D. H., Borwein, J., & López de Prado, M. (2016). Stock portfolio design and backtest overfitting. *Journal of Investment Management*, *15*(1), 1–13.

Bailey, D. H., Borwein, J. M., López de Prado, M., Salehipour, A., & Zhu, Q. J. (2016). Backtest overfitting in financial markets. *Automated Trader*, *39*(2), 52–57.

Bailey, D. H., Borwein, J. M., López de Prado, M., & Zhu, Q. J. (2014). Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance. *Notices of the American Mathematical Society*, *61*(5), 458–471.

Bailey, D. H., Borwein, J. M., López de Prado, M., & Zhu, Q. J. (2017). The probability of backtest overfitting. *Journal of Computational Finance (Risk Journals)*, *20*(4), 458–471.

Bailey, D. H., & López de Prado, M. (2012). The Sharpe ratio efficient frontier. *The Journal of Risk*, *15*(2), 3–44.

Bailey, D. H., & López de Prado, M. (2014). The deflated Sharpe ratio: Correcting for selection bias, backtest overfitting and non-normality. *Journal of Portfolio Management*, *40*(5), 94–107.

Chan, E. P. (2008). *Quantitative Trading: How to Build Your Own Algorithmic Trading Business*. John Wiley & Sons.

Glabadanidis, P. (2015). Market timing with moving averages. *International Review of Finance*, *15*(3), 387–425.

Harvey, C. R. (2017). Presidential address: The scientific outlook in financial economics. *The Journal of Finance*, *72*(4), 1399–1440.

Harvey, C. R., Liu, Y., & Zhu, H. (2016). ... and the cross-section of expected returns. *Review of Financial Studies*, *29*(1), 5–68.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: With Applications in R*. Springer.

López de Prado, M. (2018). *Advances in Financial Machine Learning*. John Wiley & Sons.

Luo, Y., Alvarez, M., Wang, S., Jussa, J., Wang, A., & Rohal, G. (2014). Seven sins of quantitative investing. *White paper, Deutsche Bank Markets Research*.

Mayer, J., Khairy, K., & Howard, J. (2010). Drawing an elephant with four complex parameters. *American Journal of Physics*, *78*(6), 648–649.

Montier, J. (2005). Seven sins of fund management. *Dresdner Kleinwort Wasserstein – Global Equity Strategy*.

Pardo, R. (2008). *The Evaluation and Optimization of Trading Strategies* (2nd ed.). John Wiley & Sons.

White, H. (2000). A reality check for data snooping. *Econometrica*, *68*(5), 1097–1126.

Zakamulin, V. (2018). Revisiting the profitability of market timing with moving averages. *International Review of Finance*, *18*(2), 317–327.

9

High-Order Portfolios

The study of economics does not seem to require any specialised gifts of an unusually high order.

— John Maynard Keynes, *Alfred Marshall*

Markowitz's mean–variance portfolio optimizes a trade-off between expected return and risk measured by the variance. However, since financial data is not Gaussian distributed, due to asymmetry and heavy tails in the distribution, it would be reasonable to also incorporate higher-order moments.

Unfortunately, designing a portfolio based on the first four moments (i.e., mean, variance, skewness, and kurtosis) brings at least two critical difficulties:

- The dimensionality of the higher-order moments grows as N^4, where N is the number of assets, with implications in the complexity of the moment computation, memory storage, and algorithmic manipulation.
- The portfolio formulations are nonconvex, further complicating the design and optimization.

High-order portfolios were formulated over half a century ago, but have only recently become a practical reality for large numbers of assets in the order of hundreds or even thousands.

9.1 Introduction

Markowitz's mean–variance portfolio (Markowitz, 1952) formulates the portfolio design as a trade-off between the expected return $\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}$ and the risk measured by the variance $\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}$ (see Chapter 7 for details):

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \frac{\lambda}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w} \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array}$$

where λ is a hyper-parameter that controls the investor's risk aversion and $\mathcal{W}$ denotes an arbitrary constraint set, such as $\mathcal{W} = \{\boldsymbol{w} \mid \mathbf{1}^\mathsf{T}\boldsymbol{w} = 1, \boldsymbol{w} \geq \mathbf{0}\}$.

Nevertheless, decades of empirical studies have clearly demonstrated that financial data do

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

not follow a Gaussian distribution (see Chapter 2 for stylized facts of financial data). This implies that it would be reasonable to also incorporate *higher-order moments* (or *higher-order statistics*), in addition to the first two moments (i.e., mean and variance), in the portfolio formulation. We will consider here the first four moments as an improvement over the first two moments. The third and fourth moments are called *skewness* and *kurtosis*, respectively. In terms of portfolio optimization, in the same way that a higher expected return and lower variance are desired, we should seek a higher skewness and a lower kurtosis (i.e., higher odd moments and lower even moments) (Scott & Horvath, 1980).

Figure 9.1 shows the skewed t probability distribution function with different degrees of skewness (controlled by the parameter γ, with $\gamma = 0$ for the symmetric case) and kurtosis (controlled by the parameter ν, with $\nu \longrightarrow \infty$ for the non-heavy-tailed case).

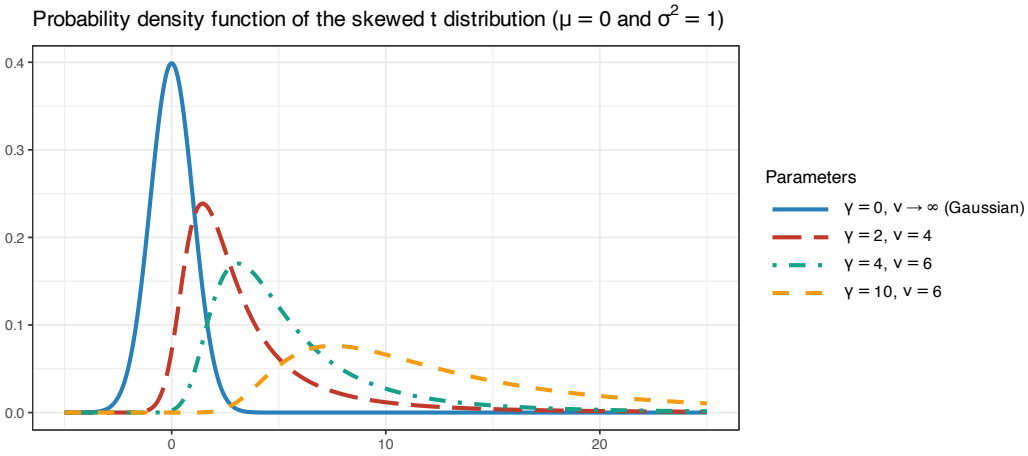


Figure 9.1 Illustration of skewness and kurtosis with the skewed t distribution.

The higher moments are definitely important and, for example, a skew-adjusted Sharpe ratio was proposed in Zakamouline and Koekebakker (2009):

$$\text{skew-adjusted-SR} = \text{SR} \times \sqrt{1 + \frac{\text{skewness}}{3}\text{SR}}.$$

Thus, the idea is to have a portfolio formulation where investors may accept lower expected return and/or higher volatility, compared to the mean–variance benchmark, in exchange for higher skewness and lower kurtosis.

9.1.1 High-Order Portfolios

As will be elaborated in the next section, the third and fourth moments of the portfolio are given by the expressions $\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Phi}(\boldsymbol{w} \otimes \boldsymbol{w})$ and $\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Psi}(\boldsymbol{w} \otimes \boldsymbol{w} \otimes \boldsymbol{w})$, respectively, where $\boldsymbol{\Phi} \in \mathbb{R}^{N \times N^2}$ is the co-skewness matrix, and $\boldsymbol{\Psi} =\in \mathbb{R}^{N \times N^3}$ is the co-kurtosis matrix.

Unfortunately, the incorporation of the third and fourth moments into the portfolio formulation brings two critical issues:

- Due to the dimensions of the co-skewness and co-kurtosis matrices, their computation, storage in memory, and algorithmic manipulation will be extremely challenging and severely limiting.
- The third moment is a nonconvex function of the portfolio $\boldsymbol{w}$. This will make the high-order portfolio formulations complicated and the design of numerical algorithms more involved.

9.1.2 Historical Perspective

Portfolio formulations incorporating high-order moments go back to the 1960s (Jean, 1971; Young & Trent, 1969). However, the estimation of such high-order moments was an absolute impasse in those early days, the main problem being the dimensionality issue (recall that the co-skewness matrix is $N \times N^2$ and the co-kurtosis matrix is $N \times N^3$), which means that the number of parameters quickly grows with the number of assets, with detrimental consequences in terms of the amount of data required for the estimation, the computational cost, and the storage needs. As a consequence, some authors have shown skepticism regarding the portfolio formulation based on high-order moments (Brandt et al., 2009):

> extending the traditional approach beyond first and second moments, when the investor's utility function is not quadratic, is practically impossible because it requires modeling [. . .] the numerous higher-order cross-moments.

We had to wait for half a century to start finding publications proposing improved estimation methods, such as introducing structure and shrinkage in the high-order parameters (Boudt et al., 2014; Martellini & Ziemann, 2010) or assuming parametric multivariate distributions, to drastically reduce the number of parameters to estimate (Birgeand & Chavez-Bedoya, 2016; X. Wang et al., 2023). As empirically shown in Martellini and Ziemann (2010), unless improved estimators are employed, incorporating third and fourth moments in the portfolio design is detrimental in terms of out-of-sample performance.

In addition, apart from the estimation difficulties, the algorithmic part of the portfolio design has been extremely challenging. Since the high-order problem formulations are nonconvex, one approach is to use meta-heuristic optimization tools, such as genetic algorithms or differential evolution (Boudt et al., 2014), which attempt to find the global solution at the expense of a very high computational cost, but this becomes prohibitive when the problem dimension grows. Local optimization methods are a reasonable practical alternative by finding a locally optimal solution with acceptable computational cost.

A method based on difference-of-convex (DC) programming was proposed to solve high-order portfolio problems to a stationary point (Dinh & Niu, 2011), but convergence is slow and it is only applicable to low-dimensional problems. An improved method was proposed based on the difference of convex sums of squares (DC-SOS) decomposition techniques (Niu & Wang, 2019), but the complexity of computing the gradients of high-order moments grows rapidly with the problem dimension. The classical gradient descent method and backtracking line search also become inapplicable when the problem dimension grows large.

Faster methods were developed in Zhou and Palomar (2021) based on the successive convex approximation (SCA) framework (Scutari et al., 2014), which solves the original difficult problem by constructing and solving a sequence of convex approximated problems whose

solutions converge to a stationary point of the original problem (see Appendix B for details on the SCA framework). The obtained algorithms converge much faster, although each iteration still requires a significant computational cost due to the computation of gradients and Hessians of high-order moments, allowing high-dimensional settings on the order of hundreds of assets, but not higher.

In Birgeand and Chavez-Bedoya (2016) and X. Wang et al. (2023), the computational cost of high-order moments was significantly reduced via a multivariate parametric model of the data, and in X. Wang et al. (2023) even faster numerical methods were developed that can be employed in very large dimensions with hundreds or thousands of assets.

Thus, practical high-order portfolios are now a reality after more than half a century of research by the scientific community.

9.2 High-Order Moments

The first two moments of a random variable are enough to characterize its distribution only if it follows a Gaussian or normal distribution; otherwise, higher-order moments, also termed higher-order statistics, are necessary.

The first four moments of a random variable X are

- the mean or first moment (measure of location): $\bar{X} \triangleq \mathbb{E}[X]$;
- the variance or second central moment (measure of dispersion): $\mathbb{E}[(X - \bar{X})^2]$;
- the *skewness*[1] or third central moment (measure of asymmetry): $\mathbb{E}[(X - \bar{X})^3]$; and
- the *kurtosis*[2] or fourth central moment (measure of thickness of tails): $\mathbb{E}[(X - \bar{X})^4]$.

Next, we explore the expressions of the high-order statistics for the portfolio return.

9.2.1 Nonparametric Case

In the portfolio context with a universe of N assets, we denote by $\boldsymbol{r} \in \mathbb{R}^N$ the returns of the N assets and by $\boldsymbol{w} \in \mathbb{R}^N$ the portfolio weights. Then, the return of this portfolio is $\boldsymbol{w}^\mathsf{T}\boldsymbol{r}$ and the first four moments are given by

$$\begin{aligned}
\phi_1(\boldsymbol{w}) &\triangleq \mathbb{E}\left[\boldsymbol{w}^\mathsf{T}\boldsymbol{r}\right] = \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}, \\
\phi_2(\boldsymbol{w}) &\triangleq \mathbb{E}\left[\left(\boldsymbol{w}^\mathsf{T}\bar{\boldsymbol{r}}\right)^2\right] = \boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}, \\
\phi_3(\boldsymbol{w}) &\triangleq \mathbb{E}\left[\left(\boldsymbol{w}^\mathsf{T}\bar{\boldsymbol{r}}\right)^3\right] = \boldsymbol{w}^\mathsf{T}\boldsymbol{\Phi}(\boldsymbol{w} \otimes \boldsymbol{w}), \\
\phi_4(\boldsymbol{w}) &\triangleq \mathbb{E}\left[\left(\boldsymbol{w}^\mathsf{T}\bar{\boldsymbol{r}}\right)^4\right] = \boldsymbol{w}^\mathsf{T}\boldsymbol{\Psi}(\boldsymbol{w} \otimes \boldsymbol{w} \otimes \boldsymbol{w}),
\end{aligned} \tag{9.1}$$

[1] The skewness is actually defined as the standardized or normalized third moment $\mathbb{E}\left[\left((X - \bar{X})/\mathrm{Std}[X]\right)^3\right]$, where $\mathrm{Std}[X]$ is the standard deviation of X.

[2] The kurtosis is similarly defined as the standardized or normalized fourth moment $\mathbb{E}\left[\left((X - \bar{X})/\mathrm{Std}[X]\right)^4\right]$.

where $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{r}] \in \mathbb{R}^N$ is the mean vector, $\bar{\boldsymbol{r}} = \boldsymbol{r} - \boldsymbol{\mu}$ are the centered returns, $\boldsymbol{\Sigma} = \mathbb{E}\left[\bar{\boldsymbol{r}}\bar{\boldsymbol{r}}^\top\right] \in \mathbb{R}^{N \times N}$ is the covariance matrix, $\boldsymbol{\Phi} = \mathbb{E}\left[\bar{\boldsymbol{r}}(\bar{\boldsymbol{r}} \otimes \bar{\boldsymbol{r}})^\top\right] \in \mathbb{R}^{N \times N^2}$ is the co-skewness matrix, and $\boldsymbol{\Psi} = \mathbb{E}\left[\bar{\boldsymbol{r}}(\bar{\boldsymbol{r}} \otimes \bar{\boldsymbol{r}} \otimes \bar{\boldsymbol{r}})^\top\right] \in \mathbb{R}^{N \times N^3}$ is the co-kurtosis matrix.

The gradients and Hessians of the moments, required by numerical algorithms, are given by

$$\begin{aligned} \nabla\phi_1(\boldsymbol{w}) &= \boldsymbol{\mu}, \\ \nabla\phi_2(\boldsymbol{w}) &= 2\,\boldsymbol{\Sigma}\boldsymbol{w}, \\ \nabla\phi_3(\boldsymbol{w}) &= 3\,\boldsymbol{\Phi}(\boldsymbol{w} \otimes \boldsymbol{w}), \\ \nabla\phi_4(\boldsymbol{w}) &= 4\,\boldsymbol{\Psi}(\boldsymbol{w} \otimes \boldsymbol{w} \otimes \boldsymbol{w}) \end{aligned} \tag{9.2}$$

and

$$\begin{aligned} \nabla^2\phi_1(\boldsymbol{w}) &= \boldsymbol{0}, \\ \nabla^2\phi_2(\boldsymbol{w}) &= 2\,\boldsymbol{\Sigma}, \\ \nabla^2\phi_3(\boldsymbol{w}) &= 6\,\boldsymbol{\Phi}(\boldsymbol{I} \otimes \boldsymbol{w}), \\ \nabla^2\phi_4(\boldsymbol{w}) &= 12\,\boldsymbol{\Psi}(\boldsymbol{I} \otimes \boldsymbol{w} \otimes \boldsymbol{w}), \end{aligned} \tag{9.3}$$

respectively (Zhou & Palomar, 2021).

Complexity Analysis

The main problem with high-order moments is the sheer number of elements required to characterize them. This has direct implications on the computational cost, as well as the memory cost.

To simplify the analysis, rather than characterizing the exact cost or complexity, we are just interested in how the complexity grows with the dimensionality N. This can be done with the "big O" notation, which measures the order of complexity. To be specific, we say that the complexity is $f(N) = O\left(g(N)\right)$, as $N \to \infty$, if there exists a positive real number M and N_0 such that $|f(N)| \leq Mg(N)$ for all $N \geq N_0$.

Let us start by looking at the four parameters $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Phi}$, and $\boldsymbol{\Psi}$. From their dimensions, we can infer that their complexity is $O(N)$, $O(N^2)$, $O(N^3)$, and $O(N^4)$, respectively. The computation of the portfolio moments in (9.1) has the same complexity order. Regarding the gradients in (9.2), their computation has complexity $O(1)$, $O(N^2)$, $O(N^3)$, and $O(N^4)$, respectively. Finally, the complexity of the computation of the Hessians in (9.3) is $O(1)$, $O(1)$, $O(N^3)$, and $O(N^4)$, respectively.

For example, when $N = 200$, storing the co-kurtosis matrix $\boldsymbol{\Psi}$ requires almost 12 GB of memory (assuming a floating-point number is represented with 64 bits or 8 bytes).

Summarizing, while the complexity order of the first and second moments in Markowitz's portfolio is $O(N^2)$, further incorporating the third and fourth moments increases the complexity order to $O(N^4)$, which severely limits the practical application to scenarios with small number of assets.

9.2.2 Structured Moments

One way to reduce the number of parameters to be estimated in the high-order moments is by introducing some structure in the high-order moment matrices via factor modeling (see Chapter 3). This will, however, make the estimation process much more complicated due to the intricate structure in the matrices.

Consider a single market-factor model of the returns,

$$\boldsymbol{r}_t = \boldsymbol{\alpha} + \boldsymbol{\beta} r_t^{\text{mkt}} + \boldsymbol{\epsilon}_t,$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the so-called "alpha" and "beta," respectively, r_t^{mkt} is the market index, and $\boldsymbol{\epsilon}_t$ the residual. Then, the moments can be written (Martellini & Ziemann, 2010) as

$$\begin{aligned}
\boldsymbol{\mu} &= \boldsymbol{\alpha} + \boldsymbol{\beta}\phi_1^{\text{mkt}}, \\
\boldsymbol{\Sigma} &= \boldsymbol{\beta}\boldsymbol{\beta}^{\mathsf{T}}\phi_2^{\text{mkt}} + \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}, \\
\boldsymbol{\Phi} &= \boldsymbol{\beta}\left(\boldsymbol{\beta}^{\mathsf{T}} \otimes \boldsymbol{\beta}^{\mathsf{T}}\right)\phi_3^{\text{mkt}} + \boldsymbol{\Phi}_{\boldsymbol{\epsilon}}, \\
\boldsymbol{\Psi} &= \boldsymbol{\beta}\left(\boldsymbol{\beta}^{\mathsf{T}} \otimes \boldsymbol{\beta}^{\mathsf{T}} \otimes \boldsymbol{\beta}^{\mathsf{T}}\right)\phi_4^{\text{mkt}} + \boldsymbol{\Psi}_{\boldsymbol{\epsilon}},
\end{aligned} \tag{9.4}$$

where ϕ_i^{mkt} denotes the ith moment of the market factor, and $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}$, $\boldsymbol{\Phi}_{\boldsymbol{\epsilon}}$, and $\boldsymbol{\Psi}_{\boldsymbol{\epsilon}}$ are the covariance, co-skewness, and co-kurtosis matrices of the residuals $\boldsymbol{\epsilon}_t$, respectively.

Alternatively, we can assume a multi-factor model of the returns,

$$\boldsymbol{r}_t = \boldsymbol{\alpha} + \boldsymbol{B}\boldsymbol{f}_t + \boldsymbol{\epsilon}_t,$$

where $\boldsymbol{f}_t \in \mathbb{R}^K$ contains the K factors (typically with $K \ll N$). Then, the moments can be written (Boudt et al., 2014) as

$$\begin{aligned}
\boldsymbol{\mu} &= \boldsymbol{\alpha} + \boldsymbol{B}\boldsymbol{\phi}_1^{\text{factors}}, \\
\boldsymbol{\Sigma} &= \boldsymbol{\beta}\boldsymbol{\Phi}_2^{\text{factors}}\boldsymbol{\beta}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}, \\
\boldsymbol{\Phi} &= \boldsymbol{\beta}\boldsymbol{\Phi}_3^{\text{factors}}\left(\boldsymbol{\beta}^{\mathsf{T}} \otimes \boldsymbol{\beta}^{\mathsf{T}}\right) + \boldsymbol{\Phi}_{\boldsymbol{\epsilon}}, \\
\boldsymbol{\Psi} &= \boldsymbol{\beta}\boldsymbol{\Phi}_4^{\text{factors}}\left(\boldsymbol{\beta}^{\mathsf{T}} \otimes \boldsymbol{\beta}^{\mathsf{T}} \otimes \boldsymbol{\beta}^{\mathsf{T}}\right) + \boldsymbol{\Psi}_{\boldsymbol{\epsilon}},
\end{aligned} \tag{9.5}$$

where $\boldsymbol{\phi}_1^{\text{factors}}$ denotes the mean, and $\boldsymbol{\Phi}_2^{\text{factors}}$, $\boldsymbol{\Phi}_3^{\text{factors}}$, and $\boldsymbol{\Phi}_4^{\text{factors}}$, the covariance matrix, co-skewness matrix, and co-kurtosis matrix of the multiple factors in $\boldsymbol{f}_t$, respectively.

In addition to the structure provided by factor modeling, another technique is via shrinkage, see Martellini and Ziemann (2010) and Boudt, Cornilly, and Verdonck (2020).

9.2.3 Parametric Case

Multivariate Normal Distribution

A multivariate normal (or Gaussian) distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is characterized by the probability density function

$$f_{\text{mvn}}(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right).$$

A random vector $\boldsymbol{x}$ drawn from the normal distribution $f_{\text{mvn}}(\boldsymbol{x})$ is denoted by

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

Nevertheless, decades of empirical studies have clearly demonstrated that financial data do not follow a Gaussian distribution (see Chapter 2 for stylized facts of financial data). Thus, we need more general distributions that can model the skewness (i.e., asymmetry) and kurtosis (i.e., heavy tails).

Multivariate Normal Mixture Distributions

The multivariate normal can be generalized to obtain multivariate normal mixture distributions. The crucial idea is the introduction of randomness into first the covariance matrix and then the mean vector of a multivariate normal distribution via a positive mixing variable, denoted by w (McNeil et al., 2015).

A multivariate normal *variance mixture* can be represented as

$$\boldsymbol{x} = \boldsymbol{\mu} + \sqrt{w}\boldsymbol{z},$$

where $\boldsymbol{\mu}$ is referred to as the *location vector*, $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ with $\boldsymbol{\Sigma}$ referred to as the *scatter matrix*, and w is a nonnegative scalar-valued random variable independent of $\boldsymbol{z}$. Observe that the random variable w only affects the covariance matrix, but not the mean,

$$\begin{aligned} \mathbb{E}[\boldsymbol{x}] &= \boldsymbol{\mu}, \\ \text{Cov}(\boldsymbol{x}) &= \mathbb{E}[w]\boldsymbol{\Sigma}, \end{aligned}$$

hence the name variance mixture.

One important example of a normal variance mixture is the *multivariate t* distribution, obtained when w follows an inverse gamma distribution $w \sim \text{Ig}\left(\frac{\nu}{2}, \frac{\nu}{2}\right)$, which is equivalent to saying that $\tau = 1/w$ follows a gamma distribution $\tau \sim \text{Gamma}\left(\frac{\nu}{2}, \frac{\nu}{2}\right)$.[3] This can be represented in a hierarchical structure as

$$\begin{aligned} \boldsymbol{x} \mid \tau &\sim \mathcal{N}\left(\boldsymbol{\mu}, \frac{1}{\tau}\boldsymbol{\Sigma}\right), \\ \tau &\sim \text{Gamma}\left(\frac{\nu}{2}, \frac{\nu}{2}\right). \end{aligned}$$

The multivariate t distribution (also called Student's t distribution) is widely used to model heavy tails (via the parameter ν) in finance and other areas. However, it cannot capture the asymmetries observed in financial data.

A more general case of normal variance mixture is the *multivariate symmetric generalized hyperbolic* distribution, obtained when w follows a generalized inverse Gaussian (GIG) distribution, which contains the inverse gamma distribution as a particular case. But this

[3] Gamma(a, b) represents the gamma distribution of shape a and rate b, which has the pdf

$$f(\tau \mid a, b) = b^a \tau^{a-1} \frac{\exp(-b\tau)}{\Gamma(a)},$$

where $\Gamma(a)$ is the gamma function, $\Gamma(a) = \int_0^\infty t^{a-1} e^{-t}\, \mathrm{d}t$.

distribution still cannot capture asymmetries since it is a variance mixture. In order to model asymmetry, the mixture has to affect the mean as well.

A multivariate normal *mean–variance mixture* can be represented as

$$\boldsymbol{x} = \boldsymbol{m}(w) + \sqrt{w}\boldsymbol{z},$$

where $\boldsymbol{m}(w)$ is now some function of w and the rest is as before for variance mixtures. A typical example of $\boldsymbol{m}(w)$ is $\boldsymbol{m}(w) = \boldsymbol{\mu} + w\boldsymbol{\gamma}$, leading to

$$\begin{aligned}\mathbb{E}[\boldsymbol{x}] &= \boldsymbol{\mu} + \mathbb{E}[w]\boldsymbol{\gamma},\\ \mathrm{Cov}(\boldsymbol{x}) &= \mathbb{E}[w]\boldsymbol{\Sigma} + \mathrm{var}(w)\boldsymbol{\gamma}\boldsymbol{\gamma}^\intercal.\end{aligned}$$

An example of a mean–variance mixture is the *multivariate generalized hyperbolic* (GH) distribution, obtained when $\boldsymbol{m}(w) = \boldsymbol{\mu} + w\boldsymbol{\gamma}$ and w follows a GIG distribution. If the GIG distribution is further particularized to an inverse gamma distribution, then the *multivariate skewed t* distribution (or *generalized hyperbolic multivariate skewed t* (ghMST)) is obtained, which can be conveniently represented in a hierarchical structure as

$$\begin{aligned}\boldsymbol{x} \mid \tau &\sim \mathcal{N}\left(\boldsymbol{\mu} + \frac{1}{\tau}\boldsymbol{\gamma}, \frac{1}{\tau}\boldsymbol{\Sigma}\right),\\ \tau &\sim \mathrm{Gamma}\left(\frac{\nu}{2}, \frac{\nu}{2}\right).\end{aligned}$$

Other distributions more general than the ghMST could be considered, such as the restricted multivariate skewed t (rMST) distribution and the unrestricted multivariate skewed t (uMST) distribution; see X. Wang et al. (2023) and references therein for more details. However, these more complex distributions are significantly more complicated to fit to data (e.g., the uMST can only be fitted for $N < 10$ in practice due to the computational complexity) and do not seem to provide any advantage in modeling the asymmetries of financial data. Figure 9.2 shows the goodness of fit (via the out-of-sample likelihood of the fit) of several multivariate distributions from the simplest Gaussian to the most complicated uMST. The skewed t distribution seems to obtain a good fit while preserving its simplicity.

Parametric Moments

The advantage of assuming a parametric model for the data is that the computation of the moments simplifies a great deal. To be specific, under the multivariate skewed t distribution, the first four moments are conveniently simplified (Birgeand & Chavez-Bedoya, 2016; X. Wang et al., 2023) to

$$\begin{aligned}\phi_1(\boldsymbol{w}) &= \boldsymbol{w}^\intercal\boldsymbol{\mu} + a_1\boldsymbol{w}^\intercal\boldsymbol{\gamma},\\ \phi_2(\boldsymbol{w}) &= a_{21}\boldsymbol{w}^\intercal\boldsymbol{\Sigma}\boldsymbol{w} + a_{22}(\boldsymbol{w}^\intercal\boldsymbol{\gamma})^2,\\ \phi_3(\boldsymbol{w}) &= a_{31}(\boldsymbol{w}^\intercal\boldsymbol{\gamma})^3 + a_{32}(\boldsymbol{w}^\intercal\boldsymbol{\gamma})\boldsymbol{w}^\intercal\boldsymbol{\Sigma}\boldsymbol{w},\\ \phi_4(\boldsymbol{w}) &= a_{41}(\boldsymbol{w}^\intercal\boldsymbol{\gamma})^4 + a_{42}(\boldsymbol{w}^\intercal\boldsymbol{\gamma})^2\boldsymbol{w}^\intercal\boldsymbol{\Sigma}\boldsymbol{w} + a_{43}(\boldsymbol{w}^\intercal\boldsymbol{\Sigma}\boldsymbol{w})^2,\end{aligned} \tag{9.6}$$

where $a_1 = \frac{\nu}{\nu-2}$, $a_{21} = a_1$, $a_{22} = \frac{2\nu^2}{(\nu-2)^2(\nu-4)}$, $a_{31} = \frac{16\nu^3}{(\nu-2)^3(\nu-4)(\nu-6)}$, $a_{32} = \frac{6\nu^2}{(\nu-2)^2(\nu-4)}$, $a_{41} = \frac{(12\nu+120)\nu^4}{(\nu-2)^4(\nu-4)(\nu-6)(\nu-8)}$, $a_{42} = \frac{6(2\nu+4)\nu^3}{(\nu-2)^3(\nu-4)(\nu-6)}$, and $a_{43} = \frac{3\nu^2}{(\nu-2)(\nu-4)}$.

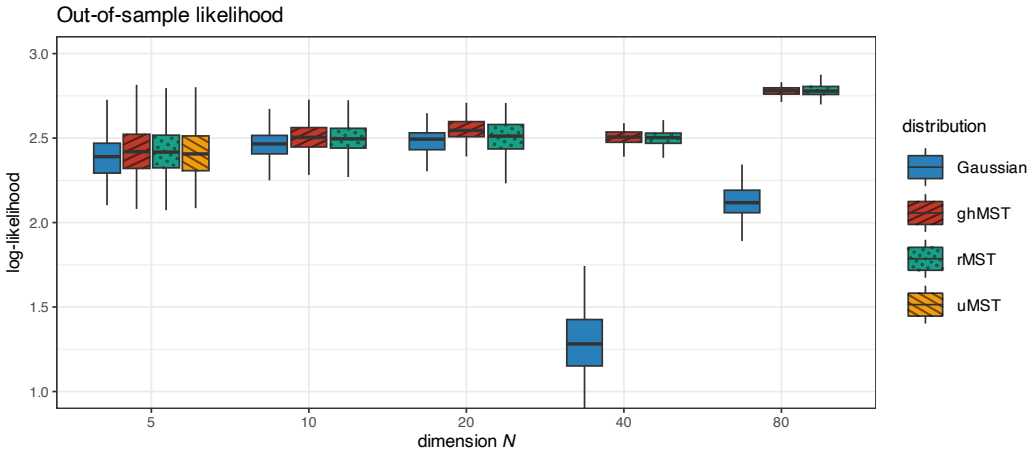


Figure 9.2 Likelihood of different fitted multivariate distributions for S&P 500 daily stock returns.

It is important to emphasize that now $\boldsymbol{\mu}$ refers to the location vector and not the mean; similarly, $\boldsymbol{\Sigma}$ now refers to the scatter matrix and not the covariance matrix.

The gradients and Hessians are given by

$$\begin{aligned}
\nabla\phi_1(\boldsymbol{w}) &= \boldsymbol{\mu} + a_1\boldsymbol{\gamma}, \\
\nabla\phi_2(\boldsymbol{w}) &= 2a_{21}\boldsymbol{\Sigma}\boldsymbol{w} + 2a_{22}(\boldsymbol{w}^\top\boldsymbol{\gamma})\boldsymbol{\gamma}, \\
\nabla\phi_3(\boldsymbol{w}) &= 3a_{31}(\boldsymbol{w}^\top\boldsymbol{\gamma})^2\boldsymbol{\gamma} + a_{32}\left((\boldsymbol{w}^\top\boldsymbol{\Sigma}\boldsymbol{w})\boldsymbol{\gamma} + 2(\boldsymbol{w}^\top\boldsymbol{\gamma})\boldsymbol{\Sigma}\boldsymbol{w}\right), \\
\nabla\phi_4(\boldsymbol{w}) &= 4a_{41}(\boldsymbol{w}^\top\boldsymbol{\gamma})^3\boldsymbol{\gamma} \\
&\quad + 2a_{42}\left((\boldsymbol{w}^\top\boldsymbol{\gamma})^2\boldsymbol{\Sigma}\boldsymbol{w} + (\boldsymbol{w}^\top\boldsymbol{\Sigma}\boldsymbol{w})(\boldsymbol{w}^\top\boldsymbol{\gamma})\boldsymbol{\gamma}\right) + 4a_{43}(\boldsymbol{w}^\top\boldsymbol{\Sigma}\boldsymbol{w})\boldsymbol{\Sigma}\boldsymbol{w}
\end{aligned} \tag{9.7}$$

and

$$\begin{aligned}
\nabla^2\phi_1(\boldsymbol{w}) &= \mathbf{0}, \\
\nabla^2\phi_2(\boldsymbol{w}) &= 2a_{21}\boldsymbol{\Sigma} + 2a_{22}\boldsymbol{\gamma}\boldsymbol{\gamma}^\top, \\
\nabla^2\phi_3(\boldsymbol{w}) &= 6a_{31}(\boldsymbol{w}^\top\boldsymbol{\gamma})\boldsymbol{\gamma}\boldsymbol{\gamma}^\top + 2a_{32}\left(\boldsymbol{\gamma}\boldsymbol{w}^\top\boldsymbol{\Sigma} + \boldsymbol{\Sigma}\boldsymbol{w}\boldsymbol{\gamma}^\top + (\boldsymbol{w}^\top\boldsymbol{\gamma})\boldsymbol{\Sigma}\right), \\
\nabla^2\phi_4(\boldsymbol{w}) &= 12a_{41}(\boldsymbol{w}^\top\boldsymbol{\gamma})^2\boldsymbol{\gamma}\boldsymbol{\gamma}^\top \\
&\quad + 2a_{42}\left(2(\boldsymbol{w}^\top\boldsymbol{\gamma})\boldsymbol{\Sigma}\boldsymbol{w}\boldsymbol{\gamma}^\top + (\boldsymbol{w}^\top\boldsymbol{\gamma})^2\boldsymbol{\Sigma} + 2(\boldsymbol{w}^\top\boldsymbol{\gamma})\boldsymbol{\gamma}\boldsymbol{w}^\top\boldsymbol{\Sigma} + (\boldsymbol{w}^\top\boldsymbol{\Sigma}\boldsymbol{w})\boldsymbol{\gamma}\boldsymbol{\gamma}^\top\right) \\
&\quad + 4a_{43}\left(2\boldsymbol{\Sigma}\boldsymbol{w}\boldsymbol{w}^\top\boldsymbol{\Sigma} + (\boldsymbol{w}^\top\boldsymbol{\Sigma}\boldsymbol{w})\boldsymbol{\Sigma}\right),
\end{aligned} \tag{9.8}$$

respectively (X. Wang et al., 2023).

9.2.4 L-Moments

One property of higher-order moments is that they fully characterize the distribution function of the random variable. In addition, the first four moments convey descriptive properties of the random variable such as location, dispersion, asymmetry, and thickness of tails.

Interestingly, the so-called "L-moments" also characterize the distribution of a random variable and similarly convey descriptive properties (Hosking, 1990). In addition, they are *linear* functions of the order statistics and are easier to estimate in practice.

Let X be a random variable and $X_{1:n} \leq X_{2:n} \leq \cdots \leq X_{n:n}$ be the *order statistics* of a random sample of size n drawn from the distribution of X. The *L-moments* of X are defined as

$$\lambda_r = \frac{1}{r} \sum_{k=0}^{r-1} (-1)^k \binom{r-1}{k} \mathbb{E}[X_{r-k:r}], \quad r = 1, 2, \ldots$$

In particular, the first four L-moments are

$$\begin{aligned}
\lambda_1 &= \mathbb{E}[X], \\
\lambda_2 &= \frac{1}{2}\mathbb{E}[X_{2:2} - X_{1:2}], \\
\lambda_3 &= \frac{1}{3}\mathbb{E}[X_{3:3} - 2X_{2:3} + X_{1:3}] \\
&= \frac{1}{3}\mathbb{E}[(X_{3:3} - X_{2:3}) - (X_{2:3} - X_{1:3})], \\
\lambda_4 &= \frac{1}{4}\mathbb{E}[X_{4:4} - 3X_{3:4} + 3X_{2:4} - X_{1:4}] \\
&= \frac{1}{4}\mathbb{E}[(X_{4:4} - X_{1:4}) - 3(X_{3:4} - X_{2:4})].
\end{aligned} \tag{9.9}$$

These moments provide descriptive information similar to the regular moments (Hosking, 1990):

- The *L-location*, λ_1, is identical to the mean $\bar{X}$.
- The *L-scale*, λ_2, measures the expected difference between any two realizations. It resembles the variance expressed as $\sigma^2 = \frac{1}{2}\mathbb{E}[(X_{2:2} - X_{1:2})^2]$.
- The *L-skewness*, λ_3, is the expected difference of differences. It provides a measure of the asymmetry less sensitive to extreme tails than the regular skewness $\mathbb{E}[(X - \bar{X})^3]$, which makes its estimation more accurate in practice.
- The *L-kurtosis*, λ_4, measures the expected exceedance of the largest difference. It is a measure of how thick the tails are, similar to the regular kurtosis $\mathbb{E}[(X - \bar{X})^4]$, but, again, less sensitive to extreme tails, which makes its estimation more accurate in practice.

A direct estimation of the moments in (9.9) from a set of observations would be computationally demanding (if not insurmountable), because the possible numbers of combinations of two, three, and four values from a sample can be quite large even for a relatively small sample size. Fortunately, there is a much simpler way to cover all the possible combinations, leading to the following estimators for the L-moments in terms of the sample values in ascending order,

$x_{(1)} \le x_{(2)} \le \cdots \le x_{(n)}$ (Q. J. Wang, 1996):

$$\begin{aligned}
\hat{\lambda}_1 &= \frac{1}{C_1^n} \sum_{i=1}^{n} x_{(i)} = \frac{1}{n} \sum_{i=1}^{n} x_{(i)}, \\
\hat{\lambda}_2 &= \frac{1}{2} \frac{1}{C_2^n} \sum_{i=1}^{n} \left(C_1^{i-1} - C_1^{n-i} \right) x_{(i)}, \\
\hat{\lambda}_3 &= \frac{1}{3} \frac{1}{C_3^n} \sum_{i=1}^{n} \left(C_2^{i-1} - 2C_1^{i-1} C_1^{n-i} + C_2^{n-i} \right) x_{(i)}, \\
\hat{\lambda}_4 &= \frac{1}{4} \frac{1}{C_4^n} \sum_{i=1}^{n} \left(C_3^{i-1} - 3C_2^{i-1} C_1^{n-i} + 3C_1^{i-1} C_2^{n-i} - C_3^{n-i} \right) x_{(i)},
\end{aligned}$$

where $C_k^m \triangleq \binom{m}{k} = \frac{m!}{k!(m-k)!}$ is the number of combinations of any k items from m items (equals zero when $k > m$).

Figure 9.3 compares the moments and L-moments of the S&P 500 index returns. The L-moments clearly convey a similar information to the regular moments. In addition, they seem to be more stable (in the sense that they do not exhibit jumps as high as the regular moments).

It seems that the L-moments may be superior to the regular moments in the sense that they are more stable and convey similar useful information. However, when it comes to the portfolio design, they come with the additional difficulty of requiring sorted returns, whose ordering depends on the portfolio $\boldsymbol{w}$:

$$\boldsymbol{w}^\mathsf{T} \boldsymbol{r}_1, \boldsymbol{w}^\mathsf{T} \boldsymbol{r}_2, \ldots, \boldsymbol{w}^\mathsf{T} \boldsymbol{r}_T \quad \longrightarrow \quad \boldsymbol{w}^\mathsf{T} \boldsymbol{r}_{\tau(1)} \le \boldsymbol{w}^\mathsf{T} \boldsymbol{r}_{\tau(2)} \le \cdots \le \boldsymbol{w}^\mathsf{T} \boldsymbol{r}_{\tau(T)},$$

where $\tau(\cdot)$ denotes a permutation of the T observations so that the portfolio returns are sorted in increasing order.

Thus, given the permutation $\tau(\cdot)$, the portfolio moments can be expressed as

$$\begin{aligned}
\hat{\phi}_1(\boldsymbol{w}) &= \frac{1}{n} \sum_{t=1}^{T} \boldsymbol{w}^\mathsf{T} \boldsymbol{r}_{\tau(t)}, \\
\hat{\phi}_2(\boldsymbol{w}) &= \frac{1}{2} \frac{1}{C_2^T} \sum_{t=1}^{T} \left(C_1^{t-1} - C_1^{T-t} \right) \boldsymbol{w}^\mathsf{T} \boldsymbol{r}_{\tau(t)}, \\
\hat{\phi}_3(\boldsymbol{w}) &= \frac{1}{3} \frac{1}{C_3^T} \sum_{t=1}^{T} \left(C_2^{t-1} - 2C_1^{t-1} C_1^{T-t} + C_2^{T-t} \right) \boldsymbol{w}^\mathsf{T} \boldsymbol{r}_{\tau(t)}, \\
\hat{\phi}_4(\boldsymbol{w}) &= \frac{1}{4} \frac{1}{C_4^T} \sum_{t=1}^{T} \left(C_3^{t-1} - 3C_2^{t-1} C_1^{T-t} + 3C_1^{t-1} C_2^{T-t} - C_3^{T-t} \right) \boldsymbol{w}^\mathsf{T} \boldsymbol{r}_{\tau(t)}.
\end{aligned} \tag{9.10}$$

9.3 High-Order Portfolio Formulations

We now explore different formulations involving high-order moments. Recall that higher-order moments will make the formulations nonconvex, unlike the mean–variance formulation.

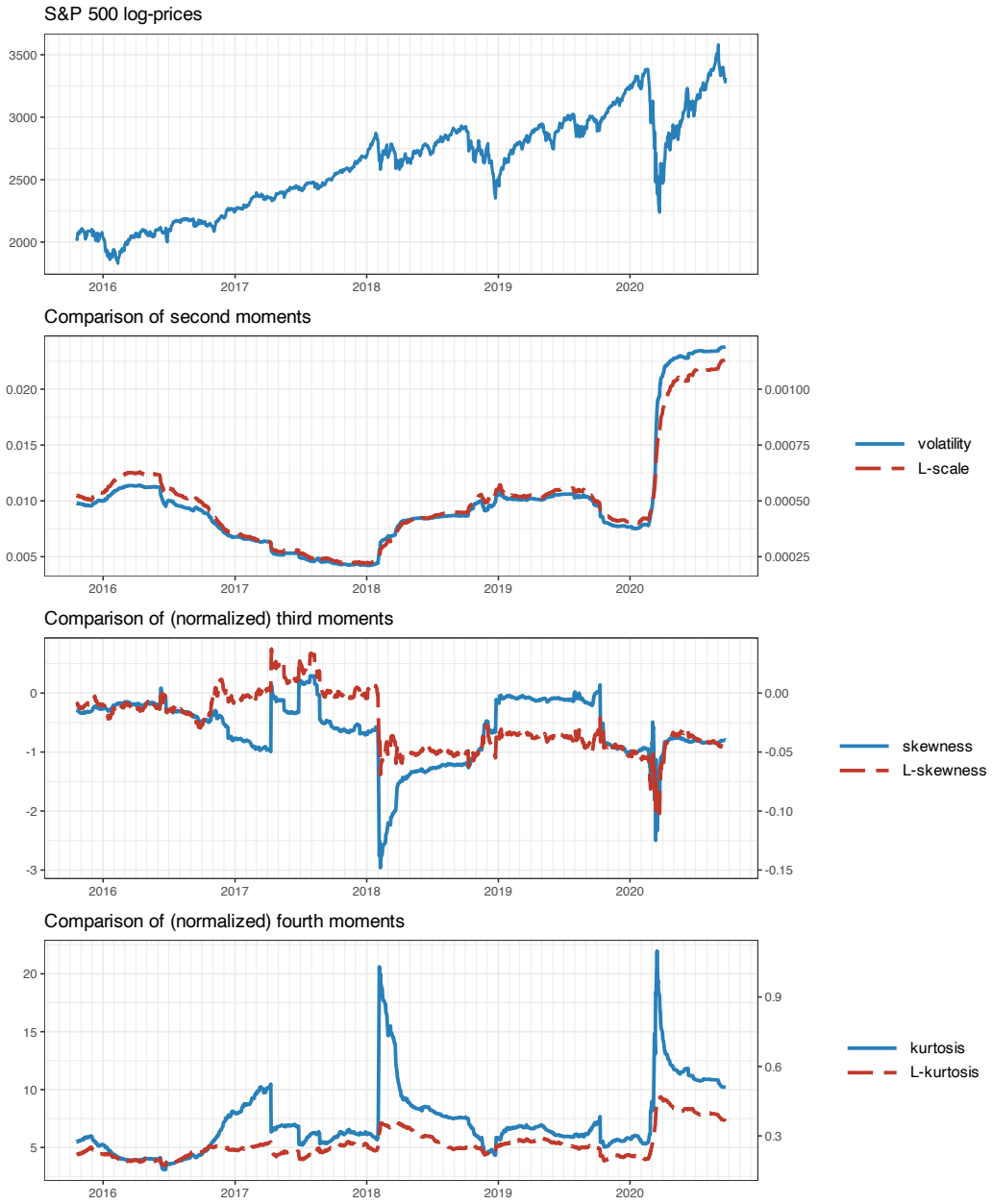


Figure 9.3 Moments and L-moments of the S&P 500 index in a rolling-window fashion.

As explored in Section 9.2, there are several options for the expressions of the moments $\phi_1(\boldsymbol{w})$, $\phi_2(\boldsymbol{w})$, $\phi_3(\boldsymbol{w})$, and $\phi_4(\boldsymbol{w})$:

- *nonparametric moments*: as in (9.1);
- *factor model structured moments*: as in (9.4) for a single factor or (9.5) for multiple factors;

- *parametric moments*: following the multivariate skewed t as in (9.6); and
- *L-moments*: as in (9.10).

9.3.1 MVSK Portfolios

A natural and straightforward way to incorporate higher-order moments in Markowitz's mean–variance framework is by optimizing a weighted combination of the first four moments, called the *mean–variance–skewness–kurtosis* (MVSK) portfolio:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & -\lambda_1\phi_1(\boldsymbol{w}) + \lambda_2\phi_2(\boldsymbol{w}) - \lambda_3\phi_3(\boldsymbol{w}) + \lambda_4\phi_4(\boldsymbol{w}) \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}. \end{array} \tag{9.11}$$

The hyper-parameters λ_1, λ_2, λ_3, and λ_4 are chosen, as usual, according to the investor's risk aversion. Observe that a reasonable investor seeks higher values of the first and third moments (i.e., mean and skewness), $\phi_1(\boldsymbol{w})$ and $\phi_3(\boldsymbol{w})$, and lower values of the second and fourth moments (i.e., variance and kurtosis), $\phi_2(\boldsymbol{w})$ and $\phi_4(\boldsymbol{w})$ (Briec et al., 2007; Martellini & Ziemann, 2010; Scott & Horvath, 1980). An interesting case of (9.11) arises when $\lambda_1 = 0$, that is, ignoring the mean like in the global minimum variance portfolio (GMVP); see Section 6.5.1 in Chapter 6.

A convenient choice for the hyper-parameters is according to the constant relative risk aversion (Martellini & Ziemann, 2010):

$$\begin{aligned} \lambda_1 &= 1, \\ \lambda_2 &= \frac{\gamma}{2}, \\ \lambda_3 &= \frac{\gamma(\gamma+1)}{6}, \\ \lambda_4 &= \frac{\gamma(\gamma+1)(\gamma+2)}{24}, \end{aligned}$$

where $\gamma \geq 0$ is the risk aversion parameter.

As is customary, this high-order portfolio design could be alternatively formulated with any of the moments as constraints. For example, the feasibility problem

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{find}} & \boldsymbol{w} \\ \text{subject to} & \phi_1(\boldsymbol{w}) \geq \alpha_1, \\ & \phi_2(\boldsymbol{w}) \leq \alpha_2, \\ & \phi_3(\boldsymbol{w}) \geq \alpha_3, \\ & \phi_4(\boldsymbol{w}) \leq \alpha_4, \end{array}$$

where the hyper-parameters are given by α_1, α_2, α_3, and α_4, denoting the investor's preference.

Efficient numerical algorithms specifically designed to solve the MVSK formulation in (9.11) are discussed in Section 9.4, based on Zhou and Palomar (2021) and X. Wang et al. (2023).

Expected-Utility Approximations

Expected utility theory in the context of portfolio design was explored in Section 7.3 of Chapter 7. The idea is to maximize the expected value of a utility function, $\mathbb{E}\left[U(\boldsymbol{w}^\mathsf{T}\boldsymbol{r})\right]$, where $U(\cdot)$ denotes some utility function, instead of the mean–variance objective $\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \frac{\lambda}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}$.

High-order portfolios were considered in Young and Trent (1969) using the following approximation for the geometric mean of the returns:

$$\mathbb{E}\left[\log\left(1+\boldsymbol{w}^\mathsf{T}\boldsymbol{r}\right)\right] \approx \log\left(1+\phi_1(\boldsymbol{w})\right) - \frac{\phi_2(\boldsymbol{w})}{2\phi_1^2(\boldsymbol{w})} + \frac{\phi_3(\boldsymbol{w})}{3\phi_1^3(\boldsymbol{w})} - \frac{\phi_4(\boldsymbol{w})}{4\phi_1^4(\boldsymbol{w})},$$

where the approximation up to the first two terms coincides to that in (7.16). High-order expansions were also considered for arbitrary expected utilities (Jean, 1971). More recently, Martellini and Ziemann (2010) considered high-order approximations of expected utilities with structured estimators of the moments as in (9.4).

9.3.2 Making Portfolios Efficient

The *shortage function* is an important quantity in multi-objective optimization related to the efficient frontier and the Pareto-optimal points (see Section A.7 in Appendix A).

The shortage function measures the distance between the moments of a portfolio and the efficient frontier along a given direction. Based on this concept, given a reference portfolio $\boldsymbol{w}^0$ and a direction vector $\boldsymbol{g}$, we can optimize a portfolio by pushing the reference portfolio towards the efficient frontier along that direction (Briec et al., 2007; Jurczenko et al., 2006):

$$\begin{array}{ll} \underset{\boldsymbol{w},\,\delta\geq 0}{\text{maximize}} & \delta \\ \text{subject to} & \phi_1(\boldsymbol{w}) \geq \phi_1(\boldsymbol{w}^0) + \delta g_1, \\ & \phi_2(\boldsymbol{w}) \leq \phi_2(\boldsymbol{w}^0) - \delta g_2, \\ & \phi_3(\boldsymbol{w}) \geq \phi_3(\boldsymbol{w}^0) + \delta g_3, \\ & \phi_4(\boldsymbol{w}) \leq \phi_4(\boldsymbol{w}^0) - \delta g_4. \end{array} \tag{9.12}$$

Observe that this formulation is always feasible. In the case that the reference portfolio $\boldsymbol{w}^0$ was already on the efficient frontier, then the solution will be $\boldsymbol{w} = \boldsymbol{w}^0$ and $\delta = 0$.

9.3.3 Portfolio Tilting

The formulation in (9.12) to improve a given reference portfolio $\boldsymbol{w}^0$ can be further extended by introducing a measure of portfolio optimality.

Suppose that the reference portfolio $\boldsymbol{w}^0$ is obtained as the solution to the minimization of some cost function $\xi(\cdot)$:

$$\boldsymbol{w}^0 = \arg\min_{\boldsymbol{w}\in\mathcal{W}} \xi(\boldsymbol{w}).$$

Some illustrative examples of the cost function $\xi(\cdot)$ are

- the Herfindahl index of the portfolio weights to promote diversity (see Section 7.1.5 in Chapter 7):

$$\xi(\boldsymbol{w}) = \sum_{i=1}^{N} w_i^2;$$

- equalization of risk contributions (see the risk parity portfolio in Chapter 11):

$$\xi(\boldsymbol{w}) = \sum_{i=1}^{N} \left(\frac{w_i(\boldsymbol{\Sigma}\boldsymbol{w})_i}{\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}} - \frac{1}{N} \right)^2;$$

- diversification ratio (see Section 6.5 in Chapter 6):

$$\xi(\boldsymbol{w}) = -\frac{\boldsymbol{w}^\mathsf{T}\boldsymbol{\sigma}}{\sqrt{\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}}};$$

- tracking error of a benchmark portfolio $\boldsymbol{w}^\mathrm{b}$ (see index tracking in Chapter 13):

$$\xi(\boldsymbol{w}) = \sqrt{(\boldsymbol{w} - \boldsymbol{w}^\mathrm{b})^\mathsf{T}\boldsymbol{\Sigma}(\boldsymbol{w} - \boldsymbol{w}^\mathrm{b})}.$$

The so-called MVSK *portfolio tilting* is formulated (Boudt, Cornilly, Holle, & Willems, 2020) as

$$\begin{aligned} \underset{\boldsymbol{w},\,\delta\geq 0}{\text{maximize}} \quad & \delta \\ \text{subject to} \quad & \xi(\boldsymbol{w}) \leq \xi(\boldsymbol{w}^0) + \kappa, \\ & \phi_1(\boldsymbol{w}) \geq \phi_1(\boldsymbol{w}^0) + g_1(\delta), \\ & \phi_2(\boldsymbol{w}) \leq \phi_2(\boldsymbol{w}^0) - g_2(\delta), \\ & \phi_3(\boldsymbol{w}) \geq \phi_3(\boldsymbol{w}^0) + g_3(\delta), \\ & \phi_4(\boldsymbol{w}) \leq \phi_4(\boldsymbol{w}^0) - g_4(\delta), \end{aligned} \tag{9.13}$$

where $g_i(\delta)$ are increasing functions of δ, and $\kappa > 0$ is the "sacrifice parameter" to allow for some loss of optimality with respect to the reference portfolio (according to the cost function $\xi(\cdot)$) in exchange for getting closer to the efficient frontier.

One simple way to choose the hyper-parameters is proportional to the reference values, for example:

$$\begin{aligned} \kappa &= 0.01 \times \xi(\boldsymbol{w}^0), \\ g_1(\delta) &= \delta \times \phi_1(\boldsymbol{w}^0), \\ g_2(\delta) &= \delta \times \phi_2(\boldsymbol{w}^0), \\ g_3(\delta) &= \delta \times \phi_3(\boldsymbol{w}^0), \\ g_4(\delta) &= \delta \times \phi_4(\boldsymbol{w}^0). \end{aligned}$$

Efficient numerical algorithms specifically designed to solve the MVSK tilting portfolio formulation were developed in Zhou and Palomar (2021).

9.3.4 Polynomial Goal Programming MVSK Portfolio

Another possible way to obtain a trade-off among the moments can be formulated as the so-called *polynomial goal programming* into which the investor's preferences and objectives

are incorporated. The formulation is based on minimizing the distance to some reference moments measured with a polynomial (Lai, 1991):

$$\begin{array}{ll} \underset{\boldsymbol{w},\boldsymbol{d}\geq\mathbf{0}}{\text{minimize}} & \left|\frac{d_1}{\phi_1^0}\right|^{\lambda_1} + \left|\frac{d_2}{\phi_2^0}\right|^{\lambda_2} + \left|\frac{d_3}{\phi_3^0}\right|^{\lambda_3} + \left|\frac{d_4}{\phi_4^0}\right|^{\lambda_4} \\ \text{subject to} & \phi_1(\boldsymbol{w}) + d_1 \geq \phi_1^0, \\ & \phi_2(\boldsymbol{w}) - d_2 \leq \phi_2^0, \\ & \phi_3(\boldsymbol{w}) + d_3 \geq \phi_3^0, \\ & \phi_4(\boldsymbol{w}) - d_4 \leq \phi_4^0, \end{array}$$

where $\boldsymbol{d}$ denotes the deviation from the so-called "aspired levels" of the moments ϕ_1^0, ϕ_2^0, ϕ_3^0, and ϕ_4^0, which can be obtained, for example, as the extreme values $\phi_i^0 = \max(\min)_{\boldsymbol{w}\in\mathcal{W}} \phi_i(\boldsymbol{w})$. Observe that these aspired levels are not jointly achievable by a single portfolio and that is where the vector variable $\boldsymbol{d} \geq \mathbf{0}$ comes into play to relax the problem. If the aspired levels could be achieved by a portfolio $\boldsymbol{w}^0$, then the optimal solution would simply be $\boldsymbol{w} = \boldsymbol{w}^0$ and $\boldsymbol{d} = \mathbf{0}$.

One particular case of this polynomial goal programming is when using the Minkowski distance (where the exponents are set to $\lambda_i = 1/p$):

$$\begin{array}{ll} \underset{\boldsymbol{w},\boldsymbol{d}\geq\mathbf{0}}{\text{minimize}} & \left(\sum_{i=1}^{4}\left|\frac{d_i}{\phi_i^0}\right|^p\right)^{1/p} \\ \text{subject to} & \phi_1(\boldsymbol{w}) + d_1 \geq \phi_1^0, \\ & \phi_2(\boldsymbol{w}) - d_2 \leq \phi_2^0, \\ & \phi_3(\boldsymbol{w}) + d_3 \geq \phi_3^0, \\ & \phi_4(\boldsymbol{w}) - d_4 \leq \phi_4^0. \end{array}$$

9.3.5 L-Moment Portfolios

We now turn to the L-moments in (9.10) obtained from the sorted portfolio returns

$$\boldsymbol{w}^\top \boldsymbol{r}_{\tau(1)} \leq \boldsymbol{w}^\top \boldsymbol{r}_{\tau(2)} \leq \cdots \leq \boldsymbol{w}^\top \boldsymbol{r}_{\tau(T)},$$

where the permutation $\tau(\cdot)$ is a critical component that makes the problem nonconvex and difficult to handle.

Plugging the expressions of the L-moments in terms of sorted portfolio returns, as in (9.10), into the MSVK portfolio formulation in (9.11) leads to

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \sum_{i=1}^{T} v_i \boldsymbol{w}^\top \boldsymbol{r}_{\tau(i)} \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array}$$

for properly chosen weights v_i.

The function in the objective $\sum_{i=1}^{T} v_i \boldsymbol{w}^\top \boldsymbol{r}_{\tau(i)}$ involving ordered values is called *ordered weighted averaging* (OWA) and was studied in the 1990s. It turns out that such a problem can be reformulated in terms of auxiliary integer (actually binary) variables, which shows that the problem is in general a nonconvex integer problem (Yager, 1996).

To be specific, the OWA problem

$$\begin{array}{ll} \underset{\boldsymbol{w},\{x_t\}}{\text{maximize}} & \sum_{i=1}^{T} v_i x_{(i)} \\ \text{subject to} & x_t = \boldsymbol{w}^\top \boldsymbol{r}_t, \qquad t = 1, \ldots, T, \\ & \boldsymbol{w} \in \mathcal{W} \end{array} \tag{9.14}$$

is equivalent to the mixed-integer linear program (Yager, 1996)

$$\begin{array}{lll} \underset{\boldsymbol{w},\{x_t\},\{y_t\},\{z_{ij}\}}{\text{maximize}} & \displaystyle\sum_{i=1}^{T} v_i y_i & \\ \text{subject to} & x_t = \boldsymbol{w}^\top \boldsymbol{r}_t, & t = 1, \ldots, T, \\ & \boldsymbol{w} \in \mathcal{W}, & \\ & y_1 \le y_2 \le \cdots \le y_T, & \\ & y_i \mathbf{1} \le \boldsymbol{x} + M\boldsymbol{z}_i, & i = 1, \ldots, T, \\ & \mathbf{1}^\top \boldsymbol{z}_i \le i - 1, & \\ & z_{ij} \in \{0, 1\}, & \end{array}$$

where the weights v_i are assumed to be nonnegative and M is a sufficiently large constant (much larger than any possible value that any of the x_t or y_t can take).

If the weights v_i are positive and decreasing, it was shown (Ogryczak, 2000) that the OWA objective function is a concave piecewise linear function,

$$\sum_{i=1}^{T} v_i x_{(i)} = \min_{\tau \in \Pi} \left(\sum_{i=1}^{T} v_{\tau(i)} x_i \right),$$

where $\tau(\cdot)$ is a permutation and Π the set of all possible $T!$ permutations for a set of length T. Then the OWA problem (9.14) can be rewritten as the linear program

$$\begin{array}{ll} \underset{\boldsymbol{w},s}{\text{maximize}} & s \\ \text{subject to} & s \le \sum_{t=1}^{T} v_{\tau(t)} \boldsymbol{w}^\top \boldsymbol{r}_t, \qquad \text{for all } \tau \in \Pi, \\ & \boldsymbol{w} \in \mathcal{W}. \end{array}$$

This formulation, unfortunately, has $T!$ constraints involved in all possible permutations, which makes its usefulness questionable. An efficient dual implementation was considered in Ogryczak and Sliwinski (2003). An alternative is based on relaxing the set of permutations to its convex hull, which does not change the minimum value (Chassein & Goerigk, 2015).

Yet another reformulation of the OWA problem (9.14) is in terms of the cumulative ordered values, defined as $\bar{x}_i = \sum_{j=1}^{i} x_{(i)}$, which allows us to write

$$\sum_{i=1}^{T} v_i x_{(i)} = \sum_{i=1}^{T} v'_i \bar{x}_i,$$

where $v'_i = v_i - v_{i+1}$, for $i = 1, \ldots, T-1$, and $v'_T = v_T$. It was shown (Ogryczak & Sliwinski, 2003) that

$$\bar{x}_i = \max_{y_i} \left\{ i y_i - \mathbf{1}^\top (\boldsymbol{x} - y_i \mathbf{1})^+ \right\}.$$

Thus, if the weights v_i are positive and decreasing, then $v_i' > 0$ for $i = 1, \ldots, T$ and the OWA formulation can be written as the following problem (Ogryczak & Sliwinski, 2003):

$$\begin{array}{ll} \underset{\boldsymbol{w},\boldsymbol{x},\boldsymbol{y}}{\text{maximize}} & \displaystyle\sum_{i=1}^{T} v_i' \left(i y_i - \mathbf{1}^{\mathsf{T}} (\boldsymbol{x} - y_i \mathbf{1})^{+} \right) \\ \text{subject to} & x_t = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{r}_t, \qquad t = 1, \ldots, T, \\ & \boldsymbol{w} \in \mathcal{W}, \end{array}$$

which can be easily rewritten as a linear program:

$$\begin{array}{ll} \underset{\boldsymbol{w},\boldsymbol{x},\boldsymbol{y},\boldsymbol{s} \geq \mathbf{0}}{\text{maximize}} & \displaystyle\sum_{i=1}^{T} v_i' \left(i y_i - \mathbf{1}^{\mathsf{T}} s_i \right) \\ \text{subject to} & x_t = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{r}_t, \qquad t = 1, \ldots, T, \\ & s_i \geq \boldsymbol{x} - y_i \mathbf{1}, \quad i = 1, \ldots, T, \\ & \boldsymbol{w} \in \mathcal{W}. \end{array}$$

Nevertheless, if the weights v_i are not positive and decreasing, then the problem cannot be simplified as above.

9.4 Algorithms

We will focus only on the MVSK portfolio formulation (9.11), whose objective function,

$$f(\boldsymbol{w}) = -\lambda_1 \phi_1(\boldsymbol{w}) + \lambda_2 \phi_2(\boldsymbol{w}) - \lambda_3 \phi_3(\boldsymbol{w}) + \lambda_4 \phi_4(\boldsymbol{w}), \tag{9.15}$$

is nonconvex due to the higher-order moments.

It is convenient to split the nonconvex MVSK objective function (9.15) into convex and nonconvex terms:

$$f(\boldsymbol{w}) = f_{\text{cvx}}(\boldsymbol{w}) + f_{\text{ncvx}}(\boldsymbol{w}),$$

where

$$\begin{aligned} f_{\text{cvx}}(\boldsymbol{w}) &= -\lambda_1 \phi_1(\boldsymbol{w}) + \lambda_2 \phi_2(\boldsymbol{w}), \\ f_{\text{ncvx}}(\boldsymbol{w}) &= -\lambda_3 \phi_3(\boldsymbol{w}) + \lambda_4 \phi_4(\boldsymbol{w}). \end{aligned}$$

One can, of course, always use some off-the-shelf general nonconvex solver (typically referred to as a nonlinear solver) capable of dealing with the nonconvex MVSK portfolio formulation (9.11). In the following, however, we will develop ad hoc methods that are much more efficient and do not require the use of a general nonlinear solver. In particular, we will explore the *successive convex approximation* (SCA) framework as well as the majorization–minimization (MM) framework for the derivation of the numerical methods (see Appendix B for details on numerical methods).

9.4.1 Algorithms via the SCA Framework

Preliminaries on SCA

The successive convex approximation method (or framework) approximates a difficult optimization problem by a sequence of simpler convex approximated problems. For details,

the reader is referred to Section B.8 in Appendix B, the original paper in Scutari et al. (2014), and the comprehensive book chapter in Scutari and Sun (2018). We now give a concise description.

Suppose the following (difficult) problem is to be solved:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathcal{X}, \end{array}$$

where $f(\cdot)$ is the (possibly nonconvex) objective function and $\mathcal{X}$ is a convex set. Instead of attempting to directly obtain a solution $\boldsymbol{x}^\star$ (either a local or global solution), the SCA method will produce a sequence of iterates $\boldsymbol{x}^0, \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots$ that will converge to $\boldsymbol{x}^\star$.

More specifically, at iteration k, the SCA approximates the objective function $f(\boldsymbol{x})$ by a surrogate function around the current point $\boldsymbol{x}^k$, denoted by $\tilde{f}\left(\boldsymbol{x}; \boldsymbol{x}^k\right)$. One may be tempted to solve the sequence of (simpler) problems

$$\boldsymbol{x}^{k+1} = \underset{\boldsymbol{x} \in \mathcal{X}}{\arg\min}\ \tilde{f}\left(\boldsymbol{x}; \boldsymbol{x}^k\right), \qquad k = 0, 1, 2, \ldots$$

Unfortunately, the previous sequence of updates may not converge and a smoothing step is necessary to introduce some memory in the process, which will avoid undesired oscillations. Thus, the correct sequence of problems in the SCA method is

$$\left.\begin{array}{l} \hat{\boldsymbol{x}}^{k+1} = \underset{\boldsymbol{x} \in \mathcal{X}}{\arg\min}\ \tilde{f}\left(\boldsymbol{x}; \boldsymbol{x}^k\right) \\ \boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \gamma^k \left(\hat{\boldsymbol{x}}^{k+1} - \boldsymbol{x}^k\right) \end{array}\right\} \qquad k = 0, 1, 2, \ldots,$$

where $\{\gamma^k\}$ is a properly designed sequence with $\gamma^k \in (0, 1]$ (Scutari et al., 2014).

In order to guarantee converge of the iterates, the surrogate function $\tilde{f}\left(\boldsymbol{x}; \boldsymbol{x}^k\right)$ has to satisfy the following technical conditions (Scutari et al., 2014):

- $\tilde{f}\left(\boldsymbol{x}; \boldsymbol{x}^k\right)$ must be strongly convex on the feasible set $\mathcal{X}$; and
- $\tilde{f}\left(\boldsymbol{x}; \boldsymbol{x}^k\right)$ must be differentiable with $\nabla \tilde{f}\left(\boldsymbol{x}^k; \boldsymbol{x}^k\right) = \nabla f\left(\boldsymbol{x}^k\right)$, that is, the gradients of $\tilde{f}$ and f must coincide at $\boldsymbol{x}^k$ (not at every $\boldsymbol{x}$).

SCA Applied to MVSK Portfolio Design

Following the SCA framework, we will leave $f_{\text{cvx}}(\boldsymbol{w})$ untouched, since it is already (strongly) convex, and we will construct a quadratic convex approximation (another option would be a linear approximation) of the nonconvex term $f_{\text{ncvx}}(\boldsymbol{w})$ around the point $\boldsymbol{w} = \boldsymbol{w}^k$ as

$$\begin{aligned} \tilde{f}_{\text{ncvx}}\left(\boldsymbol{w}; \boldsymbol{w}^k\right) = f_{\text{ncvx}}\left(\boldsymbol{w}^k\right) + \nabla f_{\text{ncvx}}\left(\boldsymbol{w}^k\right)^{\mathsf{T}} \left(\boldsymbol{w} - \boldsymbol{w}^k\right) \\ + \frac{1}{2}\left(\boldsymbol{w} - \boldsymbol{w}^k\right)^{\mathsf{T}} \left[\nabla^2 f_{\text{ncvx}}\left(\boldsymbol{w}^k\right)\right]_{\text{PSD}} \left(\boldsymbol{w} - \boldsymbol{w}^k\right), \end{aligned}$$

where $[\boldsymbol{\Xi}]_{\text{PSD}}$ denotes the projection of the matrix $\boldsymbol{\Xi}$ onto the set of positive semidefinite matrices. In practice, this projection can be obtained by first computing the eigenvalue decomposition of the matrix, $\boldsymbol{\Xi} = \boldsymbol{U}\text{Diag}\left(\boldsymbol{\lambda}\right)\boldsymbol{U}^{\mathsf{T}}$, and then projecting the eigenvalues onto the nonnegative orthant $[\boldsymbol{\Xi}]_{\text{PSD}} = \boldsymbol{U}\text{Diag}\left(\boldsymbol{\lambda}^+\right)\boldsymbol{U}^{\mathsf{T}}$, where $(\cdot)^+ = \max(0, \cdot)$. [As a technical

detail, if $\lambda_2 = 0$, then the matrix $[\boldsymbol{\Xi}]_{\text{PSD}}$ should be made positive definite so that the overall approximation is strongly convex (Zhou & Palomar, 2021).]

Thus, a quadratic convex approximation of $f(\boldsymbol{w})$ in (9.15) is obtained as

$$\begin{aligned}\tilde{f}\left(\boldsymbol{w};\boldsymbol{w}^k\right) &= f_{\text{cvx}}(\boldsymbol{w}) + \tilde{f}_{\text{ncvx}}\left(\boldsymbol{w};\boldsymbol{w}^k\right)\\ &= \frac{1}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{Q}^k\boldsymbol{w} + \boldsymbol{w}^\mathsf{T}\boldsymbol{q}^k + \text{constant},\end{aligned}$$

where

$$\begin{aligned}\boldsymbol{Q}^k &= \lambda_2\nabla^2\phi_2\left(\boldsymbol{w}^k\right) + \left[\nabla^2 f_{\text{ncvx}}\left(\boldsymbol{w}^k\right)\right]_{\text{PSD}},\\ \boldsymbol{q}^k &= -\lambda_1\nabla\phi_1\left(\boldsymbol{w}^k\right) + \nabla f_{\text{ncvx}}\left(\boldsymbol{w}^k\right) - \left[\nabla^2 f_{\text{ncvx}}\left(\boldsymbol{w}^k\right)\right]_{\text{PSD}}\boldsymbol{w}^k,\end{aligned}$$

and the gradients and Hessians of the moments can be obtained from either the nonparametric expressions in (9.2)–(9.3) or the parametric ones in (9.7)–(9.8).

Finally, the SCA-based algorithm can be implemented by successively solving, for $k = 0, 1, 2, \ldots$, the convex quadratic problems

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \frac{1}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{Q}^k\boldsymbol{w} + \boldsymbol{w}^\mathsf{T}\boldsymbol{q}^k\\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}.\end{array} \tag{9.16}$$

This SCA-based quadratic approximated MVSK (SCA-Q-MVSK) method is summarized in Algorithm 9.1. More details and similar algorithms for other formulations, such as MVSK portfolio tilting, can be found in Zhou and Palomar (2021).

Algorithm 9.1: SCA-Q-MVSK method to solve the MVSK portfolio in (9.11).

Choose initial point $\boldsymbol{w}^0 \in \mathcal{W}$ and sequence $\{\gamma^k\}$;
Set $k \leftarrow 0$;
repeat
  Calculate $\nabla f_{\text{ncvx}}\left(\boldsymbol{w}^k\right)$ and $\left[\nabla^2 f_{\text{ncvx}}\left(\boldsymbol{w}^k\right)\right]_{\text{PSD}}$;
  Solve the problem (9.16) and keep solution as $\hat{\boldsymbol{w}}^{k+1}$;
  $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k + \gamma^k(\hat{\boldsymbol{w}}^{k+1} - \boldsymbol{w}^k)$;
  $k \leftarrow k + 1$;
until convergence;

9.4.2 Algorithms via the MM Framework

Preliminaries on MM

The majorization–minimization method (or framework), similarly to SCA, approximates a difficult optimization problem by a sequence of simpler approximated problems. We now give a concise description. For details, the reader is referred to Section B.7 in Appendix B, as well as the concise tutorial in Hunter and Lange (2004), the long tutorial with applications in Sun et al. (2017), and the convergence analysis in Razaviyayn et al. (2013).

Suppose the following (difficult) problem is to be solved:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{X}, \end{aligned}$$

where $f(\cdot)$ is the (possibly nonconvex) objective function and $\mathcal{X}$ is a (possibly nonconvex) set. Instead of attempting to directly obtain a solution $\boldsymbol{x}^\star$ (either a local or global solution), the MM method will produce a sequence of iterates $\boldsymbol{x}^0, \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots$ that will converge to $\boldsymbol{x}^\star$.

More specifically, at iteration k, the MM approximates the objective function $f(\boldsymbol{x})$ by a surrogate function around the current point $\boldsymbol{x}^k$ (essentially, a tangent upper bound), denoted by $u\left(\boldsymbol{x}; \boldsymbol{x}^k\right)$, leading to the sequence of (simpler) problems

$$\boldsymbol{x}^{k+1} = \arg\min_{\boldsymbol{x} \in \mathcal{X}} u\left(\boldsymbol{x}; \boldsymbol{x}^k\right), \qquad k = 0, 1, 2, \ldots$$

In order to guarantee convergence of the iterates, the surrogate function $u\left(\boldsymbol{x}; \boldsymbol{x}^k\right)$ has to satisfy the following technical conditions (Razaviyayn et al., 2013; Sun et al., 2017):

- *upper bound property*: $u\left(\boldsymbol{x}; \boldsymbol{x}^k\right) \geq f\left(\boldsymbol{x}\right)$;
- *touching property*: $u\left(\boldsymbol{x}^k; \boldsymbol{x}^k\right) = f\left(\boldsymbol{x}^k\right)$; and
- *tangent property*: $u\left(\boldsymbol{x}; \boldsymbol{x}^k\right)$ must be differentiable with $\nabla u\left(\boldsymbol{x}^k; \boldsymbol{x}^k\right) = \nabla f\left(\boldsymbol{x}^k\right)$, that is, the gradients of u and f must coincide at $\boldsymbol{x}^k$ (not at every $\boldsymbol{x}$).

The surrogate function $u\left(\boldsymbol{x}; \boldsymbol{x}^k\right)$ is also referred to as the *majorizer* because it is an upper bound of the original function. The fact that, at each iteration, first the majorizer is constructed and then it is minimized gives the name *majorization–minimization* to the method.

MM Applied to MVSK Portfolio Design

Following the MM framework, we will leave $f_{\text{cvx}}(\boldsymbol{w})$ untouched, since it is already convex, and we will construct a majorizer (i.e., a tangent upper-bound surrogate function) of the nonconvex term $f_{\text{ncvx}}(\boldsymbol{w})$ around the point $\boldsymbol{w} = \boldsymbol{w}^k$. For example, a linear function plus a quadratic regularizer,

$$\tilde{f}_{\text{ncvx}}\left(\boldsymbol{w}; \boldsymbol{w}^k\right) = f_{\text{ncvx}}\left(\boldsymbol{w}^k\right) + \nabla f_{\text{ncvx}}\left(\boldsymbol{w}^k\right)^{\mathsf{T}} \left(\boldsymbol{w} - \boldsymbol{w}^k\right) + \frac{\tau_{\text{MM}}}{2} \|\boldsymbol{w} - \boldsymbol{w}^k\|_2^2,$$

where τ_{MM} is a positive constant properly chosen so that $\tilde{f}_{\text{ncvx}}\left(\boldsymbol{w}; \boldsymbol{w}^k\right)$ upper-bounds $f_{\text{ncvx}}(\boldsymbol{w})$ (Zhou & Palomar, 2021).

Thus, a quadratic convex approximation of $f(\boldsymbol{w})$ (albeit using only first-order or linear information about the nonconvex term f_{ncvx}) is obtained as

$$\begin{aligned} \tilde{f}\left(\boldsymbol{w}; \boldsymbol{w}^k\right) &= f_{\text{cvx}}(\boldsymbol{w}) + \tilde{f}_{\text{ncvx}}\left(\boldsymbol{w}; \boldsymbol{w}^k\right) \\ &= -\lambda_1 \phi_1(\boldsymbol{w}) + \lambda_2 \phi_2(\boldsymbol{w}) + \nabla f_{\text{ncvx}}\left(\boldsymbol{w}^k\right)^{\mathsf{T}} \boldsymbol{w} + \frac{\tau_{\text{MM}}}{2} \|\boldsymbol{w} - \boldsymbol{w}^k\|_2^2 + \text{constant}, \end{aligned}$$

where the gradient of the nonconvex function f_{ncvx} can be obtained from the gradients of $\phi_3(\boldsymbol{w})$ and $\phi_4(\boldsymbol{w})$, either the nonparametric expressions in (9.2) or the parametric ones in (9.7). It is worth pointing out that, owing to the linear approximation of the nonconvex term $f_{\text{ncvx}}(\boldsymbol{w})$, the Hessians are not even required, which translates into huge savings in computation time and storage memory.

Finally, the MM-based algorithm can be implemented by successively solving, for $k = 0, 1, 2, \ldots$, the convex problems

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & -\lambda_1\phi_1(\boldsymbol{w}) + \lambda_2\phi_2(\boldsymbol{w}) + \nabla f_{\text{ncvx}}\left(\boldsymbol{w}^k\right)^\mathsf{T}\boldsymbol{w} + \frac{\tau_{\text{MM}}}{2}\|\boldsymbol{w} - \boldsymbol{w}^k\|_2^2 \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}. \end{array} \tag{9.17}$$

We denote the solution to each majorized problem in (9.17) as $\text{MM}(\boldsymbol{w}^k)$, so that $\boldsymbol{w}^{k+1} = \text{MM}(\boldsymbol{w}^k)$.

Unfortunately, due to the upper-bound requirement of the nonconvex function $\tilde{f}_{\text{ncvx}}\left(\boldsymbol{w}; \boldsymbol{w}^k\right)$, the constant τ_{MM} ends up being too large in practice. This means that the approximation is not tight enough and the method requires many iterations to converge. For that reason, it is necessary to resort to some acceleration techniques.

We consider a quasi-Newton acceleration technique called SQUAREM (Varadhan & Roland, 2008) that works well in practice. Instead of taking the solution to the majorized problem in (9.17), $\text{MM}(\boldsymbol{w}^k)$, as the next point $\boldsymbol{w}^{k+1}$, the acceleration technique takes two steps and combines them in a sophisticated way, with a final third step to guarantee feasibility. The details are as follows:

$$\begin{aligned} \text{difference first update:} \quad & \boldsymbol{r}^k = R(\boldsymbol{w}^k) = \text{MM}(\boldsymbol{w}^k) - \boldsymbol{w}^k \\ \text{difference of differences:} \quad & \boldsymbol{v}^k = R(\text{MM}(\boldsymbol{w}^k)) - R(\boldsymbol{w}^k) \\ \text{stepsize:} \quad & \alpha^k = -\max\left(1, \|\boldsymbol{r}^k\|_2/\|\boldsymbol{v}^k\|_2\right) \\ \text{actual step taken:} \quad & \boldsymbol{y}^k = \boldsymbol{w}^k - \alpha^k\boldsymbol{r}^k \\ \text{final update on actual step:} \quad & \boldsymbol{w}^{k+1} = \text{MM}(\boldsymbol{y}^k). \end{aligned} \tag{9.18}$$

The choice of the stepsize α^k can be further refined to get a more robust algorithm with a much faster convergence (X. Wang et al., 2023). The last step, $\boldsymbol{w}^{k+1} = \text{MM}(\boldsymbol{y}^k)$, can also be simplified to avoid having to solve the majorized problem a third time.

This quasi-Newton accelerated MM linear (as in using linear information or gradient only of the nonconvex term) MVSK (Acc-MM-L-MVSK) method is summarized in Algorithm 9.2. More details and similar algorithms for other formulations, such as MVSK portfolio tilting, can be found in X. Wang et al. (2023).[4]

Algorithm 9.2: Acc-MM-L-MVSK method to solve the MVSK portfolio in (9.11).

1: Choose initial point $\boldsymbol{w}^0 \in \mathcal{W}$ and proper constant τ_{MM} for majorized problem (9.17);
2: Set $k \leftarrow 0$;
3: **repeat**
4: Calculate $\nabla f_{\text{ncvx}}\left(\boldsymbol{w}^k\right)$;
5: Compute the quantities $\boldsymbol{r}^k$, $\boldsymbol{v}^k$, α^k, $\boldsymbol{y}^k$, and current solution $\boldsymbol{w}^{k+1}$ as in (9.18), which requires solving the majorized problem (9.17) three times;
6: $k \leftarrow k + 1$;
7: **until** convergence;

[4] The R package `highOrderPortfolios` (Zhou et al., 2022) implements Algorithms 9.1 and 9.2 based on Zhou and Palomar (2021) and X. Wang et al. (2023).

9.4.3 Numerical Experiments

We now perform an empirical study and comparison of the computational cost of the two methods SCA-Q-MVSK and Acc-MM-L-MVSK described in Algorithms 9.1 and 9.2, respectively. In addition, for the computation of the gradients and Hessians of the moments, we consider both the nonparametric expressions in (9.2)–(9.3) and the parametric ones in (9.7)–(9.8).

In fact, the Acc-MM-L-MVSK method considered in the numerical results is actually an improvement over Algorithm 9.2 that does not require the computation of the constant τ_{MM} in (9.17) (which already takes on the order of 10 seconds for $N = 100$). For details refer to X. Wang et al. (2023).

Convergence and Computation Time

Figure 9.4 shows the convergence of different MVSK portfolio optimization methods for a universe size of $N = 100$. In this case, both the nonparametric and parametric expressions for the moments can be employed. As a reference, the benchmark computation of the solution via an off-the-shelf nonlinear solver requires around 5 seconds with nonparametric moments and around 0.5 seconds with parametric moments.

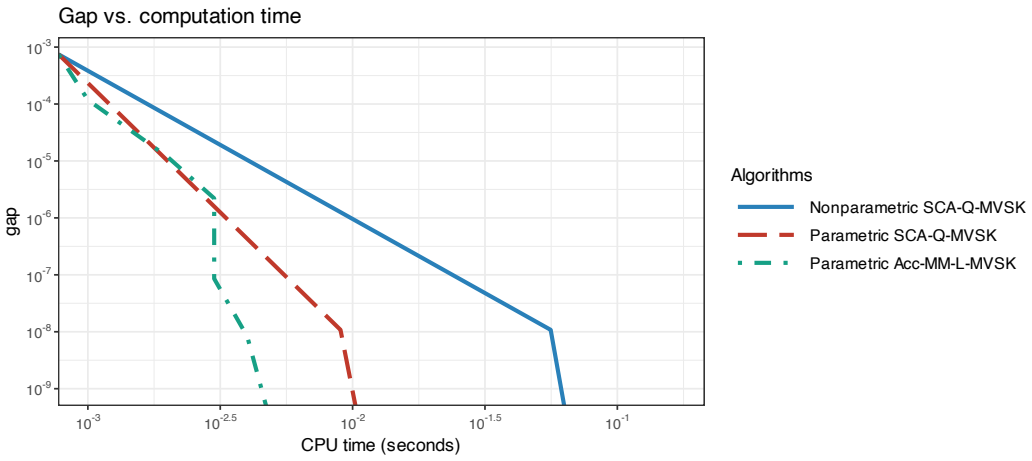


Figure 9.4 Convergence of different MVSK portfolio optimization algorithms for $N = 100$.

Figure 9.5 similarly shows the convergence of different MVSK portfolio optimization methods for a universe size of $N = 400$. In this case, however, due to the larger universe size, only the parametric computation of the moments is feasible. As a reference, the benchmark computation of the solution via an off-the-shelf nonlinear solver requires around 1 minute.

Figure 9.6 shows a boxplot of the computation time vs. the universe dimension N for different MVSK portfolio optimization methods (in the parametric case).

More extensive numerical comparisons can be found in Zhou and Palomar (2021) and X. Wang et al. (2023).

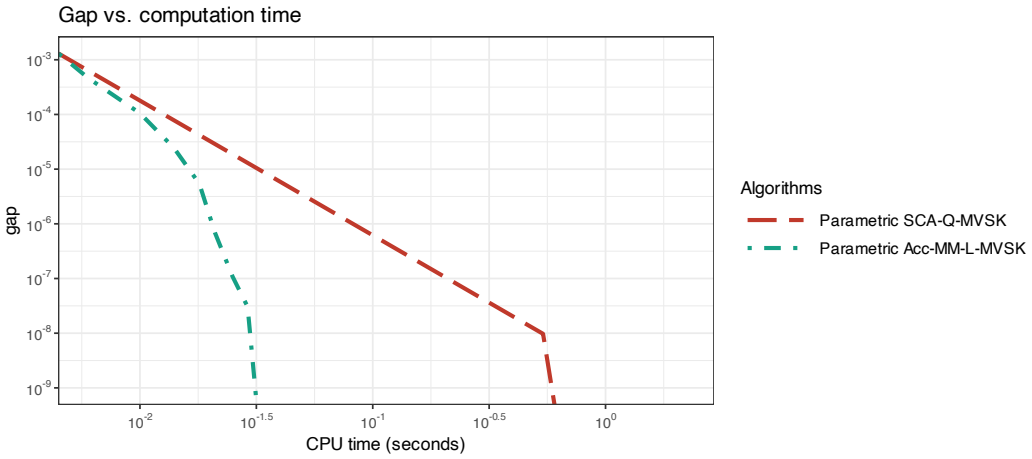


Figure 9.5 Convergence of different MVSK portfolio optimization algorithms for $N = 400$.

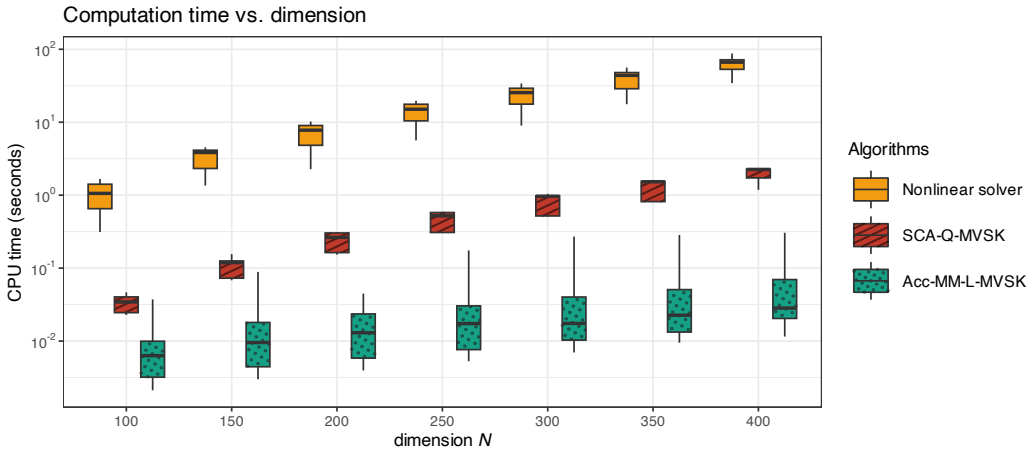


Figure 9.6 Computation time of different (parametric) MVSK portfolio optimization algorithms.

Summarizing, it seems clear that parametric computation of the moments is the most appropriate (due to the otherwise high computational cost). In addition, the Acc-MM-L-MVSK method described in Algorithm 9.2 – to be exact, the more sophisticated version proposed in X. Wang et al. (2023)) – provides the fastest convergence by orders of magnitude.

Portfolio Backtest

We compare the performance of some portfolios based on different moments, namely, the global maximum return portfolio (GMRP; see Section 6.4.2 in Chapter 6), the global minimum variance portfolio (GMVP; see Section 6.5.1 in Chapter 6), and the MVSK portfolio in (9.11) (setting $\lambda_1 = 0$ to resemble the GMVP).

Figure 9.7 shows the cumulative P&L and drawdown during 2016–2019 over a random

universe of 20 stocks from the S&P 500, whereas Table 9.1 gives the numerical values over the whole period. A small improvement of the MVSK portfolio over the GMVP can be observed.

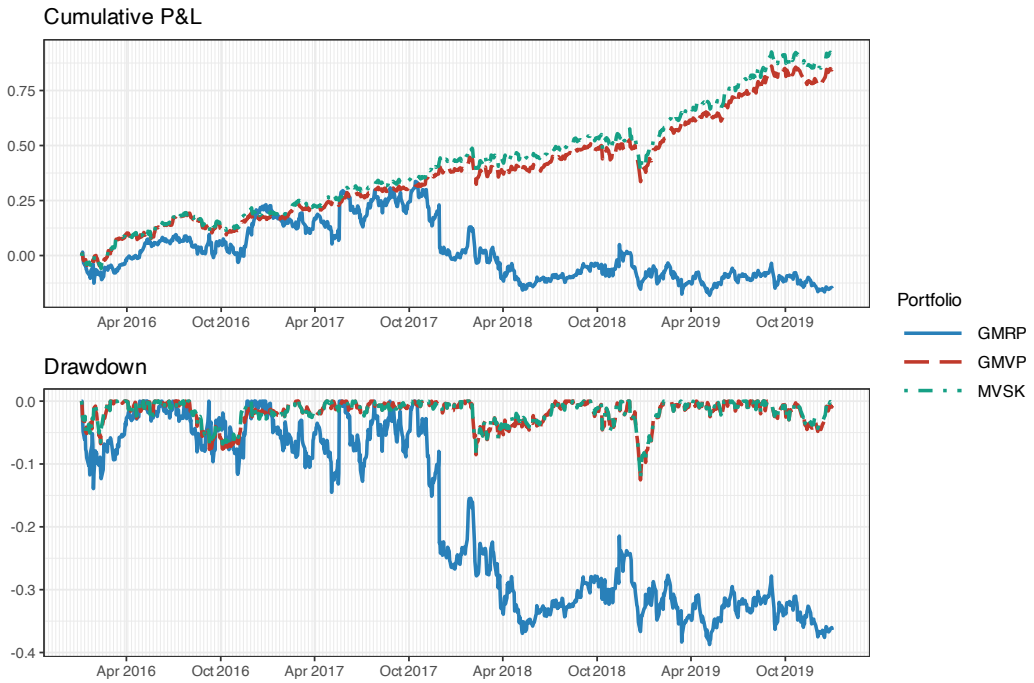


Figure 9.7 Backtest of high-order portfolios: cumulative P&L and drawdown.

Table 9.1 Backtest of high-order portfolios: performance measures.

Portfolio	Sharpe ratio	Annual return	Annual volatility	Sortino ratio	Max drawdown	CVaR (0.95)
GMRP	−0.01	0%	27%	−0.02	39%	4%
GMVP	1.47	16%	11%	2.12	13%	2%
MVSK	1.56	17%	11%	2.26	12%	2%

Multiple Portfolio Backtests

We now consider multiple randomized backtests (see Section 8.4 in Chapter 8) for a better characterization of the performance of the portfolios. We take a dataset of $N = 20$ stocks over the period 2015–2020 and generate 100 resamples each with $N = 8$ randomly selected stocks and a random period of two years. Then we perform a walk-forward backtest with a lookback window of 1 year, reoptimizing the portfolio every month.

Figure 9.8 shows the boxplots of the Sharpe ratio and maximum drawdown, and Table 9.2 shows the backtest results in table form with different performance measures over the whole period. Again, one can observe a modest improvement of the MVSK portfolio over the GMVP.

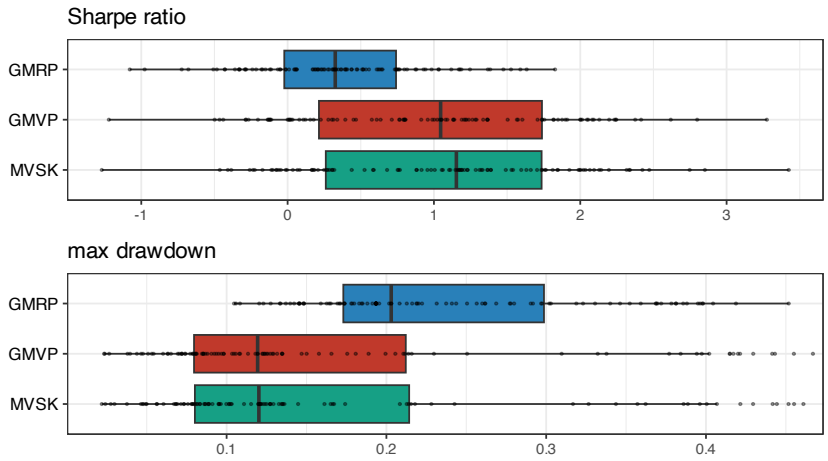


Figure 9.8 Multiple randomized backtest of high-order portfolios: Sharpe ratio and maximum drawdown.

Table 9.2 Multiple randomized backtest of high-order portfolios: performance measures.

Portfolio	Sharpe ratio	Annual return	Annual volatility	Sortino ratio	Max drawdown	CVaR (0.95)
GMRP	0.33	9%	26%	0.45	20%	4%
GMVP	1.05	13%	13%	1.48	12%	2%
MVSK	1.15	14%	13%	1.58	12%	2%

9.5 Summary

- Markowitz's portfolio is formulated in terms of the mean and variance of the returns (first and second moments), but financial data is not Gaussian distributed and higher orders may be necessary for a proper characterization.

- High-order portfolios attempt to capture the non-Gaussianity by incorporating the skewness and kurtosis (third and fourth moments) in the formulation to better model the asymmetry and heavy tails of the distribution.

- High-order portfolios go back to the 1960s. However, the estimation and manipulation of such high-order moments was an impossibility in those early days. For a universe of N assets, the number of parameters increases at a rate of N^4, which rapidly becomes unmanageable in terms of computational complexity and memory storage. In addition, the portfolio formulations are nonconvex, adding to the difficulty of designing optimal portfolios.

- There is a wide variety of portfolio formulations incorporating high orders, such as MVSK

portfolios, portfolio tilting, polynomial-goal formulations, and even using alternative linear moments (L-moments).

- Efficient algorithms are now readily available based on mature iterative algorithmic frameworks.
- Consequently, after over half a century of research by the scientific community, all the challenges have been overcome, and designing high-order portfolios can now be easily achieved with hundreds or even thousands of assets. The decision to incorporate high-order moments now rests in the hands of the trader.

Exercises

9.1 (Non-Gaussian return distribution)

a. Download market data for one asset.
b. Plot the histograms for different frequencies of returns.
c. Try to fit a Gaussian distribution.
d. Assess the asymmetry as well as the thickness of the tails for these histograms (use Q–Q plots, compute skewness and kurtosis, etc.).

9.2 (Computation of portfolio sample moments)

a. Download market data corresponding to N assets during a period with T observations, $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T \in \mathbb{R}^N$.
b. Estimate the mean vector, covariance matrix, co-skewness matrix, and co-kurtosis matrix of the data via sample means.
c. Design some portfolio, such as the $1/N$ portfolio, and compute the four moments of the portfolio returns (i.e., mean, variance, skewness, and kurtosis).
d. Additionally, compute the gradient and Hessian of the four portfolio moments.
e. Repeat the whole process for different values of N, while keeping track of the computational cost, and make a final plot of complexity vs. N.

9.3 (Comparison of nonparametric, structured, and parametric moments)

a. Download market data corresponding to N assets during a period with T observations, $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T \in \mathbb{R}^N$.
b. Design some portfolio, such as the $1/N$ portfolio.
c. Estimate the mean, variance, skewness, and kurtosis of the portfolio returns in the following ways:
   - nonparametric moments: via a direct sample mean estimation of the mean vector, covariance matrix, co-skewness matrix, and co-kurtosis matrix;
   - structured moments: via fitting a single market-factor model to the returns;
   - parametric moments: via fitting a multivariate skew t distribution to the returns.
d. Repeat the whole process for different values of N, while keeping track of the computational cost, and make a final plot of complexity vs. N.

9.4 (Sanity check of parametric moment expressions)

a. Generate synthetic data according to a multivariate skew t distribution.
b. Design some portfolio, such as the $1/N$ portfolio.
c. Estimate the mean, variance, skewness, and kurtosis of the portfolio returns in the following ways:
   - nonparametric moments: first estimate via sample means the mean vector, covariance matrix, co-skewness matrix, and co-kurtosis matrix of the data, then evaluate the portfolio moments (as well as gradients and Hessians);
   - parametric moments: first fit a multivariate skew t distribution to these synthetic returns, then evaluate the moments with the parametric expressions (as well as gradients and Hessians).
d. Compare the nonparametric and parametric estimations.
e. Repeat the whole process for different numbers of data samples T, and make a final plot of estimators vs. T.

9.5 (L-moments)

a. Download market data for one asset.
b. Compute the first four moments (i.e., mean, variance, skewness, and kurtosis) in a rolling-window fashion and plot them over time.
c. Compute the first four L-moments (i.e., L-location, L-scale, L-skewness, and L-kurtosis) in a rolling-window fashion and plot them over time.
d. Try different values for the lookback window and compare the regular moments with the L-moments.

9.6 (MVSK portfolios)

a. Download market data corresponding to N assets during a period with T observations, $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T \in \mathbb{R}^N$.
b. Fit a multivariate skew t distribution to the data.
c. Design a traditional mean–variance portfolio.
d. Design a high-order MVSK portfolio.
e. Compare their performance. Try to obtain a clear performance improvement via the introduction of higher orders.

9.7 (Portfolio tilting)

a. Download market data corresponding to N assets during a period with T observations, $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T \in \mathbb{R}^N$.
b. Fit a multivariate skew t distribution to the data.
c. Design some portfolio as a reference.
d. Use the portfolio tilting formulation to improve the reference portfolio.
e. Compare their performance. Try to obtain a clear performance improvement via tilting.

References

Birgeand, J., & Chavez-Bedoya, L. (2016). Portfolio optimization under generalized hyperbolic skewed t distribution and exponential utility. *Quantitative Finance*, *16*(7), 1019–1036.

Boudt, K., Cornilly, D., Holle, F. V., & Willems, J. (2020). Algorithmic portfolio tilting to harvest higher moment gains. *Heliyon*, *6*(3), e03516.

Boudt, K., Cornilly, D., & Verdonck, T. (2020). A coskewness shrinkage approach for estimating the skewness of linear combinations of random variables. *Journal of Financial Econometrics*, *18*(1), 1–23.

Boudt, K., Lu, W., & Peeters, B. (2014). Higher order comoments of multifactor models and asset allocation. *Finance Research Letters*, *13*, 225–233.

Brandt, M. W., Santa-Clara, P., & Valkanov, R. (2009). Parametric portfolio policies: Exploiting characteristics in the cross section of equity returns. *Review of Financial Studies*, *22*(9), 3411–3447.

Briec, W., Kerstens, K., & Jokung, O. (2007). Mean–variance–skewness portfolio performance gauging: A general shortage function and dual approach. *Management Science*, *53*(1), 135–149.

Chassein, A., & Goerigk, M. (2015). Alternative formulations for the ordered weighted averaging objective. *Information Processing Letters*, *115*, 604–608.

Dinh, T. P., & Niu, Y.-S. (2011). An efficient DC programming approach for portfolio decision with higher moments. *Computational Optimization and Applications*, *50*(3), 525–554.

Hosking, J. R. M. (1990). L-moments: Analysis and estimation of distributions using linear combinations of order statistics. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, *52*(1), 105–124.

Hunter, D. R., & Lange, K. (2004). A tutorial on MM algorithms. *The American Statistician*, *58*, 30–37.

Jean, W. H. (1971). The extension of portfolio analysis to three or more parameters. *Journal of Financial and Quantitative Analysis*, *6*(1), 505–515.

Jurczenko, E., Maillet, B., & Merlin, P. (2006). Hedge fund portfolio selection with higher-order moments: A nonparametric mean–variance–skewness–kurtosis efficient frontier. In E. Jurczenko & B. Maillet (Eds.), *Multi-moment Asset Allocation and Pricing Models* (pp. 51–66). John Wiley & Sons.

Lai, T.-Y. (1991). Portfolio selection with skewness: A multiple- objective approach. *Review of Quantitative Finance and Accounting*, *1*(3), 293–305.

Markowitz, H. M. (1952). Portfolio selection. *The Journal of Finance*, *7*(1), 77–91.

Martellini, L., & Ziemann, V. (2010). Improved estimates of higher-order comoments and implications for portfolio selection. *The Review of Financial Studies*, *23*(4), 1467–1502.

McNeil, A. J., Frey, R., & Embrechts, P. (2015). *Quantitative Risk Management*. Princeton University Press.

Niu, Y.-S., & Wang, Y.-J. (2019). Higher-order moment portfolio optimization via the difference-of-convex programming and sums-of-squares. *Preprint. Available at arXiv*. https://arxiv.org/abs/1906.01509v2

Ogryczak, W. (2000). Multiple criteria linear programming model for portfolio selection. *Annals of Operations Research*, *97*, 143–162.

Ogryczak, W., & Sliwinski, T. (2003). On solving linear programs with the ordered weighted averaging objective. *European Journal of Operational Research*, *148*, 80–91.

Razaviyayn, M., Hong, M., & Luo, Z. (2013). A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, *23*(2), 1126–1153.

Scott, R., & Horvath, P. (1980). On the direction of preference for moments of higher order than the variance. *The Journal of Finance*, *35*(4), 915–919.

Scutari, G., Facchinei, F., Song, P., Palomar, D. P., & Pang, J.-S. (2014). Decomposition by partial linearization: Parallel optimization of multi-agent systems. *IEEE Transactions on Signal Processing*, *62*(3), 641–656.

Scutari, G., & Sun, Y. (2018). Parallel and distributed successive convex approximation methods for big-data optimization. In F. Facchinei & J. S. Pang (Eds.), *Multi-agent Optimization* (pp. 141–308). Springer.

Sun, Y., Babu, P., & Palomar, D. P. (2017). Majorization–minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, *65*(3), 794–816.

Varadhan, R., & Roland, C. (2008). Simple and globally convergent methods for accelerating the convergence of any EM algorithm. *Scandinavian Journal of Statistics*, *35*(2), 335–353.

Wang, Q. J. (1996). Direct sample estimators of L moments. *Water Resources Research*, *32*(12), 3617–3619.

Wang, X., Zhou, R., Ying, J., & Palomar, D. P. (2023). Efficient and scalable parametric high-order portfolios design via the skew-*t* distribution. *IEEE Transactions on Signal Processing*, *71*, 3726–3740.

Yager, R. R. (1996). Constrained OWA aggregation. *Fuzzy Sets and Systems*, *81*(1), 89–101.

Young, W. E., & Trent, R. H. (1969). Geometric mean approximations of individual security and portfolio performance. *Journal of Financial and Quantitative Analysis*, *4*(2), 179–199.

Zakamouline, V., & Koekebakker, S. (2009). Portfolio performance evaluation with generalized Sharpe ratios: Beyond the mean and variance. *Journal of Banking and Finance*, *33*(7), 1242–1254.

Zhou, R., & Palomar, D. P. (2021). Solving high-order portfolios via successive convex approximation algorithms. *IEEE Transactions on Signal Processing*, *69*, 892–904.

Zhou, R., Wang, X., & Palomar, D. P. (2022). *highOrderPortfolios: Design of High-Order Portfolios via Mean, Variance, Skewness, and Kurtosis* [R package]. https://CRAN.R-project.org/package=highOrderPortfolios

10

Portfolios with Alternative Risk Measures

"I try all things, I achieve what I can."

— Herman Melville, *Moby Dick*

Markowitz's mean–variance portfolio optimizes a trade-off between expected return and risk measured by the variance. The higher the variance, the more uncertainty, which is undesired, and vice versa. In principle, this makes sense and follows our intuitive expectation of a measure of risk.

However, as already indicated by Markowitz, the variance and volatility are very simplistic measures of risk. To start with, they penalize both the unwanted losses and the desired gains. In addition, the shape of the distribution function of the returns is being ignored. Rather than focusing on the width of the middle part of the distribution (as the volatility does), it is the tail of the distribution that characterizes the big losses.

This chapter explores a variety of alternative and more sophisticated measures proposed over the past seven decades (such as downside risk, semi-variance, value-at-risk, conditional value-at-risk, expected shortfall, and drawdown) and, more importantly, how to incorporate such measures in the portfolio formulation in a manageable way.

10.1 Introduction

Markowitz's mean–variance portfolio (Markowitz, 1952) formulates the portfolio design as a trade-off between the expected return $\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}$ and the risk measured by the variance $\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}$ (see Chapter 7 for details):

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \frac{\lambda}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w} \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array}$$

where λ is a hyper-parameter that controls the investor's risk aversion and $\mathcal{W}$ denotes an arbitrary constraint set, such as $\mathcal{W} = \{\boldsymbol{w} \mid \mathbf{1}^\mathsf{T}\boldsymbol{w} = 1, \boldsymbol{w} \geq \mathbf{0}\}$.

Nevertheless, it has been well recognized over decades of research and experimentation that

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

measuring the portfolio risk with the variance $\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}$ or, similarly, the volatility $\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}$ may not be the best choice for out-of-sample performance. Markowitz himself recognized and stressed the limitations of the mean–variance analysis (Markowitz, 1959). As a consequence, academics and practitioners have explored alternative risk measures that satisfy desirable properties, notably the family of coherent risk measures (Artzner et al., 1999).

This chapter explores a variety of risk measures alternative to the variance, namely, the downside risk, semi-variance, semi-deviation, value-at-risk (VaR), conditional value-at-risk (CVaR), expected shortfall (ES), and drawdown. Particular emphasis is placed on how to incorporate such alternative risk measures in the portfolio formulation itself.

10.2 Alternative Risk Measures

The return obtained by portfolio $\boldsymbol{w}$ is $R = \boldsymbol{w}^\top \boldsymbol{r}$, where $\boldsymbol{r}$ denotes the vector of random returns of the N assets (see Chapter 6 for details). Since the portfolio return R is a random variable, a proper full characterization is provided by the probability distribution function (pdf). For simplicity and convenience, the information contained in the pdf is typically condensed into a few key numbers, such as the mean (expected return) and the standard deviation (as a measure of risk). However, determining the appropriate quantity that should be used as a measure of risk has been a subject of scientific investigation since the 1950s (McNeil et al., 2015).

As a consequence, academics and practitioners have explored over decades alternative risk measures that satisfy desirable properties, notably the family of coherent risk measures that satisfy four basic properties: translation invariance, monotonicity, subadditivity, and positive homogeneity (Artzner et al., 1999).

Figure 10.1 illustrates the meaning of the most popular measures of risk in the context of the pdf of the portfolio return, namely, the variance/volatility, the semi-variance/semi-deviation, the VaR, and the CVaR.

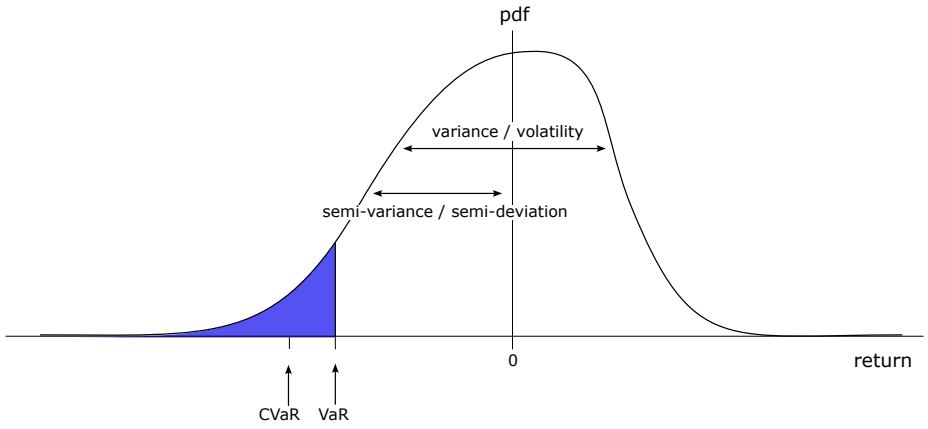


Figure 10.1 Illustration of return distribution and measures of risk.

In the following, we describe in detail variations of downside risk, VaR, CVaR, and drawdown.

As will be discussed later, drawdown is fundamentally different from all the other measures in that it is not invariant to the order of the returns.

10.2.1 Downside Risk

Economists have long recognized that investors care differently about downside losses vs. upside gains (Roy, 1952). Markowitz himself advocated using the semi-variance as a measure of risk, rather than the variance (Markowitz, 1959).

Downside risk generally refers to risk measures that quantify the losses below a certain threshold. A number of studies indicate that downside risk measures are more meaningful than symmetric measures such as the variance or volatility (Ang et al., 2006; Estrada, 2006). Nevertheless, if the distribution of returns is not sufficiently asymmetric, downside risk measures may not withstand scrutiny in terms of providing an advantage over the variance or volatility (Grootveld & Hallerbach, 1999).

Semi-variance and Semi-deviation

The variance of the return random variable R is

$$\sigma^2 = \mathbb{E}\left[(R-\mu)^2\right],$$

where $\mu = \mathbb{E}[R]$ is the mean. The *semi-variance* (SV) is similarly defined but only taking into account when the random variable is below the mean:

$$\mathrm{SV} = \mathbb{E}\left[((\mu - R)^+)^2\right], \tag{10.1}$$

where the operator $(\cdot)^+ = \max(0, \cdot)$ only keeps the nonnegative part.

Similarly to the volatility (the square root of the variance), the *semi-deviation* is defined as the square root of the semi-variance. A related performance measure is the *Sortino ratio*, defined as the ratio of the expected return to the semi-deviation (similarly to the Sharpe ratio, which uses the volatility instead).

LPM

The *lower partial moment* (LPM) is a generalization of the semi-variance (Bawa, 1975; Fishburn, 1977):

$$\mathrm{LPM}_\alpha = \mathbb{E}\left[((\tau - R)^+)^\alpha\right], \tag{10.2}$$

where the parameter τ is termed the disaster level (minimum acceptable return) and the parameter α reflects the investor's feeling about falling short of τ, namely, $\alpha > 1$ naturally fits a risk-averse investor, $\alpha = 1$ corresponds to a neutral investor, and $0 < \alpha < 1$ is suitable for risk-seeking behavior.

By changing the parameters α and τ in (10.2) most downside measures used in practice can be formed. For instance, setting $\alpha = 2$ and $\mu = \mathbb{E}[R]$ yields the semi-variance (10.1).

In the same way that in the traditional modern portfolio theory it is common to plot the mean–volatility trade-off achieved by portfolios (see Section 7.1.1 in Chapter 7 for details), we can similarly plot the mean–risk trade-off where the risk is given by $\mathrm{LPM}_\alpha^{1/\alpha}$.

10.2.2 Tail Measures: VaR, CVaR, and EVaR

Tail measures, as the name indicates, focus on the tail of the distribution. They are typically defined in terms of the loss, which can be taken as the opposite of the portfolio return $R = \boldsymbol{w}^\top \boldsymbol{r}$:

$$\xi = -\boldsymbol{w}^\top \boldsymbol{r}.$$

Differently from the variance/volatility and semi-variance/semi-deviation, which attempt to measure the dispersion of the pdf of the portfolio return, tail measures focus on the tail of the distribution that represents the big losses (the left tail of the return distribution or the right tail of the loss distribution).

The *value-at-risk* (VaR) as a risk measure was proposed in the early 1990s by J. P. Morgan and denotes the maximum loss with a specified confidence level (McNeil et al., 2015):

$$\text{VaR}_\alpha = \inf \{\xi_0 : \Pr [\xi \leq \xi_0] \geq \alpha\}, \tag{10.3}$$

where α is the confidence level, for example, $\alpha = 0.95$ for 95%. In other words, VaR_α is the quantile function $q_\alpha(F)$ for the distribution function F (defined as $F(x_0) = \Pr[x \leq x_0]$). However, this measure does not consider the distribution shape of losses exceeding the VaR, is nonconvex, and is not a coherent measure (it lacks the subadditivity property) (McNeil et al., 2015).

The *conditional value-at-risk* (CVaR), also called *expected shortfall* (ES) and *expected tail loss* (ETL), builds on the VaR by taking into account the shape of the losses exceeding the VaR through the average:

$$\text{CVaR}_\alpha = \mathbb{E} [\xi \mid \xi \geq \text{VaR}_\alpha] . \tag{10.4}$$

The CVaR is a coherent risk measure and therefore satisfies several desirable properties (Rockafellar & Uryasev, 2002).

The *entropic VaR* (EVaR) is the tightest possible upper bound that can be obtained from the Chernoff inequality for the VaR (Ahmadi-Javid, 2012):

$$\text{EVaR}_\alpha = \inf_{z>0} \left\{ z^{-1} \log \left(\frac{1}{1-\alpha} \mathbb{E} [\exp(z\xi)] \right) \right\}. \tag{10.5}$$

The EVaR is also a coherent risk measure (Ahmadi-Javid, 2012) and satisfies other properties, such as strong monotonicity (Ahmadi-Javid & Fallah-Tafti, 2019).

Some interesting connections among the tail measures are:

- the monotonicity relationship:

$$\text{VaR}_\alpha \leq \text{CVaR}_\alpha \leq \text{EVaR}_\alpha;$$

- the "average VaR" expression:

$$\text{CVaR}_\alpha = \frac{1}{1-\alpha} \int_\alpha^1 \text{VaR}_u \, \mathrm{d}u$$

- limiting behavior: the VaR, CVaR, and EVaR all tend to the maximal value of the support of the pdf as $\alpha \to 1$.

Figure 10.2 illustrates the VaR, CVaR, and EVaR (as well as the maximal value) in the context of the pdf of the loss.

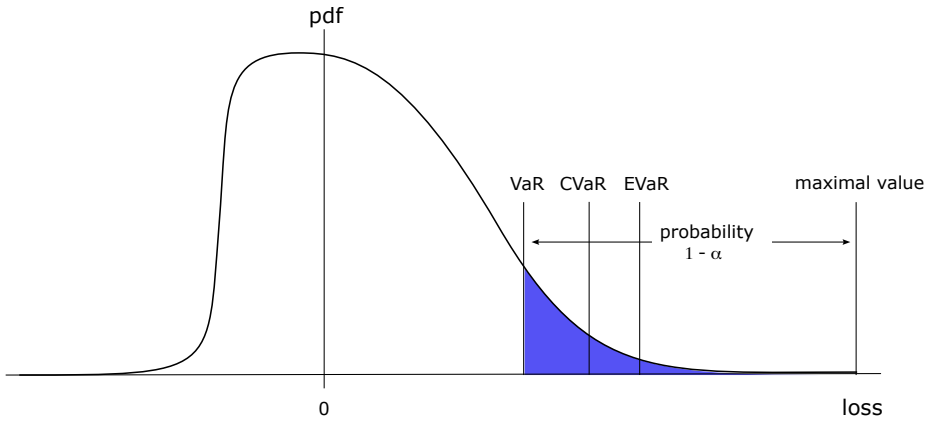


Figure 10.2 Illustration of loss distribution and tail measures (VaR, CVaR, and EVaR).

For the Gaussian distribution with mean μ and standard deviation σ, these tail measures can be further written as

$$\begin{aligned}
\mathrm{VaR}_\alpha &= \mu + \sigma\,\Phi^{-1}(\alpha), \\
\mathrm{CVaR}_\alpha &= \mu + \sigma\frac{\phi\left(\Phi^{-1}(\alpha)\right)}{1-\alpha}, \\
\mathrm{EVaR}_\alpha &= \mu + \sigma\sqrt{-2\log(1-\alpha)},
\end{aligned}$$

where Φ denotes the standard normal distribution function, ϕ its density function, and $\Phi^{-1}(\alpha)$ the α-quantile of Φ. Clearly, minimizing any of these measures under a Gaussian distribution is tantamount to minimizing the standard deviation σ.

Convex Characterization of CVaR

To use the CVaR in a portfolio formulation, it is helpful to first obtain a convex representation. To start with, the CVaR can be rewritten as

$$\begin{aligned}
\mathrm{CVaR}_\alpha &= \frac{1}{1-\alpha}\mathbb{E}\left[\xi \times I\{\xi \geq \mathrm{VaR}_\alpha\}\right] \\
&= \mathrm{VaR}_\alpha + \frac{1}{1-\alpha}\mathbb{E}\left[(\xi - \mathrm{VaR}_\alpha)^+\right],
\end{aligned}$$

which requires knowledge of the VaR.

Interestingly, the CVaR can be similarly written in a variational form without knowledge of the VaR (Rockafellar & Uryasev, 2000):

$$\mathrm{CVaR}_\alpha = \inf_\tau \left\{\tau + \frac{1}{1-\alpha}\mathbb{E}\left[(\xi-\tau)^+\right]\right\},$$

where the optimal τ is precisely the VaR.

In the context of optimization of the portfolio $\boldsymbol{w}$, we can conveniently write:

$$\begin{aligned} \mathrm{VaR}_\alpha(\boldsymbol{w}) &\in \arg\min_\tau F_\alpha(\boldsymbol{w},\tau), \\ \mathrm{CVaR}_\alpha(\boldsymbol{w}) &= \inf_\tau F_\alpha(\boldsymbol{w},\tau), \end{aligned} \tag{10.6}$$

where $F_\alpha(\boldsymbol{w},\tau)$ is the convex auxiliary function

$$F_\alpha(\boldsymbol{w},\tau) = \tau + \frac{1}{1-\alpha}\mathbb{E}\left[(-\boldsymbol{w}^\mathsf{T}\boldsymbol{r} - \tau)^+\right].$$

Convexity is easily established since the term $-\boldsymbol{w}^\mathsf{T}\boldsymbol{r} - \tau$ is linear, the operator $(\cdot)^+$ is the maximum of two convex functions (one constant and one linear) and hence convex, and the expectation is just a convexity-preserving nonnegative sum (see Appendix A for details on convexity).

From Downside Risk to CVaR

The CVaR is intimately related to the downside risk in the form of LPM (10.2) with $\alpha = 1$. This can be seen by rewriting

$$\begin{aligned} \mathrm{LPM}_1 &= \mathbb{E}\left[(\tau - R) \times I\{R \le \tau\}\right] \\ &= \mathbb{E}\left[(\xi - (-\tau)) \times I\{\xi \ge -\tau\}\right], \end{aligned}$$

where $I\{\cdot\}$ is the indicator function and we have used the fact that the loss is $\xi = -R$.

If we now choose the disaster level as $\tau = -\mathrm{VaR}_\alpha$, we can further write

$$\begin{aligned} \mathrm{LPM}_1 &= \mathbb{E}\left[(\xi - \mathrm{VaR}_\alpha) \times I\{\xi \ge \mathrm{VaR}_\alpha\}\right] \\ &= (1-\alpha)\mathbb{E}\left[(\xi - \mathrm{VaR}_\alpha) \mid \xi \ge \mathrm{VaR}_\alpha\right], \end{aligned}$$

which bears a striking resemblance to the CVaR in (10.4):

$$\mathrm{CVaR}_\alpha = \mathbb{E}\left[\xi \mid \xi \ge \mathrm{VaR}_\alpha\right].$$

Basically, by choosing $\tau = -\mathrm{VaR}_\alpha$ and ignoring the scaling factor $(1-\alpha)$, the downside risk LPM_1 measures the expected value of the tail shifted to the origin (excess loss above VaR_α), whereas the CVaR measures the expected value of the tail (loss above VaR_α). The main difference is that the VaR chooses the disaster level automatically whereas for the downside risk it is fixed a priori.

10.2.3 Drawdown

The *drawdown* (or underwater curve) attempts to measure the amount of suffering of an investor constantly monitoring the cumulative return or wealth. As such, it focuses entirely on the downside events while ignoring the upside movements. In addition, the measure of loss is always in relation to the past maximum (mimicking the human psychology).

The *high watermark* is the historical peak of the value $X(t)$ up to time t:

$$\mathrm{HWM}(t) = \max_{1 \le \tau \le t} X(\tau).$$

The drawdown at time t is defined as the decline from a historical peak of the value:

- *absolute drawdown*:

$$D(t) = \mathrm{HWM}(t) - X(t);$$

- *normalized drawdown*:

$$\bar{D}(t) = \frac{\mathrm{HWM}(t) - X(t)}{\mathrm{HWM}(t)}.$$

Figure 10.3 illustrates the net asset value (NAV) curve of the S&P 500 and the corresponding (normalized) drawdown, which is typically depicted as negative numbers going down.

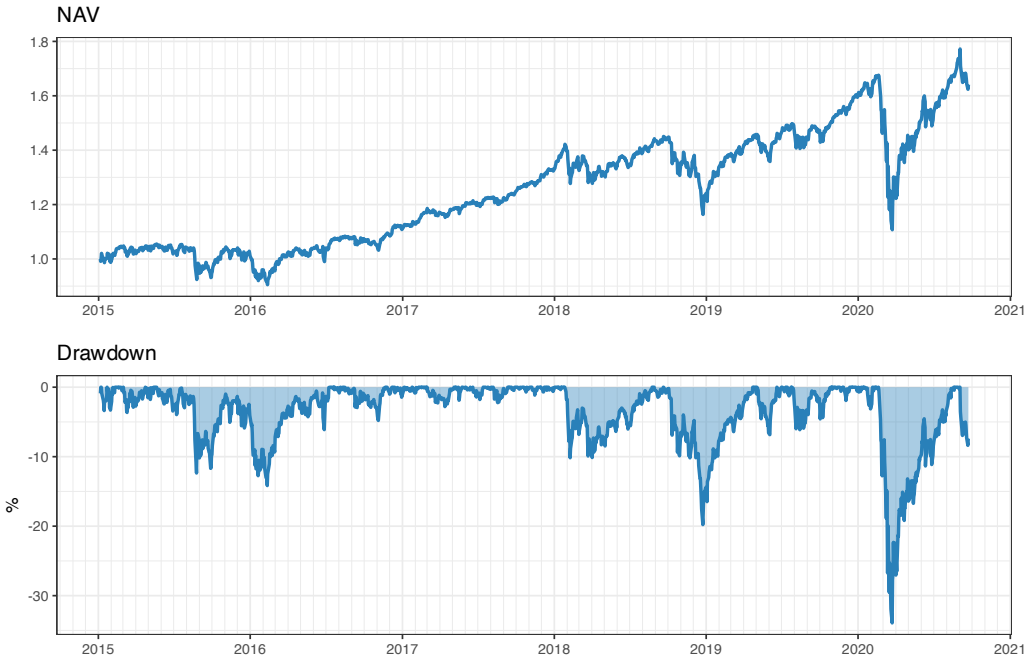


Figure 10.3 Illustration of NAV and corresponding drawdown.

Path Dependency

It is important to remark that the drawdown is a *path-dependent measure*. This means that it depends on the temporal order in which the returns happen. This is in sharp contrast to the previously considered measures, which are agnostic to the ordering of the returns.

For illustration purposes, Figure 10.4 shows the best and worst possible ordering of the returns corresponding to the original ordering in Figure 10.3. The difference is extreme: in the best case the drawdown reaches about 12.5%, whereas in the worst case it goes to virtually 100% (the original drawdown reached 34%).

Single-Number Summarization

In practice, it is convenient to summarize the whole drawdown curve, which spans a period $t = 1, \ldots, T$, into a single number. This can be done in different ways:

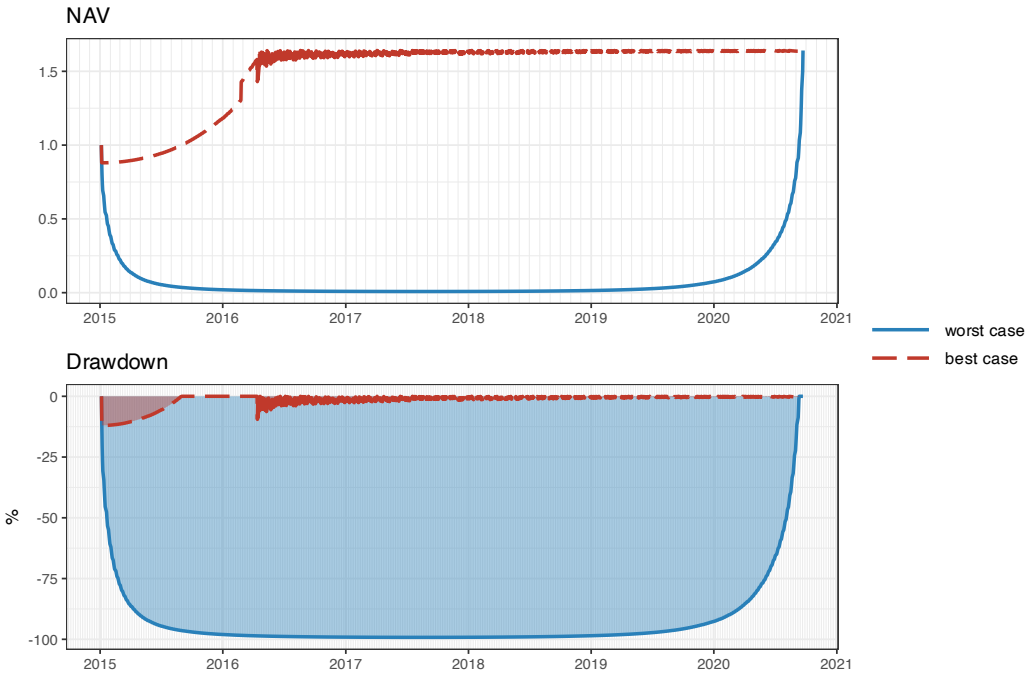


Figure 10.4 Effect of ordering of returns in the cumulative return and drawdown.

- *maximum drawdown* (Max-DD):

$$\text{Max-DD} = \max_{1 \le t \le T} D(t);$$

- *average drawdown* (Ave-DD):

$$\text{Ave-DD} = \frac{1}{T} \sum_{1 \le t \le T} D(t);$$

- *CVaR of drawdown* (CVaR-DD) or *conditional drawdown at risk* (CDaR):

$$\text{CDaR}_\alpha = \mathbb{E}\left[D(t) \mid D(t) \ge \text{VaR}_\alpha\right],$$

  where VaR_α is the value at risk of the drawdown $D(t)$ with confidence level α.

10.3 Downside Risk Portfolios

We now consider portfolio formulations based on downside risk measures.

10.3.1 Formulation

Using the downside risk measure LPM in (10.2) instead of the usual variance $\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}$ leads to the mean–downside risk formulation:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\top \boldsymbol{\mu} - \lambda\, \mathbb{E}\left[\left((\tau - \boldsymbol{w}^\top \boldsymbol{r})^+\right)^\alpha\right] \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}. \end{array} \tag{10.7}$$

Note that this problem can be similarly formulated by moving either the expected return or risk term to the constraints (see Chapter 7).

In practice, the expectation operator in the LPM has to be approximated by the sample mean over T observations $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T$:

$$\mathbb{E}\left[\left((\tau - \boldsymbol{w}^\mathsf{T}\boldsymbol{r})^+\right)^\alpha\right] \approx \frac{1}{T}\sum_{t=1}^{T}\left((\tau - \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t)^+\right)^\alpha.$$

Note that this is the same technique used to approximate the variance:

$$\mathbb{E}\left[\left(\boldsymbol{w}^\mathsf{T}(\boldsymbol{r} - \boldsymbol{\mu})\right)^2\right] \approx \frac{1}{T}\sum_{t=1}^{T}\left(\boldsymbol{w}^\mathsf{T}(\boldsymbol{r}_t - \boldsymbol{\mu})\right)^2 = \boldsymbol{w}^\mathsf{T}\hat{\boldsymbol{\Sigma}}\boldsymbol{w},$$

where $\hat{\boldsymbol{\Sigma}}$ is the sample covariance matrix (see Chapter 3 for details on estimating the covariance matrix).

Thus, the mean–downside risk formulation is finally written as

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda\frac{1}{T}\sum_{t=1}^{T}\left((\tau - \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t)^+\right)^\alpha \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W},\end{array}$$

where all the return observations $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T$ appear explicitly and cannot be condensed into a convenient matrix (unlike in the case of the variance).

From an optimization perspective, it is convenient to rewrite the formulation without the nondifferentiable operator $(\cdot)^+$ as

$$\begin{array}{lll}\underset{\boldsymbol{w},\{s_t\}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda\frac{1}{T}\sum_{t=1}^{T} s_t^\alpha & \\ \text{subject to} & 0 \leq s_t \geq \tau - \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t, \qquad & t = 1, \ldots, T, \\ & \boldsymbol{w} \in \mathcal{W}. & \end{array} \tag{10.8}$$

Interestingly, the class of optimization problems for common choices of α (assuming that the constraints in $\mathcal{W}$ are linear) are all convex and can be optimally solved, namely:

- linear program for $\alpha = 1$;
- quadratic program for $\alpha = 2$ (semi-variance portfolio); and
- general convex program for $\alpha = 3$.

10.3.2 Semi-variance Portfolios

As previously mentioned, the variance has a very convenient expression in terms of the covariance matrix $\boldsymbol{\Sigma}$:

$$\mathbb{E}\left[\left(\boldsymbol{w}^\mathsf{T}(\boldsymbol{r} - \boldsymbol{\mu})\right)^2\right] = \boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w},$$

where

$$\boldsymbol{\Sigma} = \mathbb{E}\left[(\boldsymbol{r} - \boldsymbol{\mu})\,(\boldsymbol{r} - \boldsymbol{\mu})^\mathsf{T}\right].$$

The question is whether the semi-variance enjoys a similar property (even as an approximation) so that we can write

$$\mathbb{E}\left[\left((\tau - \boldsymbol{w}^\top \boldsymbol{r})^+\right)^2\right] \approx \boldsymbol{w}^\top \boldsymbol{M} \boldsymbol{w}$$

for some conveniently defined matrix $\boldsymbol{M}$.

In fact, Markowitz himself suggested using the following matrix (Markowitz, 1959), which provides an exact semi-variance:

$$\boldsymbol{M}(\boldsymbol{w}) = \mathbb{E}\left[(\tau \mathbf{1} - \boldsymbol{r})(\tau \mathbf{1} - \boldsymbol{r})^\top \times I\{\tau > \boldsymbol{w}^\top \boldsymbol{r}\}\right],$$

where $\mathbf{1}$ denotes the all-one vector and $I\{\cdot\}$ the indicator function. However, this matrix is endogenous in the sense that it depends on the portfolio $\boldsymbol{w}$ and it is therefore not appropriate for portfolio optimization.

Unfortunately, the semi-variance cannot be written via an exogenous matrix $\boldsymbol{M}$ (independent of $\boldsymbol{w}$). Nevertheless, this has not stopped authors from proposing good heuristic approximations (Estrada, 2008) such as

$$\boldsymbol{M} = \mathbb{E}\left[(\tau \mathbf{1} - \boldsymbol{r})^+ \left((\tau \mathbf{1} - \boldsymbol{r})^+\right)^\top\right].$$

Many other practical approaches have been proposed over the past few decades, cf. Estrada (2008).

Summarizing, the mean–semi-variance formulation can be obtained by setting $\alpha = 2$ in (10.7), but it can also be conveniently approximated (similarly to the mean–variance formulation) as

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\top \boldsymbol{\mu} - \frac{\lambda}{2} \boldsymbol{w}^\top \boldsymbol{M} \boldsymbol{w} \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}. \end{array}$$

10.3.3 Numerical Experiments

We now compare different versions of downside risk portfolios based on (10.8), namely, for $\alpha = 1$, $\alpha = 2$ (with and without the approximation), and $\alpha = 3$. To focus on the effect of the risk measure, we ignore the expected return term in the optimization (effectively letting $\lambda \to \infty$) and also include the GMVP as a reference benchmark.

Figure 10.5 shows boxplots of the Sharpe ratio and maximum drawdown for 200 realizations of 50 randomly chosen stocks from the S&P 500 during 2015–2020, reoptimizing the portfolios every month with a lookback of one year.

10.4 Tail-Based Portfolios

We will now focus on CVaR and EVaR portfolios (and the limiting case of the worst-case portfolio) since they can be conveniently formulated as convex problems.

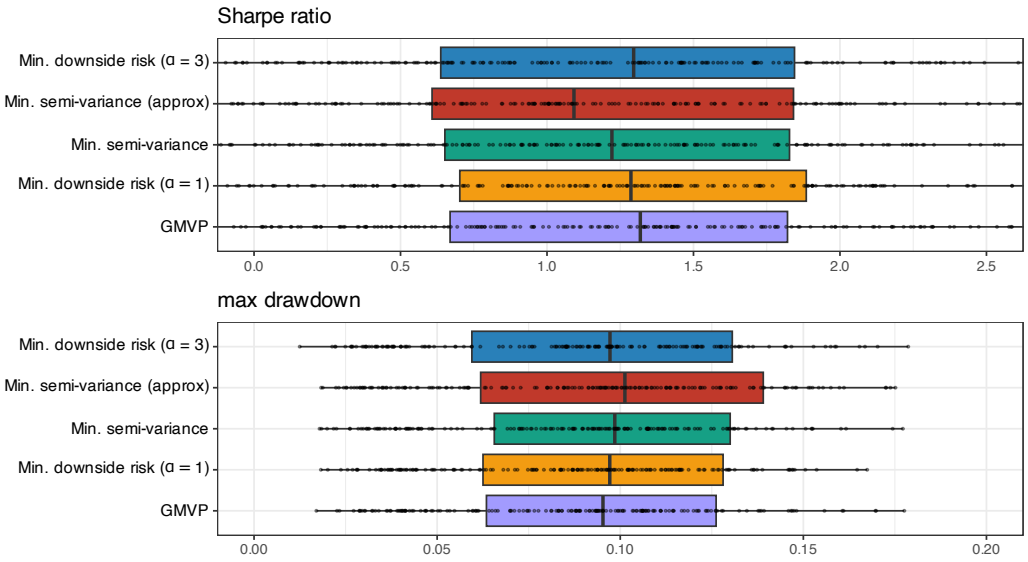


Figure 10.5 Backtest performance of different downside risk portfolios.

10.4.1 Formulation for CVaR Portfolios

The mean–CVaR formulation replaces the usual variance term $\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}$ by the CVaR in (10.4) as a measure of risk:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\top \boldsymbol{\mu} - \lambda \, \text{CVaR}_\alpha(\boldsymbol{w}) \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}. \end{array} \tag{10.9}$$

As usual, this problem can be similarly formulated by moving either the expected return or risk term to the constraints (see Chapter 7).

To write the problem in convex form, we will use the variational convex representation of the CVaR in (10.6):

$$\text{CVaR}_\alpha(\boldsymbol{w}) = \inf_\tau \left\{ \tau + \frac{1}{1-\alpha} \mathbb{E}\left[(-\boldsymbol{w}^\top \boldsymbol{r} - \tau)^+ \right] \right\}.$$

This leads to the convex mean–CVaR formulation:

$$\begin{array}{ll} \underset{\boldsymbol{w}, \tau}{\text{maximize}} & \boldsymbol{w}^\top \boldsymbol{\mu} - \lambda \left(\tau + \frac{1}{1-\alpha} \mathbb{E}\left[(-\boldsymbol{w}^\top \boldsymbol{r} - \tau)^+ \right] \right) \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array}$$

where the auxiliary variable τ has been conveniently moved from the inner minimization to the outer maximization (and we have tacitly assumed that the set $\mathcal{W}$ is convex).

In practice, the expectation operator is approximated by the sample mean over T observations $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T$:

$$\mathbb{E}\left[(-\boldsymbol{w}^\top \boldsymbol{r} - \tau)^+ \right] \approx \frac{1}{T} \sum_{t=1}^{T} (-\boldsymbol{w}^\top \boldsymbol{r}_t - \tau)^+.$$

Thus, the mean–CVaR formulation is finally given as

$$\begin{array}{ll} \underset{\boldsymbol{w},\tau}{\text{maximize}} & \boldsymbol{w}^\top\boldsymbol{\mu} - \lambda\left(\tau + \frac{1}{1-\alpha}\frac{1}{T}\sum_{t=1}^{T}(-\boldsymbol{w}^\top\boldsymbol{r}_t - \tau)^+\right) \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}. \end{array}$$

From an optimization perspective, it is convenient to rewrite the formulation without the nondifferentiable operator $(\cdot)^+$ by introducing the T auxiliary variables $\boldsymbol{u} = (u_1, \ldots, u_T)$ as

$$\begin{array}{ll} \underset{\boldsymbol{w},\tau,\boldsymbol{u}}{\text{maximize}} & \boldsymbol{w}^\top\boldsymbol{\mu} - \lambda\left(\tau + \frac{1}{1-\alpha}\frac{1}{T}\sum_{t=1}^{T} u_t\right) \\ \text{subject to} & 0 \leq u_t \geq -\boldsymbol{w}^\top\boldsymbol{r}_t - \tau, \qquad t = 1, \ldots, T, \\ & \boldsymbol{w} \in \mathcal{W}. \end{array} \tag{10.10}$$

This problem is a linear program (assuming that the set $\mathcal{W}$ is described via linear constraints) and can be conveniently solved with an LP solver.

It is important to bear in mind that the tail events in the CVaR formulation (10.10) happen with low probability (by definition) and, therefore, very few samples (if any) will contribute to the characterization of the CVaR. For instance, if $\alpha = 0.99$ and we have $T = 200$ observations, then only 2 samples out of the 200 will characterize the tail, which is too few samples for a proper characterization of the shape of the tail. This effect is further exacerbated as the dimension N becomes large. As a consequence, the CVaR portfolio may not be numerically stable and alternative methods have been proposed, such as based on some parametric distribution of the returns (Gaussian or elliptical distributions), alternative estimation methods for CVaR (Hong et al., 2014; Nadarajah et al., 2014), use of worst-case characterizations of CVaR (Zhu & Fukushima, 2009), and sophisticated tail characterizations based on extreme value theory (McNeil & Frey, 2000).

10.4.2 Formulation for EVaR Portfolios

Similarly to (10.9), the mean–EVaR formulation replaces the usual variance term $\boldsymbol{w}^\top\boldsymbol{\Sigma}\boldsymbol{w}$ by the EVaR in (10.5) as a measure of risk:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\top\boldsymbol{\mu} - \lambda\,\text{EVaR}_\alpha(\boldsymbol{w}) \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}. \end{array} \tag{10.11}$$

Using the change of variable $t = z^{-1}$ in the EVaR (10.5), the problem can be written (Ahmadi-Javid & Fallah-Tafti, 2019) as

$$\begin{array}{ll} \underset{\boldsymbol{w},t>0}{\text{maximize}} & \boldsymbol{w}^\top\boldsymbol{\mu} - \lambda\left(t\,\log\left(\frac{1}{1-\alpha}\mathbb{E}\left[\exp(-t^{-1}\boldsymbol{w}^\top\boldsymbol{r})\right]\right)\right) \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array}$$

which is a convex problem (assuming that the set $\mathcal{W}$ is convex) because the log-sum-exp function is convex and the perspective $t f(\boldsymbol{x}/t)$ of a function $f(\boldsymbol{x})$ preserves convexity (see Appendix A for details on convexity).

In practice, the expectation operator is approximated by the sample mean over T observations

$\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T$ and the mean–EVaR formulation is finally written as

$$\begin{array}{ll} \underset{\boldsymbol{w}, t>0}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda \left(t \log \left(\sum_{t'=1}^{T} \left[\exp(-t^{-1}\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_{t'}) \right] \right) - t \log \left((1-\alpha)T \right) \right) \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}. \end{array} \tag{10.12}$$

This problem can be solved in practice in a variety of ways (see Appendix B for a discussion on algorithms, solvers, and modeling frameworks):

- via a general-purpose solver (since the problem is convex, it will find an optimal solution);
- via a tailored interior-point method for convex problems (Ahmadi-Javid & Fallah-Tafti, 2019);
- via a convex modeling framework that can recognize the convexity of the log-sum-exp function and then performing bisection over t;
- via a convex modeling framework that can recognize both the convexity of the log-sum-exp function and the convexity-preserving property of the perspective operator;
- via a convex reformulation in terms of the *exponential cone* $\mathcal{K}_{\text{exp}}$ (Chares, 2007),[1] which some solvers and modeling frameworks can recognize (Cajas, 2021):[2]

$$\begin{array}{ll} \underset{\boldsymbol{w}, t>0, s, \boldsymbol{u}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda \left(s - t \log \left((1-\alpha)T \right) \right) \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \\ & t \geq u_1 + \cdots + u_T, \\ & (-\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_{t'} - s, t, u_{t'}) \in \mathcal{K}_{\text{exp}}, \quad t' = 1, \ldots, T, \end{array}$$

where

$$\mathcal{K}_{\text{exp}} \triangleq \left\{ (a,b,c) \mid c \geq b\, e^{a/b}, b > 0 \right\} \cup \left\{ (a,b,c) \mid a \leq 0, b = 0, c \geq 0 \right\}.$$

10.4.3 Formulation for the Worst-Case Portfolio

The VaR, CVaR, and EVaR measures all tend to the maximal value of the support of the pdf of the loss as $\alpha \to 1$. In practice, this implies focusing attention on the worst realization of the return or loss.

---

[1] The convex constraint involving the log-sum-exp function

$$s \geq t \log \left(e^{x_1/t} + e^{x_2/t} \right),$$

for $t > 0$, can be rewritten in terms of the exponential cone $\mathcal{K}_{\text{exp}}$ as (Chares, 2007)

$$\begin{array}{c} t \geq u_1 + u_2, \\ (x_i - s, t, u_i) \in \mathcal{K}_{\text{exp}}, \qquad i = 1, 2. \end{array}$$

[2] Some solvers like the Embedded COnic Solver (ECOS) solver (https://github.com/embotech/ecos) or MOSEK (www.mosek.com) are able to handle problems with the exponential cone. Some modeling frameworks like CVXR (https://cvxr.rbind.io) can also accept the exponential cone.

This worst-case risk leads to the following formulation (Young, 1998):

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda \max_{1\le t\le T}\{-\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t\} \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W} \end{array}$$

or, without the nondifferentiable maximum operator,

$$\begin{array}{lll} \underset{\boldsymbol{w},\tau}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda\ \tau & \\ \text{subject to} & \tau \ge -\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t, \qquad & t = 1,\ldots,T, \\ & \boldsymbol{w} \in \mathcal{W}. & \end{array} \tag{10.13}$$

This problem is again a linear program (assuming that the set $\mathcal{W}$ is described via linear constraints) and can be conveniently solved with an LP solver.

10.4.4 Numerical Experiments

We now compare CVaR, EVaR, and worst-case portfolios based on (10.10), (10.12), and (10.13), respectively. To focus on the effect of the risk measure, we ignore the expected return term in the optimization (effectively letting $\lambda \to \infty$) and also include the GMVP as a reference benchmark.

A word of caution is necessary. Since these portfolio formulations are based on a nonparametric computation of the risk (observed returns directly instead of some covariance matrix) and the tail events happen with low probability (by definition), we cannot expect a good characterization of the true tail. The most extreme case is the worst-case portfolio which is defined by a single data point. For CVaR, very few observations (if any) may occur on the tail, and this can be exacerbated with larger values of α. EVaR may be slightly better in this regard since it uses all the observations. For this reason, alternative, more stable, methods have been proposed, as previously mentioned, based on parametric models or sophisticated tail characterizations based on extreme value theory.

Figure 10.6 shows boxplots of the Sharpe ratio and maximum drawdown for 200 realizations of 50 randomly chosen stocks, from the S&P 500 during 2015–2020, reoptimizing the portfolios every month with a lookback of one year. It is difficult to draw conclusions from this numerical experiment, but the EVaR portfolios seem to produce better results than the CVaR ones, as expected.

10.5 Drawdown Portfolios

Drawdown portfolios can be approached via statistical modeling of the returns based on dynamic programming (Cvitanic & Karatzas, 1995; Grossman & Zhou, 1993), as well as from a more data-driven perspective based on a sample-path (realization) of portfolio returns (Chekhlov et al., 2004), as we consider here.

As we know, the return of a portfolio $\boldsymbol{w}$ at time t is given by $R_t = \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t$, where $\boldsymbol{r}_t$ denotes linear returns or approximately log-returns. However, since the drawdown is derived from the cumulative return, we need the *portfolio cumulative return*:

$$R_t^{\text{cum}} = \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}},$$

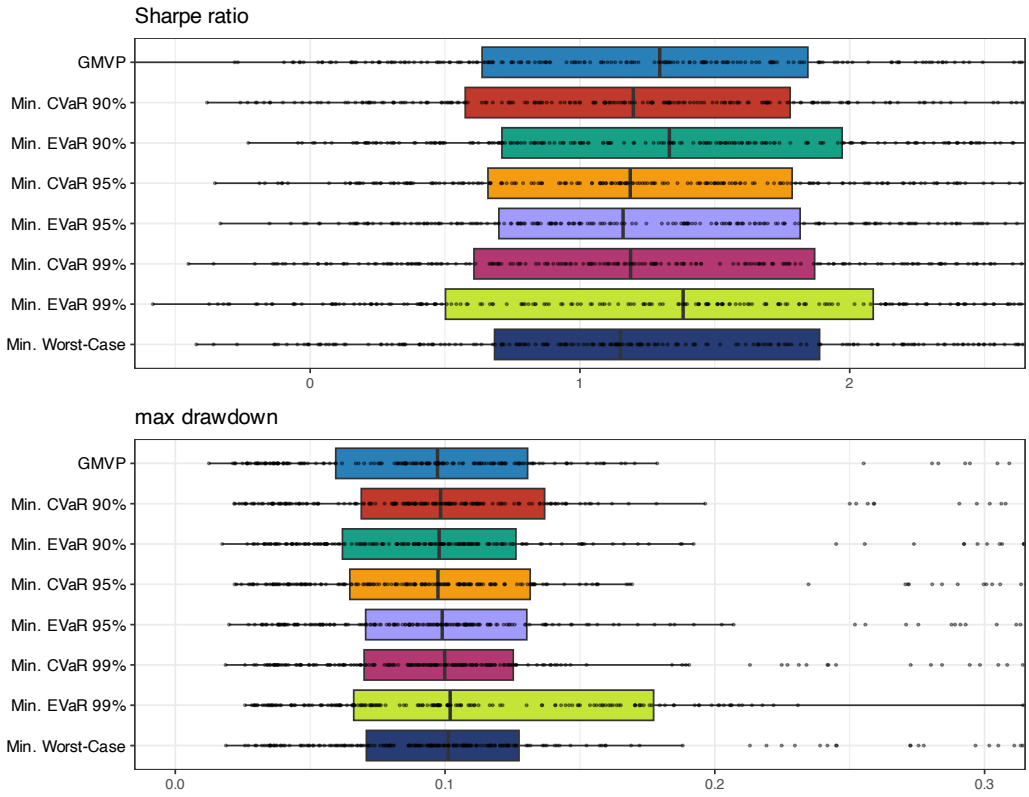


Figure 10.6 Backtest performance of CVaR and EVaR portfolios.

where the cumulative returns of the assets can be computed as

$$\boldsymbol{r}_t^{\text{cum}} = \sum_{\tau=1}^{t} \boldsymbol{r}_\tau.$$

Depending on whether we use linear or log-returns in $\boldsymbol{r}_t$, this expression corresponds to the uncompounded linear returns or compounded log-returns, respectively (see Section 6.1.3 in Chapter 6 for details). Note that the value of the portfolio is $1 + R_t^{\text{cum}}$.

The *absolute drawdown* is

$$D_t(\boldsymbol{w}) = \max_{1 \le \tau \le t} \boldsymbol{w}^\mathsf{T} \boldsymbol{r}_\tau^{\text{cum}} - \boldsymbol{w}^\mathsf{T} \boldsymbol{r}_t^{\text{cum}}$$

and a constraint of the form $D_t(\boldsymbol{w}) \le \alpha$ can be written as the linear (after adding some dummy variables) constraint

$$\boldsymbol{w}^\mathsf{T} \boldsymbol{r}_t^{\text{cum}} \ge \max_{1 \le \tau \le t} \boldsymbol{w}^\mathsf{T} \boldsymbol{r}_\tau^{\text{cum}} - \alpha.$$

Similarly, the *normalized drawdown* is

$$\bar{D}_t(\boldsymbol{w}) = \frac{\max\limits_{1\le\tau\le t} \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_\tau^{\text{cum}} - \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}}}{1 + \max\limits_{1\le\tau\le t} \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_\tau^{\text{cum}}}$$

and a constraint of the form $\bar{D}_t(\boldsymbol{w}) \le \alpha$ can be written as the linear (again after adding some dummy variables) constraint

$$\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}} \ge (1-\alpha) \max_{1\le\tau\le t} \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_\tau^{\text{cum}} - \alpha.$$

10.5.1 Formulation for the Max-DD Portfolio

The mean–Max-DD formulation replaces the usual variance term $\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}$ by the maximum drawdown as a measure of risk:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda\,\text{Max-DD}(\boldsymbol{w}) \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}. \end{array} \tag{10.14}$$

As usual, this problem can be similarly formulated by moving either the expected return or the risk term to the constraints (see Chapter 7).

Substituting for $\text{Max-DD}(\boldsymbol{w}) = \max\limits_{1\le t\le T} D_t(\boldsymbol{w})$, where $D_t(\boldsymbol{w})$ is the drawdown at time t, leads to the problem

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda \max\limits_{1\le t\le T} \left\{ \max\limits_{1\le\tau\le t} \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_\tau^{\text{cum}} - \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}} \right\} \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array}$$

which is convex (assuming $\mathcal{W}$ is convex) because the maximum of convex functions is convex (see Appendix A for details on convexity).

Writing the problem in epigraph form and introducing the auxiliary variables s and $\boldsymbol{u}$ ($u_0 \triangleq -\infty$) to get rid of the max operators leads to

$$\begin{array}{lll} \underset{\boldsymbol{w},\boldsymbol{u},s}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda\, s & \\ \text{subject to} & \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}} \le u_t \le s + \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}}, & t = 1,\ldots,T, \\ & u_{t-1} \le u_t, & \\ & \boldsymbol{w} \in \mathcal{W}, & \end{array} \tag{10.15}$$

which is a linear program (assuming $\mathcal{W}$ only contains linear constraints).

It is important to remark that, by definition, the worst drawdown is given by a single data point, which makes this risk measure extremely sensitive. In other words, minute changes in the portfolio weights and the specific time period examined (recall that the drawdown is path-dependent) may produce totally different values of the maximum drawdown. This makes this measure of risk not very reliable, which can be mitigated by using instead the average drawdown or the conditional drawdown-at-risk considered next. As an alternative, if the distribution is close to Gaussian, then the mean–variance framework may be sufficient (see Chapter 7), whereas if the distribution shows some skewness and heavy tails, then high-order portfolios can be used (see Chapter 9).

10.5.2 Formulation for the Ave-DD Portfolio

We can repeat the same procedure with the average drawdown instead:

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda \dfrac{1}{T}\displaystyle\sum_{t=1}^{T}\left(\max_{1\le\tau\le t} \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_\tau^{\text{cum}} - \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}}\right)\\ \text{subject to} & \boldsymbol{w}\in\mathcal{W},\end{array}$$

which is also convex (assuming $\mathcal{W}$ is convex) because the maximum of convex functions is convex (see Appendix A for details on convexity).

Writing the problem in epigraph form and introducing the auxiliary variables s and $\boldsymbol{u}$ ($u_0 \triangleq -\infty$) to get rid of the max operator leads to

$$\begin{array}{ll}\underset{\boldsymbol{w},\boldsymbol{u},s}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda\, s\\ \text{subject to} & \frac{1}{T}\sum_{t=1}^{T} u_t \le \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}} + s,\\ & \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}} \le u_t, \qquad t=1,\ldots,T,\\ & u_{t-1}\le u_t,\\ & \boldsymbol{w}\in\mathcal{W},\end{array} \tag{10.16}$$

which is a linear program (assuming $\mathcal{W}$ only contains linear constraints).

10.5.3 Formulation for the CVaR-DD Portfolio

To formulate the mean–CVaR-DD portfolio, we will make use of the following variational convex representation of the CVaR-DD (or CDaR) (Chekhlov et al., 2004, 2005):

$$\text{CVaR-DD}(\boldsymbol{w}) = \inf_{\tau}\left\{\tau + \frac{1}{1-\alpha}\frac{1}{T}\sum_{t=1}^{T}(D_t(\boldsymbol{w})-\tau)^+\right\}.$$

This leads to the mean–CVaR-DD formulation:

$$\begin{array}{ll}\underset{\boldsymbol{w},\tau}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda\left(\tau + \dfrac{1}{1-\alpha}\dfrac{1}{T}\displaystyle\sum_{t=1}^{T}\left(\max_{1\le\tau'\le t}\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_{\tau'}^{\text{cum}} - \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}} - \tau\right)^+\right)\\ \text{subject to} & \boldsymbol{w}\in\mathcal{W},\end{array}$$

where the auxiliary variable τ has been conveniently moved from the inner minimization to the outer maximization. This problem is convex, assuming that $\mathcal{W}$ is a convex set.

After a series of manipulations to get rid of the nondifferentiable max operator and $(\cdot)^+$, and the introduction of the auxiliary variables s, $\boldsymbol{z}$, and $\boldsymbol{u}$ ($u_0 \triangleq -\infty$), the problem can be finally rewritten as

$$\begin{array}{ll}\underset{\boldsymbol{w},\tau,s,\boldsymbol{z},\boldsymbol{u}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda\, s\\ \text{subject to} & s \ge \tau + \frac{1}{1-\alpha}\frac{1}{T}\sum_{t=1}^{T} z_t,\\ & 0\le z_t \ge u_t - \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}} - \tau, \qquad t=1,\ldots,T,\\ & \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}} \le u_t,\\ & u_{t-1}\le u_t,\\ & \boldsymbol{w}\in\mathcal{W},\end{array} \tag{10.17}$$

which is a linear program (assuming $\mathcal{W}$ only contains linear constraints).

Similarly to the EVaR portfolio in (10.11)–(10.12), one can easily formulate a drawdown EVaR simply by replacing in (10.12) the loss terms $-\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t$ by $D_t(\boldsymbol{w}) = \max_{1\le\tau\le t} \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_\tau^{\text{cum}} - \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}}$.

10.5.4 Numerical Experiments

We now compare several drawdown-based portfolios, namely, that based on the minimization of the maximum drawdown formulated in (10.15), the average drawdown formulated in (10.16), and the drawdown CVaR formulated in (10.17). To focus on the effect of the risk measure, we ignore the expected return term in the optimization (effectively letting $\lambda \to \infty$) and also include the GMVP as a reference benchmark.

Similarly to the CVaR portfolios, a word of caution is necessary here. The problem boils down to the fact that the worst drawdowns happen with low probability, which translates into very few samples being used in the computation of the risk measure. This is specially true for the maximum drawdown (a single sample) and the drawdown CVaR (extremely few samples).

Figure 10.7 shows boxplots of the Sharpe ratio and maximum drawdown for 200 realizations of 50 randomly chosen stocks, from the S&P 500 during 2015–2020, reoptimizing the portfolios every month with a lookback of one year. The drawdown portfolios do not seem to outperform the simple GMVP benchmark, although more exhaustive empirical experiments would be necessary.

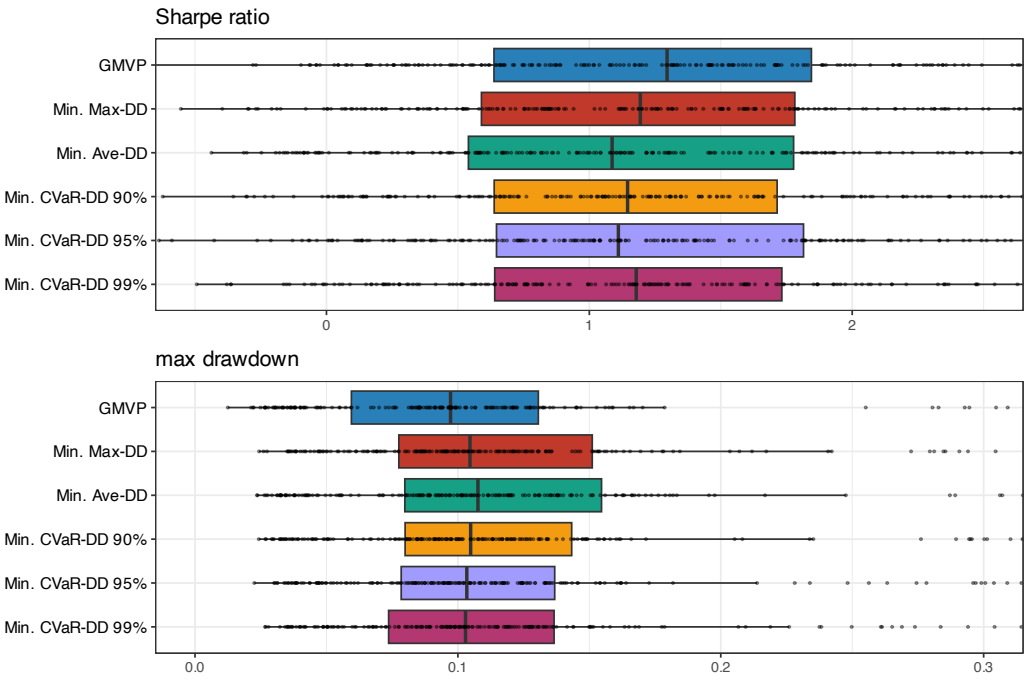


Figure 10.7 Backtest performance of drawdown portfolios.

10.6 Summary

The variance (similarly, the volatility) is a very simple way to measure the risk of a portfolio and was used in Markowitz's 1952 seminal mean–variance modern portfolio theory framework. Since then, a wide variety of alternative and more sophisticated measures of risk have been proposed, such as semi-variance, downside risk, VaR, CVaR, EVaR, drawdown, and so on.

Interestingly, many meaningful risk measures can be conveniently incorporated in the context of portfolio optimization, expressed in terms of the raw returns of the assets. Some notable examples include:

- *Downside risk portfolios*: The risk focuses on the downside losses, which can be formulated in convex form (parameterized by the parameter α):
  - $\alpha = 1$: formulated as a linear program;
  - $\alpha = 2$: semi-variance portfolio formulated as a quadratic program;
  - $\alpha = 3$: more risk averse and formulated as a convex program.
- *Tail portfolios*: The risk is measured by the tail of the distribution of the losses, which can be formulated in convex form:
  - *CVaR portfolios*: based on the mean of the tail and formulated as a linear program;
  - *EVaR portfolios*: based on a smooth approximation of the CVaR and formulated in terms of the exponential cone;
  - *worst-case portfolio*: extreme version of CVaR and EVaR portfolios and formulated as a linear program.
- *Drawdown portfolios*: The risk is based on the drawdown, which can be formulated as linear programs:
  - *maximum drawdown portfolio*: based on the single worst drawdown;
  - *average drawdown portfolio*: based on the average of all the drawdowns; and
  - *drawdown CVaR portfolio*: based on the average of the tail of the drawdowns.

Exercises

10.1 (Computing alternative measures of risk) Generate 10 000 samples following a normal distribution, plot the histogram, and compute the following measures:

- mean
- variance and standard deviation
- semi-variance and semi-deviation
- tail measures (VaR, CVaR, and EVaR) based on raw data
- tail measures (VaR, CVaR, and EVaR) based on a Gaussian approximation.

10.2 (CVaR in variational convex form) Consider the following expression for the CVaR:

$$\mathrm{CVaR}_\alpha = \mathbb{E}\left[\xi \mid \xi \geq \mathrm{VaR}_\alpha\right].$$

Show that it can be rewritten in a convex variational form as:

$$\mathrm{CVaR}_\alpha = \inf_\tau \left\{ \tau + \frac{1}{1-\alpha}\mathbb{E}\left[(\xi - \tau)^+\right] \right\},$$

where the optimal τ precisely equals VaR_α.

10.3 (Sanity check for variational computation of CVaR) Generate 10 000 samples of the random variable ξ following a normal distribution and compute the CVaR as

$$\mathrm{CVaR}_\alpha = \mathbb{E}\left[\xi \mid \xi \geq \mathrm{VaR}_\alpha\right].$$

Verify numerically that the variational expression for the CVaR gives the same result:

$$\mathrm{CVaR}_\alpha = \inf_\tau \left\{ \tau + \frac{1}{1-\alpha}\mathbb{E}\left[(\xi - \tau)^+\right] \right\}.$$

10.4 (CVaR vs. downside risk) Consider the following two measures of risk in terms of the loss random variable ξ:

- downside risk in the form of lower partial moment (LPM) with $\alpha = 1$:

$$\mathrm{LPM}_1 = \mathbb{E}\left[(\xi - \xi_0)^+\right];$$

- CVaR:

$$\mathrm{CVaR}_\alpha = \mathbb{E}\left[\xi \mid \xi \geq \mathrm{VaR}_\alpha\right].$$

Rewrite LPM_1 in the form of CVaR_α and the other way around. Hint: use $\xi_0 = \mathrm{VaR}_\alpha$.

10.5 (Log-sum-exp function as exponential cone) Show that the following convex constraint involving the perspective operator on the log-sum-exp function,

$$s \geq t \log\left(e^{x_1/t} + e^{x_2/t}\right),$$

for $t > 0$, can be rewritten in terms of the exponential cone $\mathcal{K}_{\mathrm{exp}}$ as

$$\begin{aligned} t &\geq u_1 + u_2, \\ (x_i - s, t, u_i) &\in \mathcal{K}_{\mathrm{exp}}, \qquad i = 1, 2, \end{aligned}$$

where

$$\mathcal{K}_{\mathrm{exp}} \triangleq \left\{(a,b,c) \mid c \geq b\, e^{a/b}, b > 0\right\} \cup \left\{(a,b,c) \mid a \leq 0, b = 0, c \geq 0\right\}.$$

10.6 (Drawdown and path-dependency)

a. Generate 10 000 samples of returns following a normal distribution.
b. Compute and plot the cumulative returns, and plot the drawdown.
c. Randomly reorder the original returns and plot again.
d. Repeat a few times to observe the path-dependency property of the drawdown.

10.7 (Semi-variance portfolios)

a. Download market data corresponding to N assets (e.g., stocks or cryptocurrencies) during a period with T observations, $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T \in \mathbb{R}^N$.

b. Solve the minimization of the semi-variance in a nonparametric way (reformulate it as a quadratic program):

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \frac{1}{T}\sum_{t=1}^{T}\left((\tau-\boldsymbol{w}^{\mathsf{T}}\boldsymbol{r}_t)^{+}\right)^2\\ \text{subject to} & \boldsymbol{w}\geq\mathbf{0},\quad \mathbf{1}^{\mathsf{T}}\boldsymbol{w}=1.\end{array}$$

c. Solve the parametric approximation based on the quadratic program:

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{M}\boldsymbol{w}\\ \text{subject to} & \boldsymbol{w}\geq\mathbf{0},\quad \mathbf{1}^{\mathsf{T}}\boldsymbol{w}=1,\end{array}$$

where

$$\boldsymbol{M}=\mathbb{E}\left[(\tau\mathbf{1}-\boldsymbol{r})^{+}\left((\tau\mathbf{1}-\boldsymbol{r})^{+}\right)^{\mathsf{T}}\right].$$

d. Comment on the goodness of the approximation.

10.8 (CVaR portfolios)

a. Download market data corresponding to N assets (e.g., stocks or cryptocurrencies) during a period with T observations, $\boldsymbol{r}_1,\ldots,\boldsymbol{r}_T\in\mathbb{R}^N$.
b. Solve the minimum CVaR portfolio as the following linear program for different values of the parameter α:

$$\begin{array}{lll}\underset{\boldsymbol{w},\tau,\boldsymbol{u}}{\text{minimize}} & \tau+\frac{1}{1-\alpha}\frac{1}{T}\sum_{t=1}^{T}u_t & \\ \text{subject to} & 0\leq u_t\geq-\boldsymbol{w}^{\mathsf{T}}\boldsymbol{r}_t-\tau, & t=1,\ldots,T,\\ & \boldsymbol{w}\geq\mathbf{0},\quad \mathbf{1}^{\mathsf{T}}\boldsymbol{w}=1. & \end{array}$$

c. Observe how many observations are actually used ($u_t>0$) for the different values of α.
d. Add some small perturbation or noise to the sequence of returns $\boldsymbol{r}_1,\ldots,\boldsymbol{r}_T$ and repeat the experiment to observe the sensitivity of the solutions to data perturbation.

10.9 (Mean–Max-DD formulation as an LP) The mean–Max-DD formulation replaces the usual variance term $\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}$ by the Max-DD as a measure of risk, defined as

$$\text{Max-DD}(\boldsymbol{w})=\max_{1\leq t\leq T}D_t(\boldsymbol{w}),$$

where $D_t(\boldsymbol{w})$ is the drawdown at time t. This leads to the problem formulation

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu}-\lambda\max\limits_{1\leq t\leq T}\left\{\max\limits_{1\leq\tau\leq t}\boldsymbol{w}^{\mathsf{T}}\boldsymbol{r}_{\tau}^{\text{cum}}-\boldsymbol{w}^{\mathsf{T}}\boldsymbol{r}_{t}^{\text{cum}}\right\}\\ \text{subject to} & \boldsymbol{w}\in\mathcal{W}.\end{array}$$

Show that it can be rewritten as the following problem ($u_0\triangleq-\infty$):

$$\begin{array}{ll}\underset{\boldsymbol{w},\boldsymbol{u},s}{\text{maximize}} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu}-\lambda\, s\\ \text{subject to} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{r}_t^{\text{cum}}\leq u_t\leq s+\boldsymbol{w}^{\mathsf{T}}\boldsymbol{r}_t^{\text{cum}},\quad t=1,\ldots,T,\\ & u_{t-1}\leq u_t,\\ & \boldsymbol{w}\in\mathcal{W},\end{array}$$

which is a linear program (assuming $\mathcal{W}$ only contains linear constraints).

10.10 (Mean–Ave-DD formulation as an LP) The mean–Ave-DD formulation replaces the usual variance term $\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}$ by the Ave-DD as a measure of risk, defined as

$$\text{Ave-DD} = \frac{1}{T}\sum_{1\le t\le T} D_t(\boldsymbol{w}),$$

where $D_t(\boldsymbol{w})$ is the drawdown at time t. This leads to the problem formulation

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda\,\dfrac{1}{T}\displaystyle\sum_{t=1}^{T}\left(\max_{1\le\tau\le t}\ \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_\tau^{\text{cum}} - \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}}\right)\\ \text{subject to} & \boldsymbol{w}\in\mathcal{W}.\end{array}$$

Show that it can be rewritten as the following problem ($u_0 \triangleq -\infty$):

$$\begin{array}{ll}\underset{\boldsymbol{w},\boldsymbol{u},s}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda\, s\\ \text{subject to} & \frac{1}{T}\sum_{t=1}^{T} u_t \le \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}} + s,\\ & \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}} \le u_t, \qquad t=1,\ldots,T,\\ & u_{t-1}\le u_t,\\ & \boldsymbol{w}\in\mathcal{W},\end{array}$$

which is a linear program (assuming $\mathcal{W}$ only contains linear constraints).

10.11 (Mean–CVaR-DD formulation as an LP) The mean–CVaR-DD formulation replaces the usual variance term $\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}$ by the CVaR-DD as a measure of risk, expressed in a variational form as

$$\text{CVaR-DD}(\boldsymbol{w}) = \inf_{\tau}\left\{\tau + \frac{1}{1-\alpha}\frac{1}{T}\sum_{t=1}^{T}(D_t(\boldsymbol{w})-\tau)^+\right\},$$

where $D_t(\boldsymbol{w})$ is the drawdown at time t. This leads to the problem formulation

$$\begin{array}{ll}\underset{\boldsymbol{w},\tau}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda\left(\tau + \dfrac{1}{1-\alpha}\dfrac{1}{T}\displaystyle\sum_{t=1}^{T}\left(\max_{1\le\tau'\le t}\ \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_{\tau'}^{\text{cum}} - \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}} - \tau\right)^+\right)\\ \text{subject to} & \boldsymbol{w}\in\mathcal{W}.\end{array}$$

Show that it can be rewritten as the following problem ($u_0 \triangleq -\infty$):

$$\begin{array}{ll}\underset{\boldsymbol{w},\tau,s,\boldsymbol{z},\boldsymbol{u}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \lambda\, s\\ \text{subject to} & s \ge \tau + \frac{1}{1-\alpha}\frac{1}{T}\sum_{t=1}^{T} z_t,\\ & 0\le z_t \ge u_t - \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}} - \tau, \qquad t=1,\ldots,T,\\ & \boldsymbol{w}^\mathsf{T}\boldsymbol{r}_t^{\text{cum}} \le u_t,\\ & u_{t-1}\le u_t,\\ & \boldsymbol{w}\in\mathcal{W},\end{array}$$

which is a linear program (assuming $\mathcal{W}$ only contains linear constraints).

10.12 (Mean–EVaR-DD formulation as a convex problem) The mean–EVaR-DD formulation replaces the usual variance term $\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}$ by the EVaR-DD as a measure of risk, defined as

$$\text{EVaR-DD}(\boldsymbol{w}) = \inf_{z>0}\left\{z^{-1}\log\left(\frac{1}{1-\alpha}\frac{1}{T}\sum_{t=1}^{T}\exp(zD_t(\boldsymbol{w}))\right)\right\},$$

where $D_t(\boldsymbol{w})$ is the drawdown at time t defined as

$$D_t(\boldsymbol{w}) = \max_{1 \le \tau \le t} \boldsymbol{w}^\mathsf{T} \boldsymbol{r}_\tau^{\text{cum}} - \boldsymbol{w}^\mathsf{T} \boldsymbol{r}_t^{\text{cum}}.$$

a. Write down the mean–EVaR-DD portfolio formulation in convex form.
b. Further rewrite the problem in terms of the exponential cone.

References

Ahmadi-Javid, A. (2012). Entropic value-at-risk: A new coherent risk measure. *Journal of Optimization Theory and Applications*, *155*(3), 1105–1123.

Ahmadi-Javid, A., & Fallah-Tafti, M. (2019). Portfolio optimization with entropic value-at-risk. *European Journal of Operational Research*, *279*(1), 225–241.

Ang, A., Chen, J., & Xing, Y. (2006). Downside risk. *Review of Financial Studies*, *19*(4), 1191–1239.

Artzner, P., Delbaen, F., Eber, J. M., & Heath, D. (1999). Coherent measures of risk. *Mathematical Finance*, *9*(3), 203–227.

Bawa, V. (1975). Optimal rules for ordering uncertain prospects. *Journal of Financial Economics*, *2*(1), 95–121.

Cajas, D. (2021). Entropic portfolio optimization: A disciplined convex programming framework. *SSRN Electronic Journal*. https://dx.doi.org/10.2139/ssrn.3792520

Chares, R. (2007). *Cones and Interior-point Algorithms for Structured Convex Optimization Involving Powers and Exponentials* [Doctoral dissertation, Université Catholique de Louvain and École Polytechnique de Louvain].

Chekhlov, A., Uryasev, S., & Zabarankin, M. (2004). Portfolio optimization with drawdown constraints. In P. M. Pardalos, A. Migdalas, & G. Baourakis (Eds.), *Supply Chain And Finance* (pp. 209–228). World Scientific.

Chekhlov, A., Uryasev, S., & Zabarankin, M. (2005). Drawdown measure in portfolio optimization. *International Journal of Theoretical and Applied Finance*, *8*(1), 13–58.

Cvitanic, J., & Karatzas, I. (1995). On portfolio optimization under "drawdown" constraints. *IMA Lecture Notes in Mathematics & Applications*, *65*, 77–88.

Estrada, J. (2006). Downside risk in practice. *Journal of Applied Corporate Finance*, *18*(1), 117–125.

Estrada, J. (2008). Mean–semivariance optimization: A heuristic approach. *Journal of Applied Finance*, *18*(1), 57–72.

Fishburn, P. C. (1977). Mean-risk analysis with risk associated with below-target returns. *American Economic Review*, *67*(2), 116–126.

Grootveld, H., & Hallerbach, W. (1999). Variance vs downside risk: Is there really that much difference? *European Journal of Operational Research*, *114*, 304–319.

Grossman, S. J., & Zhou, Z. (1993). Optimal investment strategies for controlling drawdowns. *Mathematical Finance*, *3*(3), 241–276.

Hong, L. J., Hu, Z., & Liu, G. (2014). Monte Carlo methods for value-at-risk and conditional value-at-risk: A review. *ACM Transactions on Modeling and Computer Simulation*, *24*(4), 1–37.

Markowitz, H. M. (1952). Portfolio selection. *The Journal of Finance*, *7*(1), 77–91.

Markowitz, H. M. (1959). *Portfolio Selection: Efficient Diversification of Investments*. John Wiley & Sons.

McNeil, A. J., & Frey, R. (2000). Estimation of tail-related risk measures for heteroscedastic financial time series: An extreme value approach. *Journal of Empirical Finance*, *7*(3–4), 271–300.

McNeil, A. J., Frey, R., & Embrechts, P. (2015). *Quantitative Risk Management*. Princeton University Press.

Nadarajah, S., Zhang, B., & Chan, S. (2014). Estimation methods for expected shortfall. *Quantitative Finance*, *14*(2), 271–291.

Rockafellar, R. T., & Uryasev, S. (2000). Optimization of conditional value-at-risk. *Journal of Risk*, *2*(3), 21–42.

Rockafellar, R. T., & Uryasev, S. (2002). Conditional value-at-risk for general loss distributions. *Journal of Banking and Finance*, *26*(7), 1443–1471.

Roy, A. (1952). Safety first and the holding of assets. *Econometrica*, *20*(3), 431–449.

Young, M. R. (1998). A minimax portfolio selection rule with linear programming solution. *Management Science*, *44*(5), 673–683.

Zhu, S., & Fukushima, M. (2009). Worst-case conditional value-at-risk with application to robust portfolio management. *Operations Research*, *57*(5), 1155–1168.

11

Risk Parity Portfolios

"To dare the impossible is no mark of a wise man."

— Euripides, *Helen*

Markowitz's mean–variance portfolio optimizes a trade-off between expected return and risk measured by the variance. Alternative measures of risk to the variance or volatility can certainly be entertained.

However, measuring the risk of the portfolio with a single number provides a limited view. Instead, a more refined characterization comes from employing a risk profile that quantifies the amount of risk contributed by each constituent asset. This refined risk characterization allows a proper control of the portfolio risk diversification and will be explored in this chapter.

11.1 Introduction

Markowitz's mean–variance portfolio (Markowitz, 1952) formulates the portfolio design as a trade-off between the expected return $\boldsymbol{w}^T\boldsymbol{\mu}$ and the risk measured by the variance $\boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w}$ (see Chapter 7 for details):

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^T\boldsymbol{\mu} - \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w} \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array}$$

where λ is a hyper-parameter that controls the investor's risk aversion and $\mathcal{W}$ denotes an arbitrary constraint set, such as $\mathcal{W} = \{\boldsymbol{w} \mid \mathbf{1}^T\boldsymbol{w} = 1, \boldsymbol{w} \geq \mathbf{0}\}$.

Nevertheless, it has been well recognized over decades of research and experimentation that measuring the portfolio risk with the variance $\boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w}$ or, similarly, the volatility $\sqrt{\boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w}}$ may not be the best choice for out-of-sample performance. One way to address this drawback is to use alternative risk measures as explored in Chapter 10. On top of that, one can add another layer of sophistication by characterizing the risk of the portfolio not just with a single number but with a risk profile that quantifies the amount of risk contributed by each constituent asset. This refined risk characterization allows for proper control of the portfolio risk diversification (Litterman, 1996; Qian, 2005, 2016; Roncalli, 2013b; Tasche, 2008).

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

This chapter introduces the risk parity portfolio from its simplest form (with a closed-form solution), through vanilla convex formulations, to the more general nonconvex formulations, providing a wide range of numerical algorithms (Feng & Palomar, 2015, 2016; Maillard et al., 2010).

11.2 From Dollar to Risk Diversification

Risk parity is an approach to investment management that focuses on allocation of risk rather than allocation of dollars or capital. In other words, it departs from the concept of dollar diversification to risk diversification. The underlying idea is to design the asset allocation so that each asset (or asset class) contributes the same risk level to the overall portfolio. This typically produces better out-of-sample risk control and can be more resistant to market downturns than traditional portfolios.

Historically, risk parity starts from the observation that traditional asset allocations, such as the market portfolio or the 60/40 portfolio in stocks/bonds, are insufficiently diversified in terms of risk contribution (Asness et al., 2012). From a risk perspective, the 60/40 portfolio is mainly equity portfolio since stocks are much more volatile than bonds and dominate the risk of the entire portfolio.

Some of the theoretical components were developed in the 1950s and 1960s, but the first risk parity fund, called the "All Weather" fund, was pioneered by Bridgewater Associates in 1996. An early analysis in terms of partial derivatives to identify "hot spots" can be found in Litterman (1996). The term "risk parity" was coined in 2005 by Edward Qian[1] (Qian, 2005) and, in recent years, it has been adopted by the asset management industry as a way of risk management. Risk parity gained significantly in popularity after the global financial crisis in 2008 (Asness et al., 2012; Maillard et al., 2010). Nevertheless, some portfolio managers have expressed skepticism about its practical application and effectiveness in all types of market conditions, especially bull markets (Anderson et al., 2012; Chaves et al., 2011).

The risk parity portfolio has received significant attention from both practitioner and academic communities, producing a large number of publications and some textbooks such as Qian (2016) for a high-level and practical perspective and Roncalli (2013b) for a more mathematical treatment.

Figure 11.1 illustrates the difference between dollar diversification and risk diversification. The $1/N$ portfolio by definition perfectly diversifies the capital allocation (i.e., portfolio weights) but does not show a good risk diversification profile (in fact, a single asset exhibits a high risk contribution). On the other hand, the risk parity portfolio precisely focuses on the diversification of the risk profile.

[1] Dr. Edward Qian is the Chief Investment Officer and Head of Research of the Multi Asset Group at PanAgora Asset Management.

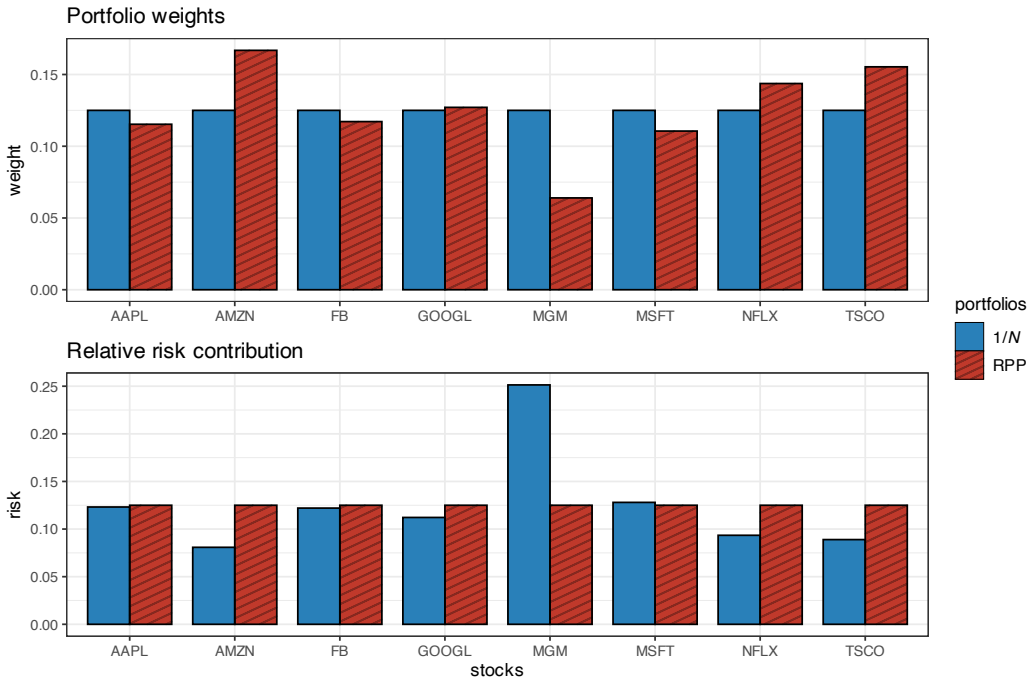


Figure 11.1 Illustration of portfolio allocation and risk allocation for the $1/N$ portfolio and risk parity portfolio.

11.3 Risk Contributions

The whole idea of the risk parity portfolio hinges on quantifying the decomposition of the portfolio risk into the sum of risk contributions from the individual assets:

$$\text{portfolio risk} = \sum_{i=1}^{N} \text{RC}_i,$$

where RC_i denotes the *risk contribution* (RC) of the ith asset to the total risk. The choice of risk measure depends on the portfolio designer, with common options being volatility, value-at-risk (VaR), or conditional VaR (CVaR). For a list of performance measures, refer to Section 6.3, and for a detailed discussion on portfolio design based on alternative performance measures, see Chapter 10. Euler's theorem offers the solution for the desired decomposition of portfolio risk, cf. Litterman (1996), Tasche (2008), and references therein.

Theorem 11.1 (Euler's homogenous function theorem) *Let a continuous and differentiable function $f : \mathbb{R}^N \to \mathbb{R}$ be a positively homogeneous function of degree one.*[2] *Then*

$$f(\boldsymbol{w}) = \sum_{i=1}^{N} w_i \frac{\partial f}{\partial w_i}. \tag{11.1}$$

[2] A function $f(\boldsymbol{w})$ is a positively homogeneous function of degree one if $f(c\boldsymbol{w}) = cf(\boldsymbol{w})$ holds for any $c > 0$. This condition is satisfied by the volatility, VaR, and CVaR, among others (but not by the variance, for example).

From Theorem 11.1, for a given risk measure $f(\boldsymbol{w})$, we can interpret the risk contribution from the ith asset as

$$\mathrm{RC}_i = w_i \frac{\partial f(\boldsymbol{w})}{\partial w_i}.$$

In addition, it is convenient to define the related *marginal risk contribution* (MRC) as

$$\mathrm{MRC}_i = \frac{\partial f(\boldsymbol{w})}{\partial w_i},$$

which evaluates the portfolio risk's sensitivity with respect to the ith asset weight, and the *relative risk contribution* (RRC) as

$$\mathrm{RRC}_i = \frac{\mathrm{RC}_i}{f(\boldsymbol{w})},$$

which satisfies $\sum_{i=1}^{N} \mathrm{RRC}_i = 1$.

The following measures of risk do satisfy Euler's requirement in Theorem 11.1 and the decomposition in (11.1) can be employed.

- For the volatility, $\sigma(\boldsymbol{w}) = \sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}$, Euler's theorem can be used with

$$\mathrm{RC}_i = w_i \frac{\partial \sigma(\boldsymbol{w})}{\partial w_i} = \frac{w_i (\boldsymbol{\Sigma}\boldsymbol{w})_i}{\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}}.$$

- For the VaR, defined as $\mathrm{VaR}_\alpha(\boldsymbol{w}) = \inf\left\{\xi_0 \mid \Pr(-\boldsymbol{w}^\top \boldsymbol{r} \leq \xi_0) \geq \alpha\right\}$ with α the confidence level (e.g., $\alpha = 0.95$), $\boldsymbol{r}$ representing the random market return vector, and $-\boldsymbol{w}^\top \boldsymbol{r}$ denoting the (random) loss of portfolio $\boldsymbol{w}$, it follows (Hallerbach, 2003) that

$$\mathrm{RC}_i = w_i \frac{\partial \mathrm{VaR}_\alpha(\boldsymbol{w})}{\partial w_i} = \mathbb{E}\left[-w_i r_i \mid -\boldsymbol{w}^\top \boldsymbol{r} = \mathrm{VaR}_\alpha(\boldsymbol{w})\right].$$

- For the CVaR, defined as $\mathrm{CVaR}_\alpha(\boldsymbol{w}) = \mathbb{E}\left[-\boldsymbol{w}^\top \boldsymbol{r} \mid -\boldsymbol{w}^\top \boldsymbol{r} \geq \mathrm{VaR}_\alpha(\boldsymbol{w})\right]$, it follows (Scaillet, 2004) that

$$\mathrm{RC}_i = w_i \frac{\partial \mathrm{CVaR}_\alpha(\boldsymbol{w})}{\partial w_i} = \mathbb{E}\left[-w_i r_i \mid -\boldsymbol{w}^\top \boldsymbol{r} \geq \mathrm{VaR}_\alpha(\boldsymbol{w})\right].$$

In practice, the VaR and CVaR risk contribution expressions are not easily computable. Interestingly, if the returns $\boldsymbol{r}$ follow a Gaussian distribution, then the VaR and CVaR can be expressed explicitly as (McNeil et al., 2015)

$$\begin{aligned}
\mathrm{VaR}_\alpha(\boldsymbol{w}) &= -\boldsymbol{w}^\top \boldsymbol{\mu} + \kappa_1(\alpha)\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}, \\
\mathrm{CVaR}_\alpha(\boldsymbol{w}) &= -\boldsymbol{w}^\top \boldsymbol{\mu} + \kappa_2(\alpha)\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}},
\end{aligned}$$

where $\kappa_1(\alpha) \triangleq \Phi^{-1}(\alpha)$, $\kappa_2(\alpha) \triangleq \frac{1}{(1-\alpha)} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\Phi^{-1}(\alpha)^2\right)$, and $\Phi(\cdot)$ is the cumulative distribution function of a zero-mean unit-variance Gaussian variable.

11.3.1 Volatility Risk Contributions

In the rest of the chapter, we will focus on the portfolio volatility, $\sigma(\boldsymbol{w}) = \sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}$, which admits the decomposition

$$\sigma(\boldsymbol{w}) = \sum_{i=1}^{N} w_i \frac{\partial \sigma}{\partial w_i} = \sum_{i=1}^{N} \frac{w_i (\boldsymbol{\Sigma} \boldsymbol{w})_i}{\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}},$$

leading to the following expressions:

$$\text{MRC}_i = \frac{(\boldsymbol{\Sigma} \boldsymbol{w})_i}{\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}}, \qquad \text{RC}_i = \frac{w_i (\boldsymbol{\Sigma} \boldsymbol{w})_i}{\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}}, \qquad \text{RRC}_i = \frac{w_i (\boldsymbol{\Sigma} \boldsymbol{w})_i}{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}.$$

11.4 Problem Formulation

The *risk parity portfolio* (RPP), also termed *equal risk portfolio* (ERP), is simply formulated as requiring the risk contributions to be equal (i.e., equalizing the individual risks):

$$\text{RRC}_i = \frac{w_i (\boldsymbol{\Sigma} \boldsymbol{w})_i}{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}} = \frac{1}{N}, \qquad i = 1, \ldots, N.$$

This is reminiscent of the $1/N$ portfolio or equally weighted portfolio (EWP) that satisfies:

$$w_i = \frac{1}{N}, \qquad i = 1, \ldots, N.$$

Thus, whereas the $1/N$ portfolio equalizes the dollar allocation, the RPP equalizes the risk contribution. Interestingly, if all the assets have roughly the same Sharpe ratios and same correlations, the RPP can be interpreted as optimal under the Markowitz mean–variance framework (Maillard et al., 2010). In addition, it can be shown that the RPP exists, is unique, and is located between the minimum variance and equally weighted portfolios (Maillard et al., 2010).

More generally, one can specify a risk profile allocation different from the uniform one, termed the *risk budgeting portfolio* (RBP):

$$\text{RRC}_i = \frac{w_i (\boldsymbol{\Sigma} \boldsymbol{w})_i}{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}} = b_i, \qquad i = 1, \ldots, N,$$

where $\boldsymbol{b} = (b_1, \ldots, b_N) \geq \boldsymbol{0}$ (normalized to $\boldsymbol{1}^\top \boldsymbol{b} = 1$) is the desired risk profile.

Thus, in its simplest form, the problem can be formulated as finding a portfolio $\boldsymbol{w} \geq \boldsymbol{0}$, with $\boldsymbol{1}^\top \boldsymbol{w} = 1$, satisfying the risk budgeting constraints

$$w_i (\boldsymbol{\Sigma} \boldsymbol{w})_i = b_i \, \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}, \qquad i = 1, \ldots, N.$$

This is a feasibility problem (there is no objective to maximize or minimize, just constraints), whose solution is not trivial. In the rest of the chapter we will consider the resolution of this problem by exploring three cases in order of difficulty:[3]

- naive diagonal formulation;

[3] The R package `riskParityPortfolio` (Cardoso & Palomar, 2021) and the Python package `riskparity.py` can solve these formulations very efficiently.

- vanilla convex formulation; and
- general nonconvex formulation.

11.4.1 Formulation with Shorting

Typically, the RPP formulation involves no shorting, so $\boldsymbol{w} \geq \boldsymbol{0}$. Allowing shorting generally requires more complicated resolution methods. Nevertheless, in the case that the shorting pattern is known a priori, the problem can be easily reformulated like the no-shorting formulation with the following trick (Spinu, 2013).

Suppose that the shorting pattern, i.e., which assets to long or short, is defined a priori in the vector $\boldsymbol{s} = (s_1, \ldots, s_N)$ with $s_i = 1$ for long positions and $s_i = -1$ for short positions. Then the desired portfolio $\boldsymbol{w}$, which follows the shorting pattern, can be related to a virtual no-shorting portfolio $\tilde{\boldsymbol{w}} \geq \boldsymbol{0}$ as $\boldsymbol{w} = \boldsymbol{s} \odot \tilde{\boldsymbol{w}}$ such that $\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w} = \tilde{\boldsymbol{w}}^\mathsf{T}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{w}}$, where $\tilde{\boldsymbol{\Sigma}} = \mathrm{Diag}(\boldsymbol{s})\boldsymbol{\Sigma}\mathrm{Diag}(\boldsymbol{s})$. Then, the risk budgeting equations become

$$\tilde{w}_i(\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{w}})_i = b_i\, \tilde{\boldsymbol{w}}^\mathsf{T}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{w}}, \qquad i = 1, \ldots, N.$$

11.4.2 Formulation with Group Risk Parity

The idea of group risk parity is to consider the risk contributions of several assets belonging to the same group (e.g., industry or sector) as a whole. For example, suppose there are K groups (with $K < N$), denoted $\mathcal{G}_1, \ldots, \mathcal{G}_K$, such that they form a partition of the N assets. We can define the risk contribution from the kth group as

$$\mathrm{RC}_{\mathcal{G}_k} = \sum_{i \in \mathcal{G}_k} w_i \frac{\partial \sigma}{\partial w_i} = \sum_{i \in \mathcal{G}_k} \frac{w_i(\boldsymbol{\Sigma}\boldsymbol{w})_i}{\sqrt{\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}}}$$

and then the risk budgeting equations become

$$\sum_{i \in \mathcal{G}_k} w_i(\boldsymbol{\Sigma}\boldsymbol{w})_i = b_k\, \boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}, \qquad i = 1, \ldots, K.$$

11.4.3 Formulation with Risk Factors

Consider the following factor modeling of the returns:

$$\boldsymbol{r}_t = \boldsymbol{\alpha} + \boldsymbol{B}\boldsymbol{f}_t + \boldsymbol{\epsilon}_t,$$

where $\boldsymbol{f}_t$ contains the K factors (typically with $K \ll N$), $\boldsymbol{\alpha}$ is the so-called "alpha," matrix $\boldsymbol{B}$ contains the so-called "betas" on the columns corresponding to the different factors, and $\boldsymbol{\epsilon}_t$ is the residual.

The risk contribution from the kth factor can be defined (Roncalli & Weisang, 2016) as

$$\mathrm{RC}_k = \frac{(\boldsymbol{B}^\mathsf{T}\boldsymbol{w})_k(\boldsymbol{B}^\dagger\boldsymbol{\Sigma}\boldsymbol{w})_k}{\sqrt{\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}}},$$

where $\boldsymbol{B}^{\dagger} \triangleq \boldsymbol{B}^{\mathsf{T}}(\boldsymbol{B}^{\mathsf{T}}\boldsymbol{B})^{-1}$ is the Moore–Penrose pseudo-inverse matrix of $\boldsymbol{B}$, and the risk budgeting equations become

$$(\boldsymbol{B}^{\mathsf{T}}\boldsymbol{w})_k(\boldsymbol{B}^{\dagger}\boldsymbol{\Sigma}\boldsymbol{w})_k = b_k\, \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}, \qquad i = 1, \ldots, K.$$

11.5 Naive Diagonal Formulation

Consider the risk budgeting equations

$$w_i\, (\boldsymbol{\Sigma}\boldsymbol{w})_i = b_i\, \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}, \qquad i = 1, \ldots, N.$$

If we assume that the covariance matrix is diagonal, $\boldsymbol{\Sigma} = \mathrm{Diag}(\boldsymbol{\sigma}^2)$, then they simplify to

$$w_i^2\sigma_i^2 = b_i \sum_{j=1}^{N} w_j^2\sigma_j^2, \qquad i = 1, \ldots, N$$

or, equivalently,

$$w_i = \frac{\sqrt{b_i}}{\sigma_i}\sqrt{\sum_{j=1}^{N} w_j^2\sigma_j^2}, \qquad i = 1, \ldots, N. \tag{11.2}$$

Observe that this portfolio is inversely proportional to the assets' volatilities, which corresponds to the inverse volatility portfolio (IVolP) introduced in Section 6.5.2 of Chapter 6. Lower weights are given to high-volatility assets and higher weights to low-volatility assets, resulting in weighted constituent assets with equal volatility (for the case $b_i = 1/N$):

$$\mathrm{Std}(w_i x_i) = w_i\sigma_i = 1/N,$$

where $\mathrm{Std}(\cdot)$ denotes standard deviation.

For a general nondiagonal covariance matrix, a closed-form solution is not available and some optimization procedure has to be employed. Nevertheless, the diagonal solution in (11.2) can always be used as a "naive" solution. Interestingly, the diagonal solution is also optimal if the correlations of all assets are the same (Maillard et al., 2010).

11.5.1 Illustrative Example

Figure 11.2 shows the portfolio allocation and the risk contribution of the naive RPP and the $1/N$ portfolio. The $1/N$ portfolio has an equal capital allocation by definition, but it shows an unequal risk contribution with much more risk from the single-asset "MGM." The naive RPP does a much better job equalizing the risk contribution, but it is not perfectly equalized since it ignores the off-diagonal elements of the covariance matrix.

11.6 Vanilla Convex Formulations

Consider the risk budgeting equations

$$w_i\, (\boldsymbol{\Sigma}\boldsymbol{w})_i = b_i\, \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}, \qquad i = 1, \ldots, N$$

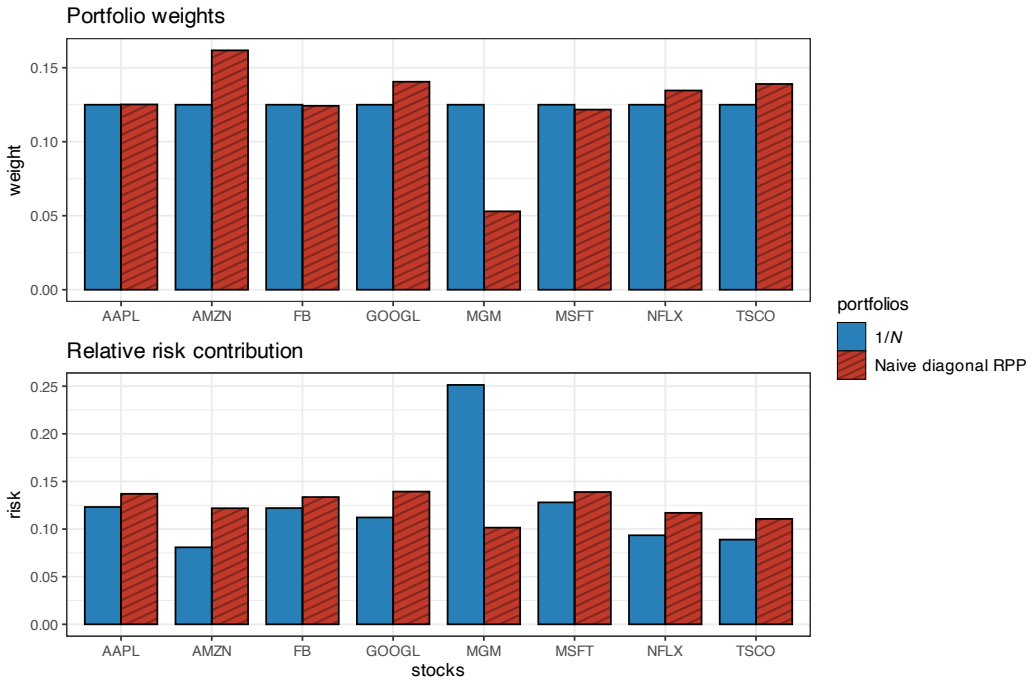


Figure 11.2 Portfolio allocation and risk contribution of the $1/N$ portfolio and naive diagonal RPP.

with $\mathbf{1}^\top \boldsymbol{w} = 1$ and $\boldsymbol{w} \geq \mathbf{0}$. If we define $\boldsymbol{x} = \boldsymbol{w}/\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}$, then we can rewrite them as $x_i \, (\boldsymbol{\Sigma} \boldsymbol{x})_i = b_i$ or, more compactly in vector form, as

$$\boldsymbol{\Sigma} \boldsymbol{x} = \boldsymbol{b}/\boldsymbol{x} \tag{11.3}$$

with $\boldsymbol{x} \geq \mathbf{0}$, from which we can recover the portfolio simply by normalizing $\boldsymbol{x}$ as $\boldsymbol{w} = \boldsymbol{x}/(\mathbf{1}^\top \boldsymbol{x})$.

Interestingly, the equations in (11.3) can be rewritten in terms of the correlation matrix $\boldsymbol{C} = \boldsymbol{D}^{-1/2} \boldsymbol{\Sigma} \boldsymbol{D}^{-1/2}$, with $\boldsymbol{D}$ a diagonal matrix containing the variances $\text{diag}(\boldsymbol{\Sigma}) = \boldsymbol{\sigma}^2$ along the main diagonal, as (Spinu, 2013)

$$\boldsymbol{C} \tilde{\boldsymbol{x}} = \boldsymbol{b}/\tilde{\boldsymbol{x}}, \tag{11.4}$$

where $\boldsymbol{x} = \tilde{\boldsymbol{x}}/\boldsymbol{\sigma}$. This has the effect of normalizing the returns with respect to the volatilities of the assets, which can help numerically as will be empirically verified later.

11.6.1 Direct Resolution via Root Finding

The system of nonlinear equations $\boldsymbol{\Sigma} \boldsymbol{x} = \boldsymbol{b}/\boldsymbol{x}$ can be interpreted as finding the roots of the nonlinear function

$$F(\boldsymbol{x}) = \boldsymbol{\Sigma} \boldsymbol{x} - \boldsymbol{b}/\boldsymbol{x},$$

that is, solving for

$$F(\boldsymbol{x}) = \mathbf{0}.$$

In practice, this can be done with a general-purpose nonlinear multivariate root finder, available in most programming languages.[4]

A similar root-finding problem can be formulated in terms of $\boldsymbol{w}$ by including the budget constraint $\mathbf{1}^\top \boldsymbol{w} = 1$ explicitly in the nonlinear function (Chaves et al., 2012):

$$F(\boldsymbol{w}, \lambda) = \begin{bmatrix} \boldsymbol{\Sigma w} - \lambda \boldsymbol{b}/\boldsymbol{w} \\ \mathbf{1}^\top \boldsymbol{w} - 1 \end{bmatrix}.$$

11.6.2 Formulations

A more interesting way to solve the risk budgeting equation in (11.3) is by unveiling the hidden convexity. This can be achieved by realizing that (11.3) precisely corresponds to the optimality conditions of some carefully chosen convex optimization problems, as is the case with the following three formulations.

- To start, consider the following convex optimization problem (Spinu, 2013):

$$\underset{\boldsymbol{x} \geq \mathbf{0}}{\text{minimize}} \quad \tfrac{1}{2} \boldsymbol{x}^\top \boldsymbol{\Sigma x} - \boldsymbol{b}^\top \log(\boldsymbol{x}). \tag{11.5}$$

  It turns out that setting the gradient of the objective function to zero precisely corresponds to (11.3):

$$\boldsymbol{\Sigma x} = \boldsymbol{b}/\boldsymbol{x}.$$

  Therefore, solving this problem is equivalent to solving the risk budgeting equations.

- A slightly different convex optimization problem is (Roncalli, 2013b)

$$\underset{\boldsymbol{x} \geq \mathbf{0}}{\text{minimize}} \quad \sqrt{\boldsymbol{x}^\top \boldsymbol{\Sigma x}} - \boldsymbol{b}^\top \log(\boldsymbol{x}). \tag{11.6}$$

  In this case, setting the gradient to zero leads to

$$\boldsymbol{\Sigma x} = \boldsymbol{b}/\boldsymbol{x} \times \sqrt{\boldsymbol{x}^\top \boldsymbol{\Sigma x}},$$

  which follows the form of the risk budgeting equation (11.3) after renormalizing the variable as $\tilde{\boldsymbol{x}} = \boldsymbol{x}/(\boldsymbol{x}^\top \boldsymbol{\Sigma x})^{1/4}$.

- Similarly, the following formulation was proposed in Maillard et al. (2010):

$$\begin{array}{ll} \underset{\boldsymbol{x} \geq \mathbf{0}}{\text{minimize}} & \sqrt{\boldsymbol{x}^\top \boldsymbol{\Sigma x}} \\ \text{subject to} & \boldsymbol{b}^\top \log(\boldsymbol{x}) \geq c, \end{array} \tag{11.7}$$

  with c an arbitrary constant. This formulation provides an alternative interpretation of the problem as minimizing the volatility (or variance) with the constraint on $\boldsymbol{b}^\top \log(\boldsymbol{x})$ controlling the diversification.

[4] In R, a general-purpose nonlinear multivariate root finder is provided by the function `multiroot()` from the package `rootSolve`. In MATLAB, we can use the function `fsolve()`.

- Another reformulation swapping the objective and constraint in (11.7) was proposed by Kaya and Lee (2012) as

$$\begin{array}{ll} \underset{\boldsymbol{x}\geq\mathbf{0}}{\text{maximize}} & \boldsymbol{b}^{\mathsf{T}}\log(\boldsymbol{x}) \\ \text{subject to} & \sqrt{\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{x}} \leq \sigma_0, \end{array} \tag{11.8}$$

  where the volatility term appears as a constraint and σ_0 is the chosen maximum level of volatility. In this case, we can define the Lagrangian (ignoring the nonnegativity constraint on $\boldsymbol{x}$ since it is automatically satisfied) as

$$L(\boldsymbol{x};\lambda) = \boldsymbol{b}^{\mathsf{T}}\log(\boldsymbol{x}) + \lambda\left(\sigma_0 - \sqrt{\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{x}}\right)$$

  and setting its gradient to zero leads to

$$\lambda\boldsymbol{\Sigma}\boldsymbol{x} = \boldsymbol{b}/\boldsymbol{x} \times \sqrt{\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{x}},$$

  which follows the form of the risk budgeting equation (11.3) after renormalizing the variable as $\tilde{\boldsymbol{x}} = \boldsymbol{x} \times \lambda^{1/2}/(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{x})^{1/4}$.

All these formulations are convex and can be solved with a general-purpose solver available in any programming language.[5] Alternatively, simple and efficient tailored algorithms can be developed.

Illustrative Example

Figure 11.3 shows the distribution of the portfolio weights and risk contribution for the $1/N$ portfolio, the naive diagonal RPP, and the vanilla convex RPP. The latter clearly achieves perfect risk equalization as opposed to the naive diagonal case.

11.6.3 Algorithms

As an alternative to using a general-purpose solver, we will now develop several practical iterative algorithms tailored to the problem formulations in (11.5) and (11.6) that produce a sequence of iterates $\boldsymbol{x}^0, \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots$

As the initial point of these iterative algorithms, we can use several options that attempt to approximately solve the system of nonlinear equations $\boldsymbol{\Sigma}\boldsymbol{x} = \boldsymbol{b}/\boldsymbol{x}$:

- *Naive diagonal solution*: Inspired by the diagonal case $\boldsymbol{\Sigma} = \text{Diag}(\boldsymbol{\sigma}^2)$, we can simply use

$$\boldsymbol{x}^0 = \sqrt{\boldsymbol{b}}/\boldsymbol{\sigma}.$$

- *Scaled heuristic* for (11.5): For any given point $\bar{\boldsymbol{x}}$, we can always find a more appropriate scaling factor t such that $\boldsymbol{x} = t\bar{\boldsymbol{x}}$ satisfies the sum of the nonlinear equations $\mathbf{1}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{x} = \mathbf{1}^{\mathsf{T}}(\boldsymbol{b}/\boldsymbol{x})$, leading to $t = \sqrt{\mathbf{1}^{\mathsf{T}}(\boldsymbol{b}/\bar{\boldsymbol{x}})/\mathbf{1}^{\mathsf{T}}\boldsymbol{\Sigma}\bar{\boldsymbol{x}}}$ and then

$$\boldsymbol{x}^0 = \bar{\boldsymbol{x}} \times \sqrt{\frac{\mathbf{1}^{\mathsf{T}}(\boldsymbol{b}/\bar{\boldsymbol{x}})}{\mathbf{1}^{\mathsf{T}}\boldsymbol{\Sigma}\bar{\boldsymbol{x}}}}.$$

[5] In R, there is a myriad of solvers, a typical example being the base function `optim()`. In MATLAB, a similar function is `fmincon()`.

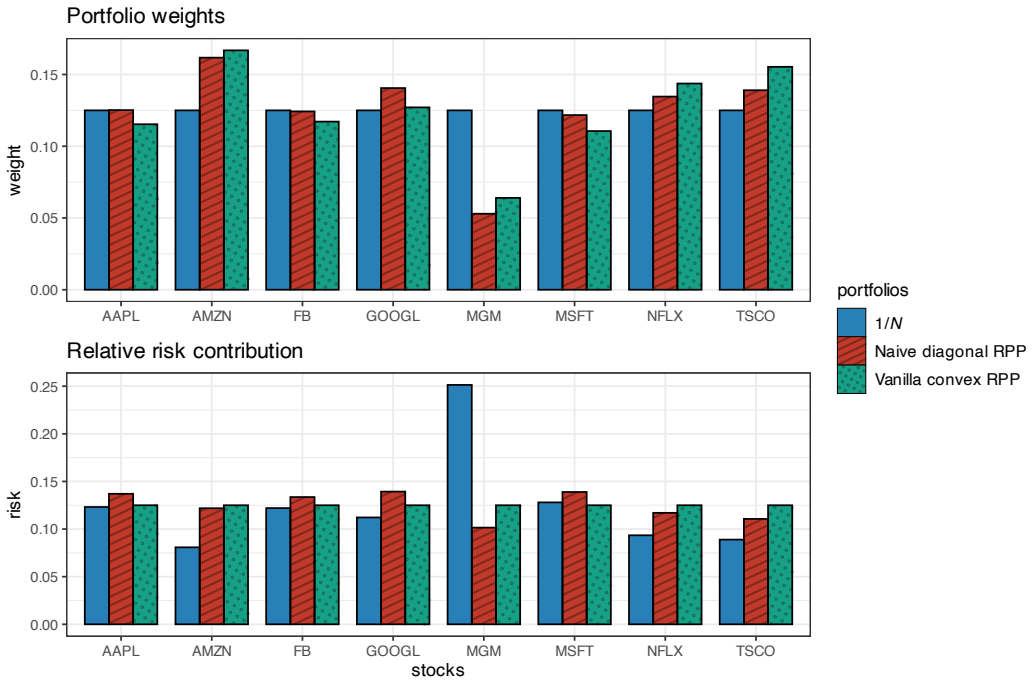


Figure 11.3 Portfolio allocation and risk contribution of the vanilla convex RPP compared to benchmarks ($1/N$ portfolio and naive diagonal RPP).

- *Scaled heuristic* for (11.6): For any given point $\bar{\boldsymbol{x}}$, we can always find a more appropriate scaling factor t such that $\boldsymbol{x} = t\bar{\boldsymbol{x}}$ satisfies the sum of the nonlinear equations $\mathbf{1}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x} = \mathbf{1}^\mathsf{T}(\boldsymbol{b}/\boldsymbol{x}) \times \sqrt{\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x}}$, leading to

$$\boldsymbol{x}^0 = \bar{\boldsymbol{x}} \times \frac{\mathbf{1}^\mathsf{T}(\boldsymbol{b}/\bar{\boldsymbol{x}})}{\mathbf{1}^\mathsf{T}\boldsymbol{\Sigma}\bar{\boldsymbol{x}}}\sqrt{\bar{\boldsymbol{x}}^\mathsf{T}\boldsymbol{\Sigma}\bar{\boldsymbol{x}}}.$$

<!—

- *scaled naive diagonal solution*: the combination of the naive diagonal solution with the improved scaling factor gives

$$\boldsymbol{x}^0 = \sqrt{\boldsymbol{b}}/\boldsymbol{\sigma} \times \sqrt{\boldsymbol{\sigma}^\mathsf{T}\sqrt{\boldsymbol{b}}/\mathbf{1}^\mathsf{T}\boldsymbol{\Sigma}(\sqrt{\boldsymbol{b}}/\boldsymbol{\sigma})};$$

—>

- *Diagonal row-sum heuristic*: Using $\mathrm{Diag}(\boldsymbol{\Sigma}\mathbf{1})$ in lieu of $\boldsymbol{\Sigma}$ leads to

$$\boldsymbol{x}^0 = \sqrt{\boldsymbol{b}}/\sqrt{\boldsymbol{\Sigma}\mathbf{1}}.$$

Newton's Method

Newton's method obtains the iterates based on the gradient ∇f and the Hessian $\mathsf{H}f$ of the objective function $f(\boldsymbol{x})$ as follows:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \mathsf{H}f(\boldsymbol{x}^k)^{-1}\nabla f(\boldsymbol{x}^k).$$

For details on Newton's method, the reader is referred to Section B.2 in Appendix B. The specific application of Newton's method to risk parity portfolio was studied in detail in Spinu (2013).

In our case, the gradient and Hessian are easily computed as follows:

- For the function $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top\boldsymbol{\Sigma}\boldsymbol{x} - \boldsymbol{b}^\top \log(\boldsymbol{x})$ in (11.5):

$$\begin{aligned}\nabla f(\boldsymbol{x}) &= \boldsymbol{\Sigma}\boldsymbol{x} - \boldsymbol{b}/\boldsymbol{x},\\ \mathsf{H} f(\boldsymbol{x}) &= \boldsymbol{\Sigma} + \mathrm{Diag}(\boldsymbol{b}/\boldsymbol{x}^2).\end{aligned}$$

- For the function $f(\boldsymbol{x}) = \sqrt{\boldsymbol{x}^\top\boldsymbol{\Sigma}\boldsymbol{x}} - \boldsymbol{b}^\top \log(\boldsymbol{x})$ in (11.6):

$$\begin{aligned}\nabla f(\boldsymbol{x}) &= \boldsymbol{\Sigma}\boldsymbol{x}/\sqrt{\boldsymbol{x}^\top\boldsymbol{\Sigma}\boldsymbol{x}} - \boldsymbol{b}/\boldsymbol{x},\\ \mathsf{H} f(\boldsymbol{x}) &= \left(\boldsymbol{\Sigma} - \boldsymbol{\Sigma}\boldsymbol{x}\boldsymbol{x}^\top\boldsymbol{\Sigma}/\boldsymbol{x}^\top\boldsymbol{\Sigma}\boldsymbol{x}\right)/\sqrt{\boldsymbol{x}^\top\boldsymbol{\Sigma}\boldsymbol{x}} + \mathrm{Diag}(\boldsymbol{b}/\boldsymbol{x}^2).\end{aligned}$$

Cyclical Coordinate Descent Algorithm

This method simply minimizes the function $f(\boldsymbol{x})$ in a cyclical manner with respect to each element x_i of the variable $\boldsymbol{x} = (x_1, \ldots, x_N)$, while holding the other elements fixed. It is a particular case of the block coordinate descent (BCD) algorithm, also called the Gauss–Seidel method; for details, the reader is referred to Section B.6 in Appendix B.

This approach makes sense when the minimization with respect to a single element x_i becomes much easier. In particular, in our case the minimizer can be conveniently derived in closed form as follows (Griveau-Billion et al., 2013):

- For the function $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top\boldsymbol{\Sigma}\boldsymbol{x} - \boldsymbol{b}^\top \log(\boldsymbol{x})$ in (11.5), the elementwise minimization becomes

$$\underset{x_i \geq 0}{\text{minimize}} \quad \tfrac{1}{2}x_i^2\boldsymbol{\Sigma}_{ii} + x_i(\boldsymbol{x}_{-i}^\top\boldsymbol{\Sigma}_{-i,i}) - b_i \log x_i,$$

  where $\boldsymbol{x}_{-i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N)$ denotes the variable $\boldsymbol{x}$ without the ith element and $\boldsymbol{\Sigma}_{-i,i}$ denotes the ith column of matrix $\boldsymbol{\Sigma}$ without the ith element. Setting the partial derivative with respect to x_i to zero gives us the second-order equation

$$\boldsymbol{\Sigma}_{ii}x_i^2 + (\boldsymbol{x}_{-i}^\top\boldsymbol{\Sigma}_{-i,i})x_i - b_i = 0$$

  with positive solution given by

$$x_i = \frac{-\boldsymbol{x}_{-i}^\top\boldsymbol{\Sigma}_{-i,i} + \sqrt{(\boldsymbol{x}_{-i}^\top\boldsymbol{\Sigma}_{-i,i})^2 + 4\boldsymbol{\Sigma}_{ii}b_i}}{2\boldsymbol{\Sigma}_{ii}}.$$

- For the function $f(\boldsymbol{x}) = \sqrt{\boldsymbol{x}^\top\boldsymbol{\Sigma}\boldsymbol{x}} - \boldsymbol{b}^\top \log(\boldsymbol{x})$ in (11.6), the derivation follows similarly, with the update for x_i given by

$$x_i = \frac{-\boldsymbol{x}_{-i}^\top\boldsymbol{\Sigma}_{-i,i} + \sqrt{(\boldsymbol{x}_{-i}^\top\boldsymbol{\Sigma}_{-i,i})^2 + 4\boldsymbol{\Sigma}_{ii}b_i\sqrt{\boldsymbol{x}^\top\boldsymbol{\Sigma}\boldsymbol{x}}}}{2\boldsymbol{\Sigma}_{ii}}.$$

The main issue with this method is that the elements of $\boldsymbol{x}$ have to be updated sequentially at each iteration, which increases the computational complexity per iteration. One might be tempted to try a parallel update, but then convergence would not be guaranteed.

Parallel Update via MM

The reason why a parallel update cannot be implemented is that the term $\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x}$ couples all the elements of $\boldsymbol{x}$. One way to decouple these elements is via the majorization–minimization (MM) framework; for details on MM, the reader is referred to Section B.7 in Appendix B.

The MM method obtains the iterates $\boldsymbol{x}^0, \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots$ by solving a sequence of simpler surrogate or *majorized* problems. In particular, the following majorizer is used for the term $\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x}$ around the current point $\boldsymbol{x} = \boldsymbol{x}^k$, that is, an upper-bound tangent at the current point:

$$\frac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x} \leq \frac{1}{2}(\boldsymbol{x}^k)^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x}^k + (\boldsymbol{\Sigma}\boldsymbol{x}^k)^\mathsf{T}(\boldsymbol{x}-\boldsymbol{x}^k) + \frac{\lambda_{\max}}{2}(\boldsymbol{x}-\boldsymbol{x}^k)^\mathsf{T}(\boldsymbol{x}-\boldsymbol{x}^k),$$

where $\lambda_{\max}$ is the largest eigenvalue of matrix $\boldsymbol{\Sigma}$. We can now solve our original problem by solving instead a sequence of majorized problems as follows:

- The majorized problem corresponding to (11.5) is

$$\underset{\boldsymbol{x}\geq\boldsymbol{0}}{\text{minimize}} \quad \tfrac{\lambda_{\max}}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{x} + \boldsymbol{x}^\mathsf{T}(\boldsymbol{\Sigma}-\lambda_{\max}\boldsymbol{I})\boldsymbol{x}^k - \boldsymbol{b}^\mathsf{T}\log(\boldsymbol{x}),$$

  from which setting the gradient to zero gives the second-order equation

$$\lambda_{\max}x_i^2 + ((\boldsymbol{\Sigma}-\lambda_{\max}\boldsymbol{I})\boldsymbol{x}^k)_i x_i - b_i = 0$$

  with positive solution

$$x_i = \frac{-((\boldsymbol{\Sigma}-\lambda_{\max}\boldsymbol{I})\boldsymbol{x}^k)_i + \sqrt{((\boldsymbol{\Sigma}-\lambda_{\max}\boldsymbol{I})\boldsymbol{x}^k)_i^2 + 4\lambda_{\max}b_i}}{2\lambda_{\max}}.$$

- Similarly, forming the majorized problem for (11.6) and setting the gradient to zero gives the second-order equation

$$\lambda_{\max}x_i^2 + ((\boldsymbol{\Sigma}-\lambda_{\max}\boldsymbol{I})\boldsymbol{x}^k)_i x_i - b_i\sqrt{(\boldsymbol{x}^k)^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x}^k} = 0$$

  with positive solution

$$x_i = \frac{-((\boldsymbol{\Sigma}-\lambda_{\max}\boldsymbol{I})\boldsymbol{x}^k)_i + \sqrt{((\boldsymbol{\Sigma}-\lambda_{\max}\boldsymbol{I})\boldsymbol{x}^k)_i^2 + 4\lambda_{\max}b_i\sqrt{(\boldsymbol{x}^k)^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x}^k}}}{2\lambda_{\max}}.$$

Parallel Update via SCA

Another way to decouple the elements of $\boldsymbol{x}$ in the term $\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x}$ is via the successive convex approximation (SCA) framework; for details on SCA, the reader is referred to Section B.8 in Appendix B.

The SCA method obtains the iterates $\boldsymbol{x}^0, \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots$ by solving a sequence of simpler surrogate

problems. In particular, the following surrogate can be used for the term $\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x}$ around the current point $\boldsymbol{x} = \boldsymbol{x}^k$:

$$\frac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x} \approx \frac{1}{2}(\boldsymbol{x}^k)^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x}^k + (\boldsymbol{\Sigma}\boldsymbol{x}^k)^\mathsf{T}(\boldsymbol{x} - \boldsymbol{x}^k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^k)^\mathsf{T}\mathrm{Diag}(\boldsymbol{\Sigma})(\boldsymbol{x} - \boldsymbol{x}^k),$$

where $\mathrm{Diag}(\boldsymbol{\Sigma})$ is a diagonal matrix containing the diagonal of $\boldsymbol{\Sigma}$. We can now solve our original problems by solving instead a sequence of surrogate problems as follows:

- The surrogate problem corresponding to (11.5) is

$$\underset{\boldsymbol{x}\geq\boldsymbol{0}}{\text{minimize}} \quad \tfrac{1}{2}\boldsymbol{x}^\mathsf{T}\mathrm{Diag}(\boldsymbol{\Sigma})\boldsymbol{x} + \boldsymbol{x}^\mathsf{T}(\boldsymbol{\Sigma} - \mathrm{Diag}(\boldsymbol{\Sigma}))\boldsymbol{x}^k - \boldsymbol{b}^\mathsf{T}\log(\boldsymbol{x}),$$

  from which setting the gradient to zero gives the second-order equation

$$\boldsymbol{\Sigma}_{ii}x_i^2 + ((\boldsymbol{\Sigma} - \mathrm{Diag}(\boldsymbol{\Sigma}))\boldsymbol{x}^k)_i x_i - b_i = 0$$

  with positive solution

$$x_i = \frac{-((\boldsymbol{\Sigma} - \mathrm{Diag}(\boldsymbol{\Sigma}))\boldsymbol{x}^k)_i + \sqrt{((\boldsymbol{\Sigma} - \mathrm{Diag}(\boldsymbol{\Sigma}))\boldsymbol{x}^k)_i^2 + 4\boldsymbol{\Sigma}_{ii}b_i}}{2\boldsymbol{\Sigma}_{ii}}.$$

- Similarly, forming the surrogate problem for (11.6) and setting the gradient to zero gives the update

$$x_i = \frac{-((\boldsymbol{\Sigma} - \mathrm{Diag}(\boldsymbol{\Sigma}))\boldsymbol{x}^k)_i + \sqrt{((\boldsymbol{\Sigma} - \mathrm{Diag}(\boldsymbol{\Sigma}))\boldsymbol{x}^k)_i^2 + 4\boldsymbol{\Sigma}_{ii}b_i\sqrt{(\boldsymbol{x}^k)^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x}^k}}}{2\boldsymbol{\Sigma}_{ii}}.$$

11.6.4 Numerical Experiments

To evaluate the algorithms empirically, we use a universe of $N = 200$ stocks from the S&P 500 and compare their convergence in terms of iterations as well as CPU time (the latter is needed because each iteration may have a different computational cost depending on the method).

Effect of Initial Point

We start by evaluating in Figure 11.4 the effect of different initial points in Newton's method for Spinu's formulation (11.5). We can clearly observe that the effect is huge. Surprisingly, the naive diagonal solution, what would have seemed to be a good starting point, is the worst possible choice. The diagonal row-sum heuristic turns out to be an excellent choice and will be used hereafter.

Comparison Between the Two Formulations (11.5) and (11.6)

We continue by comparing in Figure 11.5 the difference between solving the two formulations in (11.5) (termed Spinu) and (11.6) (termed Roncalli) for Newton's method and the cyclical coordinate descent method. We can observe that there is not much difference between the two formulations. Interestingly, the cyclical coordinate method shows the best convergence in terms of iterations, but the worst in terms of CPU time. However, a word of caution is needed here: in the cyclical update, each iteration requires updating each element sequentially and

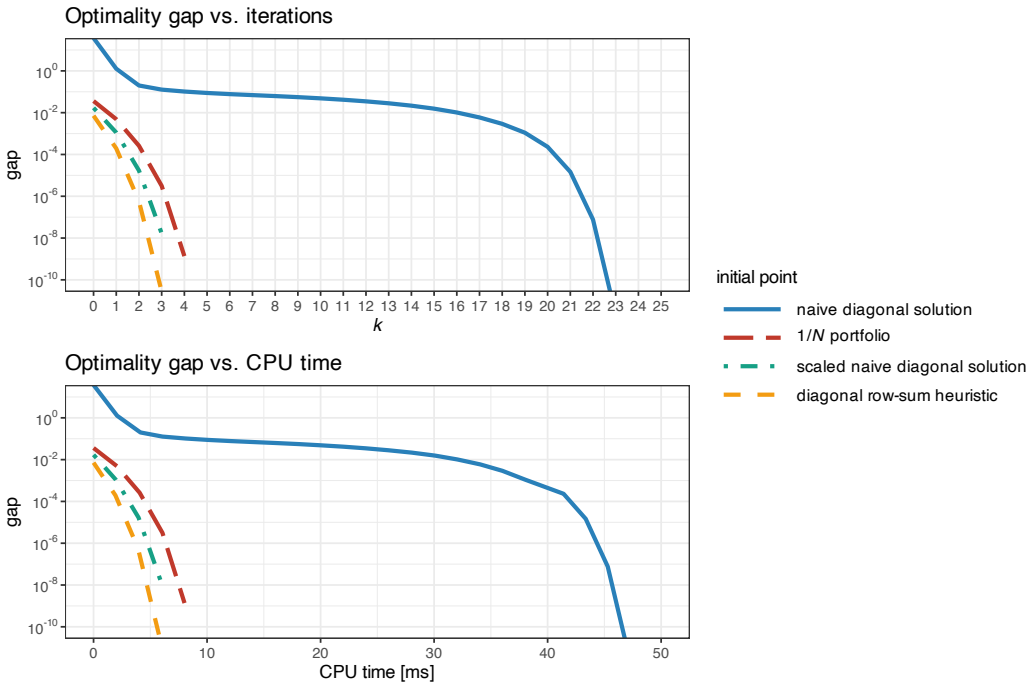


Figure 11.4 Effect of the initial point in Newton's method for Spinu's RPP formulation (11.5).

that requires a loop; it is well known that loops are slow in high-level programming languages like R or Python, but much faster in C++ or Rust. Thus, a more realistic comparison should be performed in a low-level programming language. As a first-order approximation, when implemented in a low-level programming language, one can expect the cyclical methods to perform very similarly to the parallel SCA methods.

Benefit of Reformulation in Terms of Correlation Matrix

Figure 11.6 explores the benefit of reformulating the problem in terms of the correlation matrix as in (11.4) instead of the covariance matrix – as already proposed in Spinu (2013) and numerically assessed in Choi and Chen (2022) – based on Newton's method and MM for the formulation (11.5). A modest improvement is observed and will be used hereafter.

Benefit of the Normalization Step

The solution to $\boldsymbol{\Sigma x} = \boldsymbol{b}/\boldsymbol{x}$ naturally satisfies $\boldsymbol{x}^\top \boldsymbol{\Sigma x} = \mathbf{1}^\top \boldsymbol{b} = 1$. It was suggested in Choi and Chen (2022) to enforce such normalization after each iteration with the quadratic normalization step

$$\boldsymbol{x} \leftarrow \boldsymbol{x} \times \frac{1}{\sqrt{\boldsymbol{x}^\top \boldsymbol{\Sigma x}}}.$$

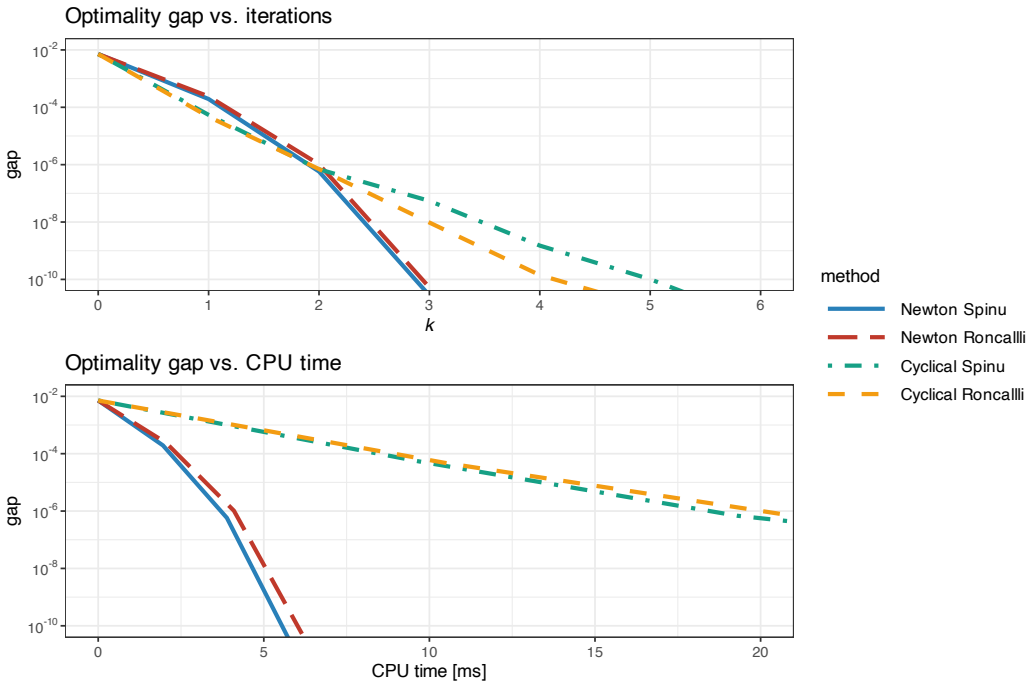


Figure 11.5 Difference between solving Spinu's RPP formulation (11.5) and Roncalli's RPP formulation (11.6) via Newton's method and the cyclical coordinate descent method.

This normalization has a complexity of $O(N^2)$, which is not insignificant. An alternative way is to enforce at each iteration $\mathbf{1}^\top \boldsymbol{\Sigma} \boldsymbol{x} = \mathbf{1}^\top (\boldsymbol{b}/\boldsymbol{x})$, leading to the linear normalization step

$$\boldsymbol{x} \leftarrow \boldsymbol{x} \times \sqrt{\frac{\boldsymbol{b}^\top (1/\boldsymbol{x})}{(\mathbf{1}^\top \boldsymbol{\Sigma})\boldsymbol{x}}},$$

which has a complexity of $O(N)$.

Figure 11.7 explores the benefit of introducing the normalization step (either quadratic or linear) for the formulation (11.5) based on the correlation matrix. A small improvement is observed in the case of the linear normalization.

Final Comparison of Selected Methods

Figure 11.8 shows the convergence of the Newton, MM, and SCA methods for problem (11.5), comparing the original version with the improved one (i.e., using the correlation matrix instead of the covariance matrix and using the linear normalization step). It seems that the best method is the improved SCA.

Finally, it is insightful to study the convergence CPU time as a function of the problem dimension N, as shown in Figure 11.9. Again, the improved SCA method seems the best, followed by the original SCA and the improved MM.

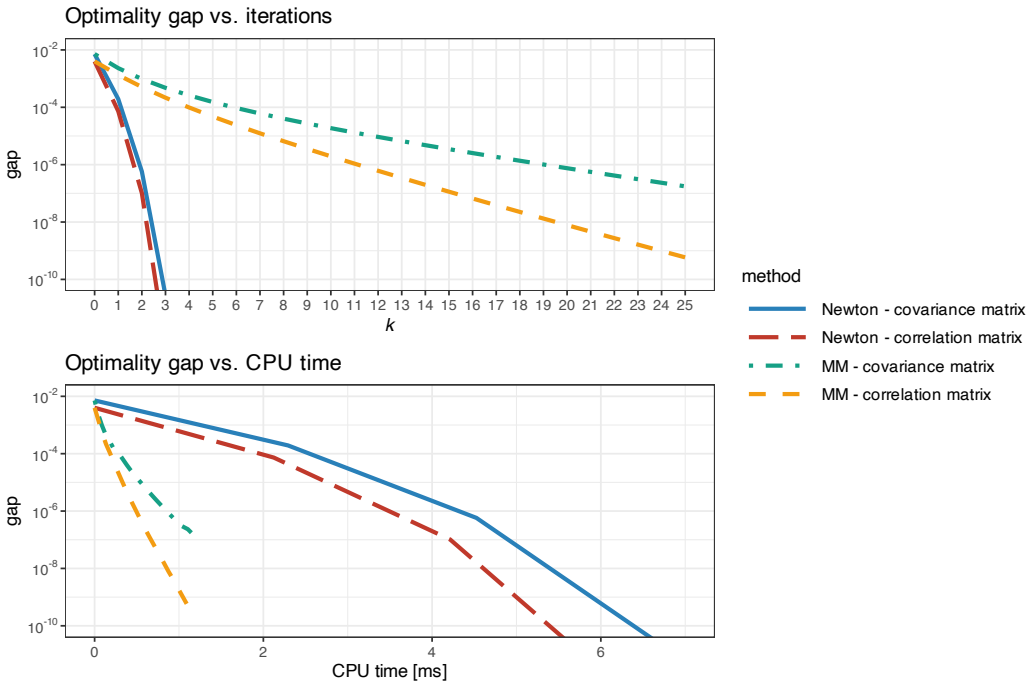


Figure 11.6 Effect of formulating the RPP problem in terms of the correlation matrix and the covariance matrix.

11.7 General Nonconvex Formulations

Risk parity places risk management at the heart of the strategy by building risk-diversified portfolios while ignoring the expected return. It has been criticized precisely because it focuses on managing risk concentration rather than portfolio performance. However, the expected return can also be taken into account within the risk parity paradigm (Roncalli, 2013a). This is often referred to as the *enhanced* risk parity portfolio.

Section 11.6 was devoted to the vanilla formulation, that is, just with the portfolio constraints $\mathbf{1}^\top \boldsymbol{w} = 1$ and $\boldsymbol{w} \geq \mathbf{0}$, which can be reformulated in convex form and optimally solved in order to obtain a portfolio satisfying the risk budgeting equations: $w_i (\boldsymbol{\Sigma w})_i = b_i \boldsymbol{w}^\top \boldsymbol{\Sigma w}$, $i = 1, \ldots, N$. However, in practice, portfolio managers always have additional constraints (e.g., turnover constraint, market-neutral constraint, maximum-position constraint) as listed in Section 6.2 and also additional objective functions such as the maximization of the expected return or minimization of the overall variance or volatility. In such realistic scenarios, unfortunately, the convex reformulations in Section 11.6 do not hold anymore and we need to resort to more complicated nonconvex formulations.

For the general case incorporating additional constraints and/or objective functions, the risk budgeting equations can only be approximately satisfied:

$$w_i (\boldsymbol{\Sigma w})_i \approx b_i \boldsymbol{w}^\top \boldsymbol{\Sigma w}, \qquad i = 1, \ldots, N.$$

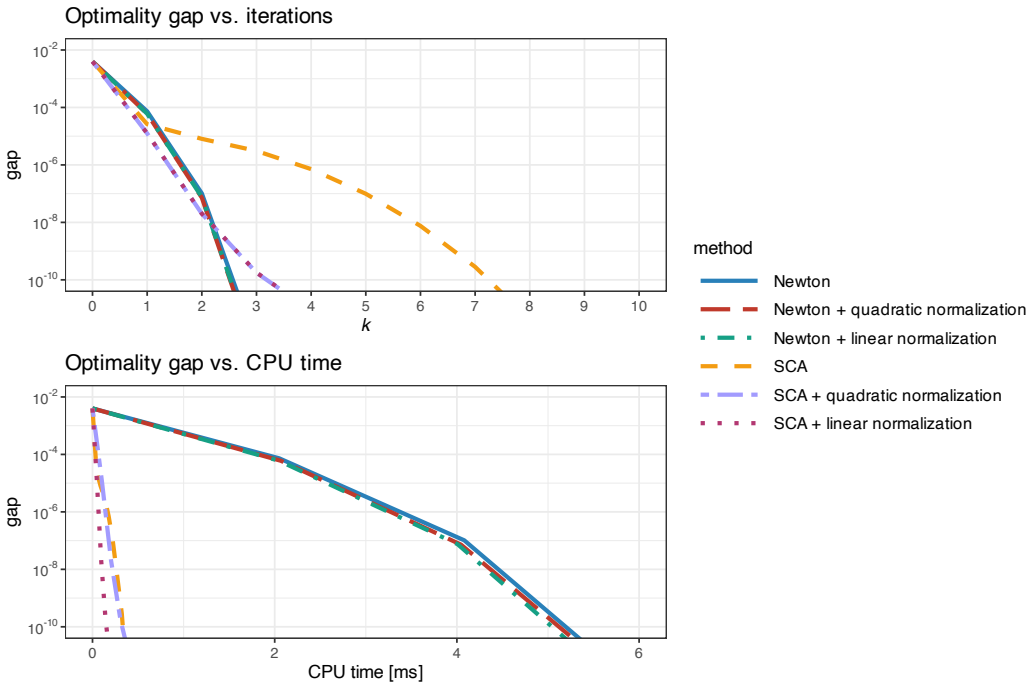


Figure 11.7 Effect of introducing a normalization step (quadratic or linear) in RPP algorithms.

Since the risk budgeting constraints can only be satisfied approximately, we need to define a measure of the approximation error, but there is a multitude of possible choices, such as:

- sum of squared relative risk-contribution errors:

$$\sum_{i=1}^{N} \left(\frac{w_i \left(\boldsymbol{\Sigma} \boldsymbol{w} \right)_i}{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}} - b_i \right)^2;$$

- sum of squared risk-contribution errors:

$$\sum_{i=1}^{N} \left(\frac{w_i \left(\boldsymbol{\Sigma} \boldsymbol{w} \right)_i}{\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}} - b_i \sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}} \right)^2;$$

- sum of squared volatility-scaled risk-contribution errors:

$$\sum_{i=1}^{N} \left(w_i \left(\boldsymbol{\Sigma} \boldsymbol{w} \right)_i - b_i \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w} \right)^2.$$

These three error definitions are the same up to a scaling factor; but in terms of numerical algorithms they show different convergence behaviors (notice that the gradients and Hessians are different). We recommend the use of the relative risk contributions since they are normalized by definition between 0 and 1, and do not suffer from numerical issues.

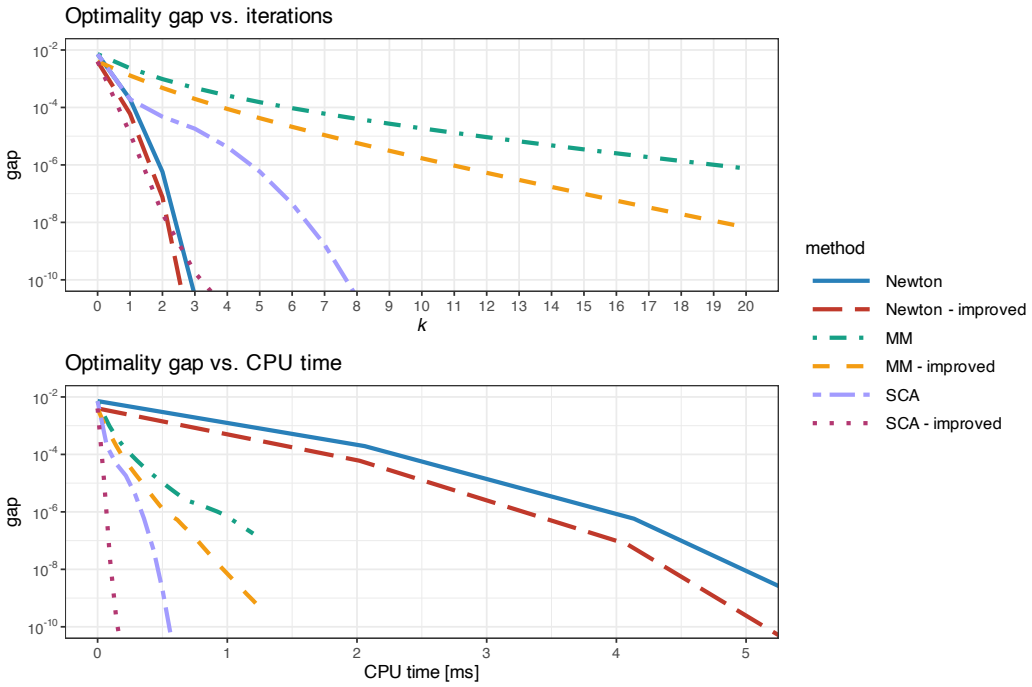


Figure 11.8 Convergence of different algorithms for the vanilla convex RPP.

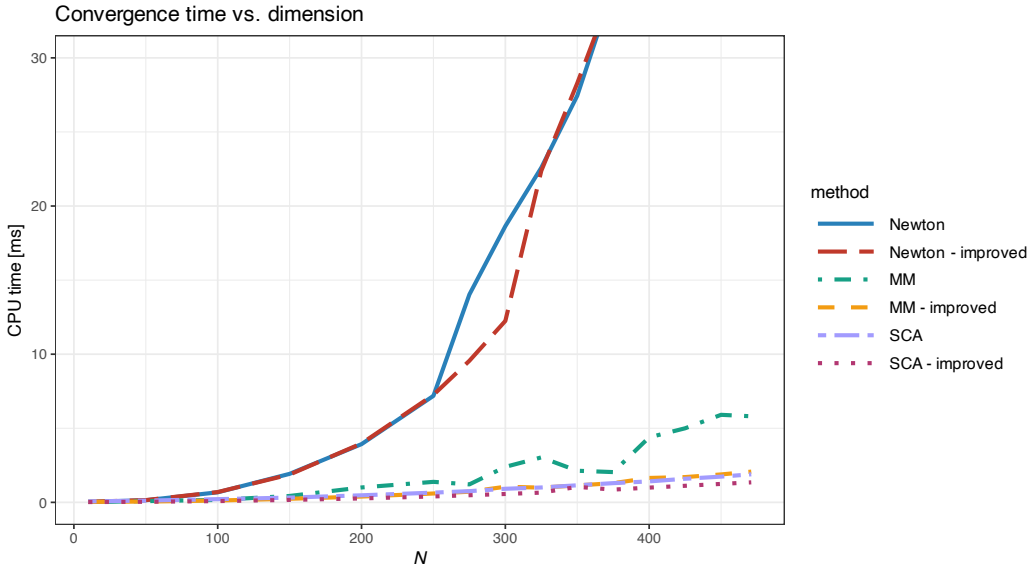


Figure 11.9 Computational cost vs. dimension N of different algorithms for the vanilla convex RPP.

Another alternative to assess the concentration of a risk measure f is via the *Herfindahl index*

(Roncalli, 2013b):

$$h(\boldsymbol{w}) \triangleq \sum_{i=1}^{N} \left(\frac{w_i \frac{\partial f}{\partial w_i}}{f(\boldsymbol{w})} \right)^2,$$

which satisfies $1/N \leq h(\boldsymbol{w}) \leq 1$, with $h(\boldsymbol{w}) = 1$ indicating that the risk is totally concentrated on a single asset and $h(\boldsymbol{w}) = 1/N$ corresponding to the risk being equally distributed among all the assets. Hence, the smaller the Herfindahl index is, the more diversified the risk is. For the case of the volatility, it becomes

$$h(\boldsymbol{w}) = \sum_{i=1}^{N} \left(\frac{w_i (\boldsymbol{\Sigma w})_i}{\boldsymbol{w}^\top \boldsymbol{\Sigma w}} \right)^2.$$

These expressions are all based on the ℓ_2-norm; however, many alternative norms could be used instead, such as the ℓ_1-norm, ℓ_∞-norm, or Huber's robust penalty function. This leads to a multitude of possible portfolio formulations, some of which have been explored in the literature as described next.

11.7.1 Formulations

One of the earlier proposed formulations (Maillard et al., 2010) penalizes the differences between the volatility-scaled risk-contribution errors as

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \displaystyle\sum_{i,j=1}^{N} \left(w_i \left(\boldsymbol{\Sigma w}\right)_i - w_j \left(\boldsymbol{\Sigma w}\right)_j \right)^2 \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array} \tag{11.9}$$

which can obviously be generalized to include the risk budget profile $\boldsymbol{b} = (b_1, \ldots, b_N)$ simply by using the normalized terms $w_i \left(\boldsymbol{\Sigma w}\right)_i / b_i$ instead of $w_i \left(\boldsymbol{\Sigma w}\right)_i$.

To avoid the double-index summation in the formulation (11.9), the following alternative reformulation, with the additional dummy variable θ, reduces the number of terms from N^2 to N:

$$\begin{array}{ll} \underset{\boldsymbol{w},\theta}{\text{minimize}} & \displaystyle\sum_{i=1}^{N} \left(w_i \left(\boldsymbol{\Sigma w}\right)_i - \theta \right)^2 \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}. \end{array} \tag{11.10}$$

Note that the optimal θ for a given $\boldsymbol{w}$ is $\theta = \frac{1}{N} \sum_{i=1}^{N} w_i \left(\boldsymbol{\Sigma w}\right)_i = \frac{1}{N} \boldsymbol{w}^\top \boldsymbol{\Sigma w}$.

The following formulation based on the relative risk-contribution errors was proposed in Bruder and Roncalli (2012):

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \displaystyle\sum_{i=1}^{N} \left(\frac{w_i \left(\boldsymbol{\Sigma w}\right)_i}{\boldsymbol{w}^\top \boldsymbol{\Sigma w}} - b_i \right)^2 \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}. \end{array} \tag{11.11}$$

Another alternative formulation is based on the minimization of the Herfindahl index (Roncalli,

2013b):

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \displaystyle\sum_{i=1}^{N}\left(\frac{w_i(\boldsymbol{\Sigma}\boldsymbol{w})_i}{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}}\right)^2 \\ \text{subject to} & \boldsymbol{w}\in\mathcal{W},\end{array} \tag{11.12}$$

which can be seen as a particular case of (11.11) with $b_i = 0$.

The formulations (11.9) and (11.10) are not recommended as they tend to suffer from numerical issues because the terms $w_i\,(\boldsymbol{\Sigma}\boldsymbol{w})_i$ squared can become very small (in order to make them work, the covariance matrix $\boldsymbol{\Sigma}$ has to be artificially scaled up) (Mausser & Romanko, 2014). For this reason, the formulations (11.11) and (11.12), based on the normalized terms $w_i\,(\boldsymbol{\Sigma}\boldsymbol{w})_i\,/(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w})$, are preferred for numerical stability.

Unified Formulation

A general risk parity formulation was proposed in Feng and Palomar (2015) as

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \displaystyle\sum_{i=1}^{N} g_i(\boldsymbol{w})^2 + \lambda F(\boldsymbol{w}) \\ \text{subject to} & \boldsymbol{w}\in\mathcal{W},\end{array} \tag{11.13}$$

where $g_i(\boldsymbol{w})$ denotes an arbitrary concentration error measure for the ith asset, for example,

$$g_i(\boldsymbol{w}) = \frac{w_i\,(\boldsymbol{\Sigma}\boldsymbol{w})_i}{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}} - b_i,$$

the function $F(\boldsymbol{w})$ denotes an arbitrary preference function, for example,

$$F(\boldsymbol{w}) = -\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu} + \frac{1}{2}\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w},$$

and λ is a trade-off hyper-parameter.

This formulation is general enough to embrace all the previously given formulations. The design of an effective algorithm is nevertheless nontrivial due to the nonconvexity of the term $\sum_{i=1}^{N} g_i(\boldsymbol{w})^2$.

Alternative Convex Formulation

An interesting convex reformulation as a second-order cone program (SOCP) was proposed in Mausser and Romanko (2014). Consider the formulation (11.10) after solving for θ:

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \displaystyle\sum_{i=1}^{N}\left(\frac{1}{N}\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w} - w_i\,(\boldsymbol{\Sigma}\boldsymbol{w})_i\right)^2 \\ \text{subject to} & \boldsymbol{w}\in\mathcal{W}.\end{array}$$

Instead of using the sum of the squares in the objective, we can focus on the worst-case performance:

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \sqrt{\frac{1}{N}\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w}} - \min\limits_i \sqrt{w_i\,(\boldsymbol{\Sigma}\boldsymbol{w})_i} \\ \text{subject to} & \boldsymbol{w}\in\mathcal{W}.\end{array}$$

At this point, by defining the dummy variable $\boldsymbol{z} = \boldsymbol{\Sigma w}$ and identifying the first term in the objective as the variable p and the second term as the variable t, we can rewrite the problem as the following SOCP:

$$\begin{array}{lll} \underset{\boldsymbol{w},\boldsymbol{z},p,t}{\text{minimize}} & p - t & \\ \text{subject to} & \boldsymbol{z} = \boldsymbol{\Sigma w}, & \\ & N p^2 \geq \boldsymbol{w}^\mathsf{T} \boldsymbol{\Sigma w}, & \\ & t^2 \leq w_i z_i, & i = 1, \ldots, N, \\ & p, t \geq 0, & \\ & \boldsymbol{z} \geq \mathbf{0}, & \\ & \boldsymbol{w} \in \mathcal{W}. & \end{array}$$

Note that the constraints $t^2 \leq w_i z_i$ are called *hyperbolic constraints* (also known as *rotated second-order constraints*) and can be written as the following second-order cone constraint (Lobo et al., 1998):

$$\left\| \begin{bmatrix} 2t \\ w_i - z_i \end{bmatrix} \right\|_2 \leq w_i + z_i.$$

Similarly, constraints of the form $w_i(\boldsymbol{\Sigma w})_i \geq b_i \boldsymbol{w}^\mathsf{T} \boldsymbol{\Sigma w}$ can be rewritten as second-order cone constraints:

$$\left\| \begin{bmatrix} 2\sqrt{b_i}\boldsymbol{\Sigma}^{1/2}\boldsymbol{w} \\ w_i - (\boldsymbol{\Sigma w})_i \end{bmatrix} \right\|_2 \leq w_i + (\boldsymbol{\Sigma w})_i.$$

Unfortunately, solving an SOCP has a higher complexity than solving a QP, as described in Section B.1, Appendix B. Therefore, as reported in Mausser and Romanko (2014), this formulation does not seem to be advantageous or competitive in terms of deriving fast practical algorithms.

Illustrative Example

Figure 11.10 shows the distribution of the portfolio weights and risk contribution for the $1/N$ portfolio, the naive diagonal RPP, the vanilla convex RPP, and the general nonconvex RPP (in this case including an upper bound on the weights of 0.15 for illustration purposes).

11.7.2 Algorithms

The previous nonconvex formulations can be solved with any general-purpose solver, cf. Mausser and Romanko (2014). As a more convenient and efficient alternative, we will now develop several practical iterative algorithms tailored to the problem formulations in (11.9)–(11.13) that produce a sequence of iterates $\boldsymbol{w}^0, \boldsymbol{w}^1, \boldsymbol{w}^2, \ldots$

As the initial point of these iterative algorithms, one option is to use the solution provided by the vanilla convex formulation considered in Section 11.6 (effectively ignoring any additional constraints in $\mathcal{W}$ and any additional objective function). However, some additional step is necessary to make sure that point is feasible taking into account all the constraints in $\mathcal{W}$. A simpler alternative may be to simply use the $1/N$ portfolio.

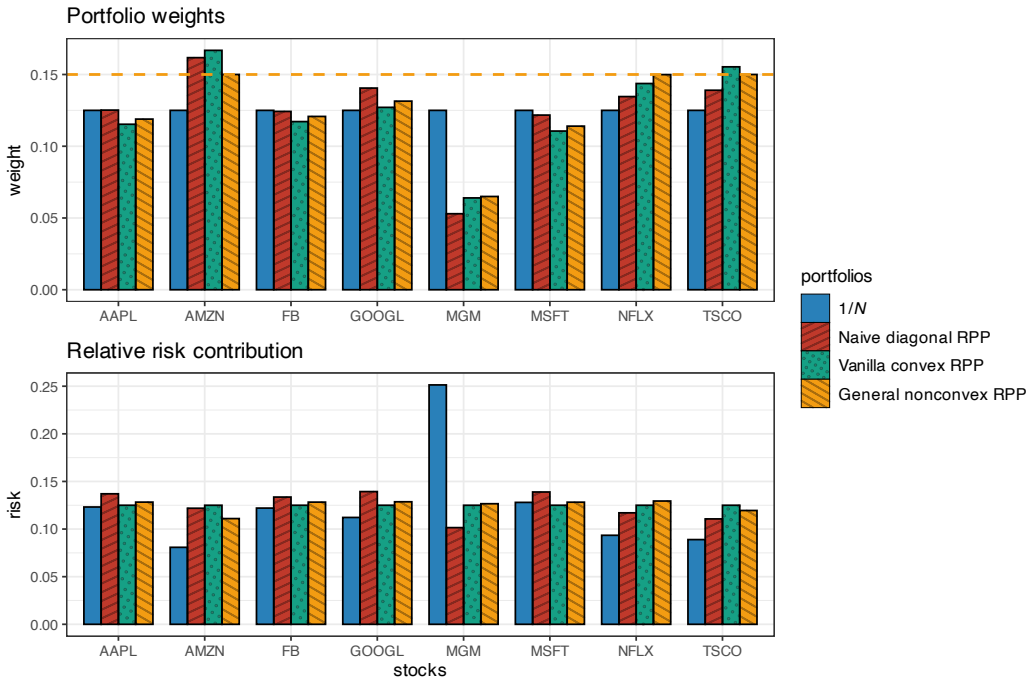


Figure 11.10 Portfolio allocation and risk contribution of general nonconvex RPP (with $w_i \leq 0.15$) compared to benchmarks ($1/N$ portfolio, naive diagonal RPP, and vanilla convex RPP).

SCA Method

The SCA method can again be employed in this nonconvex case to develop efficient algorithms as proposed in Feng and Palomar (2015); for details on SCA, the reader is referred to Section B.8 in Appendix B.

Consider the objective function in the unified formulation (11.13):

$$U(\boldsymbol{w}) = \sum_{i=1}^{N} g_i(\boldsymbol{w})^2 + \lambda F(\boldsymbol{w}).$$

One convenient way to convexify this function producing a quadratic function, which will be amenable for a QP solver, is to linearize the terms $g_i(\boldsymbol{w})$ around the current point $\boldsymbol{w}^k$:

$$g_i(\boldsymbol{w}) \approx g_i(\boldsymbol{w}^k) + \nabla g_i(\boldsymbol{w}^k)^\mathsf{T} \left(\boldsymbol{w} - \boldsymbol{w}^k\right).$$

This produces the following surrogate function for $U(\boldsymbol{w}, \boldsymbol{w}^k)$:

$$\tilde{U}(\boldsymbol{w}, \boldsymbol{w}^k) = \sum_{i=1}^{N} \left(g_i(\boldsymbol{w}^k) + \nabla g_i(\boldsymbol{w}^k)^\mathsf{T} \left(\boldsymbol{w} - \boldsymbol{w}^k\right)\right)^2 + \lambda F(\boldsymbol{w}) + \frac{\tau}{2} \left\|\boldsymbol{w} - \boldsymbol{w}^k\right\|_2^2,$$

where the last term is a regularizer to obtain a strongly convex function as required by SCA.

Finally, the approximated QP can be written (ignoring unnecessary constant terms) as

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \frac{1}{2}\boldsymbol{w}^{\mathsf{T}}\boldsymbol{Q}^k\boldsymbol{w} + \boldsymbol{w}^{\mathsf{T}}\boldsymbol{q}^k + \lambda F(\boldsymbol{w}) \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array} \tag{11.14}$$

where

$$\begin{aligned} \boldsymbol{Q}^k &\triangleq 2\left(\boldsymbol{J}^k\right)^{\mathsf{T}}\boldsymbol{J}^k + \tau\boldsymbol{I}, \\ \boldsymbol{q}^k &\triangleq 2\left(\boldsymbol{J}^k\right)^{\mathsf{T}}\boldsymbol{g}^k - \boldsymbol{Q}^k\boldsymbol{w}^k, \end{aligned}$$

and the vector of functions g_i and its Jacobian matrix at point $\boldsymbol{w}^k$ are

$$\begin{aligned} \boldsymbol{g}^k &\triangleq \left[g_1(\boldsymbol{w}^k), \ldots, g_N(\boldsymbol{w}^k)\right]^{\mathsf{T}}, \\ \boldsymbol{J}^k &\triangleq \begin{bmatrix} \nabla g_1(\boldsymbol{w}^k)^{\mathsf{T}} \\ \vdots \\ \nabla g_N(\boldsymbol{w}^k)^{\mathsf{T}} \end{bmatrix}. \end{aligned}$$

This approximated QP can be solved directly with a QP solver or, depending on the constraints in $\mathcal{W}$, even in closed form as developed in Feng and Palomar (2015).

The overall SCA method, termed *Successive Convex optimization for RIsk Parity portfolio* (SCRIP), is summarized in Algorithm 11.1. For additional details and more advanced algorithms, the reader is referred to (Feng & Palomar, 2015).

Algorithm 11.1: SCRIP to solve problem (11.13).

1: Choose initial point $\boldsymbol{w}^0 \in \mathcal{W}$ and sequence $\{\gamma^k\}$;
2: Set $k \leftarrow 0$;
3: **repeat**
4: Calculate risk concentration terms $\boldsymbol{g}^k$ and Jacobian matrix $\boldsymbol{J}^k$ for current point $\boldsymbol{w}^k$;
5: Solve approximated QP problem (11.14) and keep solution as $\hat{\boldsymbol{w}}^{k+1}$;
6: $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k + \gamma^k(\hat{\boldsymbol{w}}^{k+1} - \boldsymbol{w}^k)$;
7: $k \leftarrow k + 1$;
8: **until** convergence;

Alternate Linearization Method

The *alternate linearization method* (ALM) was proposed in Bai et al. (2016) to solve the formulation in (11.10), whose objective is

$$F(\boldsymbol{w}, \theta) = \sum_{i=1}^{N} \left(w_i \left(\boldsymbol{\Sigma}\boldsymbol{w}\right)_i - \theta\right)^2 = \sum_{i=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{M}_i\boldsymbol{w} - \theta\right)^2,$$

where matrix $\boldsymbol{M}_i$ is defined to be of the same size as $\boldsymbol{\Sigma}$ containing the same ith row, $[\boldsymbol{M}_i]_{i,:} = [\boldsymbol{\Sigma}]_{i,:}$, and zeros elsewhere.

The idea is to introduce an additional variable $\boldsymbol{y}$ and then define

$$F(\boldsymbol{w}, \boldsymbol{y}, \theta) = \sum_{i=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{M}_i \boldsymbol{y} - \theta\right)^2,$$

subject to $\boldsymbol{y} = \boldsymbol{w}$. Then, the algorithm optimizes $\boldsymbol{w}$ and $\boldsymbol{y}$ sequentially (and also θ) by minimizing the following two QP approximations of $F(\boldsymbol{w}, \boldsymbol{y}, \theta)$:

$$Q^1(\boldsymbol{w}, \boldsymbol{y}^k, \theta) \triangleq F(\boldsymbol{w}, \boldsymbol{y}^k, \theta) + \nabla_2 F(\boldsymbol{y}^k, \boldsymbol{y}^k, \theta)^{\mathsf{T}} (\boldsymbol{w} - \boldsymbol{y}^k) + \frac{1}{2\mu} \|\boldsymbol{w} - \boldsymbol{y}^k\|_2^2,$$

$$Q^2(\boldsymbol{w}^{k+1}, \boldsymbol{y}, \theta) \triangleq F(\boldsymbol{w}^{k+1}, \boldsymbol{y}, \theta) + \nabla_1 F(\boldsymbol{w}^{k+1}, \boldsymbol{w}^{k+1}, \theta)^{\mathsf{T}} (\boldsymbol{y} - \boldsymbol{w}^{k+1}) + \frac{1}{2\mu} \|\boldsymbol{y} - \boldsymbol{w}^{k+1}\|_2^2,$$

where μ is some chosen positive scalar and

$$\nabla_1 F(\boldsymbol{w}, \boldsymbol{y}, \theta) = 2 \sum_{i=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{M}_i \boldsymbol{y} - \theta\right) \boldsymbol{M}_i \boldsymbol{y},$$

$$\nabla_2 F(\boldsymbol{w}, \boldsymbol{y}, \theta) = 2 \sum_{i=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{M}_i \boldsymbol{y} - \theta\right) \boldsymbol{M}_i^{\mathsf{T}} \boldsymbol{w}.$$

11.7.3 Numerical Experiments

To evaluate the algorithms empirically, we use a universe of $N = 200$ stocks from the S&P 500 and compare their convergence in terms of iterations as well as CPU time. We include the upper bound on the weights $w_i \leq 0.008$ so that the formulation cannot be framed in the convex case considered in Section 11.6, and we use the simple initial point $w_i = 1/N$ so that we can better observe the convergence.

Convergence for Formulation in (11.10)

We start with the nonconvex formulation in (11.10), which, in fact, is not recommended since the terms $w_i \left(\boldsymbol{\Sigma} \boldsymbol{w}\right)_i$ squared can become very small and lead to numerical issues; a practical heuristic is to scale up the covariance matrix $\boldsymbol{\Sigma}$ by some large number like 10^4 (Mausser & Romanko, 2014). For this reason, the formulations based on the normalized terms $w_i \left(\boldsymbol{\Sigma} \boldsymbol{w}\right)_i / (\boldsymbol{w}^{\mathsf{T}} \boldsymbol{\Sigma} \boldsymbol{w})$ are preferred for numerical stability, like in formulation (11.11).

Figure 11.11 shows the convergence of two general-purpose nonlinear solvers as well as the methods ALM and SCA. It is clear that the nonlinear solvers are less efficient as they do not exploit the problem structure. The SCA method is clearly the fastest to converge.

Convergence for Formulation in (11.11)

We now focus on the formulation in (11.11), which is preferred because the risk terms are normalized and do not suffer from numerical issues. Figure 11.12 shows the convergence of two general-purpose nonlinear solvers as well as the SCA method, which exhibits a much faster convergence.

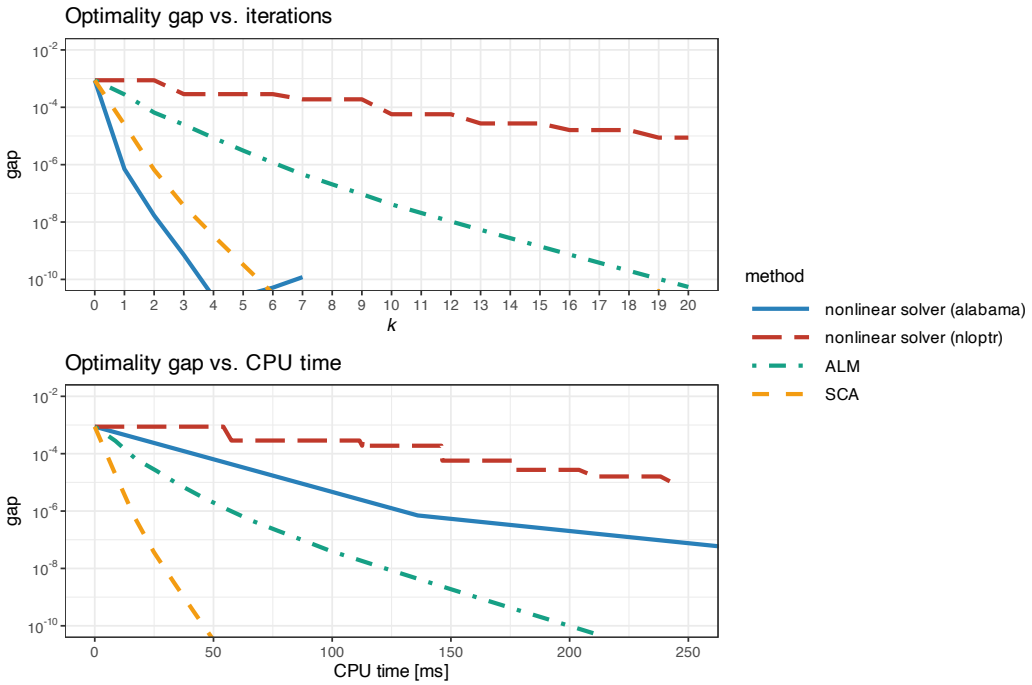


Figure 11.11 Convergence of different algorithms for the nonconvex RPP formulation (11.10).

11.8 Summary

Diversification is a crucial principle in portfolio design, exemplified by the well-known phrase, "don't put all your eggs in one basket." Some key points are:

- The widely used $1/N$ portfolio (or equally weighted portfolio) effectively diversifies capital allocation. However, a more advanced strategy is to diversify risk allocation, as implemented by risk parity portfolios.
- In risk parity portfolios, the risk measure of interest (e.g., volatility) is expressed as the sum of individual risk contributions from each asset. This provides a more refined control of the risk compared to simply using a single risk value for the overall portfolio.
- Risk parity formulations can be classified into three levels of complexity:
  - *naive diagonal formulation*: the covariance matrix is assumed diagonal and the solution simplifies to the inverse-volatility portfolio (which ignores the assets' correlations);
  - *vanilla convex formulations*: simple long-only portfolios are considered and the problems can be rewritten in convex form with efficient algorithms; and
  - *general nonconvex formulations*: admit any realistic constraint and extended objective functions at the expense of becoming nonconvex problems that require a more careful resolution (but efficient iterative algorithms can still be derived).

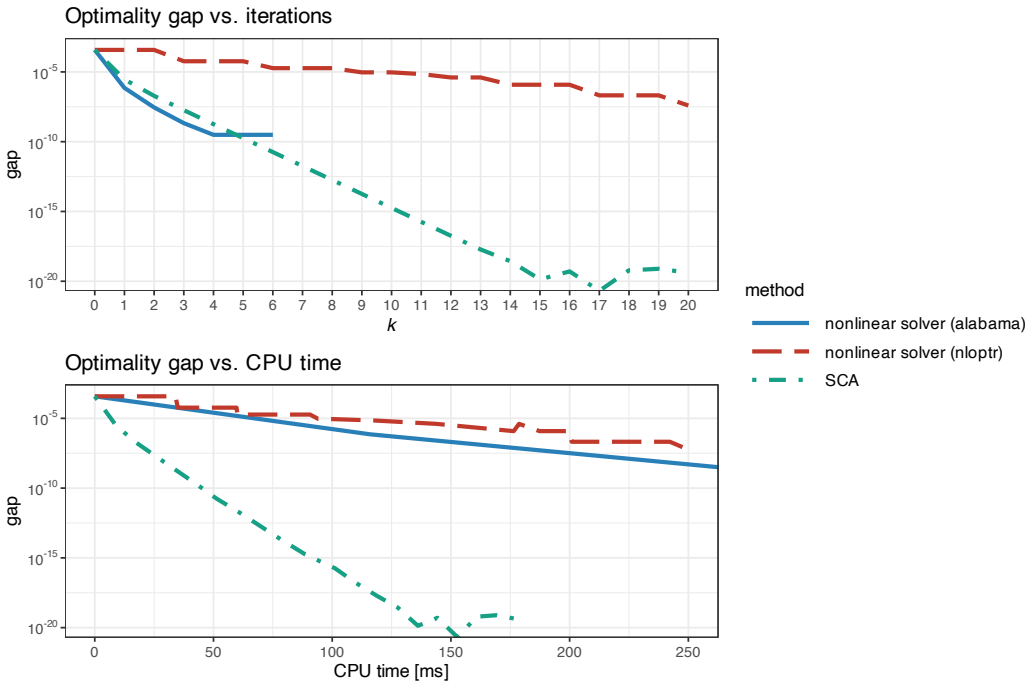


Figure 11.12 Convergence of different algorithms for the nonconvex RPP formulation (11.11) .

Exercises

11.1 (Change of variable) Show why $\boldsymbol{\Sigma x} = \boldsymbol{b}/\boldsymbol{x}$ can be equivalently solved as $\boldsymbol{Cx} = \boldsymbol{b}/\boldsymbol{x}$, where $\boldsymbol{C}$ is the correlation matrix defined as $\boldsymbol{C} = \boldsymbol{D}^{-1/2}\boldsymbol{\Sigma}\boldsymbol{D}^{-1/2}$ with $\boldsymbol{D}$ a diagonal matrix containing $\text{diag}(\boldsymbol{\Sigma})$ along the main diagonal. Would it be possible to use instead $\boldsymbol{C} = \boldsymbol{M}^{-1/2}\boldsymbol{\Sigma}\boldsymbol{M}^{-1/2}$, where $\boldsymbol{M}$ is not necessarily a diagonal matrix?

11.2 (Naive diagonal risk parity portfolio) If the covariance matrix is diagonal, $\boldsymbol{\Sigma} = \boldsymbol{D}$, then the system of nonlinear equations $\boldsymbol{\Sigma x} = \boldsymbol{b}/\boldsymbol{x}$ has the closed-form solution $\boldsymbol{x} = \sqrt{\boldsymbol{b}/\text{diag}(\boldsymbol{D})}$. Explore whether a closed-form solution can be obtained for the rank-one plus diagonal case $\boldsymbol{\Sigma} = \boldsymbol{u}\boldsymbol{u}^\mathsf{T} + \boldsymbol{D}$.

11.3 (Vanilla convex risk parity portfolio) The solution to the formulation

$$\begin{array}{ll} \underset{\boldsymbol{x}\geq\boldsymbol{0}}{\text{maximize}} & \boldsymbol{b}^\mathsf{T}\log(\boldsymbol{x}) \\ \text{subject to} & \sqrt{\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma x}} \leq \sigma_0 \end{array}$$

is

$$\lambda\boldsymbol{\Sigma x} = \boldsymbol{b}/\boldsymbol{x} \times \sqrt{\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma x}}.$$

Can you solve for λ and rewrite the solution in a more compact way without λ?

11.4 (Newton's method) Newton's method requires computing the direction $\boldsymbol{d} = \mathsf{H}^{-1}\nabla f$ or, equivalently, solving the system of linear equations $\mathsf{H}\,\boldsymbol{d} = \nabla f$ for $\boldsymbol{d}$. Explore whether a more

efficient solution is possible by exploiting the structure of the gradient and Hessian:

$$\nabla f = \boldsymbol{\Sigma}\boldsymbol{x} - \boldsymbol{b}/\boldsymbol{x},$$
$$\mathsf{H} = \boldsymbol{\Sigma} + \mathrm{Diag}(\boldsymbol{b}/\boldsymbol{x}^2).$$

11.5 (MM algorithm) The MM algorithm requires the computation of the largest eigenvalue $\lambda_{\max}$ of matrix $\boldsymbol{\Sigma}$, which can be obtained from the eigenvalue decomposition of the matrix. A more efficient alternative is the *power iteration method*. Program both methods and compare their computational complexity.

11.6 (Coordinate descent vs. SCA methods) Consider the vanilla convex formulation

$$\underset{\boldsymbol{x}\geq\mathbf{0}}{\text{minimize}} \quad \tfrac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x} - \boldsymbol{b}^\mathsf{T}\log(\boldsymbol{x}).$$

Implement the cyclical coordinate descent method and the parallel SCA method in a high-level programming language (e.g., R, Python, Julia, or MATLAB) and compare the convergence against the CPU time for these two methods. Then, re-implement these two methods in a low-level programming language (e.g., C, C++, C#, or Rust) and compare the convergence again. Comment on the difference observed.

References

Anderson, R. M., Bianchi, S. W., & Goldberg, L. R. (2012). Will my risk parity strategy outperform? *Financial Analysts Journal*, *68*(6), 75–93.

Asness, C. S., Frazzini, A., & Pedersen, L. H. (2012). Leverage aversion and risk parity. *Financial Analysts Journal*, *68*(1), 47–59.

Bai, X., Scheinberg, K., & Tütüncü, R. (2016). Least-squares approach to risk parity in portfolio selection. *Quantitative Finance*, *16*(3), 357–376.

Bruder, B., & Roncalli, T. (2012). Managing risk exposures using the risk budgeting approach. *MPRA paper 37246*.

Cardoso, J. V. M., & Palomar, D. P. (2021). *riskParityPortfolio: Design of Risk Parity Portfolios* [R package]. https://CRAN.R-project.org/package=riskParityPortfolio

Chaves, D. B., Hsu, J. C., Li, F., & Shakernia, O. (2011). Risk parity portfolio vs. other asset allocation heuristic portfolios. *Journal of Investing*, *20*(1), 108–118.

Chaves, D. B., Hsu, J. C., Li, F., & Shakernia, O. (2012). Efficient algorithms for computing risk parity portfolio weights. *Journal of Investing*, *21*(3), 150–163.

Choi, J., & Chen, R. (2022). Improved iterative methods for solving risk parity portfolio. *Journal of Derivatives and Quantitative Studies*, *30*(2), 114–124.

Feng, Y., & Palomar, D. P. (2015). SCRIP: Successive convex optimization methods for risk parity portfolios design. *IEEE Transactions on Signal Processing*, *63*(19), 5285–5300.

Feng, Y., & Palomar, D. P. (2016). A signal processing perspective on financial engineering. *Foundations and Trends in Signal Processing, Now Publishers*, *9*(1–2), 1–231.

Griveau-Billion, T., Richard, J.-C., & Roncalli, T. (2013). A fast algorithm for computing high-dimensional risk parity portfolios. *SSRN Electronic Journal*. https://dx.doi.org/10.2139/ssrn.2325255

Hallerbach, W. G. (2003). Decomposing portfolio value-at-risk: A general analysis. *Journal of Risk*, *5*(2), 1–18.

Kaya, H., & Lee, W. (2012). Demystifying risk parity. *SSRN Electronic Journal*. https://dx.doi.org/10.2139/ssrn.1987770

Litterman, R. (1996). Hot spots and hedges. *Journal of Portfolio Management*, *22*, 52–75.

Lobo, M. S., Vandenberghe, L., Boyd, S., & Lebret, H. (1998). Applications of second-order cone programming. *Linear Algebra and Applications*, *284*(1–3), 193–228.

Maillard, S., Roncalli, T., & Teiletche, J. (2010). The properties of equally weighted risk contribution portfolios. *Journal of Portfolio Management*, *36*(4), 60–70.

Markowitz, H. M. (1952). Portfolio selection. *The Journal of Finance*, *7*(1), 77–91.

Mausser, H., & Romanko, O. (2014). Computing equal risk contribution portfolios. *Journal of Research and Development*, *58*(4), 5:1–5:12.

McNeil, A. J., Frey, R., & Embrechts, P. (2015). *Quantitative Risk Management*. Princeton University Press.

Qian, E. (2005). Risk parity portfolios: Efficient portfolios through true diversification. *PanAgora Asset Management*.

Qian, E. (2016). *Risk Parity Fundamentals*. CRC Press.

Roncalli, T. (2013a). Introducing expected returns into risk parity portfolios: A new framework for tactical and strategic asset allocation. *SSRN Electronic Journal*. https://dx.doi.org/10.2139/ssrn.2321309

Roncalli, T. (2013b). *Introduction to Risk Parity and Budgeting*. Chapman & Hall/CRC.

Roncalli, T., & Weisang, G. (2016). Risk parity portfolios with risk factors. *Quantitative Finance*, *16*(3), 377–388.

Scaillet, O. (2004). Nonparametric estimation and sensitivity analysis of expected shortfall. *Mathematical Finance*, *14*(1), 115–129.

Spinu, F. (2013). An algorithm for computing risk parity weights. *SSRN Electronic Journal*. https://dx.doi.org/10.2139/ssrn.2297383

Tasche, D. (2008). Capital allocation to business units and sub-portfolios: The Euler principle. *Preprint. Available at arXiv*. https://doi.org/10.48550/arXiv.0708.2542

12

Graph-Based Portfolios

Too much and too little wine. Give him none, he cannot find truth; give him too much, the same.

— Blaise Pascal, *Pensées*

Amid an overload of information in the modern era, graphs provide a convenient and compact way to represent big data, analyze the structure of large networks, and extract patterns that may otherwise go unnoticed. In the context of financial data, graphs of assets provide key information for modern portfolio design that may be incorporated, for example, into the basic mean–variance portfolio formulation (which obtains the portfolio as a trade-off between the expected return and the risk measured by the variance). Nevertheless, exactly how to use this graph information in the portfolio optimization process is still an open question. This chapter explores some attempts in the literature.

12.1 Introduction

Markowitz's mean–variance portfolio (Markowitz, 1952) formulates the portfolio design as a trade-off between the expected return $\boldsymbol{w}^\T\boldsymbol{\mu}$ and the risk measured by the variance $\boldsymbol{w}^\T\boldsymbol{\Sigma}\boldsymbol{w}$ (see Chapter 7 for details):

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\T\boldsymbol{\mu} - \frac{\lambda}{2}\boldsymbol{w}^\T\boldsymbol{\Sigma}\boldsymbol{w} \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W},\end{array}$$

where λ is a hyper-parameter that controls the investor's risk-aversion and $\mathcal{W}$ denotes an arbitrary constraint set, such as $\mathcal{W} = \{\boldsymbol{w} \mid \mathbf{1}^\T\boldsymbol{w} = 1, \boldsymbol{w} \geq \mathbf{0}\}$.

Unfortunately, Markowitz's mean–variance portfolio is severely affected by the ever-present estimation errors in the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The question is whether this portfolio design can be improved with knowledge of the graph of assets by somehow capitalizing on the key connections revealed by the graph connectivity pattern.

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

12.1.1 Graphs and Distance Matrices

Graph-based portfolios are constructed from the graph information of the assets, which can be conveniently encoded in the form of a distance matrix containing the distance between each pair of assets. There are numerous methods to acquire this graph information. We start with two simple approaches and then consider two more sophisticated graph estimation methods from Chapter 5.

A popular and simple way to obtain a distance matrix $\boldsymbol{D}$ is directly from the assets' correlations (Mantegna, 1999) as

$$D_{ij} = \sqrt{\frac{1}{2}(1 - \rho_{ij})}, \tag{12.1}$$

where ρ_{ij} is the correlation between assets i and j; an alternative is $D_{ij} = 1 - \rho_{ij}^2$. This is equivalent to the Euclidean distance between the standardized columns of the data matrix $\boldsymbol{X}$. In more detail, denote the de-meaned and normalized ith column of the data matrix by $\tilde{\boldsymbol{x}}_i = (\boldsymbol{x}_i - \mu_i)/\sigma_i$, where μ_i and σ_i are the mean and volatility of $\boldsymbol{x}_i$, respectively. Then, the empirical value of the correlation can be written as $\rho_{ij} = \frac{1}{T}\tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{x}}_j$, where T is the number of observations, and the normalized squared Euclidean distance between $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$ is $\frac{1}{T}\|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j\|_2^2 = 2(1 - \rho_{ij})$, which coincides with D_{ij} in (12.1) up to a scaling factor. Needless to say, other distance functions could be used to compute the distance matrix; for example, the p-norm or Minkowski metric $D_{ij} = \|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j\|_p$, where $\|\boldsymbol{a}\|_p \triangleq \left(\sum_{t=1}^{T} |a_i|^p\right)^{1/p}$, which for $p = 1$ becomes the Manhattan distance and for $p = 2$ the Euclidean distance.

A drawback of such a correlation-based distance matrix is that each element only contains information involving two assets, ignoring the rest of the assets. A more holistic definition is to compute a new distance matrix $\tilde{\boldsymbol{D}}$ with elements containing the Euclidean distance between pairs of columns of $\boldsymbol{D}$ (López de Prado, 2016):

$$\tilde{D}_{ij} = \|\boldsymbol{d}_i - \boldsymbol{d}_j\|_2, \tag{12.2}$$

where $\boldsymbol{d}_i$ is the ith column of $\boldsymbol{D}$. Thus, each element of $\tilde{\boldsymbol{D}}$ is a function of the entire correlation matrix rather than a particular correlation value like in $\boldsymbol{D}$. In other words, D_{ij} is the distance between two assets while $\tilde{D}_{ij}$ indicates the closeness in similarity of these assets with the rest of the universe.

Example 12.1 (Distance matrix of a toy example) Consider the following correlation matrix of three variables (López de Prado, 2016):

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0.7 & 0.2 \\ 0.7 & 1 & -0.2 \\ 0.2 & -0.2 & 1 \end{bmatrix}.$$

The corresponding correlation-based distance matrix (12.1) is

$$\boldsymbol{D} = \begin{bmatrix} 0 & 0.3873 & 0.6325 \\ 0.3873 & 0 & 0.7746 \\ 0.6325 & 0.7746 & 0 \end{bmatrix}$$

and the Euclidean distance matrix of correlation distances (12.2) is

$$\tilde{\boldsymbol{D}} = \begin{bmatrix} 0 & 0.5659 & 0.9747 \\ 0.5659 & 0 & 1.1225 \\ 0.9747 & 1.1225 & 0 \end{bmatrix}.$$

As an alternative to these heuristic definitions of graph distance matrices, Chapter 5 offers an extensive overview of graphs and presents a wide variety of graph estimation methods derived from data. For financial time series corresponding to T observations of N assets, with the observations denoted $\boldsymbol{x}^{(t)} \in \mathbb{R}^N$, for $t = 1, \ldots, T$, the recommended methods in Section 5.6 are:[1]

- *Heavy-tailed Markov random field (MRF) with degree control* in (5.13) (Cardoso et al., 2021):

$$\begin{array}{ll} \underset{\boldsymbol{w} \geq \boldsymbol{0}}{\text{maximize}} & \log \operatorname{gdet}(\mathcal{L}(\boldsymbol{w})) - \dfrac{N+\nu}{T} \displaystyle\sum_{t=1}^{T} \log\left(\nu + (\boldsymbol{x}^{(t)})^\mathsf{T} \mathcal{L}(\boldsymbol{w}) \boldsymbol{x}^{(t)}\right) \\ \text{subject to} & \mathfrak{d}(\boldsymbol{w}) = \boldsymbol{1}, \end{array} \tag{12.3}$$

  where $\operatorname{gdet}(\cdot)$ denotes the generalized determinant of a matrix (defined as the product of nonzero eigenvalues), the weight vector $\boldsymbol{w}$ is a compact representation of the graph information, $\mathcal{L}(\boldsymbol{w})$ is the Laplacian operator that produces the Laplacian matrix $\boldsymbol{L}$ from the weights, $\mathfrak{d}(\boldsymbol{w})$ is the degree operator that gives the degrees of the nodes, and ν is a hyper-parameter that controls the degree of heavy-tailness.

- *k-component heavy-tailed MRF with degree control* (Cardoso et al., 2021):

$$\begin{array}{ll} \underset{\boldsymbol{w} \geq \boldsymbol{0}, \boldsymbol{F} \in \mathbb{R}^{N \times k}}{\text{maximize}} & \log \operatorname{gdet}(\mathcal{L}(\boldsymbol{w})) - \dfrac{N+\nu}{T} \displaystyle\sum_{t=1}^{T} \log\left(\nu + (\boldsymbol{x}^{(t)})^\mathsf{T} \mathcal{L}(\boldsymbol{w}) \boldsymbol{x}^{(t)}\right) \\ & \qquad + \gamma \operatorname{Tr}\left(\boldsymbol{F}^\mathsf{T} \mathcal{L}(\boldsymbol{w}) \boldsymbol{F}\right) \\ \text{subject to} & \mathfrak{d}(\boldsymbol{w}) = \boldsymbol{1}, \quad \boldsymbol{F}^\mathsf{T} \boldsymbol{F} = \boldsymbol{I}, \end{array} \tag{12.4}$$

  which incorporates the regularization term $\operatorname{Tr}(\boldsymbol{F}^\mathsf{T} \mathcal{L}(\boldsymbol{w}) \boldsymbol{F})$, controlled by the hyper-parameter γ; the additional variable $\boldsymbol{F}$ is to enforce the low-rank property in the Laplacian matrix (i.e., enforce $\boldsymbol{L} = \mathcal{L}(\boldsymbol{w})$ to be rank $N - k$), which corresponds to a k-component graph, that is, a graph with k clusters.

After solving these graph learning formulations, the result is contained in the graph Laplacian matrix $\boldsymbol{L} = \mathcal{L}(\boldsymbol{w})$, from which a matrix akin to the correlation matrix can be obtained as

$$\boldsymbol{C} = \operatorname{abs}(\boldsymbol{L}),$$

where $\operatorname{abs}(\cdot)$ denotes the elementwise absolute value (basically, it keeps the diagonal elements equal to 1 and makes the off-diagonal elements nonnegative). Then, the distance matrix can be obtained as in (12.1): $D_{ij} = \sqrt{\frac{1}{2}(1 - C_{ij})}$.

[1] The R package `fingraph`, based on Cardoso et al. (2021), contains efficient implementations for the two formulations (12.3) and (12.4); namely, the functions `learn_regular_heavytail_graph()` for the heavy-tailed MRF (12.3) and `learn_kcomp_heavytail_graph()` for the k-component heavy-tailed MRF (12.4) (Cardoso & Palomar, 2023).

12.2 Hierarchical Clustering and Dendrograms

Clustering is a multivariate statistical data analysis used in many fields, including machine learning, data mining, pattern recognition, bioinformatics, and financial markets. It is an unsupervised classification method that groups elements into clusters of similar characteristics based on the data.

Hierarchical clustering refers to the formation of a recursive nested clustering. It builds a binary tree of data points that represents nested groups of similar points based on a measure of distance. On the other hand, *partitional clustering* finds all the clusters simultaneously as a partition of the data points without imposing any hierarchical structure. One benefit of hierarchical clustering is that it allows exploring data on different levels of granularity.

A *dendrogram* is simply a visual representation of such a tree that encodes the successive or hierarchical clustering. It provides a highly interpretable complete description of the hierarchical clustering in a graphical format. This is one of the main reasons for the popularity of hierarchical clustering methods. For instance, in Example 12.1, items #1 and #2 have the smallest distance of 0.5659 and would be clustered together first, followed by item #3. This is illustrated in the dendrogram in Figure 12.1, which shows in a tree structure this bottom-up hierarchical clustering process.

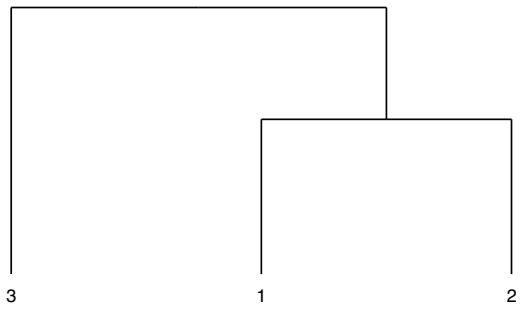


Figure 12.1 Dendrogram of toy Example 12.1.

12.2.1 Basic Procedure

Hierarchical clustering requires a distance matrix $\boldsymbol{D}$ and proceeds to sequentially cluster items based on distance (Hastie et al., 2009; James et al., 2013). Strategies for hierarchical clustering can be divided into two basic paradigms: agglomerative (bottom-up) and divisive (top-down). Bottom-up strategies start with every item representing a singleton cluster and then combine the clusters sequentially, reducing the number of clusters at each step until only one cluster is left. Alternatively, top-down strategies start with all the items in one cluster and recursively divide each cluster into smaller clusters. Each level of the hierarchy represents a particular grouping of the data into disjoint clusters of observations. The entire hierarchy represents an ordered sequence of such groupings.

Consider the bottom-up approach in which, at each step, the two closest (least dissimilar) clusters are merged into a single cluster, producing one less cluster at the next higher level.

This process requires a measure of dissimilarity between two clusters and different definitions of the distance between clusters can produce radically different dendrograms. In addition, there are different ways to merge the clusters, termed *linkage clustering*; most notably:[2]

- *single linkage*: the distance between two clusters is the minimum distance between any two points in the clusters, which is related to the so-called minimum spanning tree;
- *complete linkage*: the distance between two clusters is the maximum of the distance between any two points in the clusters;
- *average linkage*: the distance between two clusters is the average of the distance between any two points in the clusters; and
- *Ward's method*: the distance between two clusters is the increase of the squared error from when two clusters are merged, which is related to distances between centroids of clusters (Ward, 1963).

The effect of the linkage clustering method is very important as it significantly affects the hierarchical clustering obtained (Raffinot, 2018a). Single linkage tends to produce a "chaining" effect in the tree (described in Section 12.3.1) and an imbalance of large and small groups, whereas complete linkage typically produces more balanced groups, and average linkage is an intermediate case. Ward's method often produces similar results to average linkage. Figure 12.2 shows the dendrograms corresponding to the four different linkage methods for some S&P 500 stocks corresponding to different sectors during 2015–2020. Various numbers of clusters can be achieved by cutting the tree at distinct heights; for instance, four clusters can be acquired through suitable cuts, although they differ depending on the linkage methods used.

12.2.2 Number of Clusters

Traversing the dendrogram top-down, one starts with a giant cluster containing all the items all the way down to N singleton clusters, each containing a single item. In practice, however, it may not be necessary or convenient to deal with N singleton clusters, in order to avoid overfitting of the data.

Representing the data by fewer clusters necessarily loses certain fine details, but achieves simplification. However, choosing the best number of clusters is not easy: too few clusters and the algorithm may fail to find true patterns; too many clusters and it may discover spurious patterns that do not really exist. Thus, it is convenient to have an automatic way of detecting the most appropriate number of clusters to avoid overfitting.

One popular way to determine the number of clusters is the so-called *gap statistic* (Tibshirani et al., 2001). Basically, it compares the logarithm of the empirical within-cluster dissimilarity and the corresponding one for uniformly distributed data, which is a distribution with no obvious clustering.

[2] Hierarchical clustering is a fundamental method available in most programming languages; for example, the base function `hclust()` in R and the method `scipy.cluster.hierarchy.linkage()` from the Python library `scipy`.

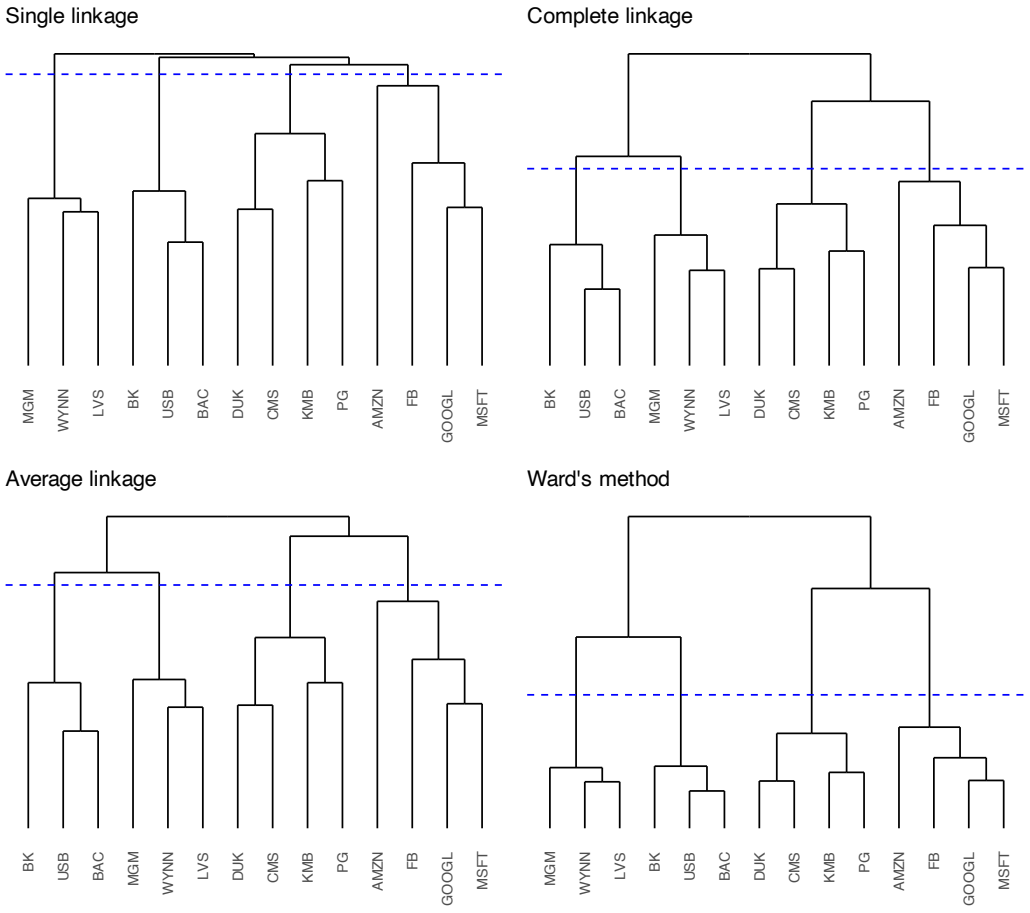


Figure 12.2 Dendrograms of S&P 500 stocks (with cut to produce four clusters).

12.2.3 Quasi-diagonalization of Correlation Matrix

Hierarchical clustering can be used to reorder the items in the correlation matrix so that similar assets (i.e., items that cluster early) are placed closer to each other, whereas dissimilar assets are placed far apart. This is called *matrix seriation* or *matrix quasi-diagonalization* and it is a very old statistical technique used to rearrange the data to show the inherent clusters clearly. This rearranges the original correlation matrix of assets into a quasi-diagonal correlation matrix, revealing similar assets as blocks along the main diagonal, whereas the correlation matrix of the original randomly ordered items presents no visual pattern.

To illustrate this process, Figure 12.3 shows the heatmaps of the original correlation matrix (with randomly ordered stocks) and the quasi-diagonal version (i.e., after a reordering of the stocks to better visualize the correlated stocks along the diagonal blocks) based on the hierarchical clustering (using single linkage). From the quasi-digonal correlation matrix one can clearly identify four clusters, which can also be observed in the dendrograms in Figure 12.2.

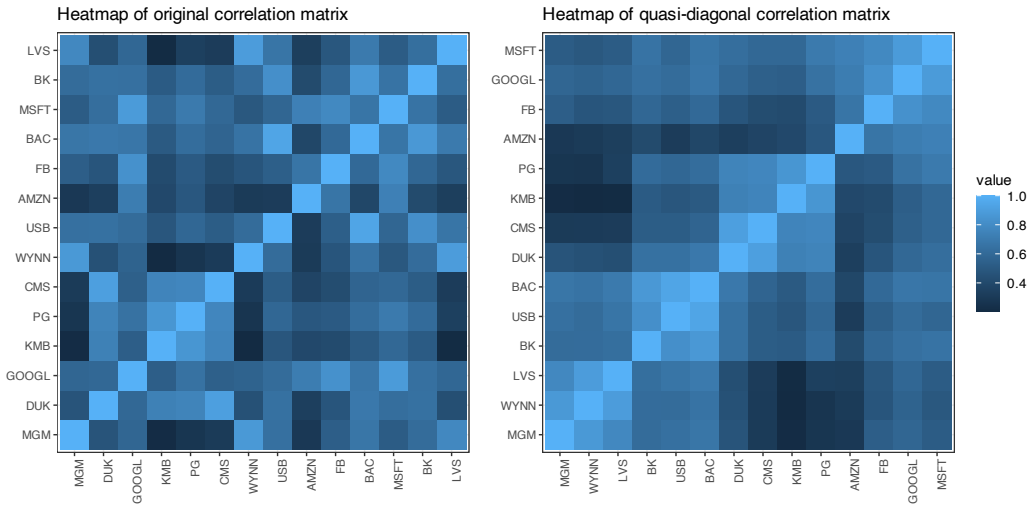


Figure 12.3 Effect of seriation in the correlation matrix of S&P 500 stocks.

12.3 Hierarchical Clustering-Based Portfolios

An alternative to using the noisy estimates of the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ in the mean–variance formulation is to design portfolios based on the graph of assets. This may produce more robust portfolios, more diversified, and with better risk-adjusted performance. Indeed, once the assets are hierarchically clustered, a variety of portfolios can be conveniently designed without relying so much on the noisy estimates of the mean vector and covariance matrix.

Hierarchical clustering aims to achieve diversification by a mechanism that distributes capital weights appropriately across the hierarchically nested clusters. The diversification scheme indicates which stocks are isolated far away from the bulk of stocks and should receive a higher weight as they are contributors to diversification. The whole process can be visualized based on the hierarchical tree layout.

The general description of hierarchical clustering-based portfolios is simple. The idea is to start with the total capital at the top and then proceed to allocate the capital through the hierarchical clustering along the dendrogram in a top-down manner. The starting point is the top giant cluster, which contains all the items, and the method proceeds down sequentially, where each cluster is successively divided into two sub-clusters. Every time a cluster is divided into two sub-clusters, the weight of the cluster has to be split into the two sub-clusters and the portfolios for the sub-clusters have to be designed.

The different hierarchical clustering-based portfolios (considered next in detail) are fully described by the following characteristics:

1. *distance matrix*: used to define the graph (e.g., correlation-based distance matrix in (12.1), distance matrix of the columns of such distance matrix as in (12.2), or obtained from some more sophisticated graph learning procedure);

2. *linkage method*: employed in the hierarchical clustering process (e.g., single, complete, average, or Ward);
3. *clustering stopping criterion*: used to stop the hierarchical clustering process (e.g., all the way down to single-item clusters or early stopping based on the gap statistic);
4. *splitting criterion*: to recursively split the assets (e.g., simply bisection or based on the dendrogram);
5. *intra-weight allocation*: used within clusters; and
6. *inter-weight allocation*: used across clusters.

The following subsections explore in detail several hierarchical clustering-based portfolios, namely, the hierarchical $1/N$ portfolio, the hierarchical risk parity portfolio, and the hierarchical equal risk contribution portfolio.

12.3.1 Hierarchical $1/N$ *Portfolio*

In 2011, a simple graph-based portfolio termed the *cluster-based waterfall portfolio* was proposed in Papenbrock (2011). The approach is based on the hierarchical tree obtained from the correlation-based distance matrix in (12.1). The allocation process goes over the dendrogram in a top-down manner, splitting the weights equally at each splitting point (i.e., the $1/N$ portfolio or equally weighted portfolio from Section 6.4.3). Figure 12.4 illustrates this top-down process of equal weight splitting.

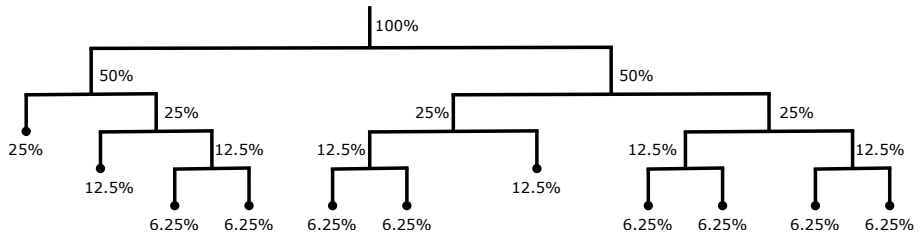


Figure 12.4 Illustration of the hierarchical $1/N$ portfolio construction in a top-down manner.

It is important to realize that the choice of linkage method has a significant effect on the concentration of the weight allocation, as illustrated in Figure 12.5. Basically, single linkage suffers from the "chaining" effect by which assets are branched out one by one, resulting in very high weights on some stocks; complete linkage produces more even groups and weights; average linkage lies somewhere in between, and Ward's method is similar to complete linkage producing slightly even more balanced groups and more equal weights. Note that the regular $1/N$ portfolio lies on the extreme of totally equalized weights, which amounts to not using any graph information at all, and is hence referred to as the *naive* $1/N$ portfolio, as opposed to the hierarchical $1/N$ portfolio, which is based on the graph information.

Summary of the hierarchical $1/N$ portfolio (Papenbrock, 2011):

1. *distance matrix*: correlation-based distance matrix $D_{ij} = \sqrt{\frac{1}{2}(1 - \rho_{ij})}$ as in (12.1);

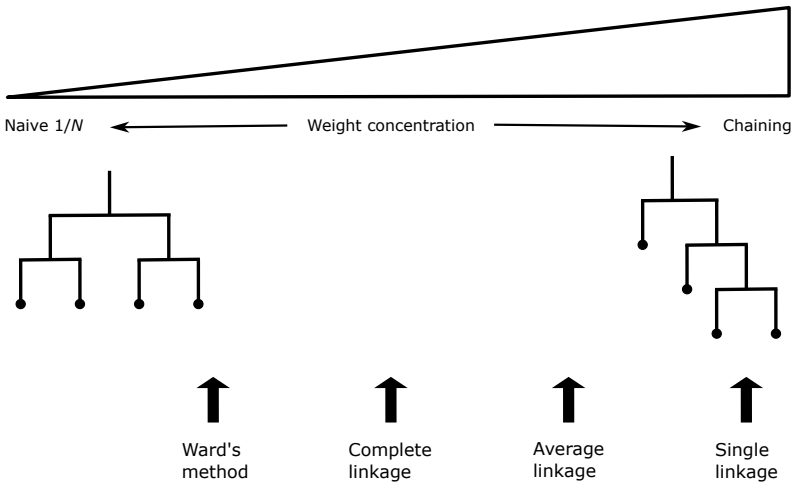


Figure 12.5 Chaining effect of different linkage methods on the hierarchical $1/N$ allocation.

2. *linkage method*: recommended methods are single linkage (for high-risk investors willing to take high stakes in single assets) and Ward's method (for risk-averse investors with less concentration on single assets);
3. *clustering stopping criterion*: all the way down to single-item clusters;
4. *splitting criterion*: following the dendrogram;
5. *intra-weight allocation*: $1/N$ portfolio; and
6. *inter-weight allocation*: $1/N$ portfolio with $N = 2$, that is, 50%–50% split at each branching.

Numerical Results

We compare different versions of the hierarchical $1/N$ portfolio along with some benchmarks. We first compare the hierarchical $1/N$ portfolio with the four different linkage methods along with the naive $1/N$ portfolio (illustrated in Figure 12.5). Figure 12.6 shows the portfolio weights: in this particular case the average linkage coincides with Ward's method. Figure 12.7 shows the cumulative P&L and drawdown of the portfolios for a single illustrative backtest: as already indicated in the original publication (Papenbrock, 2011), Ward's method is a good choice and will be used subsequently.

Now we compare the use of the correlation-based distance matrix in (12.1), which was the choice in Papenbrock (2011), with the correlation-based distance-of-distance matrix in (12.2), as used in López de Prado (2016), as well as the graphs estimated via (12.3) and (12.4). Figure 12.8 compares the portfolio allocations, whereas Figure 12.9 shows the cumulative P&L and drawdown of the portfolios for a single illustrative backtest. In this particular case, it seems that the simple correlation-based distance-of-distance matrix in (12.2) shows a better drawdown, but more exhaustive backtests are necessary before concluding anything.

Finally, we compare the selected version of the hierarchical $1/N$ portfolio (using Ward's

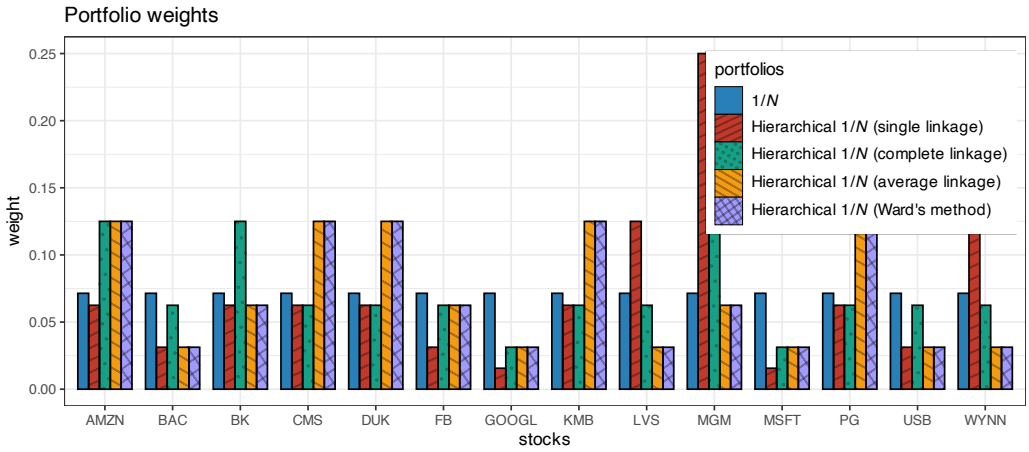


Figure 12.6 Portfolio allocation of hierarchical $1/N$ portfolios with different linkage methods.

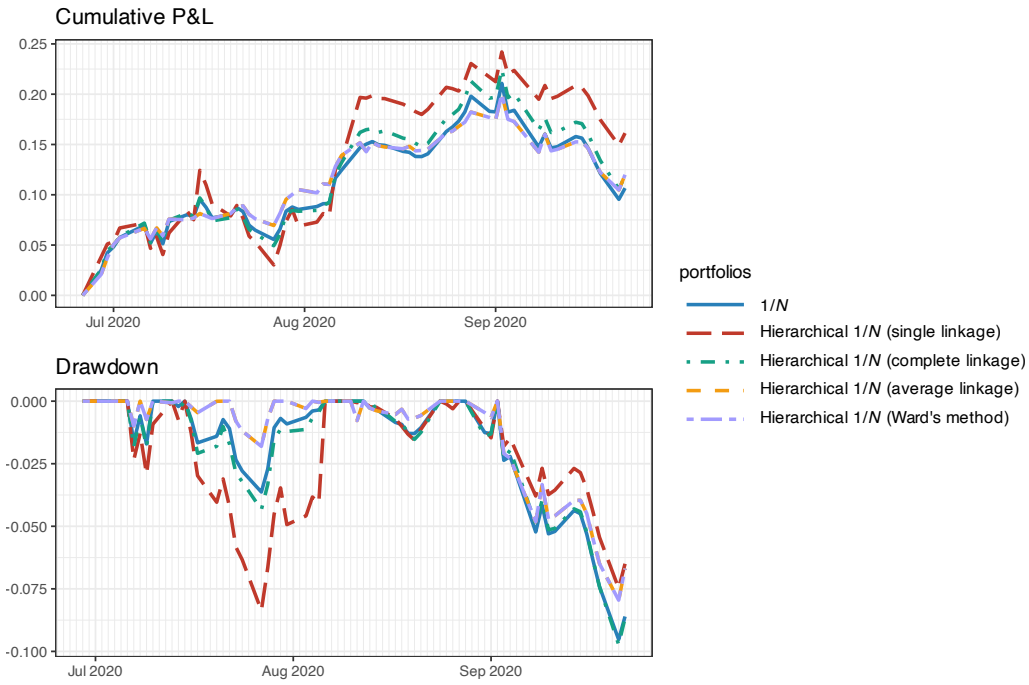


Figure 12.7 Backtest performance of hierarchical $1/N$ portfolios with different linkage methods.

method for the linkage and the correlation-based distance-of-distance matrix in (12.2)) with the following benchmarks: naive $1/N$ portfolio, global minimum variance portfolio (GMVP), and Markowitz mean–variance portfolio (MVP). Figure 12.10 compares the portfolio allocations, from where we can clearly see the difference in diversification. Figure 12.11 shows the cumulative P&L and drawdown of the portfolios for a single illustrative backtest: as expected,

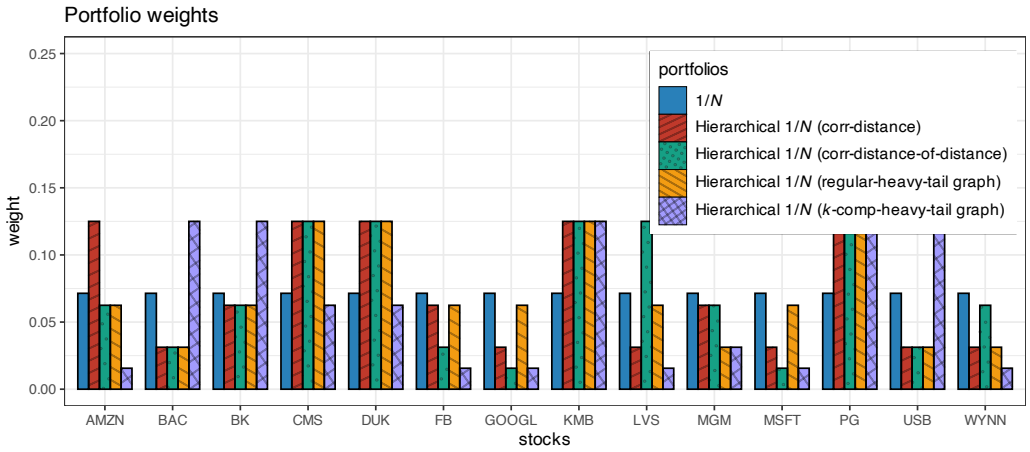


Figure 12.8 Portfolio allocation of hierarchical $1/N$ portfolios with different distance matrices.

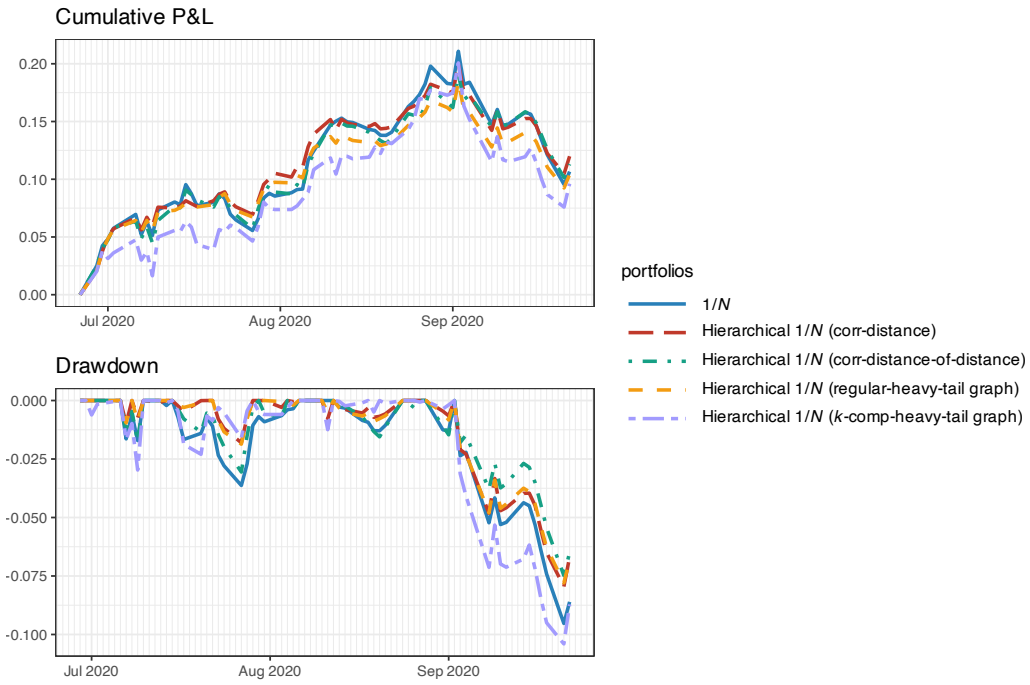


Figure 12.9 Backtest performance of hierarchical $1/N$ portfolios with different distance matrices.

the MVP shows the worst drawdown (due to the sensitivity to the estimation of $\boldsymbol{\mu}$), followed by GMVP and $1/N$ portfolio, with the hierarchical $1/N$ portfolio having the mildest drawdown. Nevertheless, the reader is reminded that this is just a single anecdotal backtest and a proper empirical evaluation based on multiple randomized backtests is necessary, as explored further in Section 12.4.

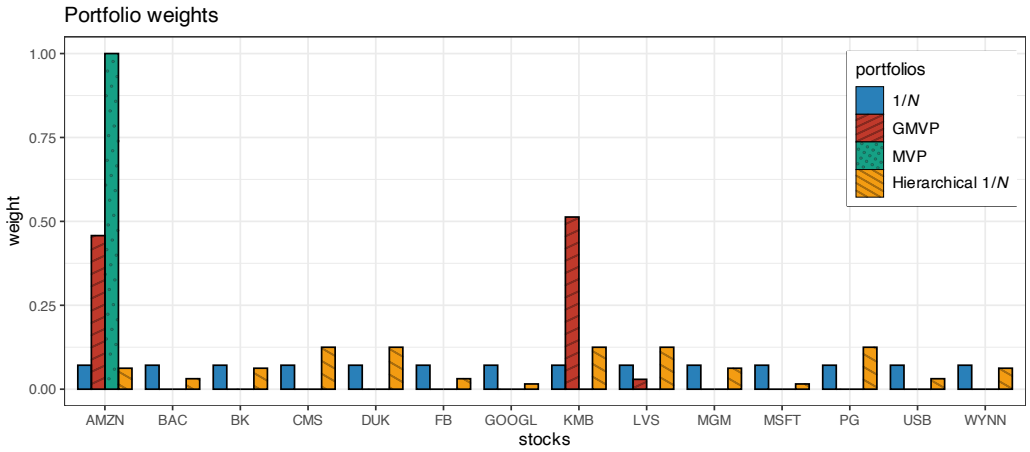


Figure 12.10 Portfolio allocation of hierarchical $1/N$ portfolio along benchmarks.

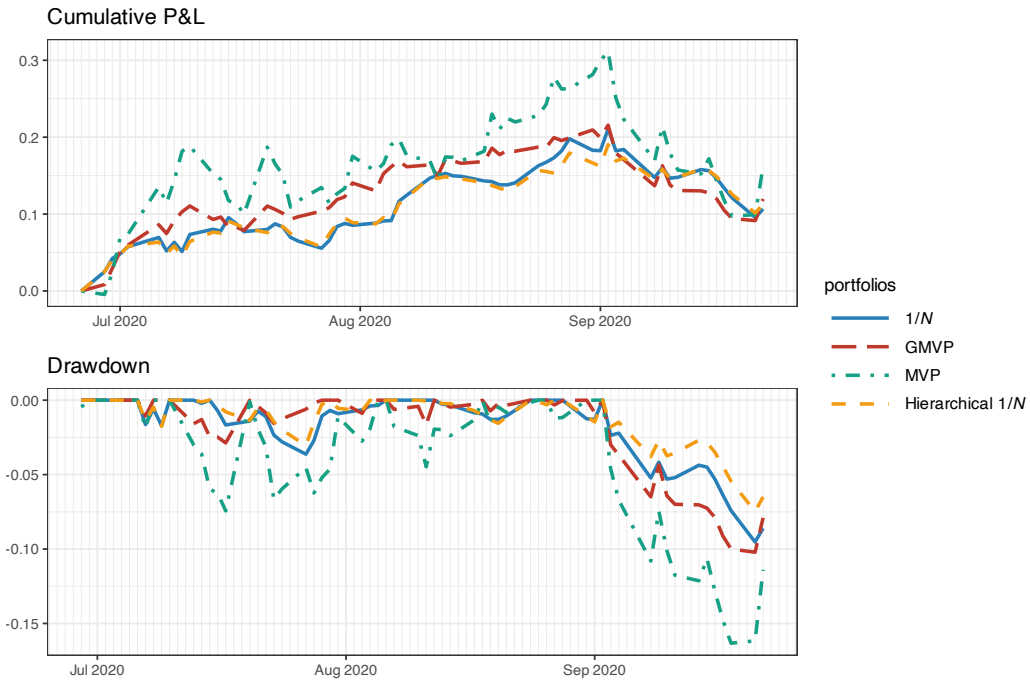


Figure 12.11 Backtest performance of hierarchical $1/N$ portfolio along benchmarks.

12.3.2 Hierarchical Risk Parity (HRP) Portfolio

In 2016, a graph-based portfolio was proposed (López de Prado, 2016) called the *hierarchical risk parity* (HRP) portfolio. The method is based on the hierarchical tree obtained from the correlation-based distance-of-distance matrix in (12.2) and the single linkage method. The

allocation process goes over the dendrogram in a top-down manner, splitting the weights based on the inverse-variance portfolio (IVarP).

Before going into the details of HRP, let us start by recalling that the global minimum variance portfolio with budget constraint is formulated (see Section 6.5.1 in Chapter 6) as

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w} \\ \text{subject to} \quad & \boldsymbol{w}^\top \mathbf{1} = 1, \end{aligned}$$

with solution

$$\boldsymbol{w} = \frac{\boldsymbol{\Sigma}^{-1}\mathbf{1}}{\mathbf{1}^\top \boldsymbol{\Sigma}^{-1}\mathbf{1}}.$$

If the covariance matrix $\boldsymbol{\Sigma}$ is diagonal, with diagonal elements denoted by $\boldsymbol{\sigma}^2 = \text{diag}(\boldsymbol{\Sigma})$, then the solution simplifies to the inverse-variance portfolio:

$$\boldsymbol{w} = \frac{\boldsymbol{\sigma}^{-2}}{\mathbf{1}^\top \boldsymbol{\sigma}^{-2}}, \tag{12.5}$$

which, for the particular case of $N = 2$ assets, becomes

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \sigma_1^{-2}/(\sigma_1^{-2} + \sigma_2^{-2}) \\ \sigma_2^{-2}/(\sigma_1^{-2} + \sigma_2^{-2}) \end{bmatrix} = \begin{bmatrix} \sigma_2^{2}/(\sigma_1^{2} + \sigma_2^{2}) \\ \sigma_1^{2}/(\sigma_1^{2} + \sigma_2^{2}) \end{bmatrix}. \tag{12.6}$$

We are now ready to go over the HRP portfolio design process. Similarly to the hierarchical $1/N$ portfolio, the allocation process goes over the dendrogram in a top-down manner, splitting the weights at each splitting point. A first difference in the HRP method proposed in López de Prado (2016) is that the splitting is based on bisection (i.e., dividing the items into two halves) rather than the natural structure given by the dendrogram, as illustrated in Figure 12.12. With bisection splitting, the dendrogram still contributes with the ordering of the assets. A priori, one cannot say which of the two methods will perform better and this will be empirically evaluated later. The second main difference is that instead of basing the weight splitting on the $1/N$ portfolio with $N = 2$, which leads to a 50%–50% split of the budget, the weight splitting is based on the IVarP with $N = 2$ in (12.6). In order to compute the variances of the two branches σ_1^2 and σ_2^2, the method uses, at each branch, the IVarP (12.5) (which presumably is not too far from the optimal GMVP since we are dealing with a quasi-diagonal matrix). Algorithm 12.1 formalizes HRP portfolio construction.

Summary of the HRP portfolio (López de Prado, 2016):

1. *distance matrix*: correlation-based distance-of-distance matrix in (12.2);
2. *linkage method*: single linkage;
3. *clustering stopping criterion*: all the way down to single-item clusters;
4. *splitting criterion*: directly bisection ignoring the grouping sizes from the dendrogram;
5. *intra-weight allocation*: IVarP; and
6. *inter-weight allocation*: IVarP for $N = 2$ as in (12.6).

The HRP portfolio can be interpreted as a refined version of the inverse-variance portfolio. Indeed, at each step, given two subsets, the portfolio weights are scaled based on the inverse of the variances while ignoring the correlation between the subsets (presumably these subsets are

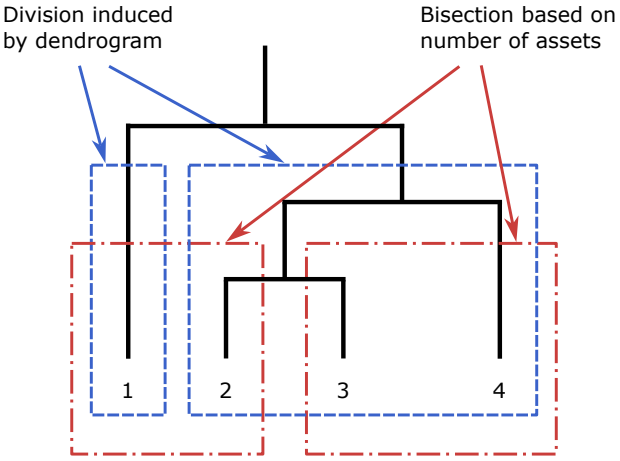


Figure 12.12 Comparison of bisection splitting and dendrogram-based splitting.

not highly correlated). The only point where correlations are considered is in the computation of the variances of each of the subsets. In fact, when the covariance matrix is diagonal, then the inverse-variance portfolio, the global minimum variance portfolio, and the HRP portfolio coincide. A connection between the HRP portfolio and the minimum-variance portfolio is further outlined in Section 12.3.4.

Algorithm 12.1: HRP portfolio.

Initialize list of groups of items $\mathcal{L} = \{L_0\}$, with $L_0 = \{1, \ldots, N\}$, and unit weights $\boldsymbol{w} = \mathbf{1}$;
repeat
  Choose a tuple $L_i \in \mathcal{L}$ with $|L_i| > 1$;
  Partition L_i into two subsets, $L_i^{(1)}$ and $L_i^{(2)}$; two options are:
  - bisection split: following the dendrogram ordering, simply split it into two subsets of approximately equal size: $|L_i^{(1)}| = \text{round}(|L_i|/2)$;
  - dendrogram split: follow the dendrogram split of this set into two subsets (i.e., not only ordering but size of subsets too);

  Define the variance of each subset $L_i^{(j)}$, $j = 1, 2$, as $(\tilde{\sigma}_i^2)^{(j)} = \tilde{\boldsymbol{w}}_i^{(j)\mathsf{T}} \boldsymbol{\Sigma}_i^{(j)} \tilde{\boldsymbol{w}}_i^{(j)}$, where $\boldsymbol{\Sigma}_i^{(j)}$ is the covariance matrix of the elements in $L_i^{(j)}$ and $\tilde{\boldsymbol{w}}_i^{(j)}$ is the inverse-variance portfolio in (12.5);
  Compute the split factor as in (12.6): $\alpha_i = (\tilde{\sigma}_i^2)^{(2)} / ((\tilde{\sigma}_i^2)^{(1)} + (\tilde{\sigma}_i^2)^{(2)})$;
  Rescale elements $n \in L_i^{(1)}$ in $\boldsymbol{w}$ by a factor of α_i and elements $n \in L_i^{(2)}$ in $\boldsymbol{w}$ by a factor of $(1 - \alpha_i)$;
until $|L_i| = 1, \forall L_i \in L$;

Numerical Results

We now compare the HRP portfolios (both with bisection split and dendrogram split) with the following benchmarks: global minimum variance portfolio (GMVP) and inverse-variance portfolio (IVarP).

Figure 12.13 shows the portfolio weights. The GMVP concentrates most of the weights into two assets, whereas the rest are more diversified. The HRP portfolios do not differ too much from the IVarP. Figure 12.14 shows the cumulative P&L and drawdown of the portfolios for a single illustrative backtest. The HRP portfolios seem to slightly improve on the IVarP; in addition, the graphs estimated via (12.3) and (12.4) seem to be better than the correlation-based distance-of-distance matrix in (12.2).

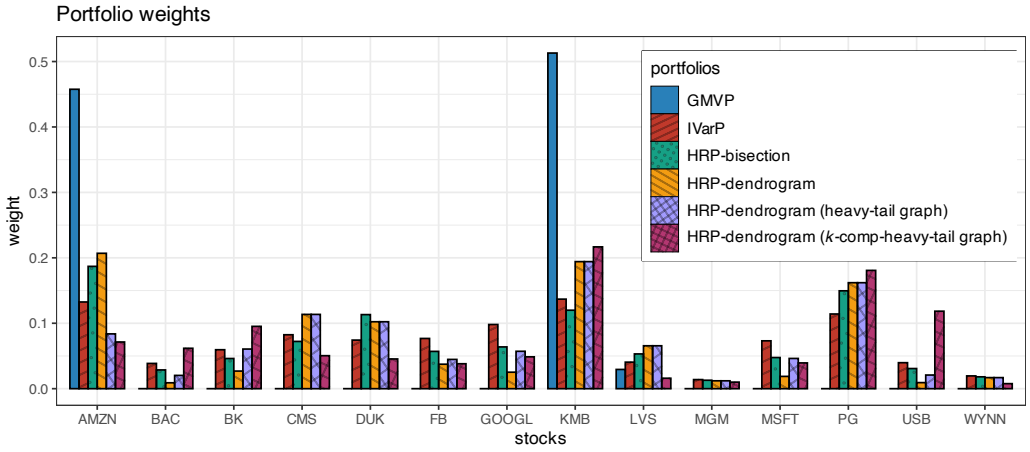


Figure 12.13 Portfolio allocation of HRP portfolios and benchmarks.

12.3.3 Hierarchical Equal Risk Contribution (HERC) Portfolio

In 2018, a refined and extended version of the hierarchical $1/N$ portfolio of Section 12.3.1 and the HRP portfolio of Section 12.3.2 was proposed in (Raffinot, 2018b) called the *hierarchical equal risk contribution* (HERC) portfolio. Basically, the two main differences are:

- Early stopping in the hierarchical clustering to automatically select the appropriate number of clusters based on the gap statistic (Tibshirani et al., 2001), a technique already used in hierarchical clustering asset allocation (Raffinot, 2018a), as opposed to clustering all the way down to single assets as typically done in hierarchical clustering.

- Use of a general equal risk contribution to split the weights at each splitting point (which can be based on alternative measures of risk to the variance used in (12.6), such as standard deviation, conditional value-at-risk, conditional drawdown-at-risk, and so on:

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \mathrm{RC}_1/(\mathrm{RC}_1 + \mathrm{RC}_2) \\ \mathrm{RC}_2/(\mathrm{RC}_1 + \mathrm{RC}_2) \end{bmatrix}, \tag{12.7}$$

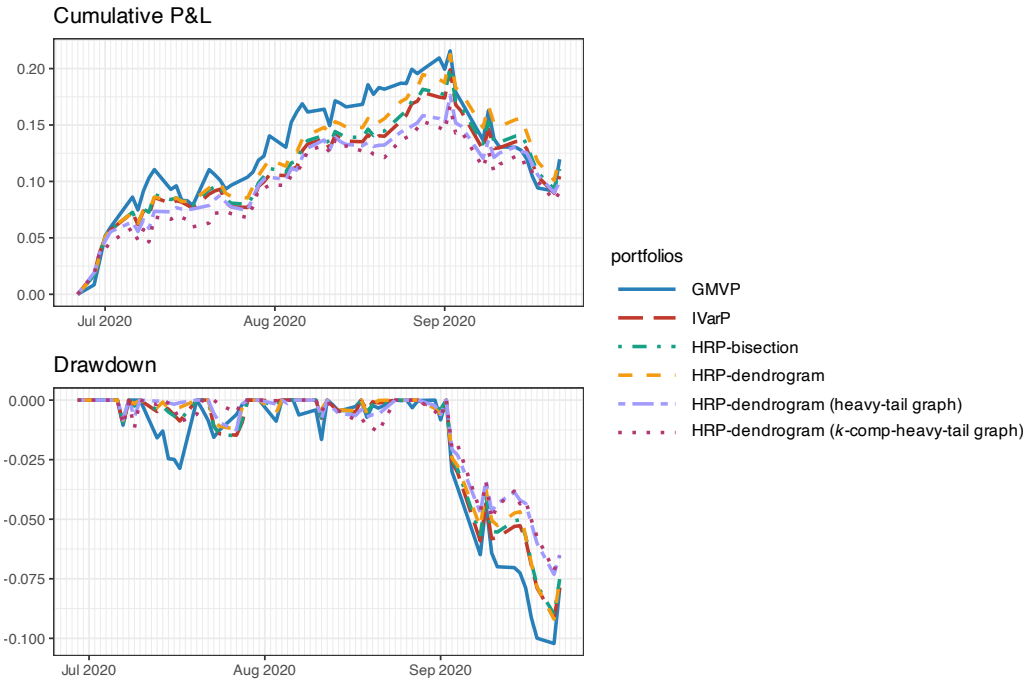


Figure 12.14 Backtest performance of HRP portfolios and benchmarks.

where RC_i denotes the risk contribution of of the ith cluster. Observe that if $\text{RC}_i = 1/\sigma_i^2$, then we recover (12.6).

The main findings in Raffinot (2018b) can be summarized as follows:

> [T]he hierarchical $1/N$ portfolio is difficult to beat, but HERC portfolios based on downside risk measures achieve statistically better risk-adjusted performances, especially those based on the conditional drawdown-at-risk.

Figure 12.15 illustrates the effect of early stopping in the hierarchical clustering process with a hierarchical $1/N$ portfolio. In particular a toy dendrogram is considered with five assets grouped into three and two clusters.

Summary of the HERC portfolio (Raffinot, 2018b):

1. *distance matrix*: correlation-based distance-of-distance matrix in (12.2);
2. *linkage method*: Ward's method;
3. *clustering stopping criterion*: gap statistic (Tibshirani et al., 2001);
4. *splitting criterion*: following the dendrogram;
5. *intra-weight allocation*: $1/N$ portfolio; and
6. *inter-weight allocation*: equal risk constribution as in (12.7).

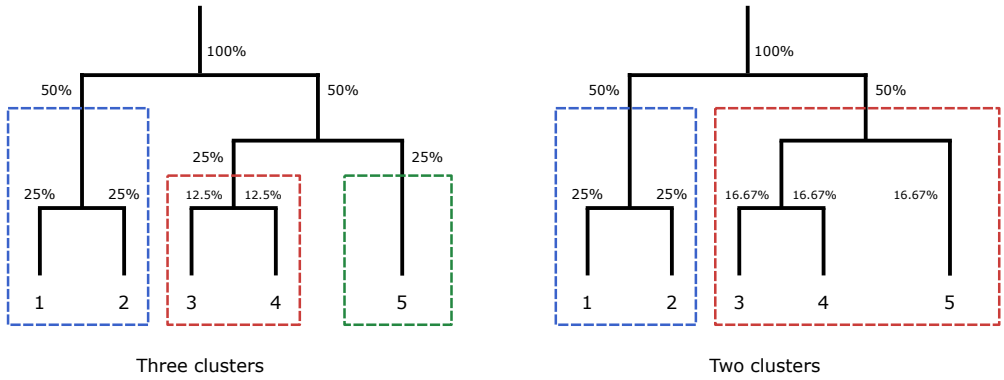


Figure 12.15 Effect of early stopping in the hierarchical clustering process.

Numerical Results

For simplicity, for the weight splitting we simply use the two risk contribution measures, $\mathrm{RC}_i = 1$ and $\mathrm{RC}_i = 1/\sigma_i^2$, leading to the 50%–50% split as in the hierarchical $1/N$ portfolio and to (12.6) as in the HRP portfolio.

We now compare the HERC portfolios (both with bisection split and dendrogram split) with the following benchmarks: $1/N$ portfolio, hierarchical $1/N$ portfolio, inverse-variance portfolio (IVarP), and HRP portfolio. Figure 12.16 shows the portfolio weights, whereas Figure 12.17 shows the cumulative P&L and drawdown of the portfolios for a single illustrative backtest. It is difficult to conclude anything from this single backtest and more exhaustive backtests are necessary.

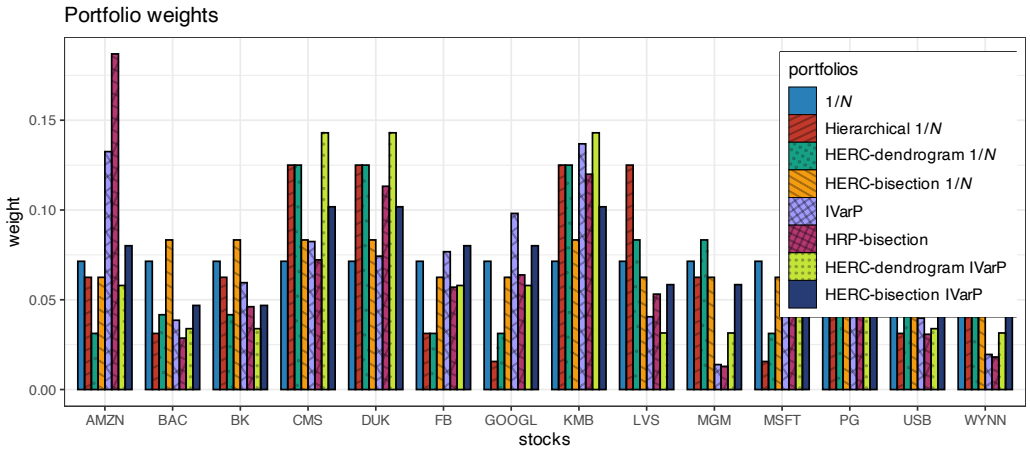


Figure 12.16 Portfolio allocation of HERC portfolios and benchmarks.

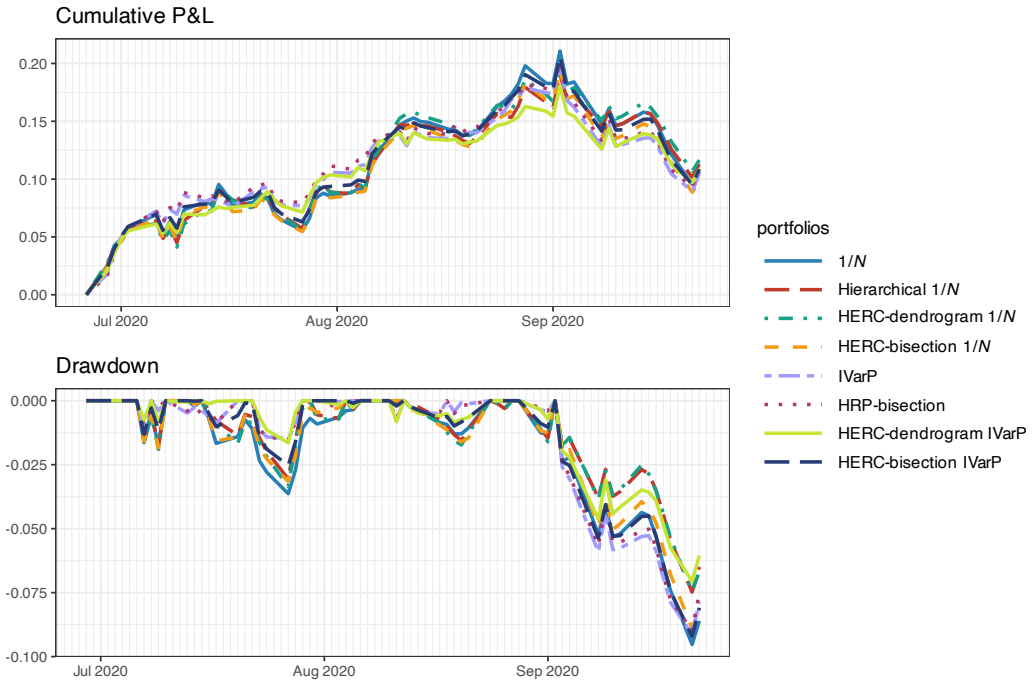


Figure 12.17 Backtest performance of HERC portfolios and benchmarks.

12.3.4 From Portfolio Risk Minimization to Hierarchical Portfolios

The essence of the hierarchical portfolios covered in this section lies in successive partitioning of the set of assets with a proper inter-set capital allocation and intra-set portfolio allocation. To be more specific, at each round of this successive partitioning, we partition a set of assets, with covariance matrix $\boldsymbol{\Sigma}$, into two subsets A and B, with covariance matrices $\boldsymbol{\Sigma}_A$ and $\boldsymbol{\Sigma}_B$, respectively. Then, the portfolio is constructed as

$$\boldsymbol{w} \propto \begin{bmatrix} \frac{1}{\nu(\boldsymbol{\Sigma}_A)} \boldsymbol{w}(\boldsymbol{\Sigma}_A) \\ \frac{1}{\nu(\boldsymbol{\Sigma}_B)} \boldsymbol{w}(\boldsymbol{\Sigma}_B) \end{bmatrix}, \tag{12.8}$$

where $\nu(\cdot)$ and $\boldsymbol{w}(\cdot)$ denote some measure of risk and portfolio construction, respectively, for a set of assets with given covariance matrix. The first component $1/\nu(\cdot)$ provides the capital allocation into the two subsets (inversely proportional to the risk), whereas the second component $\boldsymbol{w}(\cdot)$ gives the (normalized) portfolio allocation for each subset.

The particular form of $\nu(\cdot)$ and $\boldsymbol{w}(\cdot)$ in (12.8) depends on each type of hierarchical portfolio. For example, in the case of the HRP portfolio covered in Section 12.3.2:

- the intra-weight (normalized) allocation is given by the inverse-variance portfolio (12.5),

$$\boldsymbol{w}(\boldsymbol{\Sigma}) = \frac{\boldsymbol{\sigma}^{-2}}{\mathbf{1}^\top \boldsymbol{\sigma}^{-2}},$$

  where $\boldsymbol{\sigma}^2 = \text{diag}(\boldsymbol{\Sigma})$ denotes the variance of the assets, and

- the measure of risk (whose inverse gives the capital or inter-weight allocation) is measured by the variance (12.6),

$$\nu(\boldsymbol{\Sigma}) = \boldsymbol{w}(\boldsymbol{\Sigma})^{\mathsf{T}} \boldsymbol{\Sigma}\, \boldsymbol{w}(\boldsymbol{\Sigma}).$$

However, the basic structure of hierarchical portfolios (12.8) is heuristic and suboptimal, which is understandable since the motivation was not optimality but stability against estimation errors. One reason for the lack of optimality can be seen in the fact that the portfolios in each of the two groups, $\boldsymbol{w}(\boldsymbol{\Sigma}_A)$ and $\boldsymbol{w}(\boldsymbol{\Sigma}_B)$, are oblivious to each other and the two groups are only jointly adjusted in the final normalization step of (12.8) (note the notation $\propto$ which requires a normalization that depends on all the elements). On the other hand, portfolios designed based on the minimization of some properly chosen measure of risk are not heuristic by definition but optimal according to the design criterion. Can we make an explicit connection between the two paradigms? Can we somehow derive an expression similar to (12.8) but more formally justified and optimally chosen?

A positive answer was provided in Cotton (2023) as described next. Consider the global minimum variance portfolio (see Section 6.5.1 in Chapter 6):

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{w} \\ \text{subject to} & \boldsymbol{w}^{\mathsf{T}}\mathbf{1} = 1. \end{array}$$

Interestingly, by partitioning $\boldsymbol{\Sigma}$ as

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_A & \boldsymbol{\Sigma}_{AB} \\ \boldsymbol{\Sigma}_{BA} & \boldsymbol{\Sigma}_B \end{bmatrix},$$

the optimal solution can be written (up to a scaling factor) in a form reminiscent to (12.8) as

$$\boldsymbol{w} \propto \boldsymbol{\Sigma}^{-1}\mathbf{1} = \begin{bmatrix} (\boldsymbol{\Sigma}_A^{\mathrm{c}})^{-1}\boldsymbol{b}_A \\ (\boldsymbol{\Sigma}_B^{\mathrm{c}})^{-1}\boldsymbol{b}_B \end{bmatrix},$$

where $\boldsymbol{\Sigma}_A^{\mathrm{c}}$ and $\boldsymbol{\Sigma}_B^{\mathrm{c}}$ are the Schur complements of $\boldsymbol{\Sigma}_A$ and $\boldsymbol{\Sigma}_B$,[3] respectively,

$$\begin{aligned} \boldsymbol{\Sigma}_A^{\mathrm{c}} &= \boldsymbol{\Sigma}_A - \boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_B^{-1}\boldsymbol{\Sigma}_{BA}, \\ \boldsymbol{\Sigma}_B^{\mathrm{c}} &= \boldsymbol{\Sigma}_B - \boldsymbol{\Sigma}_{BA}\boldsymbol{\Sigma}_A^{-1}\boldsymbol{\Sigma}_{AB}, \end{aligned}$$

and

$$\begin{aligned} \boldsymbol{b}_A &= \mathbf{1} - \boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_B^{-1}\mathbf{1}, \\ \boldsymbol{b}_B &= \mathbf{1} - \boldsymbol{\Sigma}_{BA}\boldsymbol{\Sigma}_A^{-1}\mathbf{1}. \end{aligned}$$

Finally, this optimal solution can be further rewritten in a form similar to (12.8) as

$$\boldsymbol{w} \propto \begin{bmatrix} \frac{1}{\nu(\boldsymbol{\Sigma}_A^{\mathrm{c}}, \boldsymbol{b}_A)} \boldsymbol{w}(\boldsymbol{\Sigma}_A^{\mathrm{c}}, \boldsymbol{b}_A) \\ \frac{1}{\nu(\boldsymbol{\Sigma}_B^{\mathrm{c}}, \boldsymbol{b}_B)} \boldsymbol{w}(\boldsymbol{\Sigma}_B^{\mathrm{c}}, \boldsymbol{b}_B) \end{bmatrix}, \tag{12.9}$$

[3] The Schur complement of a block matrix is a key tool in the fields of numerical analysis, statistics, and matrix analysis. For our motivation, it suffices to know that it naturally appears in the inverse of a 2×2 matrix:

$$\begin{bmatrix} \boldsymbol{\Sigma}_A & \boldsymbol{\Sigma}_{AB} \\ \boldsymbol{\Sigma}_{BA} & \boldsymbol{\Sigma}_B \end{bmatrix}^{-1} = \begin{bmatrix} \left(\boldsymbol{\Sigma}_A - \boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_B^{-1}\boldsymbol{\Sigma}_{BA}\right)^{-1} & 0 \\ 0 & \left(\boldsymbol{\Sigma}_B - \boldsymbol{\Sigma}_{BA}\boldsymbol{\Sigma}_A^{-1}\boldsymbol{\Sigma}_{AB}\right)^{-1} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_B^{-1} \\ \boldsymbol{\Sigma}_{BA}\boldsymbol{\Sigma}_A^{-1} & \boldsymbol{I} \end{bmatrix}.$$

where the (normalized) allocation is

$$\boldsymbol{w}(\boldsymbol{\Sigma}, \boldsymbol{b}) = \frac{\boldsymbol{\Sigma}^{-1}\boldsymbol{b}}{\boldsymbol{b}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{b}}$$

and the measure of risk is

$$v(\boldsymbol{\Sigma}, \boldsymbol{b}) = (\boldsymbol{w}(\boldsymbol{\Sigma}, \boldsymbol{b}))^{\mathsf{T}}\, \boldsymbol{\Sigma}\, \boldsymbol{w}(\boldsymbol{\Sigma}, \boldsymbol{b}) = \frac{1}{\boldsymbol{b}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{b}}.$$

At this point it is important to realize that optimal portfolio in (12.9) becomes a hierarchical portfolio (12.8) if one replaces $\boldsymbol{\Sigma}_A^{\mathrm{c}} \to \boldsymbol{\Sigma}_A$ and $\boldsymbol{b}_A \to \mathbf{1}$ (and similarly for $\boldsymbol{\Sigma}_B^{\mathrm{c}}$ and $\boldsymbol{b}_B$). This observation naturally leads to the idea of transitioning from one set of parameters to the other in a smooth way to provide a continuum ranging from a hierarchical portfolio to a minimum risk portfolio as some scalar parameter γ goes from 0 to 1. There are multiple ways of doing such a smooth transition; some examples include:

- convex combination:

$$\begin{aligned}\boldsymbol{\Sigma}_A^{(\gamma)} &= (1-\gamma)\boldsymbol{\Sigma}_A + \gamma\boldsymbol{\Sigma}_A^{\mathrm{c}} = \boldsymbol{\Sigma}_A - \gamma\boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_B^{-1}\boldsymbol{\Sigma}_{BA},\\ \boldsymbol{b}_A^{(\gamma)} &= (1-\gamma)\mathbf{1} + \gamma\boldsymbol{b}_A = \mathbf{1} - \gamma\boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_B^{-1}\mathbf{1};\end{aligned}$$

- geodesic-convex combination (Wiesel, 2012):

$$\begin{aligned}\boldsymbol{\Sigma}_A^{(\gamma)} &= \boldsymbol{\Sigma}_A^{1/2}\left(\boldsymbol{\Sigma}_A^{-1/2}\boldsymbol{\Sigma}_A^{\mathrm{c}}\boldsymbol{\Sigma}_A^{-1/2}\right)^{\gamma}\boldsymbol{\Sigma}_A^{1/2},\\ \boldsymbol{b}_A^{(\gamma)} &= \boldsymbol{b}_A^{\gamma},\end{aligned}$$

  where the matrix power is defined via the power of the eigenvalues.

12.4 Numerical Experiments

After looking at several graph-based portfolios and observing their performance over single backtests, we now resort to an empirical evaluation based on multiple randomized backtests. In particular, we take a dataset of the S&P 500 stocks over the period 2015–2020 and generate 200 resamples each with $N = 50$ randomly selected stocks and a random period of two years. Then, for each resample, we perform a walk-forward backtest with a lookback window of one year, reoptimizing the portfolio every month. Note that more exhaustive backtests should be conducted with a wide variety of data before drawing any firm conclusion.

12.4.1 Splitting: Bisection vs. Dendrogram

We start by comparing the two splitting methods considered: direct bisection (after reordering the assets with the dendrogram) and following the natural splits of the dendrogram, as illustrated in Figure 12.12. In principle, one may expect that following the natural dendrogram should be more faithful to the structure in the actual data as opposed to bisection, which somehow breaks the dendrogram. Interestingly, the empirical evaluation in Figure 12.18 seems to suggest otherwise. One possible explanation is that bisection provides more balanced clusters while still using the reordering information provided by the dendrogram.

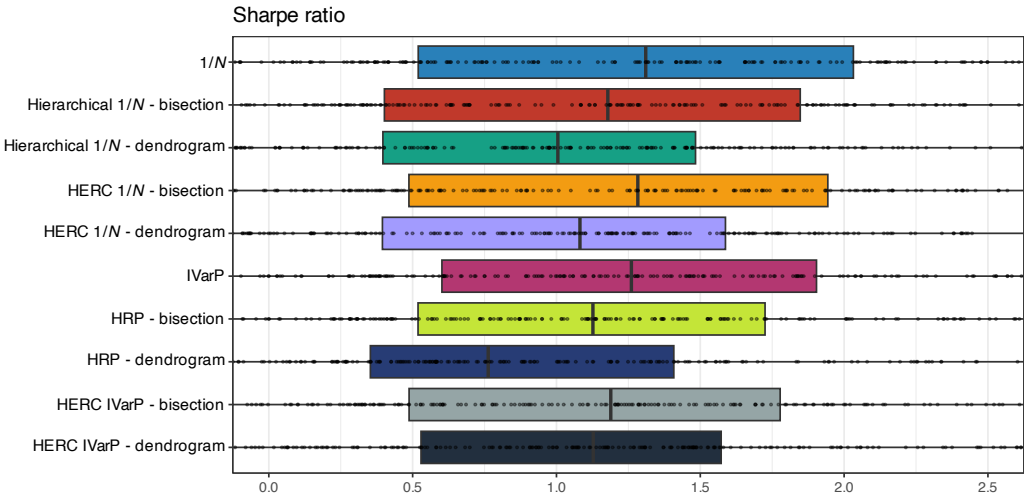


Figure 12.18 Comparison of graph-based portfolios: bisection vs. dendrogram splitting.

12.4.2 Graph Estimation: Simple vs. Sophisticated

Then we evaluate whether the more sophisticated graph estimation methods can bring any benefit to the portfolios. In principle, one may expect a positive answer, but the empirical results in Figure 12.19 show that this is not so clear and further analysis is required.

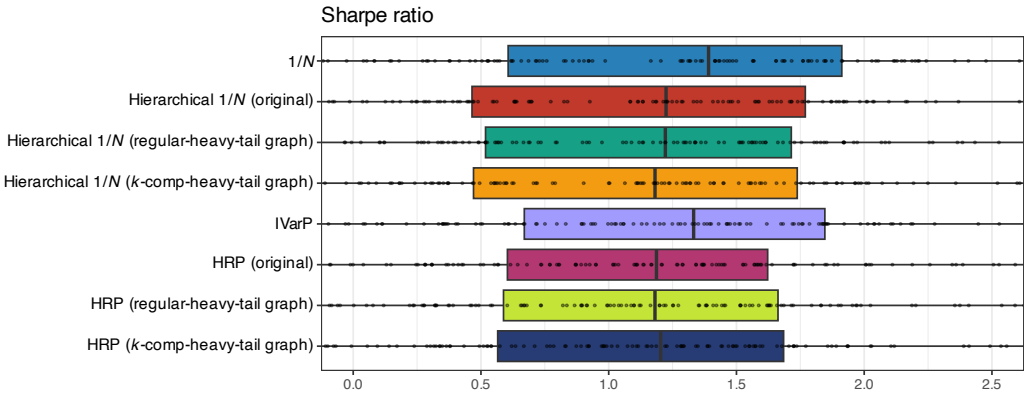


Figure 12.19 Comparison of graph-based portfolios: simple vs. sophisticated graph learning methods.

12.4.3 Final Comparison

Finally, we perform a comparison of the graph-based portfolios (using the simple graph-based distance-of-distance matrix in (12.2)), along with the benchmarks $1/N$ portfolio and IVarP:

- hierarchical $1/N$ portfolio

- HERC $1/N$ portfolio
- HRP portfolio
- HERC IVarP.

The empirical results in Table 12.1 and Figures 12.20–12.21 show that the various methods do not appear significantly different and a more exhaustive comparison is needed before drawing any clear conclusion.

Table 12.1 Comparison of selected graph-based portfolios: performance measures.

Portfolio	Sharpe ratio	Annual return	Annual volatility	Sortino ratio	Max drawdown	CVaR (0.95)
$1/N$	1.01	14%	14%	1.39	11%	2%
Hierarchical $1/N$	0.81	12%	14%	1.12	11%	2%
HERC $1/N$	0.99	14%	14%	1.31	11%	2%
IVarP	1.04	13%	12%	1.44	10%	2%
HRP	0.91	11%	12%	1.26	10%	2%
HERC IVarP	0.89	12%	13%	1.21	10%	2%

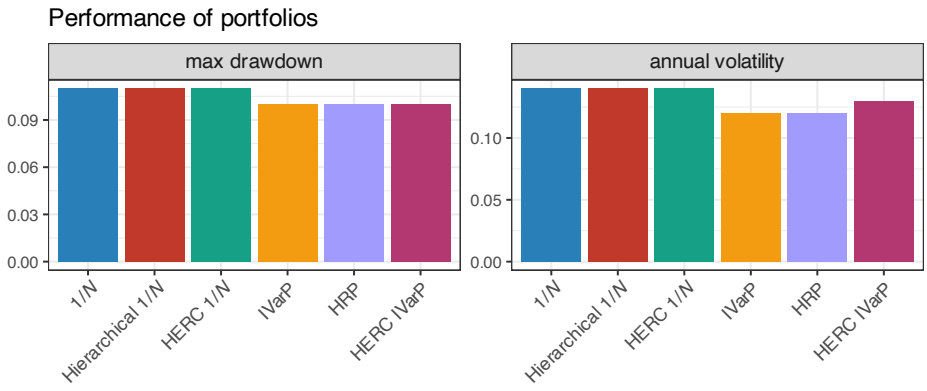


Figure 12.20 Comparison of selected graph-based portfolios: barplots of maximum drawdown and annualized volatility.

12.5 Summary

Graphs provide a convenient and compact way to represent big data, exposing the underlying structure and existing patterns that may otherwise go unnoticed. In the context of portfolio design, the following key takeaway points can be identified.

- In finance, graphs can represent the relationship among assets: each asset is a node and their pairwise relationships are represented by edges of different strength.
- Financial graphs can be learned automatically from collected data (see Chapter 5). The recommended graph learning formulations are the heavy-tailed Markov random field (12.3) or, if a clustered graph is desired, the k-component version (12.4).

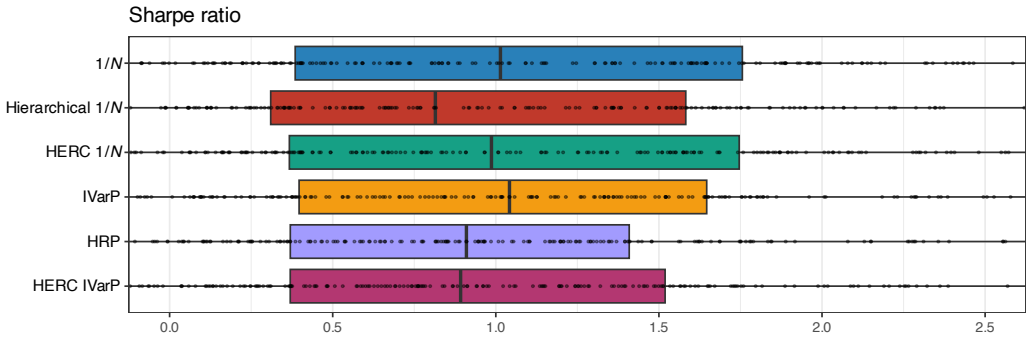


Figure 12.21 Comparison of selected graph-based portfolios: boxplots of Sharpe ratio.

- Hierarchical clustering methods can be used to partition the assets into clusters with different levels of detail from the graph information.
- Graph information of assets should be taken into account in the portfolio formulation. Further work needs to be done, but some notable examples include: the hierarchical $1/N$ portfolio, the hierarchical risk parity portfolio, and the hierarchical equal risk contribution portfolio.

Exercises

12.1 (Learning heavy-tailed financial graphs)

a. Download market data corresponding to N assets (e.g., stocks or cryptocurrencies) during a period with T observations, and form the data matrix $\boldsymbol{X} \in \mathbb{R}^{T \times N}$.

b. Learn a Gaussian MRF graph:

$$\begin{array}{ll} \underset{\boldsymbol{w} \geq \boldsymbol{0}}{\text{maximize}} & \log \operatorname{gdet}(\mathcal{L}(\boldsymbol{w})) - \operatorname{Tr}(\mathcal{L}(\boldsymbol{w})\boldsymbol{S}) \\ \text{subject to} & \mathfrak{d}(\boldsymbol{w}) = \boldsymbol{1}, \end{array}$$

where $\boldsymbol{S}$ is the sample covariance matrix of the data, $\mathcal{L}(\boldsymbol{w})$ is the Laplacian operator that produces the Laplacian matrix $\boldsymbol{L}$ from the weights $\boldsymbol{w}$, and $\mathfrak{d}(\boldsymbol{w})$ is the degree operator that gives the degrees of the nodes.

c. Learn a heavy-tailed MRF graph directly:

$$\begin{array}{ll} \underset{\boldsymbol{w} \geq \boldsymbol{0}}{\text{maximize}} & \log \operatorname{gdet}(\mathcal{L}(\boldsymbol{w})) - \dfrac{N+\nu}{T} \displaystyle\sum_{t=1}^{T} \log\left(\nu + (\boldsymbol{x}^{(t)})^{\mathsf{T}} \mathcal{L}(\boldsymbol{w}) \boldsymbol{x}^{(t)}\right) \\ \text{subject to} & \mathfrak{d}(\boldsymbol{w}) = \boldsymbol{1}, \end{array}$$

where $\boldsymbol{x}^{(t)}$ is the tth row of the data matrix $\boldsymbol{X}$.

d. Learn a heavy-tailed MRF graph by solving the sequence of Gaussianized problems for

$k = 1, 2, \ldots$

$$\begin{array}{ll} \underset{\boldsymbol{w} \geq \mathbf{0}}{\text{maximize}} & \log \operatorname{gdet}(\mathcal{L}(\boldsymbol{w})) - \operatorname{Tr}(\mathcal{L}(\boldsymbol{w})\boldsymbol{S}^k) \\ \text{subject to} & \mathfrak{d}(\boldsymbol{w}) = \mathbf{1}, \end{array}$$

where $\boldsymbol{S}^k$ is the weighted sample covariance matrix

$$\boldsymbol{S}^k = \frac{1}{T} \sum_{t=1}^{T} w_t^k \times \boldsymbol{x}^{(t)} (\boldsymbol{x}^{(t)})^\mathsf{T},$$

with weights $w_t^k = \dfrac{N + \nu}{\nu + (\boldsymbol{x}^{(t)})^\mathsf{T} \mathcal{L}(\boldsymbol{w}^k) \boldsymbol{x}^{(t)}}$.

e. Plot the graphs and compare them visually.

f. Compute the dendrogram for each of the graphs and compare them.

12.2 (Hierarchical $1/N$ portfolio)

a. Download market data corresponding to N assets during a period with T observations, and form the data matrix $\boldsymbol{X} \in \mathbb{R}^{T \times N}$.
b. Learn the graph distance matrix based on (i) a simple distance matrix from the distance between the time series of asset pairs, and (ii) a heavy-tailed MRF graph formulation.
c. Construct the hierarchical $1/N$ portfolio.
d. Plot the portfolio allocation and perform a backtest (comparing with the $1/N$ portfolio).

12.3 (Hierarchical risk parity (HRP) portfolio)

a. Download market data corresponding to N assets during a period with T observations, and form the data matrix $\boldsymbol{X} \in \mathbb{R}^{T \times N}$.
b. Learn the graph distance matrix based on (i) a simple distance matrix from the distance between the time series of asset pairs, and (ii) a heavy-tailed MRF graph formulation.
c. Construct the HRP portfolio.
d. Plot the portfolio allocation and perform a backtest (comparing with the inverse-variance portfolio).

12.4 (Hierarchical equal risk contribution (HERC) portfolio)

a. Download market data corresponding to N assets during a period with T observations, and form the data matrix $\boldsymbol{X} \in \mathbb{R}^{T \times N}$.
b. Learn the graph distance matrix based on (i) a simple distance matrix from the distance between the time series of asset pairs, and (ii) a heavy-tailed MRF graph formulation.
c. Construct the HERC portfolio.
d. Plot the portfolio allocation and perform a backtest (comparing with the hierarchical $1/N$ portfolio and HRP portfolio).

12.5 (From minimum variance portfolio to hierarchical portfolio)

a. Derive the inverse of the 2×2 block matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_A & \boldsymbol{\Sigma}_{AB} \\ \boldsymbol{\Sigma}_{BA} & \boldsymbol{\Sigma}_B \end{bmatrix},$$

identifying the Schur complements of $\boldsymbol{\Sigma}_A$ and $\boldsymbol{\Sigma}_B$ defined, respectively, as

$$\boldsymbol{\Sigma}_A^{\mathrm{c}} = \boldsymbol{\Sigma}_A - \boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_B^{-1}\boldsymbol{\Sigma}_{BA},$$
$$\boldsymbol{\Sigma}_B^{\mathrm{c}} = \boldsymbol{\Sigma}_B - \boldsymbol{\Sigma}_{BA}\boldsymbol{\Sigma}_A^{-1}\boldsymbol{\Sigma}_{AB}.$$

b. Derive the global minimum variance portfolio

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w} \\ \text{subject to} & \boldsymbol{w}^\top \mathbf{1} = 1 \end{array}$$

in the form (up to a scaling factor)

$$\boldsymbol{w} \propto \boldsymbol{\Sigma}^{-1}\mathbf{1} = \begin{bmatrix} (\boldsymbol{\Sigma}_A^{\mathrm{c}})^{-1}\boldsymbol{b}_A \\ (\boldsymbol{\Sigma}_B^{\mathrm{c}})^{-1}\boldsymbol{b}_B \end{bmatrix},$$

where

$$\boldsymbol{b}_A = \mathbf{1} - \boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_B^{-1}\mathbf{1},$$
$$\boldsymbol{b}_B = \mathbf{1} - \boldsymbol{\Sigma}_{BA}\boldsymbol{\Sigma}_A^{-1}\mathbf{1}.$$

c. Rewrite the solution in the form

$$\boldsymbol{w} \propto \begin{bmatrix} \frac{1}{\nu(\boldsymbol{\Sigma}_A^{\mathrm{c}}, \boldsymbol{b}_A)} \boldsymbol{w}\left(\boldsymbol{\Sigma}_A^{\mathrm{c}}, \boldsymbol{b}_A\right) \\ \frac{1}{\nu(\boldsymbol{\Sigma}_B^{\mathrm{c}}, \boldsymbol{b}_B)} \boldsymbol{w}\left(\boldsymbol{\Sigma}_B^{\mathrm{c}}, \boldsymbol{b}_B\right) \end{bmatrix},$$

for properly defined (normalized) allocation $\boldsymbol{w}(\boldsymbol{\Sigma}, \boldsymbol{b})$ and measure of risk $\nu(\boldsymbol{\Sigma}, \boldsymbol{b})$.

References

Cardoso, J. V. M., & Palomar, D. P. (2023). *fingraph: Learning Graphs for Financial Markets* [R package]. https://CRAN.R-project.org/package=fingraph

Cardoso, J. V. M., Ying, J., & Palomar, D. P. (2021). Graphical models in heavy-tailed markets. *Proceedings of 35th International Conference on Neural Information Processing Systems (NeurIPS)*, 19989–20001.

Cotton, P. (2023). *Schur Complementary Portfolios – A Unification of Machine Learning and Optimization-based Portfolio Construction* (tech. rep.) (See also article available at https://medium.com/geekculture/schur-complementary-portfolios-fix-hierarchical-risk-parity-28b0efa1f35f).

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: With Applications in R*. Springer.

López de Prado, M. (2016). Building diversified portfolios that outperform out of sample. *Journal of Portfolio Management*, *42*(4), 59–69.

Mantegna, R. N. (1999). Hierarchical structure in financial markets. *The European Physical Journal B – Condensed Matter and Complex Systems*, *11*, 193–197.

Markowitz, H. M. (1952). Portfolio selection. *The Journal of Finance*, *7*(1), 77–91.

Papenbrock, J. (2011). *Asset Clusters and Asset Networks in Financial Risk Management and Portfolio Optimization* [Doctoral dissertation, Karlsruher Institute für Technologie].

Raffinot, T. (2018a). Hierarchical clustering-based asset allocation. *The Journal of Portfolio Management*, *44*(2), 89–99.

Raffinot, T. (2018b). The hierarchical equal risk contribution portfolio. *SSRN Electronic Journal*. https://dx.doi.org/10.2139/ssrn.3237540

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, *63*(2), 411–423.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, *58*(301), 236–244.

Wiesel, A. (2012). Geodesic convexity and covariance estimation. *IEEE Transactions on Signal Processing*, *60*(12), 6182–6189.

13

Index Tracking Portfolios

If you have assumed a character beyond your strength, you have both played a part ill, and have fallen in an unbecoming manner: you have gone aground.

— Epictetus, *The Enchiridion*

"I never met another man I'd rather be. And even if that's a delusion, it's a lucky one."

— Charles Bukowski

Can we outsmart the market? The efficient-market hypothesis states that the price of a security already contains all the publicly available information about the future (Fama, 1970), although another line of thought supports precisely the opposite view in favor of inefficient and irrational markets (Shiller, 1981). In any case, beating the market is routinely promised by investment funds, hedge funds, financial experts, but do they keep up their promises? Empirical analysis of data shows that about 95% of funds do not outperform the market (Malkiel, 1973).

This chapter explores the topic of market or index tracking as an alternative to active investment strategies that attempt to beat the market: from heuristic and discretionary approaches, to more sophisticated optimization formulations, and even the most recent techniques to automatically choose the number of active assets in a statistically controlled way.

13.1 Active vs. Passive Strategies

Strategies followed by investors can be classified into two main families:

- *Active strategies*: the premise is that the market is not perfectly efficient (Shiller, 1981) and through expertise one can add value by choosing high-performing assets and beat the market performance. Examples include the portfolios paradigms presented in the previous chapters, namely: mean–variance portfolios (Chapter 7), high-order portfolios (Chapter 9), portfolios with alternative measures of risk (Chapter 10), risk parity portfolios (Chapter 11), and graph-based portfolios (Chapter 12).

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

- *Passive strategies*: the assumption is that the market is efficient (Fama, 1970), in the sense that prices reflect all available information, and as a consequence the market cannot be beaten in the long run (Malkiel, 1973). Passive investing methods seek to avoid the fees and limited performance that may occur with frequent trading and instead focus on infrequent rebalancing. This chapter deals with the problem of index tracking as a way of passive investment (Benidis et al., 2018a; Prigent, 2007).

13.1.1 Beating the Market

Can investors consistently beat the market? One only hears from the winners. Nobody writes articles celebrating the worst-performing mutual fund manager. However, to perform a proper data analysis, we also need to collect data from the losers and separate luck from skill.

Data clearly shows that individual investors who trade stocks directly pay a tremendous performance penalty for active trading (B. M. Barber & Odean, 2000). Perhaps expert financial managers can do better?

The percentage of fund managers who do not outperform the index can vary depending on the time period, the index being used as a benchmark, and the specific group of fund managers being analyzed. However, numerous studies have found that around 85%–95% of actively managed mutual funds run by professional fund managers fail to outperform their respective benchmark indices over the long term (Malkiel, 1973).

Some interesting and even provocative conclusions include (Malkiel, 1973):

> investors are far better off buying and holding an index fund than attempting to buy and sell individual securities or actively managed mutual funds;

and

> the market prices stocks so efficiently that a blindfolded chimpanzee throwing darts at the stock listings can select a portfolio that performs as well as those managed by the experts.

As a consequence, most investors have figured out that they are not good at stock-picking or managing trades; most professionals are not either. Thus, paying high mutual fund expenses to a manager who underperforms a benchmark makes little sense. This realization has led to the rise of indices and inexpensive *exchange-traded funds* (ETFs).

13.1.2 What is a Financial Index?

A financial index is defined as a collection of carefully selected assets to capture the value of a specific market or a segment of it. An index is effectively equivalent to a hypothetical portfolio of assets; however, one cannot invest directly in it, that is, an index is not a financial instrument that can be traded.

An index is defined by the universe of assets composing it and also the percentage of the composition. The most common type of index follows a capitalization-weighted (cap-weighted) approach:[1] the assets are weighted based on the ratio of their market capitalization (number

[1] Apart from the common cap-weighted indices (weighted according to the market capitalization), there are other

of outstanding shares multiplied by share price) to the overall capitalization of the assets that compose the index. The index value is then proportional to the weighted average of the capitalization of the underlying assets.

Standard & Poor's 500 (S&P 500) is one of the world's best-known (cap-weighted) indices and one of the most commonly used benchmarks for the U.S. stock market. Figure 13.1 shows the price of the S&P 500 index over more than a decade. As can be observed, the S&P 500 has historically risen and reasonable returns can be obtained simply by following the market without active risk management (this still holds after adjusting the price for inflation assuming a 2% annual inflation rate). Note that this period includes two severe crises: the 2007–2008 global financial crisis and the COVID-19 recession (starting around February 2020).

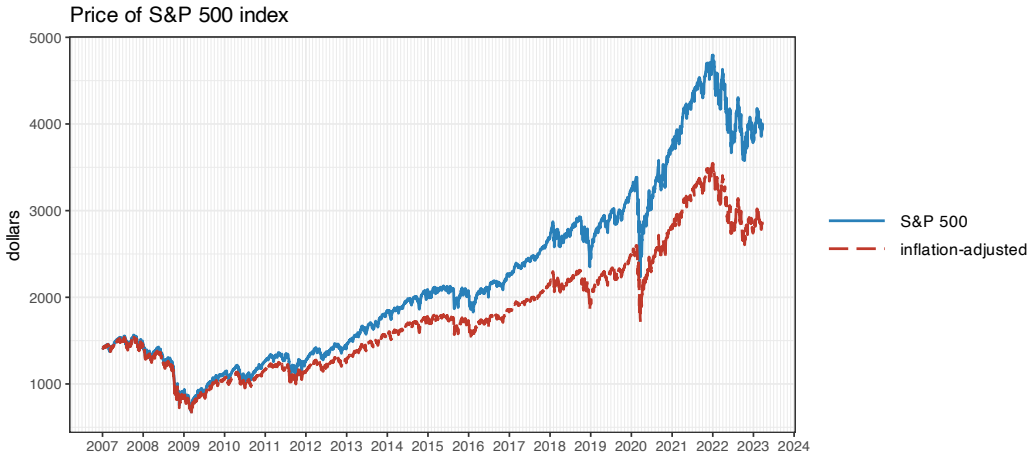


Figure 13.1 Price evolution of the S&P 500 index.

While it is difficult to give a precise number of existing financial indices, as new indices are created and old indices are retired or replaced regularly, it is safe to say that there are thousands of financial indices that cover a wide range of asset classes, sectors, and regions. Some of the most well-known indices include the S&P 500 (USA), Dow Jones Industrial Average (USA), Nasdaq Composite (USA), Hang Seng Index (Hong Kong), Shanghai Shenzhen CSI 300 Index (mainland China), Nikkei 225 (Japan), FTSE 100 (UK), DAX (Germany), and IBEX 35 (Spain), among many others.

13.1.3 Index Tracking

An index is just a definition based on a hypothetical portfolio of assets and cannot be directly traded. In practice, one has to construct a real portfolio that tracks or mimics the index as closely as possible. *Index tracking*, also known as *index replication*, is a passive portfolio management strategy that attempts to reproduce the performance of a market index.

In the current financial markets, there are a staggering number of ETFs that precisely track

types of index construction methods, such as price weighted, equal weighted, fundamentally weighted, and factor weighted.

any given index and investors can directly trade them. However, these ETFs still have to be constructed by financial managers. For example, as of 2023, there are over 250 ETFs that track the S&P 500 index, with the most popular one being the SPDR S&P 500 ETF, which happens to be the largest and oldest ETF in the world. Figure 13.2 shows the S&P 500 index together with the SPDR S&P 500 ETF (under the ticker SPY). As can be observed from the plot, its value is approximately 1/10 of the cash S&P 500 level. Nevertheless, this ratio is not exactly maintained in the long run, as it goes approximately from 1/14 to 1/10 over time during 2007–2023. This is not critical since for hedging purposes it is the short-term variations that matter.

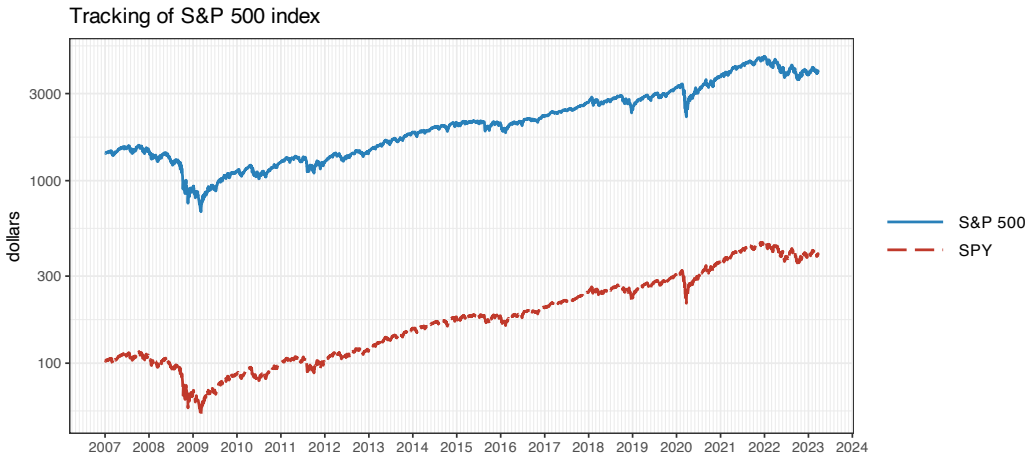


Figure 13.2 Tracking of the S&P 500 index by the SPDR S&P 500 ETF.

The most straightforward approach to create a tracking portfolio is by buying appropriate quantities of all the assets that compose the index, so-called *full replication*. This technique requires knowledge of the precise index composition, which is not always available in real time. It also needs a regular rebalancing of all the portfolio positions as the index composition is updated (including the less liquid assets), which translates into transaction costs.

Due to the aforementioned transaction cost reasons and also logistic arguments (managing a smaller portfolio is more convenient), it is advantageous from a practical standpoint to hold active positions only on a *reduced basket* of representative assets of the whole index universe. For example, instead of keeping an active portfolio of 500 assets to track the S&P 500 index, it may be better and simpler to invest in, say, only 20 assets properly selected to represent the index. This is referred to as *sparse index tracking* or *portfolio compression* (Benidis et al., 2018a, 2018b; Jansen & Van Dijk, 2002; Machkour, Palomar, & Muma, 2025; Maringer & Oyewumi, 2007; Scozzari et al., 2013; Xu et al., 2016). Interestingly, this problem is related to sparse regression techniques in statistics, as described next.

13.2 Sparse Regression

A vector is *sparse* if it has many elements equal to zero. Sparsity is a very important property in a wide variety of problems where proper control of the sparsity level is desired (Elad, 2010).

The *cardinality* of a vector $\boldsymbol{x} \in \mathbb{R}^N$, denoted by $\text{card}(\boldsymbol{x})$, refers to the number of nonzero elements:

$$\text{card}(\boldsymbol{x}) \triangleq \sum_{i=1}^{N} 1\{x_i \neq 0\},$$

where $1\{\cdot\}$ denotes the indicator function. It is often written as the ℓ_0-pseudo-norm $\|\boldsymbol{x}\|_0$, which is not a norm (in fact, it is not even convex).

13.2.1 Problem Formulation

Sparse regression refers to a regression problem with the additional requirement that the solution has to be sparse. Some common formulations of sparse regression includes:

- Regularized sparse least squares (LS):

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda \|\boldsymbol{x}\|_0,$$

  where the parameter λ can be chosen to enforce more or less sparsity in the solution.

- Constrained sparse LS:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \\ \text{subject to} \quad & \|\boldsymbol{x}\|_0 \leq k, \end{aligned}$$

  where k is a parameter to control the sparsity level.

- Sparse underdetermined system of linear equations:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \|\boldsymbol{x}\|_0 \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{aligned}$$

  where the system of linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is underdetermined (so that it has an infinite number of solutions).

Unfortunately, the cardinality operator is noncontinuous, nondifferentiable, and nonconvex, which means that developing practical algorithms under sparsity is not trivial. As a matter of fact, this has been a well-researched topic for decades and mature approximate methods are currently available, as described next.

13.2.2 Methods for Sparse Regression

The most common approach to deal with the cardinality operator is to approximate it with a more convenient form for optimization methods. Figure 13.3 shows the following two approximations (along with the indicator function):

- ℓ_1-*norm approximation*: $\|\boldsymbol{x}\|_0$ is simply approximated by $\|\boldsymbol{x}\|_1$. In the univariate case, it means that the indicator function $1\{t \neq 0\}$ is approximated by the absolute value $|t|$.
- *Concave approximation*: $\|\boldsymbol{x}\|_0$ is better approximated by a concave function (rendering the sparse regression problem nonconvex). In the univariate case, it means that the indicator function $1\{t \neq 0\}$ is approximated by a concave function, such as the log-function $\log(1+t/\varepsilon)$, where ε is a parameter controlling the accuracy of the approximation, although many other concave approximation have been proposed (Candès et al., 2008).

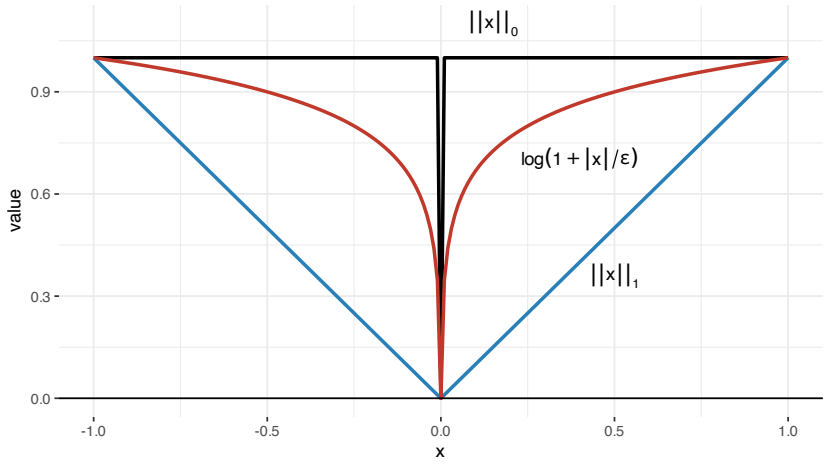


Figure 13.3 Indicator function and approximations.

ℓ_1-*Norm Approximation*

The most popular algorithm for sparse regression is undoubtedly the *LASSO* (least absolute shrinkage and selection operator) (Tibshirani, 1996), which solves the following convex quadratic problem based on the ℓ_1-norm approximation:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \\ \text{subject to} & \|\boldsymbol{x}\|_1 \leq t, \end{array}$$

where t is a parameter to control the sparsity level.

Another popular method for sparse regression is the *elastic net* (Zou & Hastie, 2005), which overcomes some of the limitations of the LASSO. In particular, if groups of variables are highly correlated, the LASSO tends to select one variable from a group and ignore the others. The elastic net is a regularized regression method that linearly combines the ℓ_1 and ℓ_2 penalties of the LASSO and ridge methods.

It is worth noting that nonnegativity constraints alone can also induce sparsity in the solution (Meinshausen, 2013).

Similarly, the sparse resolution of an underdetermined system of linear equations can be recovered in practice, under some technical conditions, as a linear program (Candès & Tao,

2005; Donoho, 2006):

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \|\boldsymbol{x}\|_1 \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{array}$$

Concave Approximation

In some cases, the ℓ_1-norm approximation is not good enough and one needs to resort to a more refined concave approximation (as illustrated in Figure 13.3):

$$\|\boldsymbol{x}\|_0 \approx \sum_{i=1}^{N} \phi(|x_i|),$$

where $\phi(\cdot)$ is an appropriate concave function, such as the popular log-function

$$\phi(t) = \log(1 + t/\varepsilon).$$

Alternative concave approximations can be used (Candès et al., 2008), such as the arc-tangent function or the ℓ_p-norm for $0 < p < 1$.

However, a concave approximation leads to a nonconvex problem formulation, such as

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda \sum_{i=1}^{N} \log\left(1 + \frac{|x_i|}{\varepsilon}\right).$$

Fortunately, we can use the majorization–minimization framework to derive iterative methods to solve problems with concave approximations of the cardinality, as explored next. Other families of algorithms have also been successfully employed such as *iteratively reweighted least squares* (IRLS) minimization (Daubechies et al., 2010).

13.2.3 Preliminaries on MM

The majorization–minimization (MM) method (or framework) approximates a difficult optimization problem by a sequence of simpler approximated problems. We now give a concise description. For details, the reader is referred to Section B.7 in Appendix B, as well as the concise tutorial in Hunter and Lange (2004), the long tutorial with applications in Sun et al. (2017), and the convergence analysis in Razaviyayn et al. (2013).

Suppose the following (difficult) problem is to be solved:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathcal{X}, \end{array}$$

where $f(\cdot)$ is the (possibly nonconvex) objective function and $\mathcal{X}$ is a (possibly nonconvex) set. Instead of attempting to directly obtain a solution $\boldsymbol{x}^\star$ (either a local or global solution), the MM method will produce a sequence of iterates $\boldsymbol{x}^0, \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots$ that will converge to $\boldsymbol{x}^\star$.

More specifically, at iteration k, MM approximates the objective function $f(\boldsymbol{x})$ by a surrogate function around the current point $\boldsymbol{x}^k$ (essentially, a tangent upper bound), denoted by $u\left(\boldsymbol{x}; \boldsymbol{x}^k\right)$, leading to the sequence of (simpler) problems:

$$\boldsymbol{x}^{k+1} = \underset{\boldsymbol{x} \in \mathcal{X}}{\operatorname{argmin}}\, u\left(\boldsymbol{x}; \boldsymbol{x}^k\right), \qquad k = 0, 1, 2, \ldots$$

In order to guarantee convergence of the iterates, the surrogate function $u\left(\boldsymbol{x};\boldsymbol{x}^k\right)$ has to satisfy the following technical conditions (Razaviyayn et al., 2013; Sun et al., 2017):

- *upper bound property*: $u\left(\boldsymbol{x};\boldsymbol{x}^k\right) \geq f\left(\boldsymbol{x}\right)$;
- *touching property*: $u\left(\boldsymbol{x}^k;\boldsymbol{x}^k\right) = f\left(\boldsymbol{x}^k\right)$; and
- *tangent property*: $u\left(\boldsymbol{x};\boldsymbol{x}^k\right)$ must be differentiable with $\nabla u\left(\boldsymbol{x}^k;\boldsymbol{x}^k\right) = \nabla f\left(\boldsymbol{x}^k\right)$, that is, the gradients of u and f must coincide at $\boldsymbol{x}^k$ (not at every $\boldsymbol{x}$).

The surrogate function $u\left(\boldsymbol{x};\boldsymbol{x}^k\right)$ is also referred to as a *majorizer* because it is an upper bound of the original function. The fact that, at each iteration, first the majorizer is constructed and then it is minimized gives the name *majorization–minimization* to the method.

13.2.4 Iterative Reweighted ℓ_1-Norm Minimization

We now employ the MM framework to derive an iterative algorithm to solve sparse regression problems. For illustration purposes, we will focus on the following formulation with a concave approximation of the cardinality operator:

$$\begin{array}{ll}\underset{\boldsymbol{x}}{\text{minimize}} & \displaystyle\sum_{i=1}^{N} \log\left(1+\frac{|x_i|}{\varepsilon}\right) \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}.\end{array} \tag{13.1}$$

Recall that many other concave functions can be used similarly (Candès et al., 2008).

In order to use MM, we need a key component: a majorizer of the concave function $\log(1+t/\varepsilon)$, which means that it has to be an upper-bound tangent on the point of interest.

Lemma 13.1 (Majorizer of the log function) *The concave function $\phi(t) = \log(1 + t/\varepsilon)$ is majorized at $t = t_0$ by its linearization:*

$$\phi(t) \leq \phi(t_0) + \phi(t_0)'(t - t_0) = \phi(t_0) + \frac{1}{\varepsilon + t_0}(t - t_0).$$

According to Lemma 13.1, the term $\log\left(1 + |x_i|/\varepsilon\right)$ is majorized at x_i^k (up to an irrelevant constant) by $\alpha_i^k\,|x_i|$ with weight $\alpha_i^k = 1/(\varepsilon + |x_i^k|)$.

Concluding, we can solve the nonconvex problem (13.1) by solving the following sequence of convex problems (Candès et al., 2008), for $k = 0, 1, 2, \ldots$:

$$\begin{array}{ll}\underset{\boldsymbol{x}}{\text{minimize}} & \displaystyle\sum_{i=1}^{N} \alpha_i^k |x_i| \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b},\end{array} \tag{13.2}$$

where the objective function is a weighted ℓ_1-norm with weights $\alpha_i^k = 1/(\varepsilon + |x_i^k|)$. That is, the concave approximation of the sparse regression in (13.1) has been effectively solved by a sequence of iterative reweighted ℓ_1-norm minimization problems as in (13.2) (Candès et al., 2008).

13.3 Sparse Index Tracking

The problem of sparse index tracking is in fact an instance of sparse regression and a variety of methods have been proposed in the literature to deal with the difficulty of cardinality constraint or sparsity control (Benidis et al., 2018a, 2018b; Jansen & Van Dijk, 2002; Maringer & Oyewumi, 2007; Scozzari et al., 2013; Xu et al., 2016).

13.3.1 Tracking Error

The return of an index or benchmark r_t^{b} is obtained from the returns of the N constituent assets $\boldsymbol{r}_t \in \mathbb{R}^N$ via the definition of the index portfolio $\boldsymbol{b}_t > \mathbf{0}$ (normalized to $\mathbf{1}^{\mathsf{T}}\boldsymbol{b}_t = 1$) as

$$\boldsymbol{r}_t^{\mathsf{T}}\boldsymbol{b}_{t-1} = r_t^{\mathrm{b}}, \quad t = 1, \ldots, T, \tag{13.3}$$

where the vector $\boldsymbol{b}_t$ denotes the proportion of capital allocated to the assets (see Section 6.1.2 for details of the notation). Note that it is also possible to define a portfolio in terms of number of shares instead of capital allocation.

If the portfolio is fixed over time, $\boldsymbol{b}_t = \boldsymbol{b}$, then the notation becomes more compact. Denoting by $\boldsymbol{X} \in \mathbb{R}^{T\times N}$ the matrix containing the return vectors $\boldsymbol{r}_t$ along the rows and by $\boldsymbol{r}^{\mathrm{b}} \in \mathbb{R}^T$ the vector containing the returns of the index, we can then write

$$\boldsymbol{X}\boldsymbol{b} = \boldsymbol{r}^{\mathrm{b}}. \tag{13.4}$$

However, it is important to remark that in most practical cases the index portfolio changes over time and the notation in (13.3) is the correct one.

The goal is to design a sparse portfolio $\boldsymbol{w}_t$ such that $\boldsymbol{r}_t^{\mathsf{T}}\boldsymbol{w}_{t-1} \approx r_t^{\mathrm{b}}$ as in (13.3). For simplicity of notation, we can assume a fixed portfolio over time, $\boldsymbol{w}_t = \boldsymbol{w}$, and then the goal can be written as the approximation $\boldsymbol{X}\boldsymbol{w} \approx \boldsymbol{r}^{\mathrm{b}}$ similarly to (13.4). Thus, probably the simplest way to define the *tracking error* (TE) is

$$\mathrm{TE} = \frac{1}{T}\left\|\boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w}\right\|_2^2, \tag{13.5}$$

which fits naturally in the context of sparse regression of Section 13.2. Further expanding this with $\boldsymbol{X}\boldsymbol{b} = \boldsymbol{r}^{\mathrm{b}}$ leads to

$$\mathrm{TE} = (\boldsymbol{b} - \boldsymbol{w})^{\mathsf{T}}\frac{1}{T}\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}(\boldsymbol{b} - \boldsymbol{w}),$$

which can be interpreted as an approximation of an alternative definition of the tracking error based on $\boldsymbol{b}$:

$$\mathrm{TE}^{\mathrm{b}} = (\boldsymbol{b} - \boldsymbol{w})^{\mathsf{T}}\boldsymbol{\Sigma}(\boldsymbol{b} - \boldsymbol{w}),$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of the returns. However, this requires knowledge of $\boldsymbol{b}$, which may or may not be available. In addition, in most practical cases the index portfolio $\boldsymbol{b}_t$ changes over time. Thus, the empirical error measure in (13.5) is preferred.

The tracking error definition in (13.5) is the most widely adopted measure in the literature (Benidis et al., 2018a, 2018b; Jansen & Van Dijk, 2002; Scozzari et al., 2013; Shapcott, 1992; Xu et al., 2016), with some exceptions that involve the portfolio defined in terms of number of

shares (Beasley et al., 2003; Maringer & Oyewumi, 2007). Other more sophisticated measures of error are considered in Section 13.4.

Fixed vs. Time-Varying Portfolio

The tracking error definition in (13.5) is very convenient because it fits naturally in the context of sparse regression of Section 13.2. However, it is important to remark that in most practical cases the index portfolio changes over time. For example, for cap-weighted indices (the most common ones), the normalized portfolio evolves by definition as

$$\boldsymbol{b}_t = \frac{\boldsymbol{b}_{t-1} \odot (\mathbf{1} + \boldsymbol{r}_t)}{\boldsymbol{b}_{t-1}^\top (\mathbf{1} + \boldsymbol{r}_t)}.$$

As a consequence, it makes little sense to try to approximate this time-varying portfolio $\boldsymbol{b}_t$ with a constant portfolio $\boldsymbol{w}$ (except for the convenience of the simplicity of notation). In addition, even if we really wanted a fixed portfolio $\boldsymbol{w}$, this would imply a frequent rebalancing (with the corresponding transaction cost) because the normalized portfolio would otherwise naturally change over time (see (6.2) in Chapter 6 for details) as

$$\boldsymbol{w}_t = \frac{\boldsymbol{w}_{t-1} \odot (\mathbf{1} + \boldsymbol{r}_t)}{\boldsymbol{w}_{t-1}^\top (\mathbf{1} + \boldsymbol{r}_t)}.$$

Interestingly, it is still possible to express a tracking error with more realistic time-varying portfolios in the same convenient form as (13.5). The key is the following approximation based on the assumption that the index is being tracked, $\boldsymbol{r}_t^\top \boldsymbol{w}_{t-1} \approx r_t^{\mathrm{b}}$:

$$\boldsymbol{w}_t \approx \frac{\boldsymbol{w}_{t-1} \odot (\mathbf{1} + \boldsymbol{r}_t)}{1 + r_t^{\mathrm{b}}} \approx \boldsymbol{w}_0 \odot \boldsymbol{\alpha}_t,$$

where $\boldsymbol{w}_0$ is the initial portfolio and

$$\boldsymbol{\alpha}_t = \prod_{t'=1}^{t} \frac{\mathbf{1} + \boldsymbol{r}_{t'}}{1 + r_{t'}^{\mathrm{b}}} \tag{13.6}$$

denotes weights based on the cumulative returns. This allows us to write the portfolio return as

$$\boldsymbol{r}_t^\top \boldsymbol{w}_{t-1} \approx \boldsymbol{r}_t^\top (\boldsymbol{w}_0 \odot \boldsymbol{\alpha}_{t-1}) = \tilde{\boldsymbol{r}}_t^\top \boldsymbol{w}_0,$$

where $\tilde{\boldsymbol{r}}_t = \boldsymbol{r}_t \odot \boldsymbol{\alpha}_{t-1}$ are properly weighted returns.

Thus, we can finally write the tracking error, similarly to (13.5), as

$$\mathrm{TE}^{\text{time-varying}} = \frac{1}{T} \left\| \boldsymbol{r}^{\mathrm{b}} - \tilde{\boldsymbol{X}} \boldsymbol{w} \right\|_2^2, \tag{13.7}$$

where now $\tilde{\boldsymbol{X}}$ contains the weighted returns $\tilde{\boldsymbol{r}}_t$ along the rows and $\boldsymbol{w}$ denotes the initial portfolio $\boldsymbol{w}_0$.

Summarizing, the formulation in (13.5) assumes a fixed portfolio (as in most of the literature), which can be considered as a rough approximation, whereas (13.7) provides a more accurate approximation.

Figure 13.4 shows the tracking error over time with $K = 20$ active assets under the approximation (or assumption) of a fixed portfolio $\boldsymbol{w}$ as in (13.5) and with the more accurate approximation of a time-varying portfolio as in (13.7), which shows improved results.

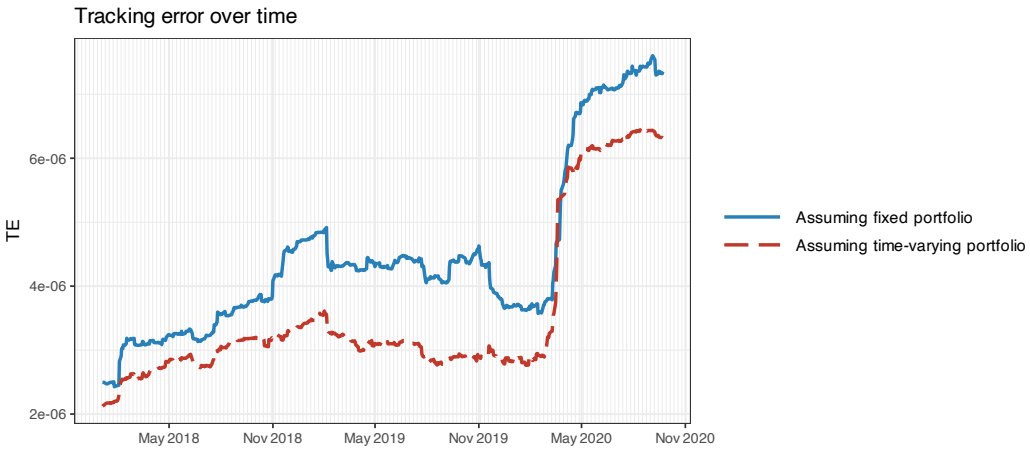


Figure 13.4 Tracking error over time of the S&P 500 index assuming fixed and time-varying portfolios.

Linear vs. Log-Returns

Computing the portfolio returns as in (13.3)–(13.4) or, simply, using the expression $\boldsymbol{X}\boldsymbol{w}$ implicitly assumes the use of linear returns in $\boldsymbol{X}$ and $\boldsymbol{r}^{\mathrm{b}}$ (recall that linear returns are additive along assets, cf. Chapter 6).

Nevertheless, one can similarly use log-returns in $\boldsymbol{X}$ and $\boldsymbol{r}^{\mathrm{b}}$ and define the tracking error in terms of log-returns. The log-returns $\boldsymbol{r}_t^{\log}$ and the linear returns $\boldsymbol{r}^{\mathrm{lin}}$ are related (see Chapter 6) as

$$\boldsymbol{r}_t^{\log} = \log\left(\mathbf{1} + \boldsymbol{r}_t^{\mathrm{lin}}\right) \approx \boldsymbol{r}_t^{\mathrm{lin}},$$

because $\log(1+x) \approx x$ for small x. Thus, the log-returns of the portfolio $\boldsymbol{w}$ can be approximated as

$$\log\left(1 + \boldsymbol{w}^{\mathsf{T}}\boldsymbol{r}_t^{\mathrm{lin}}\right) \approx \boldsymbol{w}^{\mathsf{T}}\boldsymbol{r}_t^{\log}.$$

In practice, the difference between using linear or log-returns is negligible.

Plain vs. Cumulative Returns

It is worth pointing out that minimizing the error in terms of the period-by-period returns (plain returns) does not directly imply a better track of the cumulative returns (or price) over time. The reason is that the errors in the returns can accumulate over time on a more positive or negative side. If one really wants to control the long-term deviation of the cumulative returns, then the error measure should properly reflect this, for example, by using long-term returns rather than period-by-period returns or even by using cumulative returns (Benidis et al., 2018a). However, in numerous instances, tracking an index in the short term is essential for

hedging purposes against other investments. In such cases, the objective is not to outperform the index, and it is the period-by-period returns that truly matter.

The price error can be measured by the tracking error in terms of the cumulative returns as

$$\mathrm{TE}^{\mathrm{cum}} = \frac{1}{T}\left\|\boldsymbol{r}^{\mathrm{b,cum}} - \boldsymbol{X}^{\mathrm{cum}}\boldsymbol{w}\right\|_2^2, \tag{13.8}$$

where $\boldsymbol{X}^{\mathrm{cum}}$ (similarly $\boldsymbol{r}^{\mathrm{b,cum}}$) denotes the cumulative returns, whose rows can be obtained as

$$\boldsymbol{r}_t^{\mathrm{cum}} \approx \sum_{t'=1}^{t} \boldsymbol{r}_{t'}.$$

For the case of linear returns, this expression follows from the approximation

$$\prod_{t'=1}^{t}\left(\mathbf{1} + \boldsymbol{r}_{t'}^{\mathsf{T}}\boldsymbol{w}\right) - 1 \approx \prod_{t'=1}^{t}\exp\left(\boldsymbol{r}_{t'}^{\mathsf{T}}\boldsymbol{w}\right) - 1 = \exp\left(\sum_{t'=1}^{t}\boldsymbol{r}_{t'}^{\mathsf{T}}\boldsymbol{w}\right) - 1 \approx \left(\sum_{t'=1}^{t}\boldsymbol{r}_{t'}\right)^{\mathsf{T}}\boldsymbol{w},$$

where we have used $1 + \boldsymbol{a}^{\mathsf{T}}\boldsymbol{w} \approx \exp(\boldsymbol{a}^{\mathsf{T}}\boldsymbol{w})$. For log-returns, it follows from

$$\log\left(\prod_{t'=1}^{t}\left(\mathbf{1} + \boldsymbol{r}_{t'}^{\mathsf{T}}\boldsymbol{w}\right)\right) = \sum_{t'=1}^{t}\log\left(\mathbf{1} + \boldsymbol{r}_{t'}^{\mathsf{T}}\boldsymbol{w}\right) \approx \left(\sum_{t'=1}^{t}\boldsymbol{r}_{t'}\right)^{\mathsf{T}}\boldsymbol{w}.$$

Figure 13.5 shows the tracking of the index price with $K = 20$ active assets using plain and cumulative returns in the definition of tracking error as in (13.5) and (13.8), respectively. Clearly, the cumulative returns provide a much better tracking of the index price (although not of the tracking error in terms of returns).



Figure 13.5 Tracking over time of the S&P 500 index using plain returns and cumulative returns.

Summary of Tracking Errors

Table 13.1 summarizes the different tracking error definitions previously defined (ignoring the difference between linear and log-returns). To recall, the different data matrices employed are:

- $\boldsymbol{X}$: contains the plain returns $\boldsymbol{r}_t$ along the rows;
- $\tilde{\boldsymbol{X}}$: contains the weighted plain returns $\tilde{\boldsymbol{r}}_t = \boldsymbol{r}_t \odot \boldsymbol{\alpha}_{t-1}$ along the rows (with $\boldsymbol{\alpha}_t$ in (13.6));
- $\boldsymbol{X}^{\text{cum}}$: contains the cumulative returns $\boldsymbol{r}_t^{\text{cum}} \approx \sum_{t'=1}^{t} \boldsymbol{r}_{t'}$ along the rows; and
- $\tilde{\boldsymbol{X}}^{\text{cum}}$: contains the weighted cumulative returns $\sum_{t'=1}^{t} \tilde{\boldsymbol{r}}_{t'}$.

Table 13.1 Different data matrices employed in the definition of tracking error.

	Tracking of returns	Tracking of prices
Assuming $\boldsymbol{w}$ fixed	$\boldsymbol{X}$	$\boldsymbol{X}^{\text{cum}}$
Assuming $\boldsymbol{w}$ time-varying	$\tilde{\boldsymbol{X}}$	$\tilde{\boldsymbol{X}}^{\text{cum}}$

13.3.2 Problem Formulation

Formulating the index tracking problem is quite straightforward in terms of a sparse regression problem:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \dfrac{1}{T}\left\|\boldsymbol{r}^{\text{b}} - \boldsymbol{X}\boldsymbol{w}\right\|_2^2 + \lambda\|\boldsymbol{w}\|_0 \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array} \tag{13.9}$$

where the parameter λ controls the level of sparsity in the portfolio, $\mathcal{W}$ denotes an arbitrary constraint set, such as $\mathcal{W} = \{\boldsymbol{w} \mid \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1, \boldsymbol{w} \geq \mathbf{0}\}$, and matrix $\boldsymbol{X}$ contains the returns of the assets (any version of returns as explored in Section 13.3.1).

Index tracking is a type of passive investment that avoids the expensive transaction costs incurred by frequent trading associated with active portfolio management. Therefore, it may be advantageous to explicitly control the turnover in the formulation (13.9) by adding a penalty term or constraint to control the turnover $\|\boldsymbol{w} - \boldsymbol{w}_0\|_1$, where $\boldsymbol{w}_0$ is the current portfolio (Benidis et al., 2018a, 2018b). In practice, other variations on the basic index tracking formulation (13.9) are commonly used and will be explored in Section 13.4.

Recall that the cardinality operator on the portfolio $\|\boldsymbol{w}\|_0$ is noncontinuous, nondifferentiable, and nonconvex, which means that developing practical algorithms under sparsity is not trivial. We will start with some heuristic methods and build up to the state-of-the-art solutions based on MM.

13.3.3 Methods for Sparse Index Tracking

An important observation is that if we knew a priori the active assets, then the index tracking would be a trivial convex regression problem without the difficulty of sparsity control. This leads to two-step approaches that first select the active assets and then proceed with the weight computation (Jansen & Van Dijk, 2002). Nevertheless, such two-step approaches are not optimal and one can always do better by solving the problem jointly in a single step. Several methods have been proposed in the literature; some of them are computationally

intensive, such as mixed integer programming or differential evolution techniques (Maringer & Oyewumi, 2007), while others are computationally feasible but theoretically cannot guarantee to produce a globally optimal solution, such as the iterative reweighted ℓ_1-norm optimization method (Benidis et al., 2018a, 2018b) or the projected gradient method (Xu et al., 2016).

Naive Two-Step Design

This approach first selects the desired K active assets from the universe of N assets (with $K \ll N$) in some heuristic way (Jansen & Van Dijk, 2002). This can be done, for example, based on

- the weight of the assets in the index definition (e.g., the largest K assets in $\boldsymbol{b}$);
- the market capitalization of the assets; or
- the strength of correlation between the assets and the index.

Once the active assets have been selected, then the weights are taken directly from the index definition $\boldsymbol{b}$ if available; otherwise, one can simply do a dense regression to obtain a proxy for $\boldsymbol{b}$.

In more detail, we can define the Boolean pattern vector $\boldsymbol{s} \in \mathbb{R}^N$ based on the selected assets:

$$s_i = \begin{cases} 1 & \text{if the } i\text{th asset is selected,} \\ 0 & \text{otherwise,} \end{cases}$$

so that $\mathbf{1}^\mathsf{T}\boldsymbol{s} = K$. Then, we can explicitly write the weights $\boldsymbol{w}$ proportional to the definition in $\boldsymbol{b}$ (although properly scaled so that $\mathbf{1}^\mathsf{T}\boldsymbol{w} = 1$):

$$\boldsymbol{w} = \frac{\boldsymbol{b} \odot \boldsymbol{s}}{\mathbf{1}^\mathsf{T}(\boldsymbol{b} \odot \boldsymbol{s})},$$

where $\odot$ denotes the Hadamard (elementwise) product.

Two-Step Design with Refitted Weights

This approach refines the previous one by trying to refit the weights after the selection of the active assets (Jansen & Van Dijk, 2002). To be exact, after the K assets have been selected and the Boolean pattern vector $\boldsymbol{s}$ has been computed, we can form a simple regression problem over the active assets without having to deal with the sparsity issue. Thus, the original problem (13.9) becomes

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \dfrac{1}{T}\left\|\boldsymbol{r}^{\text{b}} - \boldsymbol{X}\boldsymbol{w}\right\|_2^2 \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \\ & w_i = 0 \quad \text{if} \quad s_i = 0 \end{array}$$

or, equivalently,

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \dfrac{1}{T}\left\|\boldsymbol{r}^{\text{b}} - \boldsymbol{X}\boldsymbol{w}\right\|_2^2 \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \\ & \boldsymbol{w} \leq \boldsymbol{s}. \end{array}$$

Mixed Integer Programming Formulation

In mixed integer programming (MIP), variables are allowed to be constrained to a discrete set, making the problem nonconvex and generally with an exponential worst-case complexity. This makes this approach impractical unless the dimensionality of the problem (number of assets) is very small (Scozzari et al., 2013).

The MIP formulation of (13.9) can be written in terms of the Boolean pattern vector $\boldsymbol{s}$, which is now a variable:

$$\begin{array}{ll} \underset{\boldsymbol{w},\boldsymbol{s}}{\text{minimize}} & \frac{1}{T}\left\|\boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w}\right\|_2^2 \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \\ & \boldsymbol{w} \leq \boldsymbol{s},\ \mathbf{1}^{\mathsf{T}}\boldsymbol{s} = K,\ s_i \in \{0,1\}. \end{array}$$

Evolutionary Algorithms

Evolutionary algorithms are a family of optimization and search techniques inspired by the process of natural evolution. They operate on populations of candidate solutions, called individuals, which evolve over time while attempting to improve their fitness. Successful solutions are selected and combined to produce new candidate solutions akin to the natural selection process in living organisms.

Evolutionary algorithms have been applied to a wide range of problems, including optimization, machine learning, and game playing. They are particularly suitable for complex, multimodal, and noisy search spaces, where traditional optimization methods might struggle. Some key advantages include their ability to explore large solution spaces, robustness against local optima, and parallelization potential.

Some popular evolutionary algorithms include *genetic algorithms* and *differential evolution*. Genetic algorithms use a binary or symbolic representation for solutions and apply genetic operators like selection, crossover, and mutation to evolve populations of chromosomes. Differential evolution focuses on continuous optimization problems and is known for its ability to tackle high-dimensional, non-linear, and noisy problems.

Evolutionary algorithms have been employed for index tracking as a way to solve the complicated nonconvex mixed-integer problem formulation. Some examples include Shapcott (1992), Beasley et al. (2003), and Maringer and Oyewumi (2007). Nevertheless, the computational complexity of evolutionary algorithms can be high since a population of solutions have to be evolved over many generations to properly explore the nonconvex fitness surface.

Iterative Reweighted ℓ_1-Norm Method

Two main approaches were considered in Section 13.2 for sparse regression: based on the ℓ_1-norm approximation and based on a concave approximation. Unfortunately, the commonly used method based on the ℓ_1-norm $\|\boldsymbol{w}\|_1$ cannot be used in the context of portfolio optimization due to the portfolio constraints $\mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1$ and $\boldsymbol{w} \geq \mathbf{0}$ that lead trivially to

$$\|\boldsymbol{w}\|_1 = \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1.$$

Thus, we need to resort to a more refined concave approximation of (13.9); for example, based

on the log approximation:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \frac{1}{T}\left\|\boldsymbol{r}^{\text{b}} - \boldsymbol{X}\boldsymbol{w}\right\|_2^2 + \lambda \sum_{i=1}^{N} \log\left(1 + \frac{|w_i|}{\varepsilon}\right) \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}. \end{array} \tag{13.10}$$

This nonconvex problem can then be addressed with the MM framework to obtain a convenient iterative procedure called the *iterative reweighted ℓ_1-norm method* and summarized in Algorithm 13.1.[2]

Algorithm 13.1: Iterative reweighted ℓ_1-norm method to solve the approximated sparse index tracking problem (13.10).

1: Choose initial point $\boldsymbol{w}^0 \in \mathcal{W}$;
2: Set $k \leftarrow 0$;
3: **repeat**
4: Compute weights

$$\alpha_i^k = \frac{1}{\varepsilon + \left|w_i^k\right|};$$

5: Solve the following weighted ℓ_1-norm problem to obtain $\boldsymbol{w}^{k+1}$:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \frac{1}{T}\left\|\boldsymbol{r}^{\text{b}} - \boldsymbol{X}\boldsymbol{w}\right\|_2^2 + \lambda \sum_{i=1}^{N} \alpha_i^k |w_i| \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}; \end{array}$$

6: $k \leftarrow k + 1$;
7: **until** convergence;

Since (13.10) is a nonconvex problem, the algorithm may potentially get stuck in a local minimum. In practice, for highly sparse solutions this behavior can be observed. To deal with this issue, one can start with more dense solutions, corresponding to a small value of λ, and then sequentially increase the sparsity level by increasing λ (Benidis et al., 2018a, 2018b).

It is interesting to remark that the main step of Algorithm 13.1 requires solving an ℓ_2-norm weighted ℓ_1-norm problem, which can be done with a quadratic program solver (assuming the feasible set $\mathcal{W}$ only contains linear and quadratic terms). Nevertheless, it is possible in most practical cases, to further majorize the problem in order to obtain a closed-form solution without the need for a solver (Benidis et al., 2018a, 2018b).

[2] The R package `sparseIndexTracking` implements Algorithm 13.1 based on Benidis et al. (2018b) and Benidis et al. (2018a) as well as a number of extensions covered in Section 13.4 (Benidis & Palomar, 2019).

13.3.4 Numerical Experiments

Convergence of the Iterative Reweighted ℓ_1-Norm Method

Figure 13.6 illustrates the convergence of the iterative reweighted ℓ_1-norm method described in Algorithm 13.1. We can see that the convergence is extremely fast and that two to three iterations seem to reap most of the benefit.

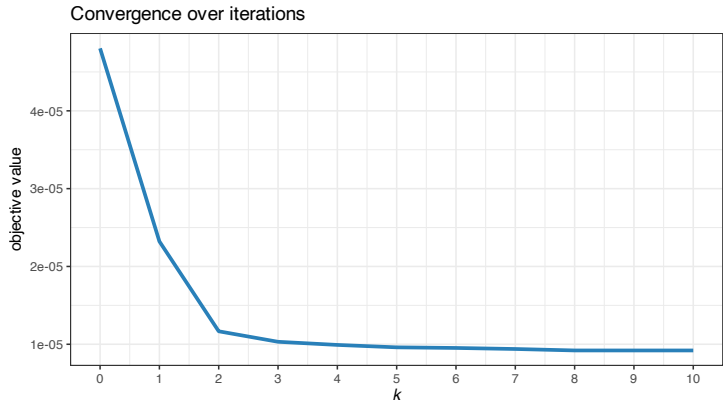


Figure 13.6 Convergence of the iterative reweighted ℓ_1-norm method for sparse index tracking.

Comparison of Algorithms

We now compare the tracking of the S&P 500 index based on different tracking methods, namely:

- the naive two-step approach with proportional weights to the index definition;
- the two-step approach with refitted weights;
- the iterative reweighted ℓ_1-norm method in Algorithm 13.1; and
- the MIP formulation (although the computational complexity is too high).

Figure 13.7 shows the tracking error over time of the S&P 500 index with $K = 20$ active assets (the MIP method is excluded due to its huge computational cost). The tracking portfolios are computed on a rolling-window basis with a lookback period of two years and recomputed every six months. To get a more complete picture, Figure 13.8 explores the trade-off of tracking error vs. the number of active assets K (including the MIP method). We can observe that, as expected, the joint designs are superior to the traditional two-step approaches. While the MIP formulation is impractical due to the huge computational cost, the iterative reweighted ℓ_1-norm method exhibits low complexity, making it a suitable approach in practice. For more exhaustive numerical comparisons, the reader is referred to Benidis et al. (2018b) and Benidis et al. (2018a).

Comparison of Formulations

Finally, we consider again the comparison between assuming a fixed portfolio and a time-varying one in the definition of tracking error as in (13.5) and (13.7), respectively. This comparison was already illustrated in Figure 13.4 in terms of tracking error over time. For a

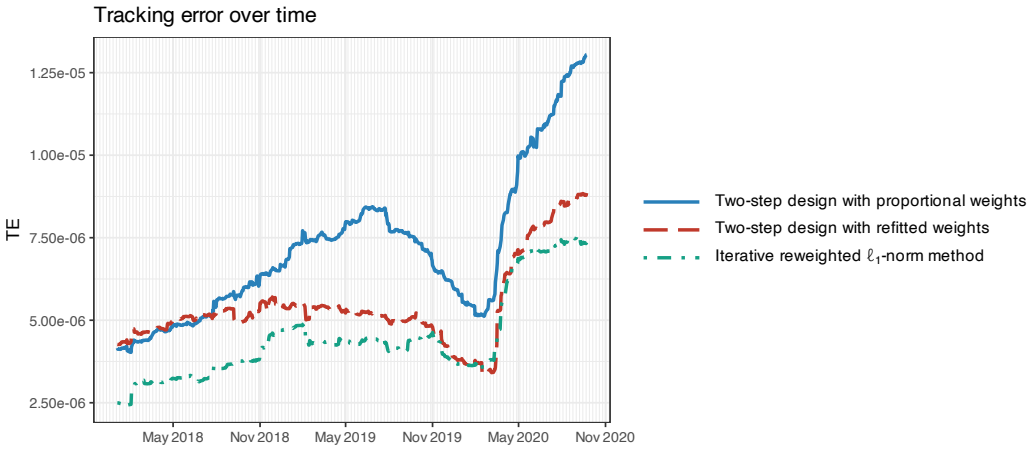


Figure 13.7 Tracking error over time of the S&P 500 index for different algorithms.

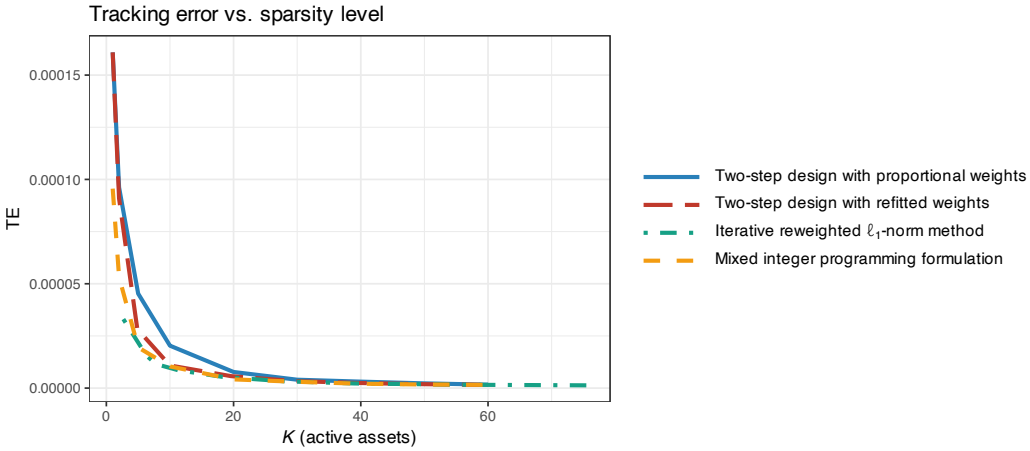


Figure 13.8 Tracking error of the S&P 500 index vs. active assets for different algorithms.

more complete comparison, Figure 13.9 shows the trade-off of tracking error vs. cardinality. Indeed, the error definition in (13.7) under a time-varying portfolio is slightly more accurate.

13.4 Enhanced Index Tracking

Enhanced index tracking refers to variations on the basic index tracking formulation (13.9). They generally attempt to increase returns by building portfolios around index-like positions but then adding tactical tilts toward specific styles or individual stocks. This has been used by professional portfolio managers for decades (Xu et al., 2022) and some examples will be explored next.

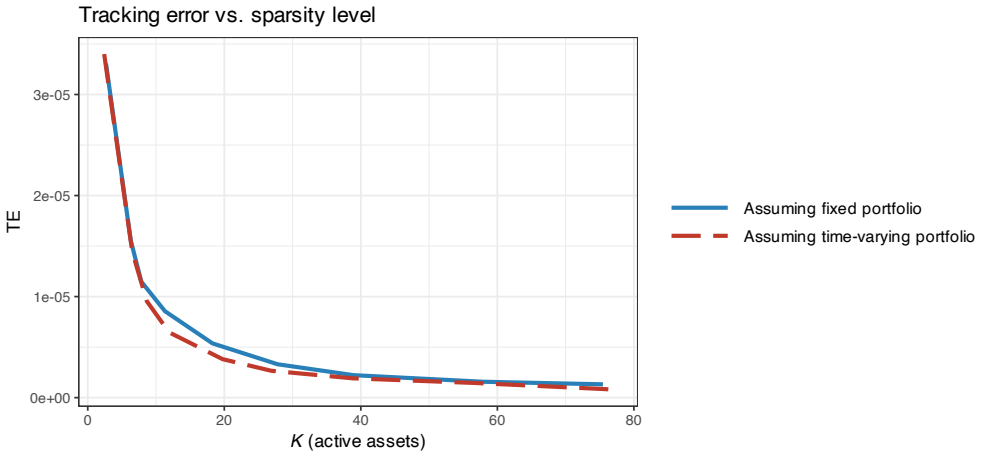


Figure 13.9 Tracking error of the S&P 500 index vs. active assets assuming fixed and time-varying portfolios.

13.4.1 Alternative Tracking Error Measures

The tracking error used in Section 13.3 is based on the ℓ_2-norm between the achieved returns $\boldsymbol{X}\boldsymbol{w}$ and the benchmark returns $\boldsymbol{r}^{\mathrm{b}}$:

$$\mathrm{TE}(\boldsymbol{w}) = \frac{1}{T}\left\|\boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w}\right\|_2^2. \tag{13.11}$$

This is a very simple way to measure the size of the tracking error and one could consider many other alternatives by either changing the norm (e.g., using the ℓ_1-norm or the ℓ_p-norm (Beasley et al., 2003)) or changing the error measure altogether, such as the excess return $\frac{1}{T}\mathbf{1}^{\mathsf{T}}\left(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{r}^{\mathrm{b}}\right)$ (Beasley et al., 2003; Dose & Cincotti, 2005), the downside risk, value at risk, or conditional value at risk (as defined in Chapter 6 and explored in detail in Chapter 10 for portfolio design). We now consider the downside risk and the ℓ_1-norm tracking error for illustration purposes.

Downside Risk Error Measure

A particularly interesting alternative is the *downside risk* that only takes into account when the achieved returns are worse than those of the benchmark,

$$\mathrm{DR}(\boldsymbol{w}) = \frac{1}{T}\left\|\left(\boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w}\right)^{+}\right\|_2^2, \tag{13.12}$$

where the operator $(\cdot)^{+} \triangleq \max(0, \cdot)$ only considers returns smaller than the benchmark. This falls into the realm of enhanced index tracking.

It is worth pointing out that while using this tracking error measure may have the benefit of possibly beating the index, it is not appropriate for hedging purposes since that requires tracking the index in both directions.

Two key observations arise regarding portfolio optimization under the downside risk $\mathrm{DR}(\boldsymbol{w})$. First, $\mathrm{DR}(\boldsymbol{w})$ is a convex function of $\boldsymbol{w}$ and, therefore, it can be effectively optimized in

practice with an appropriate solver. Second, it can be easily majorized for the purpose of using the MM method; in particular, we can choose a majorizer with the form of an ℓ_2-norm and then the methods presented in Section 13.3 can be readily employed.

Lemma 13.2 (Majorizer of the downside risk) *The downside risk function* $\mathrm{DR}(\boldsymbol{w})$ *in* (13.12) *is majorized at* $\boldsymbol{w} = \boldsymbol{w}_0$ *by* $\mathrm{TE}(\boldsymbol{w})$ *in* (13.11) *with shifted benchmark returns* $\tilde{\boldsymbol{r}}^{\mathrm{b}}$*:*

$$\mathrm{DR}(\boldsymbol{w}) \leq \frac{1}{T} \left\| \tilde{\boldsymbol{r}}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w} \right\|_2^2 ,$$

where $\tilde{\boldsymbol{r}}^{\mathrm{b}} = \boldsymbol{r}^{\mathrm{b}} + \left(\boldsymbol{X}\boldsymbol{w}_0 - \boldsymbol{r}^{\mathrm{b}} \right)^+$ *(Benidis et al., 2018a, 2018b).*

The shifted benchmark returns $\tilde{\boldsymbol{r}}^{\mathrm{b}}$ in Lemma 13.2 have an interesting interpretation: they are an improvement of the original returns $\boldsymbol{r}^{\mathrm{b}}$ for those returns that were outperformed by the nominal portfolio $\boldsymbol{w}_0$.

According to Lemma 13.2, index tracking under the downside risk can still be accomplished with Algorithm 13.1 under a slight modification: at each iteration k, use the shifted benchmark returns $\left(\tilde{\boldsymbol{r}}^{\mathrm{b}} \right)^k = \boldsymbol{r}^{\mathrm{b}} + \left(\boldsymbol{X}\boldsymbol{w}^k - \boldsymbol{r}^{\mathrm{b}} \right)^+$.

ℓ_1-Norm Tracking Error

The TE in (13.11) is based on the ℓ_2-norm, but this is a rather arbitrary choice. In fact, one may argue that the ℓ_1-norm would make more sense:

$$\mathrm{TE}_1(\boldsymbol{w}) = \frac{1}{T} \left\| \boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w} \right\|_1 . \tag{13.13}$$

This error measure is a convex function and, furthermore, it can be conveniently majorized in the form of an ℓ_2-norm as in the case of the tracking error considered in Section 13.3.

Lemma 13.3 (Majorizer of the ℓ_1-norm TE) *The ℓ_1-norm tracking error function* $\mathrm{TE}_1(\boldsymbol{w})$ *in* (13.13) *is majorized at* $\boldsymbol{w} = \boldsymbol{w}_0$ *by a weighted version of the* $\mathrm{TE}(\boldsymbol{w})$ *in* (13.11)*:*

$$\mathrm{TE}_1(\boldsymbol{w}) \leq \frac{1}{T} \left\| \boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w} \right\|_{2,\boldsymbol{\alpha}}^2 ,$$

where $\|\boldsymbol{x}\|_{2,\boldsymbol{\alpha}}^2 \triangleq \sum_{i=1}^T \alpha_i x_i^2$ *is the squared weighted ℓ_2-norm with weights* $\boldsymbol{\alpha} = 1/(2|\boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w}_0|)$*. See Section B.7 in Appendix B for details.*

The weights in the weighted ℓ_2-norm TE in Lemma 13.3 have a natural interpretation: their role is to down-weight the errors so that they grow in an approximately linear fashion like in the ℓ_1-norm, as expected.

According to Lemma 13.3, index tracking under the ℓ_1-norm TE can still be accomplished with Algorithm 13.1 under a slight modification: at each iteration k, use the weighted ℓ_2-norm with weights $\boldsymbol{\alpha}^k = 1/(2|\boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w}^k|)$.

13.4.2 Robust Tracking Error Measures

Robustness against outliers in the data is paramount in order to mitigate the effects of data contamination and avoid being sensitive or even breaking down (Section 3.5 in Chapter 3 covers the topic of robust estimators).

Any tracking error measure can be made robust to make it less sensitive to outliers. For illustration purposes, take the tracking error in (13.11) used in Section 13.3. This measure is based on the ℓ_2-norm between the achieved returns $\boldsymbol{X}\boldsymbol{w}$ and the benchmark returns $\boldsymbol{r}^{\mathrm{b}}$, but the ℓ_2-norm is not a robust measure because it is sensitive to (anomalous) large errors due the the squaring operation. Of course, one alternative is to use the ℓ_1-norm tracking error in (13.13), which is naturally robust.

Alternatively, we can make the ℓ_2-norm robust with the Huber penalty function (Huber, 2011):

$$\phi^{\mathrm{hub}}(x) = \begin{cases} x^2, & |x| \leq M, \\ M\,(2|x| - M)\,, & |x| > M, \end{cases}$$

which essentially behaves as the square function x^2 for arguments up to a magnitude of M and as a linear function otherwise (this way outliers are not amplified). We can then define a Huberized version of the tracking error:

$$\text{Hub-TE}(\boldsymbol{w}) = \frac{1}{T}\sum_{t=1}^{T} \phi^{\mathrm{hub}}(r_t^{\mathrm{b}} - \boldsymbol{X}_{t,:}\boldsymbol{w}), \tag{13.14}$$

which is a convex function. In addition, it can be conveniently majorized in the form of an ℓ_2-norm as in the case of the tracking error considered in Section 13.3.

Lemma 13.4 (Majorizer of the Huberized TE) *The Huberized tracking error function* Hub-TE($\boldsymbol{w}$) *in* (13.14) *is majorized at* $\boldsymbol{w} = \boldsymbol{w}_0$ *by a weighted version of the* TE($\boldsymbol{w}$) *in* (13.11)*:*

$$\text{Hub-TE}(\boldsymbol{w}) \leq \frac{1}{T}\left\|\boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w}\right\|_{2,\boldsymbol{\alpha}}^2 + \textit{const.},$$

where $\|\boldsymbol{x}\|_{2,\boldsymbol{\alpha}}^2 \triangleq \sum_{i=1}^{T} \alpha_i x_i^2$ *is the squared weighted* ℓ_2*-norm with weights*

$$\boldsymbol{\alpha} = \min\left(\boldsymbol{1}, \frac{M}{|\boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w}_0|}\right),$$

and the term const. refers to an irrelevant constant term (Benidis et al., 2018a, 2018b).

The weights in the Huberized TE have a very natural interpretation: their role is to down-weight the errors that are larger than M in magnitude so that the squared values grow linearly instead.

According to Lemma 13.4, index tracking under the Huberized TE can still be accomplished with Algorithm 13.1 under a slight modification: at each iteration k, use the weighted ℓ_2-norm with weights

$$\boldsymbol{\alpha}^k = \min\left(\boldsymbol{1}, \frac{M}{|\boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w}^k|}\right).$$

13.4.3 Holding Constraints

In practice, portfolios typically have holding constraints:

- *upper bounds* ($\boldsymbol{w} \leq \boldsymbol{u}$): this is to avoid risk from allocating too much budget to a single asset and to promote diversification; and

- *lower bounds* ($\boldsymbol{w} \geq \boldsymbol{l}$): this is to avoid very small positions that are irrelevant to the overall portfolio and just complicate the logistics.

These bound constraints are trivial linear inequality constraints. However, when dealing with sparsity, the lower bounds become incompatible with sparse solutions. In this case, the lower bounds should only become active for active assets:

$$w_i \begin{cases} \geq l_i, & \text{if } w_i > 0, \\ = 0, & \text{otherwise.} \end{cases}$$

These complicated nonconvex constraints can be easily incorporated into evolutionary algorithms (Beasley et al., 2003; Maringer & Oyewumi, 2007). Interestingly, they can also be approximated and solved via the MM framework in a convenient way (Benidis et al., 2018a, 2018b).

13.4.4 Group Sparsity

Stocks and other financial assets are classified and grouped together into sectors and industries. This organization is convenient for investors in order to easily diversify their investment across different sectors (which presumably are less correlated than stocks within each sector).

All the index tracking formulations considered thus far are able to control the sparsity of the portfolio via the term $\|\boldsymbol{w}\|_0$. Nevertheless, it is possible to have a more refined control by taking account of stock industry profiles. One way is by replacing the overall sparsity term $\|\boldsymbol{w}\|_0$ by a "group sparsity" term that is able to construct a portfolio composed of a specific number of stocks concentrating on a few industries, and ensures both industry-wise and within-industry sparsity (Xu et al., 2022).

13.4.5 Numerical Experiments

We now compare the tracking of the S&P 500 index based on different tracking error measures of the cumulative returns, namely:

- TE in (13.11);
- Huberized TE in (13.14);
- ℓ_1-norm TE in (13.13); and
- DR in (13.12).

Figure 13.10 shows the tracking over time with approximately $K = 20$ active assets. The tracking portfolios are computed on a rolling-window basis with a lookback period of two years and recomputed every six months. As expected, one can observe that the design based on the downside risk beats the market (suitable for investment purposes) while the other measures generally track the index in both directions (appropriate for hedging purposes).

13.5 Automatic Sparsity Control

The sparse index tracking formulation in (13.9) includes the regularization term $\lambda\|\boldsymbol{w}\|_0$ in the objective, where λ is a hyper-parameter to be chosen in order to get the desired level of

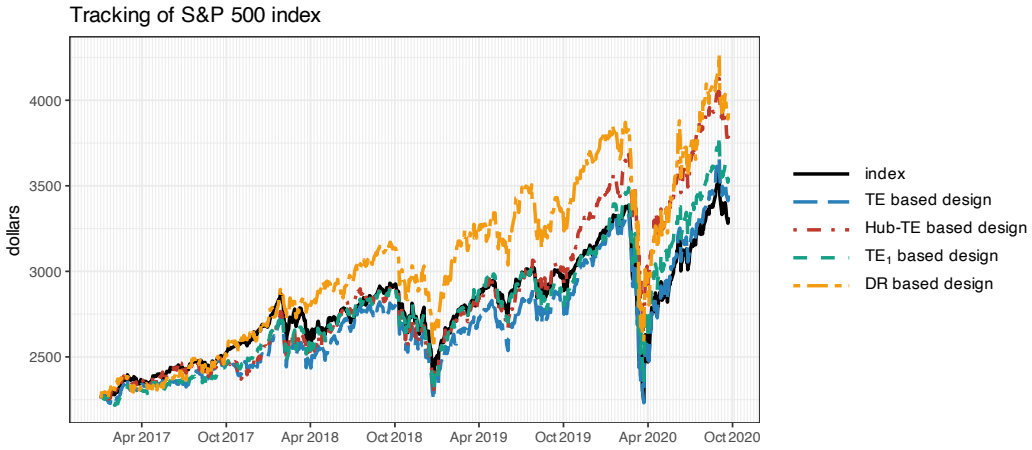


Figure 13.10 Tracking over time of the S&P 500 index under different tracking error measures.

sparsity. Alternatively, it can be formulated by moving the sparsity term to the constraints as $\|\boldsymbol{w}\|_0 \leq k$, with k denoting the desired level of sparsity. Either way, by properly adjusting the sparsity hyper-parameter, λ or k, different points on the error vs. sparsity trade-off curves in Figures 13.8–13.9 can be achieved.

In practice, however, the goal is to choose a proper operating point on the error vs. sparsity trade-off curve, preferably without having to compute the whole trade-off curve. Is there a convenient way to tune the sparsity hyper-parameter to get a proper operating point in the trade-off curve?

13.5.1 False Discovery Rate (FDR)

To properly answer the question of how to choose the operating point on the trade-off curve, we need to introduce some concepts from statistics and hypothesis testing. In particular, a key quantity when deciding whether to use a variable or not in a regression problem is the *false discovery rate* (FDR), which refers to the probability of wrongly including a variable.

In some applications, such as genomics, including the wrong variables can be catastrophic as it implies that some gene is incorrectly associated with some medical condition. In finance, for example, hundreds of papers have been written attempting to discover factors that explain the cross-section of expected returns, but are these results false? Apparently so. Based on a hypothesis testing analysis of empirical tests since 1967, it seems that most claimed research findings in financial economics are likely false (Harvey et al., 2016).

Controlling the FDR would be an ideal and sound way to decide whether to include a variable and this was achieved in the seminal paper by Benjamini and Hochberg (1995). Since then, many FDR-controlling methods have been proposed, with the most popular one being based on the concept of *knockoffs* (R. F. Barber & Candès, 2015). Knockoffs are fictitious variables that are created for the purpose of FDR control; they need to mimic the covariance structure

of the original variables and this can be computationally demanding since this requires the estimation of the covariance matrix, which can be difficult in high-dimensional settings.

More recently, a more practical method for FDR control called T-Rex (for Terminating-Random EXperiments) was proposed based on the concept of *dummies* (Machkour, Muma, & Palomar, 2025). Dummies are also fictitious variables used to control the FDR, but they are easy to generate since they do not require following the same covariance structure of the original variables. Instead, dummies can be sampled from any univariate probability distribution. The T-Rex method effectively reduces the computation time by two orders of magnitude compared to the competing methods.

13.5.2 FDR for Index Tracking

In the context of sparse index tracking, instead of selecting the sparsity level through trial and error, a more robust approach would be to precisely control the FDR. However, the interpretation of FDR is slightly different in this case because, strictly speaking, all the assets in the definition of an index are valid variables to be selected. Nevertheless, once an asset has been selected, it is typically the case that many other assets become redundant because they are highly correlated with that selected asset. In this sense we can say that selecting these irrelevant assets would be a "false discovery" and it is to be avoided.

The application of the T-Rex method to sparse index tracking with FDR control was developed in Machkour, Palomar, and Muma (2025).[3] Rather than having to fix or tune the hyper-parameter λ in (13.9), it automatically selects the assets to be included by controlling the FDR.

13.5.3 Numerical Experiments

We now compare the tracking portfolios obtained from the sparse penalized regression formulation in (13.9) with the FDR-controlling T-Rex method (Machkour, Palomar, & Muma, 2025). The tracking portfolios are computed on a rolling-window basis with a lookback period of two years and recomputed every six months.

Figure 13.11 shows the tracking error over time in terms of the plain returns (13.5) and cumulative returns (13.8), as well as the cardinality of the portfolios over time. As can be seen, the formulation in (13.9) is very sensitive to the choice of the parameter λ, producing very different results in terms of tracking error and cardinality. On the other hand, the FDR-controlling T-Rex method automatically chooses the appropriate sparsity level without having to tune any parameter. The computational cost of T-Rex is slightly higher than that of solving (13.9) for a fixed λ but lower than solving (13.9) for a whole range of values for λ.

[3] The R package `TRexSelector` (Machkour et al., 2024) implements the T-Rex method based on Machkour, Muma, and Palomar (2025) and Machkour, Palomar, and Muma (2025).

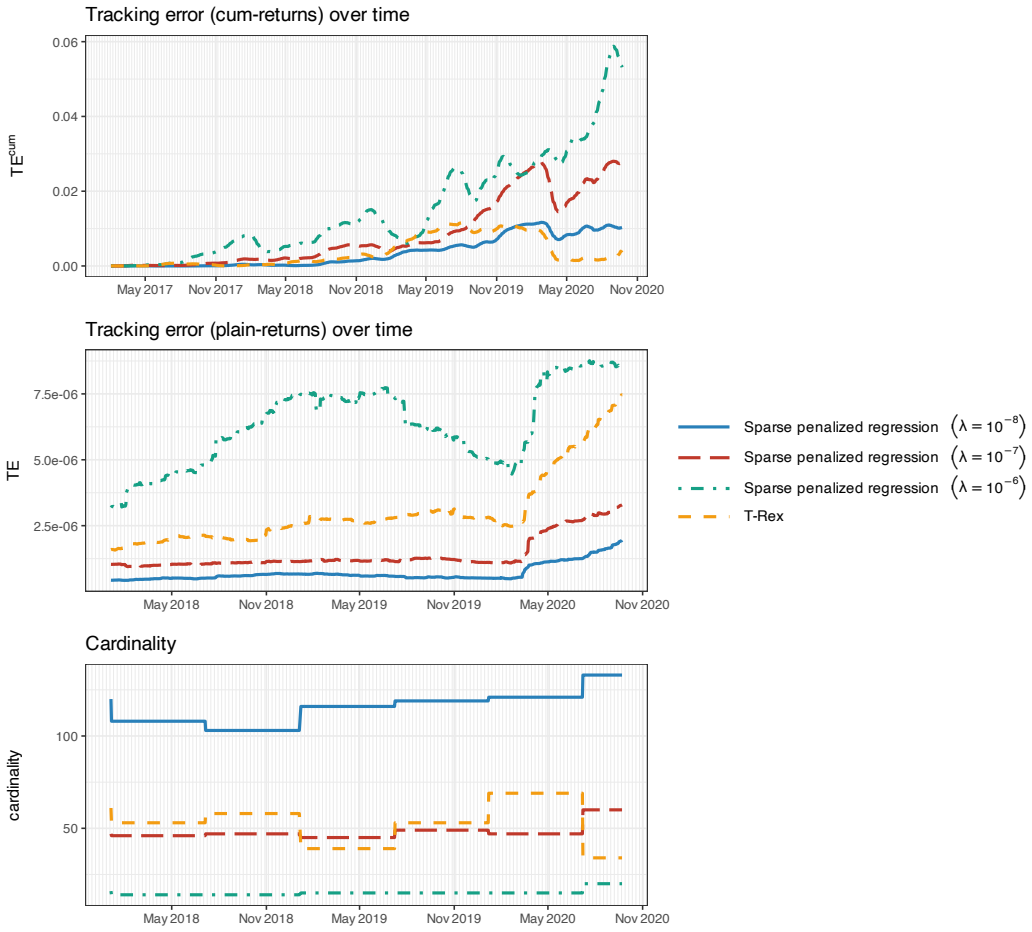


Figure 13.11 Tracking over time of the S&P 500 index with and without FDR control.

13.6 Summary

Active and passive strategies play a major role in the financial investment arena, both having theoretical and practical justifications albeit stemming from opposite views of the markets. Some key takeaways include:

- *Passive investing methods* seek to avoid the fees and limited performance that may occur with frequent active trading.
- *Index tracking* is the mainstream approach for passive investment and simply tries to mimic an index, based on the assumption that the market is efficient and cannot be beaten.
- In current markets, there are thousands of *financial indices* that cover a wide range of asset classes, sectors, and regions (e.g., the S&P 500). There are even more ETFs that precisely track any given index and investors can directly trade them (e.g., there are hundreds of ETFs that track the S&P 500 index).

- *Sparse index tracking* is closely related to a fundamental problem in statistics called sparse regression. Its goal is to approximate an index but using a small number of active assets. The mathematical problem formulation requires a tracking error measure and a mechanism to control the sparsity level.
- A variety of *tracking error measures* can be used for index tracking, such as the ℓ_2-norm tracking error (13.11), the downside risk (13.12), the ℓ_1-norm version (13.13), and the Huberized robust version (13.14), among others.
- Many algorithms have been proposed for index tracking capable of controlling the sparsity or cardinality level. The iterative reweighted ℓ_1-norm method in Algorithm 13.1 provides the best combination of tracking error while controlling the sparsity at a low computational cost.
- In practice, deciding the sparsity level is typically done by trial and error while tuning some hyper-parameter in a laborious and computationally demanding way. The recently proposed *FDR-controlling index tracking method* is able to automatically determine the sparsity based on statistically sound hypothesis testing techniques.

Exercises

13.1 (Indices and ETFs) Download price data corresponding to some financial indices (e.g., the S&P 500, Dow Jones Industrial Average, Nasdaq) and some ETFs that track each of these indices (e.g., SPY for the S&P 500 index). Plot each index along with the corresponding ETFs in a linear and a logarithmic scale. Assess the tracking capabilities.

13.2 (Active vs. passive investments) Download price data corresponding to some mutual funds and compare with appropriate financial indices. Plot the price time series and compute some performance measure, such as the Sharpe ratio, to compare their performance. Do these results support the efficient-market hypothesis, promoted by Fama, or the inefficient and irrational markets, promoted by Shiller?

13.3 (Sparse regression via ℓ_1-norm) Generate an underdetermined system of linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with $\boldsymbol{A} \in \mathbb{R}^{5\times 10}$. Then, solve the following sparse underdetermined system of linear equations via brute force (i.e., trying all possible 2^{10} patterns for the variable $\boldsymbol{x}$):

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \|\boldsymbol{x}\|_0 \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{array}$$

Finally, solve the following linear program and compare the solution with the previous one:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \|\boldsymbol{x}\|_1 \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{array}$$

13.4 (Sparse least squares) Generate an overdetermined system of linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with $\boldsymbol{A} \in \mathbb{R}^{10\times 5}$. Consider the resolution of the sparse regression problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \\ \text{subject to} & \|\boldsymbol{x}\|_0 \leq k \end{array}$$

via the following list of methods and plot the trade-off curve of regression error vs. sparsity level for each method:

a. Brute force (i.e., trying all possible 2^5 patterns for the variable $\boldsymbol{x}$).

b. ℓ_1-norm approximation:

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda\|\boldsymbol{x}\|_1.$$

c. Concave approximation using a general-purpose nonlinear solver:

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda \sum_{i=1}^{N} \log\left(1 + \frac{|x_i|}{\varepsilon}\right).$$

d. Concave approximation again, but using the iterative reweighted ℓ_1-norm method.

13.5 (Cap-weighted indices) The portfolio of a cap-weighted index is defined in terms of the market capitalization. Denoting by $\boldsymbol{p}_t$ the prices of the N assets at time t and by $\boldsymbol{n}$ the number of outstanding shares of the N assets. The capital portfolio of the assets is defined to be proportional to the market capitalization $\boldsymbol{n} \odot \boldsymbol{p}_t$, which leads to the normalized portfolio

$$\boldsymbol{b}_t = \frac{\boldsymbol{n} \odot \boldsymbol{p}_t}{\boldsymbol{n}^\mathsf{T} \boldsymbol{p}_t}.$$

Show that this normalized portfolio can also be expressed as

$$\boldsymbol{b}_t = \frac{\boldsymbol{b}_{t-1} \odot (\boldsymbol{1} + \boldsymbol{r}_t)}{\boldsymbol{b}_{t-1}^\mathsf{T} (\boldsymbol{1} + \boldsymbol{r}_t)},$$

where the returns are defined as

$$\boldsymbol{r}_t = \frac{\boldsymbol{p}_t - \boldsymbol{p}_{t-1}}{\boldsymbol{p}_{t-1}} = \frac{\boldsymbol{p}_t}{\boldsymbol{p}_{t-1}} - \boldsymbol{1}.$$

13.6 (Tracking error measures) Download price data corresponding to some financial index (e.g., the S&P 500, Dow Jones Industrial Average, Nasdaq) and some ETFs that track the index (e.g., SPY for the S&P 500 index). Compute different error tracking measures, namely the ℓ_2-norm tracking error, the downside risk, the ℓ_1-norm tracking error, and the Huberized tracking error. Finally, plot a histogram of the tracking errors as a more complete picture of the tracking performance (note that the previous error measures are summarizations of the histogram).

13.7 (Two-stage index tracking methods) Download price data corresponding to some financial index, such as the S&P 500, and the corresponding constituent N assets for some period of time. Then, construct the benchmark return vector $\boldsymbol{r}^\mathrm{b}$ and the assets' return matrix $\boldsymbol{X}$, and formulate the sparse index tracking problem

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \dfrac{1}{T}\left\|\boldsymbol{r}^\mathrm{b} - \boldsymbol{X}\boldsymbol{w}\right\|_2^2 \\ \text{subject to} & \boldsymbol{1}^\mathsf{T}\boldsymbol{w} = 1,\ \boldsymbol{w} \geq \boldsymbol{0}, \\ & \|\boldsymbol{w}\|_0 \leq K. \end{array}$$

a. Solve the problem via a naive two-stage approach: simply select the K active assets with some heuristic and then renormalize so that $\mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1$.

b. Solve the problem via a two-stage approach with refitting of weights: select the K active assets as before and then solve the convex regression problem with the selected assets.

Plot the trade-off curve of regression error vs. sparsity level K for each method.

13.8 (Sparse index tracking methods via concave sparsity approximation) Download price data corresponding to some financial index, such as the S&P 500, and the corresponding constituent N assets for some period of time. Then, construct the benchmark return vector $\boldsymbol{r}^{\mathrm{b}}$ and the assets' return matrix $\boldsymbol{X}$, and formulate the sparse index tracking problem

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \frac{1}{T}\left\|\boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w}\right\|_2^2 + \lambda\|\boldsymbol{w}\|_0 \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1,\ \boldsymbol{w} \geq \mathbf{0}\end{array}$$

for different values of the hyper-parameter λ.

a. Approximate the sparsity regularizer with the concave log-function:

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \frac{1}{T}\left\|\boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w}\right\|_2^2 + \lambda\sum_{i=1}^{N}\log\left(1 + \frac{|w_i|}{\varepsilon}\right) \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1,\ \boldsymbol{w} \geq \mathbf{0}.\end{array}$$

Then solve the problem with a general-purpose nonconvex solver.

b. Apply the majorization–minimization approach to get the iterative reweighted ℓ_1-norm method that solves sequentially, $k = 0, 1, 2, \ldots$, the following:

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \frac{1}{T}\left\|\boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w}\right\|_2^2 + \lambda\sum_{i=1}^{N}\alpha_i^k|w_i| \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1,\ \boldsymbol{w} \geq \mathbf{0},\end{array}$$

where

$$\alpha_i^k = \frac{1}{\varepsilon + \left|w_i^k\right|}.$$

Plot the trade-off curve of regression error vs. sparsity level for each method (by varying the hyper-parameter λ).

13.9 (Sparse index tracking for downside risk) Download price data corresponding to some financial index, such as the S&P 500, and the corresponding constituent N assets for some period of time. Then, construct the benchmark return vector $\boldsymbol{r}^{\mathrm{b}}$ and the assets' return matrix $\boldsymbol{X}$, and formulate the sparse index tracking problem

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \frac{1}{T}\left\|\left(\boldsymbol{r}^{\mathrm{b}} - \boldsymbol{X}\boldsymbol{w}\right)^{+}\right\|_2^2 + \lambda\|\boldsymbol{w}\|_0 \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w} = 1,\ \boldsymbol{w} \geq \mathbf{0}\end{array}$$

for different values of the hyper-parameter λ.

a. Approximate the sparsity regularizer with the concave log-function and solve the problem with a general-purpose nonconvex solver.

b. Apply the majorization–minimization approach to get the iterative reweighted ℓ_1-norm method that solves sequentially the following convex problem:

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \dfrac{1}{T}\left\|\left(\boldsymbol{r}^{\text{b}}-\boldsymbol{X}\boldsymbol{w}\right)^{+}\right\|_2^2+\lambda\displaystyle\sum_{i=1}^{N}\alpha_i^k|w_i| \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w}=1,\ \boldsymbol{w}\geq\mathbf{0},\end{array}$$

where

$$\alpha_i^k=\frac{1}{\varepsilon+\left|w_i^k\right|}.$$

c. Apply the majorization–minimization approach fully to get the iterative reweighted ℓ_1-norm method that solves sequentially the following convex problem:

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \dfrac{1}{T}\left\|\left(\tilde{\boldsymbol{r}}^{\text{b}}\right)^k-\boldsymbol{X}\boldsymbol{w}\right\|_2^2+\lambda\displaystyle\sum_{i=1}^{N}\alpha_i^k|w_i| \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w}=1,\ \boldsymbol{w}\geq\mathbf{0},\end{array}$$

where now

$$\left(\tilde{\boldsymbol{r}}^{\text{b}}\right)^k=\boldsymbol{r}^{\text{b}}+\left(\boldsymbol{X}\boldsymbol{w}^k-\boldsymbol{r}^{\text{b}}\right)^{+}.$$

Plot the trade-off curve of regression error vs. sparsity level for each method (by varying the hyper-parameter λ).

13.10 (FDR-controlling method for sparse index tracking) Download price data corresponding to some financial index, such as the S&P 500, and the corresponding constituent N assets for some period of time. Then, construct the benchmark return vector $\boldsymbol{r}^{\text{b}}$ and the assets' return matrix $\boldsymbol{X}$, and formulate the sparse index tracking problem

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \dfrac{1}{T}\left\|\boldsymbol{r}^{\text{b}}-\boldsymbol{X}\boldsymbol{w}\right\|_2^2+\lambda\|\boldsymbol{w}\|_0 \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w}=1,\ \boldsymbol{w}\geq\mathbf{0}.\end{array}$$

a. Approximate the sparsity regularizer with the ℓ_1-norm:

$$\begin{array}{ll}\underset{\boldsymbol{w}}{\text{minimize}} & \dfrac{1}{T}\left\|\boldsymbol{r}^{\text{b}}-\boldsymbol{X}\boldsymbol{w}\right\|_2^2+\lambda\|\boldsymbol{w}\|_1 \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w}=1,\ \boldsymbol{w}\geq\mathbf{0}.\end{array}$$

Then solve the problem for different values of λ and plot the trade-off curve of regression error vs. sparsity level.

b. Employ the T-Rex method to automatically choose the active assets with FDR control.

References

Barber, B. M., & Odean, T. (2000). Trading is hazardous to your wealth: The common stock investment performance of individual investors. *The Journal of Finance*, *55*(2), 773–806.

Barber, R. F., & Candès, E. J. (2015). Controlling the false discovery rate via knockoffs. *The Annals of Statistics*, *43*(5), 2055–2085.

Beasley, J. E., Meade, N., & Chang, T. J. (2003). An evolutionary heuristic for the index tracking problem. *European Journal of Operational Research*, *148*, 621–643.

Benidis, K., Feng, Y., & Palomar, D. P. (2018a). Optimization methods for financial index tracking: From theory to practice. *Foundations and Trends in Optimization, Now Publishers*, *3*(3), 171–279.

Benidis, K., Feng, Y., & Palomar, D. P. (2018b). Sparse portfolios for high-dimensional financial index tracking. *IEEE Transactions on Signal Processing*, *66*(1), 155–170.

Benidis, K., & Palomar, D. P. (2019). *sparseIndexTracking: Design of Portfolio of Stocks to Track an Index* [R package]. https://CRAN.R-project.org/package=sparseIndexTracking

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, *57*(1), 289–300.

Candès, E. J., & Tao, T. (2005). Decoding by linear programming. *IEEE Transactions on Information Theory*, *51*(12), 4203–4215.

Candès, E. J., Wakin, M. B., & Boyd, S. P. (2008). Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, *14*, 877–905.

Daubechies, I., Devore, R., Fornasier, M., & Güntürk, C. S. N. (2010). Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, *63*, 1–38.

Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, *52*(4), 1289–1306.

Dose, C., & Cincotti, S. (2005). Clustering of financial time series with application to index and enhanced index tracking portfolio. *Physica A: Statistical Mechanics and its Applications*, *355*(1), 145–151.

Elad, M. (2010). *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, *25*(2), 383–417.

Harvey, C. R., Liu, Y., & Zhu, H. (2016). ... and the cross-section of expected returns. *Review of Financial Studies*, *29*(1), 5–68.

Huber, P. J. (2011). *Robust Statistics*. Springer.

Hunter, D. R., & Lange, K. (2004). A tutorial on MM algorithms. *The American Statistician*, *58*, 30–37.

Jansen, R., & Van Dijk, R. (2002). Optimal benchmark tracking with small portfolios. *Journal of Portfolio Management*, *28*(2), 33–39.

Machkour, J., Muma, M., & Palomar, D. P. (2025). The terminating-random experiments selector: Fast high-dimensional variable selection with false discovery rate control. *Signal Processing*.

Machkour, J., Palomar, D. P., & Muma, M. (2025). FDR-controlled portfolio optimization for sparse financial index tracking. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Machkour, J., Tien, S., Palomar, D. P., & Muma, M. (2024). *TRexSelector: T-Rex Selector: High-Dimensional Variable Selection & FDR Control* [R package]. https://CRAN.R-project.org/package=TRexSelector

Malkiel, B. G. (1973). *A Random Walk Down Wall Street*. W. W. Norton.

Maringer, D., & Oyewumi, O. (2007). Index tracking with constrained portfolios. *Intelligent Systems in Accounting, Finance and Management*, *15*(1–2), 57–71.

Meinshausen, N. (2013). Sign-constrained least squares estimation for high-dimensional regression. *Electronic Journal of Statistics*, *7*, 1607–1631.

Prigent, J. L. (2007). *Portfolio Optimization and Performance Analysis*. CRC Press.

Razaviyayn, M., Hong, M., & Luo, Z. (2013). A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, *23*(2), 1126–1153.

Scozzari, A., Tardella, F., Paterlini, S., & Krink, T. (2013). Exact and heuristic approaches for the index tracking problem with UCITS constraints. *Annals of Operations Research*, *205*(1), 235–250.

Shapcott, J. (1992). *Index Tracking: Genetic Algorithms for Investment Portfolio Selection* (tech. rep.). Report EPCC-SS92-24, Edinburgh Parallel Computing Centre, The University of Edinburgh.

Shiller, R. J. (1981). Do stock prices move too much to be justified by subsequent changes in dividends? *American Economic Review*, *71*(3), 421–436.

Sun, Y., Babu, P., & Palomar, D. P. (2017). Majorization–minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, *65*(3), 794–816.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, *58*(1), 267–288.

Xu, F., Lu, Z., & Xu, Z. (2016). An efficient optimization approach for a cardinality-constrained index tracking problem. *Optimization Methods and Software*, *31*(2), 258–271.

Xu, F., Ma, J., & Lu, H. (2022). Group sparse enhanced indexation model with adaptive beta value. *Quantitative Finance*, *22*(10), 1905–1926.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, *67*(2), 301–320.

14

Robust Portfolios

"Nobody's easier to fool [. . .] than the person who is convinced that he is right."

— Haruki Murakami, 1Q84

In theory there is no difference between theory and practice, while in practice there is.

— Benjamin Brewster, *The Yale Literary Magazine*

Markowitz's mean–variance portfolio optimizes a trade-off between expected return and risk measured by the variance. This formulation requires a prior estimation of some parameters: the mean vector and covariance matrix of the assets. In ideal conditions, this would be a perfectly fine approach. In practice, however, this fails miserably due to estimation errors in these parameters, which is one of the reasons why Markowitz's portfolio has not been widely adopted by practitioners. This has been referred to as the "Markowitz optimization enigma" and portfolio optimization problems have been called "estimation-error maximizers."

The parameters of an optimization problem have to be estimated and they will inevitably contain estimation errors. The naive approach simply ignores the existence of such estimation errors and proceeds as if the parameters were perfectly known. This leads to totally unacceptable solutions due to their instability and sensitivity to errors. Fortunately, several approaches have been proposed in the literature to mitigate such sensitivity.

This chapter explores two main approaches to deal with the error sensitivity:

- *Robust optimization* is a way to formulate optimization problems such that they are aware of the possible errors in the parameters. This approach goes back to the 1990s in the operations research literature and can be effectively applied to portfolio optimization.
- *Resampling and bootstrapping methods* constitute the bread and butter in the statistics literature. They rely on a smart resampling of the data to obtain multiple solutions that are then aggregated to form the final stable solution.

The good news is that these two philosophies are very mature and do not destroy the convexity (if any) of the original portfolio formulation.

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

14.1 Introduction

Markowitz's mean–variance portfolio (Markowitz, 1952) formulates the portfolio design as a trade-off between the expected return $\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}$ and the risk measured by the variance $\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w}$ (see Chapter 7 for details):

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} - \frac{\lambda}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{w} \\ \text{subject to} & \boldsymbol{w} \in \mathcal{W}, \end{array}$$

where $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ are the parameters, λ is a hyper-parameter that controls the investor's risk aversion, and $\mathcal{W}$ denotes an arbitrary constraint set, such as $\mathcal{W} = \{\boldsymbol{w} \mid \mathbf{1}^\mathsf{T}\boldsymbol{w} = 1, \boldsymbol{w} \geq \mathbf{0}\}$.

In practice, the parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ are unknown and have to be estimated using historical data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ containing the past T observations of the assets' returns. There is a wide span of different estimators, ranging from the simplest sample estimators to the more sophisticated shrinkage heavy-tailed maximum likelihood estimators (see Chapter 3 for a variety of estimation techniques). Regardless of the estimation method employed, there will always be an estimation error that depends on the number of observations. In practice, the amount of available historical data is limited (lack of stationarity does not help either) and the estimates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ will be very noisy, particularly $\hat{\boldsymbol{\mu}}$ (Best & Grauer, 1991; Chopra & Ziemba, 1993; Michaud, 1989); see Section 3.2 in Chapter 3 for more details.

This is, in fact, the "Achilles' heel" of portfolio optimization: the estimations $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ will inevitably contain estimation noise which will lead to erratic portfolio designs. This is why Markowitz's portfolio has not been fully embraced by practitioners. This was clearly summarized by Michaud in the context of mean–variance formulations (Michaud, 1989):

- Portfolio optimization problems are "estimation-error maximizers."
- "Optimal" portfolios are financially meaningless (absence of significant investment value).

Figure 14.1 illustrates the sensitivity of the mean–variance portfolio (with $\lambda = 1$) for six different realizations of the estimation error in the parameters. As can be observed, the behavior is totally erratic because the solutions are too sensitive to the errors in the parameters. In fact, each realization is very different from the others and this is unacceptable from a practical standpoint. One cannot let the portfolio allocation depend on the flap of a butterfly's wings.

14.2 Robust Portfolio Optimization

After a brief introduction to robust optimization, we will illustrate its application to portfolio design based on the mean–variance formulation, although the techniques can be applied to other portfolio formulations.

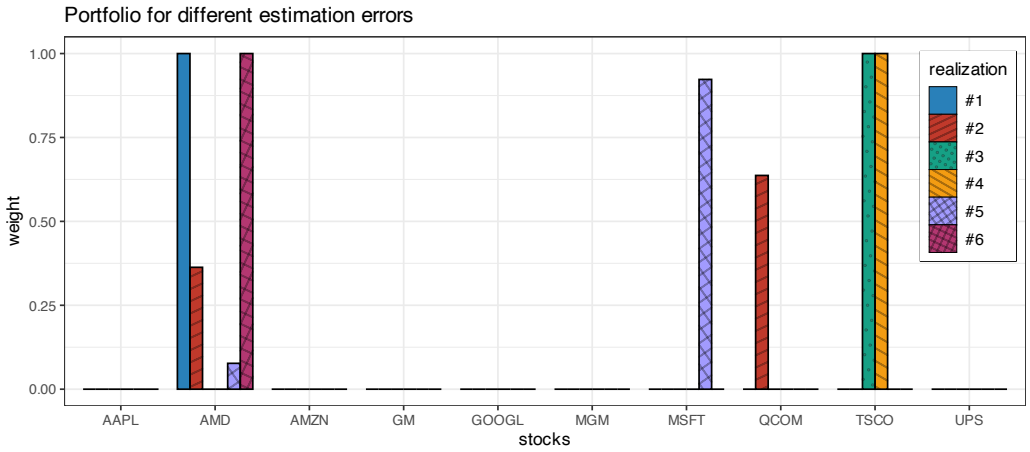


Figure 14.1 Sensitivity of the naive mean–variance portfolio.

14.2.1 Robust Optimization

Consider a general mathematical optimization problem with optimization variable $\boldsymbol{x}$ making explicit the dependency of the functions on the parameter $\boldsymbol{\theta}$:

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimize}} & f_0(\boldsymbol{x};\boldsymbol{\theta}) & \\ \text{subject to} & f_i(\boldsymbol{x};\boldsymbol{\theta}) \leq 0, & i = 1,\ldots,m, \\ & h_i(\boldsymbol{x};\boldsymbol{\theta}) = 0, & i = 1,\ldots,p, \end{array}$$

where f_0 is the objective function, $f_i,\ i = 1,\ldots,m$, are the inequality constraint functions, and $h_i,\ i = 1,\ldots,p$, are the equality constraint functions. The parameter $\boldsymbol{\theta}$ is something given externally and not to be optimized. For example, in the case of the mean–variance portfolio, we have $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$. We denote a solution to this problem by $\boldsymbol{x}^\star(\boldsymbol{\theta})$, where the dependency on $\boldsymbol{\theta}$ is again made explicit.

In practice, however, $\boldsymbol{\theta}$ is unknown and has to be estimated as $\hat{\boldsymbol{\theta}}$. The problem, of course, is that the solution obtained by solving the optimization problem using the estimated $\hat{\boldsymbol{\theta}}$, denoted by $\boldsymbol{x}^\star(\hat{\boldsymbol{\theta}})$, differs from the desired one: $\boldsymbol{x}^\star(\hat{\boldsymbol{\theta}}) \neq \boldsymbol{x}^\star(\boldsymbol{\theta})$. The question is whether it is still approximately equal, $\boldsymbol{x}^\star(\hat{\boldsymbol{\theta}}) \approx \boldsymbol{x}^\star(\boldsymbol{\theta})$, or totally different. The answer depends on each particular problem formulation; for the mean–variance portfolio formulation it turns out to be quite different (as illustrated in Figure 14.1).

There are several ways to make the problem robust to parameter errors:

- *Stochastic optimization* relies on a probabilistic modeling of the parameter (Birge & Louveaux, 2011; Prekopa, 1995; Ruszczynski & Shapiro, 2003) and may include:
  - expectations: average behavior;
  - chance constraints: probabilistic constraints.
- *Worst-case robust optimization* relies on the definition of an uncertainty set for the parameter (Ben-Tal & Nemirovski, 2008; Ben-Tal et al., 2009; Bertsimas et al., 2011; Lobo, 2000).

Stochastic Optimization

In stochastic robust optimization, the estimated parameter $\hat{\boldsymbol{\theta}}$ is modeled as a random variable that fluctuates around its true value $\boldsymbol{\theta}$. For convenience, the true value is modeled as a random variable around the estimated value (with an additive noise) as

$$\boldsymbol{\theta} = \hat{\boldsymbol{\theta}} + \boldsymbol{\delta},$$

where $\boldsymbol{\delta}$ is the estimation error modeled as a zero-mean random variable with some distribution such as Gaussian. In practice, it is critical to choose the covariance matrix (or shape) of the error term properly.

Then, rather than using a naive and wrong constraint of the form

$$f\left(\boldsymbol{x}; \hat{\boldsymbol{\theta}}\right) \leq \alpha,$$

we can use the "average constraint"

$$\mathbb{E}_{\boldsymbol{\theta}} \left[f(\boldsymbol{x}; \boldsymbol{\theta}) \right] \leq \alpha,$$

where the expectation is with respect to the random variable $\boldsymbol{\theta}$. The interpretation is that the constraint will be satisfied on average, which is a relaxation of the true constraint.

It is important to point out that this stochastic average constraint preserves convexity. That is, if $f(\cdot; \boldsymbol{\theta})$ is convex for each $\boldsymbol{\theta}$, then its expected value over $\boldsymbol{\theta}$ is also convex (the sum of convex functions preserves convexity, see Appendix A for details).

In practice, the expected value can be implemented in different ways, for example:

- *brute-force sampling*: simply sample the random variable $\boldsymbol{\theta}$ S times and use the constraint

$$\frac{1}{S} \sum_{i=1}^{S} f(\boldsymbol{x}; \boldsymbol{\theta}_i) \leq \alpha;$$

- *adaptive sampling*: sample the random variable $\boldsymbol{\theta}$ in a smart and efficient way at each iteration while the problem is being solved;
- *closed-form expression*: compute the expected value in closed form whenever possible (see Example 14.1);
- *stochastic programming*: a wide variety of numerical methods have been developed for stochastic optimization under the umbrella of stochastic programming (Birge & Louveaux, 2011; Prekopa, 1995; Ruszczynski & Shapiro, 2003).

Example 14.1 (Stochastic average constraint in closed form) Suppose we have the quadratic constraint $f(\boldsymbol{x}; \boldsymbol{\theta}) = (\boldsymbol{c}^\mathsf{T} \boldsymbol{x})^2$, where the parameter is $\boldsymbol{\theta} = \boldsymbol{c}$, modeled as $\boldsymbol{c} = \hat{\boldsymbol{c}} + \boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is zero-mean with covariance matrix $\boldsymbol{Q}$. Then the expected value is

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\theta}} \left[f(\boldsymbol{x}; \boldsymbol{\theta}) \right] &= \mathbb{E}_{\boldsymbol{\delta}} \left[\left((\hat{\boldsymbol{c}} + \boldsymbol{\delta})^\mathsf{T} \boldsymbol{x} \right)^2 \right] \\ &= \mathbb{E}_{\boldsymbol{\delta}} \left[(\hat{\boldsymbol{c}}^\mathsf{T} \boldsymbol{x})^2 + \boldsymbol{x}^\mathsf{T} \boldsymbol{\delta} \boldsymbol{\delta}^\mathsf{T} \boldsymbol{x} \right] \\ &= (\hat{\boldsymbol{c}}^\mathsf{T} \boldsymbol{x})^2 + \boldsymbol{x}^\mathsf{T} \boldsymbol{Q} \boldsymbol{x}, \end{aligned}$$

which, interestingly, has the form of the naive version plus a quadratic regularization term.

The problem with expectations and satisfying constraints on average is that there is no control about any specific realization of the estimation error. This relaxed control may not be acceptable in many applications as the constraint will be violated in many instances. The worst-case approach (considered later) tries to deal with this issue, but it may be too conservative. Chance constraints attempt to achieve a compromise between being too relaxed or overly conservative. They are also based on a statistical modeling of the estimation error, but instead of focusing on the average they focus on satisfying the constraint with some high probability (say, 95% of the cases). In this approach, the naive constraint $f(\boldsymbol{x};\hat{\boldsymbol{\theta}}) \leq \alpha$ is replaced with

$$\Pr\left[f(\boldsymbol{x};\boldsymbol{\theta}) \leq \alpha\right] \geq \epsilon,$$

where ϵ is the confidence level, say, $\epsilon = 0.95$ for 95%. Unfortunately, chance or probabilistic constraints are generally very hard to deal with and one typically has to resort to approximations (Ben-Tal & Nemirovski, 2008; Ben-Tal et al., 2009).

Worst-Case Robust Optimization

In worst-case robust optimization, the parameter $\boldsymbol{\theta}$ is not characterized statistically. Instead, it is assumed to lie within an uncertainty region near the estimated value:

$$\boldsymbol{\theta} \in \mathcal{U}_{\boldsymbol{\theta}},$$

where $\mathcal{U}_{\boldsymbol{\theta}}$ is the uncertainty set centered at $\hat{\boldsymbol{\theta}}$.

In practice, it is critical to choose the shape and size of the uncertainty set properly. Typical choices for the shape for a size of ϵ are:

- *spherical set*: $\mathcal{U}_{\boldsymbol{\theta}} = \left\{\boldsymbol{\theta} \mid \|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|_2 \leq \epsilon\right\}$;
- *box set*: $\mathcal{U}_{\boldsymbol{\theta}} = \left\{\boldsymbol{\theta} \mid \|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|_\infty \leq \epsilon\right\}$;
- *ellipsoidal set*: $\mathcal{U}_{\boldsymbol{\theta}} = \left\{\boldsymbol{\theta} \mid (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^\top \boldsymbol{S}^{-1} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \leq \epsilon^2\right\}$, where $\boldsymbol{S} \succ \boldsymbol{0}$ defines the shape of the ellipsoid.

Then, rather than using a naive and wrong constraint of the form

$$f\left(\boldsymbol{x};\hat{\boldsymbol{\theta}}\right) \leq \alpha,$$

we can use the worst-case constraint

$$\sup_{\boldsymbol{\theta} \in \mathcal{U}_{\hat{\boldsymbol{\theta}}}} f(\boldsymbol{x};\boldsymbol{\theta}) \leq \alpha.$$

The interpretation is that the constraint will be satisfied for any point inside the uncertainty set, which is a conservative approach.

It is important to point out that this worst-case constraint preserves convexity. That is, if $f(\cdot;\boldsymbol{\theta})$ is convex for each $\boldsymbol{\theta}$, then its worst case over $\boldsymbol{\theta}$ is also convex (the pointwise maximum of convex functions preserves convexity, see Appendix A for details).

In practice, the expected value can be implemented in different ways, for example:

- *brute-force sampling*: simply sample the uncertainty set $\mathcal{U}_{\boldsymbol{\theta}}$ S times and use the constraint

$$\max_{i=1,\ldots,S} f(\boldsymbol{x};\boldsymbol{\theta}_i) \leq \alpha$$

  or, equivalently, include S constraints

$$f(\boldsymbol{x};\boldsymbol{\theta}_i) \leq \alpha, \qquad i = 1,\ldots,S;$$

- *adaptive sampling algorithms*: sample the uncertainty set $\mathcal{U}_{\hat{\boldsymbol{\theta}}}$ in a smart and efficient way at each iteration while the problem is being solved;
- *closed-form expression*: compute the supremum in closed form whenever possible (see Example 14.2);
- *via Lagrange duality*: in some cases, it is possible to rewrite the worst-case supremum as an infimum that can later be combined with the outer portfolio optimization layer;
- *saddle-point optimization*: as a last resort, since the worst-case design is expressed in the form of a min–max (minimax) formulation, numerical methods specifically designed for minimax problems or related saddle-point problems can be used (Bertsekas, 1999; Tütüncü & Koenig, 2004).

Example 14.2 (Worst-case constraint in closed form) Suppose we have the quadratic constraint $f(\boldsymbol{x};\boldsymbol{\theta}) = (\boldsymbol{c}^\mathsf{T}\boldsymbol{x})^2$, where the parameter is $\boldsymbol{\theta} = \boldsymbol{c}$ and belongs to a spherical uncertainty set

$$\mathcal{U} = \{\boldsymbol{c} \mid \|\boldsymbol{c} - \hat{\boldsymbol{c}}\|_2 \leq \epsilon\}.$$

Then, the worst-case value is

$$\begin{aligned}\sup_{\boldsymbol{c}\in\mathcal{U}} |\boldsymbol{c}^\mathsf{T}\boldsymbol{x}| &= \sup_{\|\boldsymbol{\delta}\|\leq\epsilon} |(\hat{\boldsymbol{c}}+\boldsymbol{\delta})^\mathsf{T}\boldsymbol{x}| \\ &= |\hat{\boldsymbol{c}}^\mathsf{T}\boldsymbol{x}| + \sup_{\|\boldsymbol{\delta}\|\leq\epsilon} |\boldsymbol{\delta}^\mathsf{T}\boldsymbol{x}| \\ &= |\hat{\boldsymbol{c}}^\mathsf{T}\boldsymbol{x}| + \epsilon\|\boldsymbol{x}\|_2,\end{aligned}$$

where we have used the triangle inequality and the Cauchy–Schwarz inequality (with upper bound achieved by $\boldsymbol{\delta} = \epsilon\boldsymbol{x}/\|\boldsymbol{x}\|_2$). Again, this expression has the form of the naive version plus a regularization term.

As a final note, it is worth mentioning that worst-case uncertainty philosophy can be applied to probability distributions, referred to as distributional uncertainty models or distributionally robust optimization (Bertsimas et al., 2011).

14.2.2 Robust Worst-Case Portfolios

For portfolio design, we will focus on the worst-case robust optimization philosophy, due to its simplicity and convenience (Goldfarb & Iyengar, 2003; Lobo, 2000; Tütüncü & Koenig, 2004); see also textbooks such as Cornuejols and Tütüncü (2006) and Fabozzi et al. (2007).

Worst-Case Mean Vector $\boldsymbol{\mu}$

To study the uncertainty in $\boldsymbol{\mu}$, we consider the global maximum return portfolio (GMRP) for an estimated mean vector $\hat{\boldsymbol{\mu}}$ (see Chapter 6 for details):

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{maximize}} \quad & \boldsymbol{w}^\mathsf{T}\hat{\boldsymbol{\mu}} \\ \text{subject to} \quad & \mathbf{1}^\mathsf{T}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{aligned}$$

which is is known to be terribly sensitive to estimation errors.

We will instead use the worst-case formulation:

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{maximize}} \quad & \inf_{\boldsymbol{\mu}\in\mathcal{U}_{\boldsymbol{\mu}}} \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} \\ \text{subject to} \quad & \mathbf{1}^\mathsf{T}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{aligned}$$

where typical choices for the uncertainty region $\mathcal{U}_{\boldsymbol{\mu}}$ are the ellipsoidal and box sets considered next.

Lemma 14.1 (Worst-case mean vector under ellipsoidal uncertainty set) *Consider the ellipsoidal uncertainty set for $\boldsymbol{\mu}$*

$$\mathcal{U}_{\boldsymbol{\mu}} = \left\{\boldsymbol{\mu} = \hat{\boldsymbol{\mu}} + \kappa \boldsymbol{S}^{1/2}\boldsymbol{u} \mid \|\boldsymbol{u}\|_2 \leq 1\right\},$$

where $\boldsymbol{S}^{1/2}$ is the symmetric square-root matrix of the shape $\boldsymbol{S}$ (a typical choice is the covariance matrix scaled with the number of observations, $\boldsymbol{S} = (1/T)\boldsymbol{\Sigma}$) and κ determines the size of the ellipsoid. Then, the worst-case value of $\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}$ is

$$\begin{aligned} \inf_{\boldsymbol{\mu}\in\mathcal{U}_{\boldsymbol{\mu}}} \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} &= \inf_{\|\boldsymbol{u}\|\leq 1} \boldsymbol{w}^\mathsf{T}(\hat{\boldsymbol{\mu}} + \kappa \boldsymbol{S}^{1/2}\boldsymbol{u}) \\ &= \boldsymbol{w}^\mathsf{T}\hat{\boldsymbol{\mu}} + \kappa \inf_{\|\boldsymbol{u}\|\leq 1} \boldsymbol{w}^\mathsf{T}\boldsymbol{S}^{1/2}\boldsymbol{u} \\ &= \boldsymbol{w}^\mathsf{T}\hat{\boldsymbol{\mu}} - \kappa \, \|\boldsymbol{S}^{1/2}\boldsymbol{w}\|_2, \end{aligned}$$

which follows from the Cauchy–Schwarz inequality with $\boldsymbol{u} = -\boldsymbol{S}^{1/2}\boldsymbol{w}/\|\boldsymbol{S}^{1/2}\boldsymbol{w}\|_2$.

Lemma 14.2 (Worst-case mean vector under box uncertainty set) *Consider the box uncertainty set for $\boldsymbol{\mu}$*

$$\mathcal{U}_{\boldsymbol{\mu}} = \{\boldsymbol{\mu} \mid -\boldsymbol{\delta} \leq \boldsymbol{\mu} - \hat{\boldsymbol{\mu}} \leq \boldsymbol{\delta}\},$$

where $\boldsymbol{\delta}$ is the half-width of the box in all dimensions. Then, the worst-case value of $\boldsymbol{w}^\mathsf{T}\boldsymbol{\mu}$ is

$$\inf_{\boldsymbol{\mu}\in\mathcal{U}_{\boldsymbol{\mu}}} \boldsymbol{w}^\mathsf{T}\boldsymbol{\mu} = \boldsymbol{w}^\mathsf{T}\hat{\boldsymbol{\mu}} - |\boldsymbol{w}|^\mathsf{T}\boldsymbol{\delta},$$

where $|\boldsymbol{w}|$ denotes the elementwise absolute value of $\boldsymbol{w}$.

Thus, under an ellipsoidal uncertainty set as in Lemma 14.1, the robust version of the GMRP is

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{maximize}} \quad & \boldsymbol{w}^\mathsf{T}\hat{\boldsymbol{\mu}} - \kappa \, \|\boldsymbol{S}^{1/2}\boldsymbol{w}\|_2 \\ \text{subject to} \quad & \mathbf{1}^\mathsf{T}\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{aligned}$$

which is still a convex problem but now the problem complexity has increased to that of a second-order cone program (from a simple linear program in the naive formulation).

Similarly, under a box uncertainty set as in Lemma 14.2, the robust version of the GMRP is

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \boldsymbol{w}^\top \hat{\boldsymbol{\mu}} - |\boldsymbol{w}|^\top \boldsymbol{\delta} \\ \text{subject to} & \mathbf{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array}$$

which is still a convex problem and can be rewritten as a linear program after getting rid of the absolute value (see Appendix A for details). In fact, under the constraints $\mathbf{1}^\top \boldsymbol{w} = 1$ and $\boldsymbol{w} \geq \mathbf{0}$, the problem can be reduced to a naive GMRP where $\hat{\boldsymbol{\mu}}$ is replaced by $\hat{\boldsymbol{\mu}} - \boldsymbol{\delta}$.

There are other uncertainty sets that can also be considered, such as the ℓ_1-norm ball. The following example illustrates a rather interesting case of how the otherwise heuristic quintile portfolio (see Section 6.4.4 in Chapter 6 for details) can be formally derived as a robust portfolio.

Example 14.3 (Quintile portfolio as a robust portfolio) The quintile portfolio is a heuristic portfolio widely used by practitioners (see Section 6.4.4 in Chapter 6 for details). It selects the top fifth of the assets (one could also select a different fraction of the assets) over which the capital is equally allocated. This is a common-sense heuristic portfolio widely used in practice. Interestingly, it can be shown to be the optimal solution to the worst-case GMRP with an ℓ_1-norm ball uncertainty set around the estimated mean vector (Zhou & Palomar, 2020):

$$\mathcal{U}_{\mu} = \{\hat{\boldsymbol{\mu}} + \boldsymbol{e} \mid \|\boldsymbol{e}\|_1 \leq \epsilon\}.$$

Worst-Case Covariance Matrix $\boldsymbol{\Sigma}$

To study the uncertainty in $\boldsymbol{\Sigma}$, we consider the global minimum variance portfolio (GMVP) for an estimated mean vector $\hat{\boldsymbol{\Sigma}}$ (see Chapter 6 for details):

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \boldsymbol{w}^\top \hat{\boldsymbol{\Sigma}} \boldsymbol{w} \\ \text{subject to} & \mathbf{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array}$$

which is sensitive to estimation errors (albeit not as much as the previous sensitivity to errors in $\boldsymbol{\mu}$).

We will instead use the worst-case formulation:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \sup_{\boldsymbol{\Sigma} \in \mathcal{U}_{\boldsymbol{\Sigma}}} \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w} \\ \text{subject to} & \mathbf{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array}$$

where typical choices for the uncertainty region $\mathcal{U}_{\boldsymbol{\Sigma}}$ are the spherical, ellipsoidal, and box sets considered next.

Lemma 14.3 (Worst-case covariance matrix under a data spherical uncertainty set) *Consider the spherical uncertainty set for the data matrix $\boldsymbol{X} \in \mathbb{R}^{T \times N}$ containing T observations of the N assets,*

$$\mathcal{U}_{\boldsymbol{X}} = \left\{\boldsymbol{X} \mid \|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_{\mathrm{F}} \leq \epsilon\right\},$$

where ϵ determines the size of the sphere. Then, the worst-case value of $\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}$ under a

sample covariance matrix estimation $\hat{\boldsymbol{\Sigma}} = \frac{1}{T}\hat{\boldsymbol{X}}^\top \hat{\boldsymbol{X}}$ *is*

$$\begin{aligned}\sup_{\boldsymbol{X}\in\mathcal{U}_{\boldsymbol{X}}} \sqrt{\boldsymbol{w}^\top \left(\frac{1}{T}\boldsymbol{X}^\top \boldsymbol{X}\right)\boldsymbol{w}} &= \sup_{\boldsymbol{X}\in\mathcal{U}_{\boldsymbol{X}}} \frac{1}{\sqrt{T}}\|\boldsymbol{X}\boldsymbol{w}\|_2 \\ &= \sup_{\|\boldsymbol{\Delta}\|_{\mathrm{F}}\le\epsilon} \frac{1}{\sqrt{T}}\left\|(\hat{\boldsymbol{X}}+\boldsymbol{\Delta})\boldsymbol{w}\right\|_2 \\ &= \frac{1}{\sqrt{T}}\left\|\hat{\boldsymbol{X}}\boldsymbol{w}\right\|_2 + \sup_{\|\boldsymbol{\Delta}\|_{\mathrm{F}}\le\epsilon} \frac{1}{\sqrt{T}}\|\boldsymbol{\Delta}\boldsymbol{w}\|_2 \\ &= \frac{1}{\sqrt{T}}\left\|\hat{\boldsymbol{X}}\boldsymbol{w}\right\|_2 + \frac{1}{\sqrt{T}}\epsilon\|\boldsymbol{w}\|_2,\end{aligned}$$

where the third equality follows from the triangle inequality and is achieved when $\hat{\boldsymbol{X}}\boldsymbol{w}$ *and* $\boldsymbol{\Delta}\boldsymbol{w}$ *are aligned.*

Lemma 14.4 (Worst-case covariance matrix under an ellipsoidal uncertainty set) *Consider the ellipsoidal uncertainty set for* $\boldsymbol{\Sigma}$

$$\mathcal{U}_{\boldsymbol{\Sigma}} = \left\{\boldsymbol{\Sigma}\succeq\mathbf{0} \mid \left(\mathrm{vec}(\boldsymbol{\Sigma}) - \mathrm{vec}(\hat{\boldsymbol{\Sigma}})\right)^\top \boldsymbol{S}^{-1}\left(\mathrm{vec}(\boldsymbol{\Sigma}) - \mathrm{vec}(\hat{\boldsymbol{\Sigma}})\right) \le \epsilon^2\right\},$$

where $\mathrm{vec}(\cdot)$ *denotes the operator that stacks the matrix argument into a vector, matrix* $\boldsymbol{S}$ *gives the shape of the ellipsoid, and* ϵ *determines the size. Then, the worst-case value of* $\boldsymbol{w}^\top\boldsymbol{\Sigma}\boldsymbol{w}$ *given by the convex semidefinite problem*

$$\begin{aligned}\underset{\boldsymbol{\Sigma}}{\textit{maximize}} \quad & \boldsymbol{w}^\top\boldsymbol{\Sigma}\boldsymbol{w} \\ \textit{subject to} \quad & \left(\mathrm{vec}(\boldsymbol{\Sigma}) - \mathrm{vec}(\hat{\boldsymbol{\Sigma}})\right)^\top \boldsymbol{S}^{-1}\left(\mathrm{vec}(\boldsymbol{\Sigma}) - \mathrm{vec}(\hat{\boldsymbol{\Sigma}})\right) \le \epsilon^2 \\ & \boldsymbol{\Sigma}\succeq\mathbf{0},\end{aligned}$$

can be rewritten as the Lagrange dual problem:

$$\begin{aligned}\underset{\boldsymbol{Z}}{\textit{minimize}} \quad & \mathrm{Tr}\left(\hat{\boldsymbol{\Sigma}}\left(\boldsymbol{w}\boldsymbol{w}^\top + \boldsymbol{Z}\right)\right) + \epsilon\left\|\boldsymbol{S}^{1/2}\left(\mathrm{vec}(\boldsymbol{w}\boldsymbol{w}^\top) + \mathrm{vec}(\boldsymbol{Z})\right)\right\|_2 \\ \textit{subject to} \quad & \boldsymbol{Z}\succeq\mathbf{0},\end{aligned}$$

which is a convex semidefinite problem.

Lemma 14.5 (Worst-case covariance matrix under a box uncertainty set) *Consider the box uncertainty set for* $\boldsymbol{\Sigma}$

$$\mathcal{U}_{\boldsymbol{\Sigma}} = \left\{\boldsymbol{\Sigma}\succeq\mathbf{0} \mid \underline{\boldsymbol{\Sigma}} \le \boldsymbol{\Sigma} \le \overline{\boldsymbol{\Sigma}}\right\},$$

where $\underline{\boldsymbol{\Sigma}}$ *and* $\overline{\boldsymbol{\Sigma}}$ *denote the elementwise lower and upper bounds, respectively. Then, the worst-case value of* $\boldsymbol{w}^\top\boldsymbol{\Sigma}\boldsymbol{w}$*, given by the convex semidefinite problem*

$$\begin{aligned}\underset{\boldsymbol{\Sigma}}{\textit{maximize}} \quad & \boldsymbol{w}^\top\boldsymbol{\Sigma}\boldsymbol{w} \\ \textit{subject to} \quad & \underline{\boldsymbol{\Sigma}} \le \boldsymbol{\Sigma} \le \overline{\boldsymbol{\Sigma}}, \\ & \boldsymbol{\Sigma}\succeq\mathbf{0},\end{aligned}$$

can be rewritten as the Lagrange dual problem (Lobo, 2000)

$$\begin{aligned} \underset{\overline{\boldsymbol{\Lambda}},\underline{\boldsymbol{\Lambda}}}{\textit{minimize}} \quad & \operatorname{Tr}\left(\overline{\boldsymbol{\Lambda}}\,\overline{\boldsymbol{\Sigma}}\right) - \operatorname{Tr}\left(\underline{\boldsymbol{\Lambda}}\,\underline{\boldsymbol{\Sigma}}\right) \\ \textit{subject to} \quad & \begin{bmatrix} \overline{\boldsymbol{\Lambda}} - \underline{\boldsymbol{\Lambda}} & \boldsymbol{w} \\ \boldsymbol{w}^\mathsf{T} & 1 \end{bmatrix} \succeq \mathbf{0} \\ & \overline{\boldsymbol{\Lambda}} \geq \mathbf{0}, \quad \underline{\boldsymbol{\Lambda}} \geq \mathbf{0}, \end{aligned}$$

which is a convex semidefinite problem.

Thus, under a spherical uncertainty set for the data matrix as in Lemma 14.3, the robust version of the GMVP is

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \left(\left\| \hat{\boldsymbol{X}} \boldsymbol{w} \right\|_2 + \epsilon \|\boldsymbol{w}\|_2 \right)^2 \\ \text{subject to} \quad & \mathbf{1}^\mathsf{T} \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{aligned}$$

which is still a convex problem but now the problem complexity has increased to that of a second-order cone program (from a simple quadratic program in the naive formulation).

Interestingly, this problem bears a striking resemblance to a common heuristic called Tikhonov regularization:

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \left\| \hat{\boldsymbol{X}} \boldsymbol{w} \right\|_2^2 + \epsilon \|\boldsymbol{w}\|_2^2 \\ \text{subject to} \quad & \mathbf{1}^\mathsf{T} \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{aligned}$$

where the objective function can be rewritten as $\boldsymbol{w}^\mathsf{T} \frac{1}{T} \left(\hat{\boldsymbol{X}}^\mathsf{T} \hat{\boldsymbol{X}} + \epsilon \boldsymbol{I} \right) \boldsymbol{w}$, which leads to a regularized sample covariance matrix (see Section 3.6.1 in Chapter 3 for details on shrinkage).

Regarding the worst-case formulations that involve the Lagrange dual problem, the outer and inner minimizations can be simply written as a joint minimization. For example, under an ellipsoidal uncertainty set for the covariance matrix as in Lemma 14.4, the robust version of the GMVP is (Feng & Palomar, 2016)

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{W},\boldsymbol{Z}}{\text{minimize}} \quad & \operatorname{Tr}\left(\hat{\boldsymbol{\Sigma}} \left(\boldsymbol{W} + \boldsymbol{Z} \right) \right) + \epsilon \left\| \boldsymbol{S}^{1/2} \left(\operatorname{vec}(\boldsymbol{W}) + \operatorname{vec}(\boldsymbol{Z}) \right) \right\|_2 \\ \text{subject to} \quad & \begin{bmatrix} \boldsymbol{W} & \boldsymbol{w} \\ \boldsymbol{w}^\mathsf{T} & 1 \end{bmatrix} \succeq \mathbf{0} \\ & \mathbf{1}^\mathsf{T} \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \\ & \boldsymbol{Z} \succeq \mathbf{0}, \end{aligned}$$

which is still a convex problem but now the problem complexity has increased to that of a semidefinite program (from a simple quadratic program in the naive formulation). Note that the first matrix inequality is equivalent to $\boldsymbol{W} \succeq \boldsymbol{w}\boldsymbol{w}^\mathsf{T}$ and, at an optimal point, it can be shown to be satisfied with equality $\boldsymbol{W} = \boldsymbol{w}\boldsymbol{w}^\mathsf{T}$.

Worst-Case Mean Vector $\boldsymbol{\mu}$ and Covariance Matrix $\boldsymbol{\Sigma}$

The previous worst-case formulations against uncertainty in the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$ can be trivially combined under the mean–variance portfolio formulation.

For illustrative purposes, we consider the the mean–variance worst-case portfolio formulation

under box uncertainty sets for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, as in Lemmas 14.2 and 14.5, given by

$$\begin{array}{ll} \underset{\boldsymbol{w},\overline{\boldsymbol{\Lambda}},\underline{\boldsymbol{\Lambda}}}{\text{maximize}} & \boldsymbol{w}^\top\hat{\boldsymbol{\mu}} - |\boldsymbol{w}|^\top\boldsymbol{\delta} - \frac{\lambda}{2}\left(\operatorname{Tr}\big(\overline{\boldsymbol{\Lambda}}\,\overline{\boldsymbol{\Sigma}}\big) - \operatorname{Tr}\big(\underline{\boldsymbol{\Lambda}}\,\underline{\boldsymbol{\Sigma}}\big)\right) \\ \text{subject to} & \mathbf{1}^\top\boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \\ & \begin{bmatrix} \overline{\boldsymbol{\Lambda}} - \underline{\boldsymbol{\Lambda}} & \boldsymbol{w} \\ \boldsymbol{w}^\top & 1 \end{bmatrix} \succeq \mathbf{0}, \\ & \overline{\boldsymbol{\Lambda}} \geq \mathbf{0}, \quad \underline{\boldsymbol{\Lambda}} \geq \mathbf{0}, \end{array}$$

which is a convex semidefinite problem.

Other more sophisticated uncertainty sets can be considered for the mean vector and the covariance matrix, such as based on factor modeling (Goldfarb & Iyengar, 2003).

Other Worst-Case Performance Measures

Apart from the mean–variance framework for portfolio optimization, many other portfolio designs can be formulated based on other objective functions or performance measures (see Chapter 10 for details), as well as higher-order moments (see Chapter 9) or risk-parity portfolios (see Chapter 11). Each such formulation can be robustified so that the solution becomes more stable and robust against estimation errors in the parameters. For example, the worst-case Sharpe ratio can be easily managed, as well as the the worst-case value-at-risk (VaR) (El Ghaoui et al., 2003).

14.2.3 Numerical Experiments

The effectiveness of the different robust portfolio formulations depends on the shape as well as the size of the uncertainty region. These are parameters that have to be properly chosen and tuned to the type of data and nature of the financial market. Therefore, it is easy to overfit the model to the training data and extra care has to be taken.

The goal of a robust design should be in making the solution more stable and less sensitive to the error realization, but not necessarily improving the performance. In other words, one should aim at gaining robustness but not the best performance for a given backtest compared to a naive design.

We now evaluate robust portfolios under errors in the mean vector $\boldsymbol{\mu}$. Recall that robustness for the covariance matrix $\boldsymbol{\Sigma}$ is not as critical, because the bottleneck in the estimation part is on the mean vector (see Chapter 3 for details).

Sensitivity of Robust Portfolios

The extreme sensitivity of a naive mean–variance portfolio was shown in Figure 14.1. We now repeat the same numerical experiment with a robust formulation (still with $\lambda = 1$).

Figure 14.2 shows the sensitivity of a robust portfolio under an ellipsoidal uncertainty set for the mean vector $\boldsymbol{\mu}$ over six different realizations of the estimation error. Compared to Figure 14.1, it is clearly more stable and less sensitive. Many other variations in the robust formulation can be similarly considered with similar results.

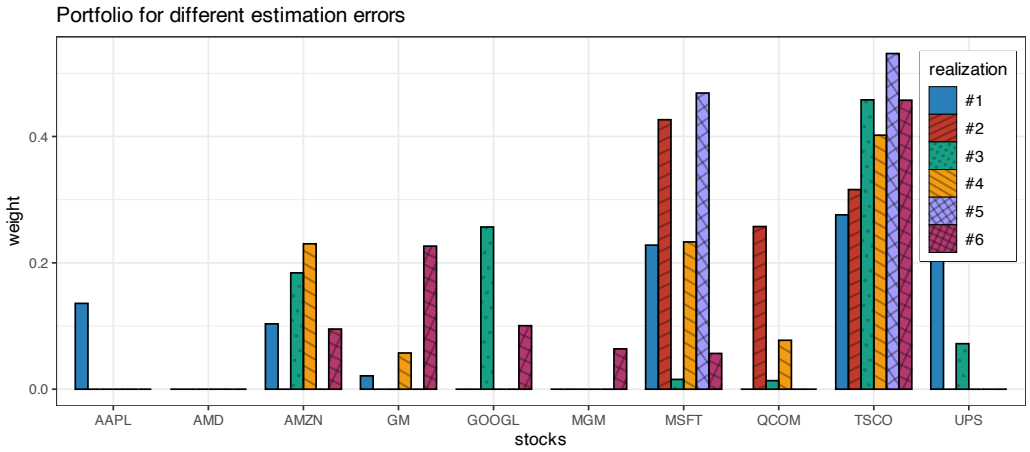


Figure 14.2 Sensitivity of the robust mean–variance portfolio under an ellipsoidal uncertainty set for $\boldsymbol{\mu}$.

Comparison of Naive vs. Robust Portfolios

Now that we have empirically observed the improved stability of mean–variance robust designs, we can assess their performance (with box and ellipsoidal uncertainty sets for the mean vector $\boldsymbol{\mu}$) in comparison with the naive design. Backtests are conducted for 50 randomly chosen stocks from the S&P 500 during 2017–2020.

Figure 14.3 shows the empirical distribution of the achieved mean–variance objective, as well as the Sharpe ratio, calculated over 1,000 Monte Carlo noisy observations. We can see that the robust designs avoid the worst-case realizations (the left tail in the distributions) at the expense of not achieving the best-case realizations (the right tail); thus, they are more stable and robust as expected.

Figure 14.4 shows the cumulative return and drawdown of the naive and robust mean–variance portfolios for an illustrative backtest. We can observe how the robust portfolios do indeed seem to be more robust. However, this is just a single backtest and more exhaustive multiple backtest are necessary.

Finally, multiple backtests are conducted for 200 realizations of 50 randomly chosen stocks from the S&P 500 from random periods during 2015–2020. Figure 14.5 shows the results in terms of the Sharpe ratio and drawdown, confirming that the robust portfolios are superior to the naive portfolio.

14.3 Portfolio Resampling

We will first give an overview of resampling methods in statistics and then apply them to portfolio optimization.

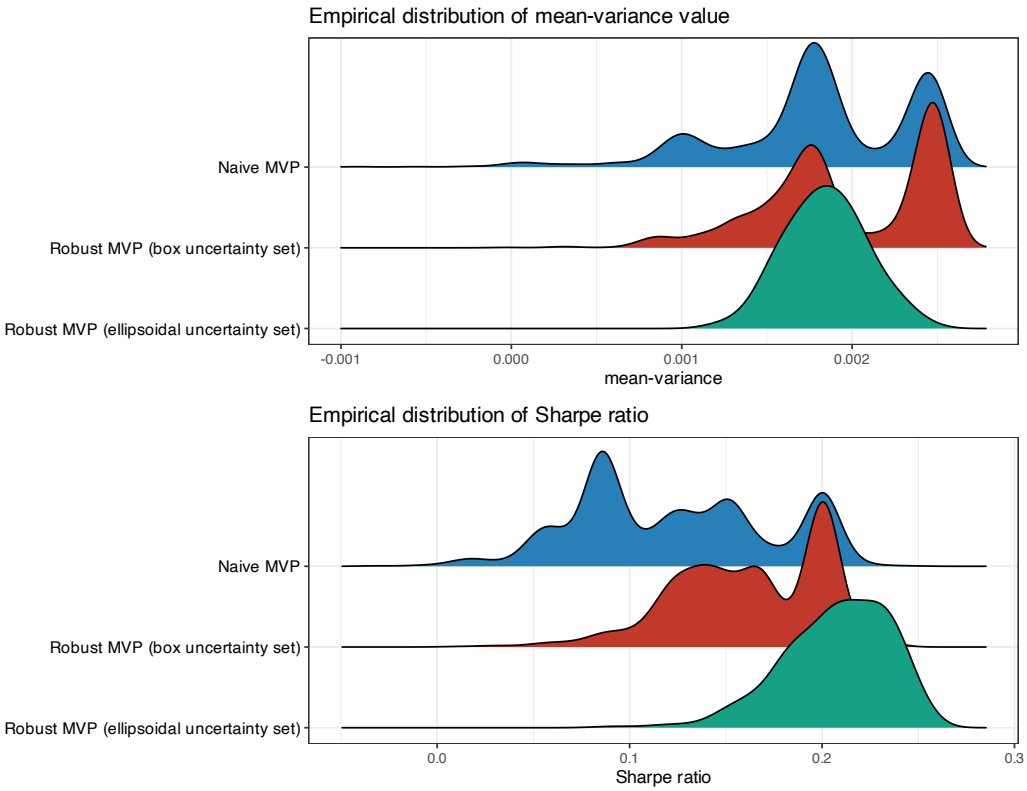


Figure 14.3 Empirical performance distribution of naive vs. robust mean–variance portfolios.

14.3.1 Resampling Methods

Estimating a parameter $\boldsymbol{\theta}$ with the value $\hat{\boldsymbol{\theta}}$ is of little use if one does not know how good that estimate is. In statistical inference, confidence intervals are key as they allow us to localize the true parameter on some interval with, say, 95% confidence. Traditionally, the derivation and analysis of confidence intervals was very theoretical with heavy use of mathematics. Resampling methods, instead, resort to computer-based numerical techniques for assessing statistical accuracy without formulas (Efron & Tibshirani, 1993).

In statistics, *resampling* is the creation of new samples based on a single observed sample block. Suppose we have n observations, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, of a random variable $\boldsymbol{x}$ from which we estimate some parameters $\boldsymbol{\theta}$ as

$$\hat{\boldsymbol{\theta}} = f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n),$$

where $f(\cdot)$ denotes the estimator. The estimation $\hat{\boldsymbol{\theta}}$ is a random variable because it is based on n random variables. It may seem that the only possible way to characterize the distribution of the estimation would be to somehow have access to more realizations of the random variable $\boldsymbol{x}$. However, this is precisely when resampling methods help to do some "magic." The most popular methods include cross-validation and the bootstrap (Efron & Tibshirani, 1993).

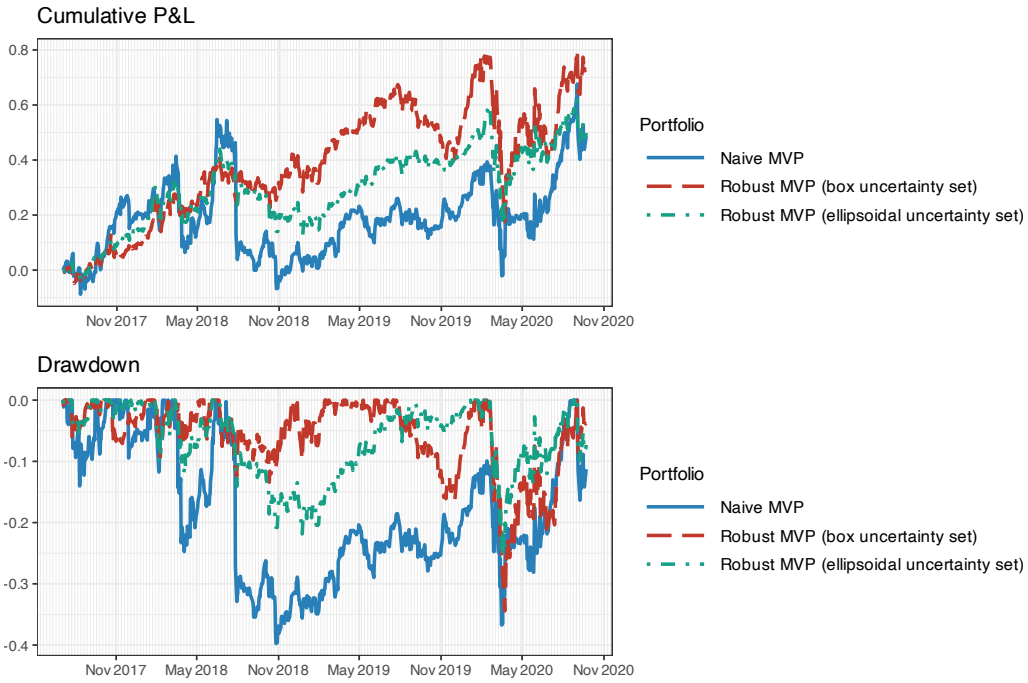


Figure 14.4 Backtest of naive vs. robust mean–variance portfolios.

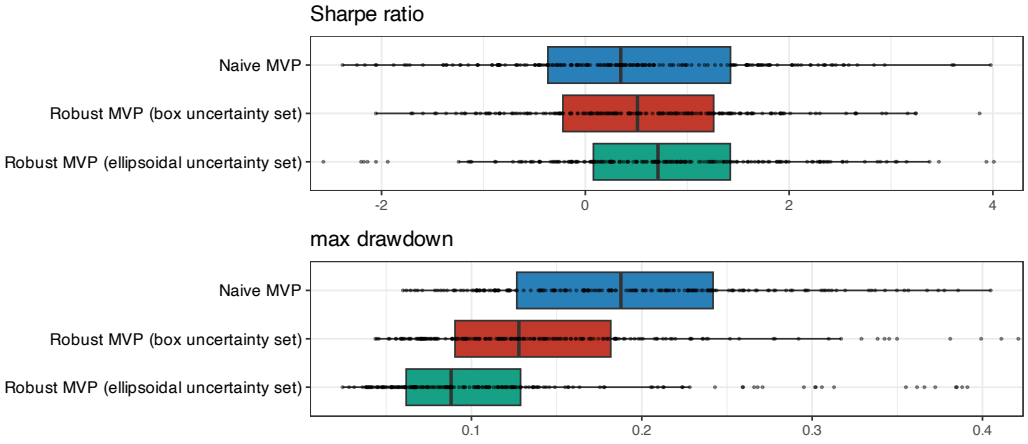


Figure 14.5 Multiple backtests of naive vs. resampled mean–variance portfolios.

Cross-Validation

Cross-validation is a type of resampling method widely used in portfolio backtesting (see Chapter 8) and machine learning (see Chapter 16). The idea is simple and consists of dividing the n observations into two groups: a training set for fitting or learning the estimator $f(\cdot)$ and a validation set for assessing its performance. This process can be repeated multiple times to provide multiple realizations of the performance value, which can then be used to

compute the empirical performance. For example, k-fold cross-validation divides the set into k subsets, where each is held back in turn as the validation set while using the others for training. Leave-one-out cross-validation is an extreme case where the original dataset of n observations is divided into $k = n$ subsets, which means that for each subset a single observation is held back from the training process and used later for validation.

The Bootstrap

The *bootstrap* is a type of resampling method proposed in 1979 by Efron, which appears truly magical but it is nevertheless based on sound statistical theory (Efron, 1979). In fact, the name itself (bootstrap) figuratively refers to the seemingly impossible task of lifting oneself by pulling on one's bootstraps.

The idea of the bootstrap is to mimic the original sampling process (from which the original n observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ were generated) by sampling these realizations n times with replacement (some samples will be selected multiple times while others will not be used). This procedure is repeated B times to obtain the bootstrap samples,

$$(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \rightarrow \left(\boldsymbol{x}_1^{*(b)}, \ldots, \boldsymbol{x}_n^{*(b)}\right), \quad b = 1, \ldots, B,$$

each of which leads to a different realization of the estimation (bootstrap replicates),

$$\hat{\boldsymbol{\theta}}^{*(b)} = f\left(\boldsymbol{x}_1^{*(b)}, \ldots, \boldsymbol{x}_n^{*(b)}\right), \quad b = 1, \ldots, B,$$

from which measures of accuracy of the estimator (bias, variance, confidence intervals, etc.) can then be empirically obtained.

The key theoretical result is that the statistical behavior of the random resampled estimates $\hat{\boldsymbol{\theta}}^{*(b)}$ compared to $\hat{\boldsymbol{\theta}}$ (taken as the true parameter) faithfully represent the statistics of the random estimates $\hat{\boldsymbol{\theta}}$ compared to the true (unknown) parameter $\boldsymbol{\theta}$. More exactly, the estimations of accuracy are asymptotically consistent as $B \rightarrow \infty$ (under some technical conditions) (Efron & Tibshirani, 1993). This is a rather surprising result that allows the empirical assessment of the accuracy of the estimator without having access to the true parameter $\boldsymbol{\theta}$.

Figure 14.6 illustrates the magic of the bootstrap to estimate the accuracy of the sample mean estimator (from $n = 100$ observations). In this case, the empirical distribution of the bias of the estimator is computed via $B = 1,000$ bootstraps, producing an accurate histogram compared to the true distribution. In practice, confidence intervals may suffice to assess the accuracy and fewer bootstraps may be used (even fewer bootstraps are necessary to compute the standard deviation of the bias).

The Jackknife

The *jackknife*, proposed in the mid-1950s by M. Quenouille, is the precursor of the bootstrap. It was derived for estimating biases and standard errors of sample estimators. Given the n observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, the ith *jackknife sample* is obtained by removing the ith data point, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_n$. This effectively produces $B = n$ bootstrap samples each with $n - 1$ observations. The jackknife can be shown to be an approximation to the bootstrap; more exactly, it makes a linear approximation to the bootstrap. Its accuracy depends on how

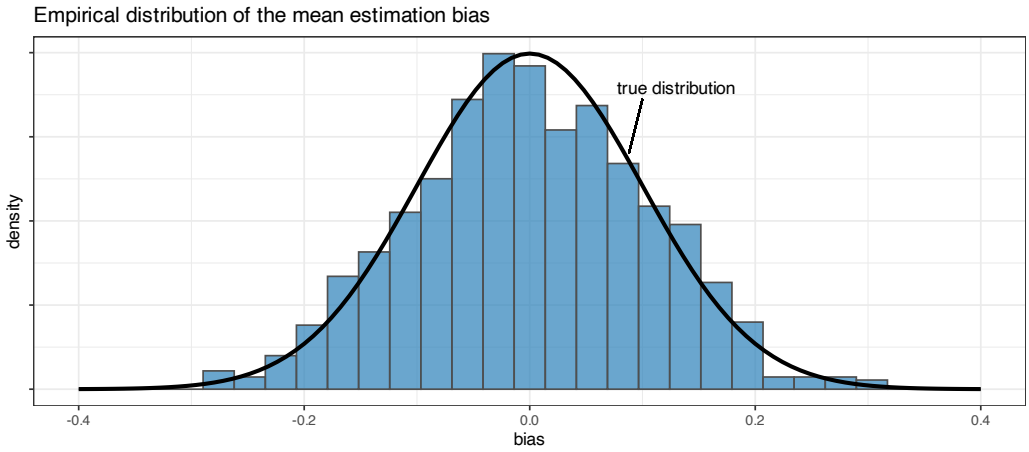


Figure 14.6 Empirical distribution of the sample mean bias via the bootstrap.

"smooth" the estimator is; for highly nonlinear functions the jackknife can be inefficient, sometimes dangerously so.

Variations of the Bootstrap

A number of variations and extensions of the basic bootstrap have been proposed over the years. Some notable examples include:

- *Parametric bootstrap*: The original bootstrap methodology mimics the true data distribution by sampling the observations with replacement. Thus, the procedure is distribution-independent or nonparametric. However, there are parametric versions of the bootstrap (Efron & Tibshirani, 1993). The idea is to make some assumption concerning the true data distribution (for example, assuming the family of Gaussian distributions), estimate the parameters of the distribution from the observed data, and then generate as much data as desired using that parametric distribution.

- *Block bootstrap*: The basic bootstrap method breaks down when the data contains structural dependency. A variety of block bootstrap methods have been proposed to deal with dependent data (Lahiri, 1999).

- *Random subspace method*: The random subspace method was proposed in the context of *decision trees* in order to decrease the correlation among trees and avoid overfitting (Ho, 1998). The idea is to let each learner use a randomly chosen subspace of the features. In fact, this was a key component in the development of *random forests* in machine learning.

- *Bag of little bootstraps*: In order to deal with large data sets with a massive number of observations, the bag of little bootstraps was proposed incorporating features of both the bootstrap and subsampling to yield a robust, computationally efficient means of assessing the quality of estimators (Kleiner et al., 2014).

Bagging

The bootstrap is as a way of assessing the accuracy of a parameter estimate or a prediction; interestingly, it can also be used to improve the estimate or prediction itself. *Bootstrap aggregating* or the acronym *bagging* refers to a method for generating multiple versions of some estimator or predictor via the bootstrap and then using these to get an aggregated version (Breiman, 1996; Hastie et al., 2009). Bagging can improve the accuracy of the basic estimator or predictor, which typically suffers from sensitivity to the realization of the random data. Mathematically, bagging is a simple average of the bootstrap replicates:

$$\hat{\boldsymbol{\theta}}^{\text{bag}} = \frac{1}{B}\sum_{b=1}^{B} \hat{\boldsymbol{\theta}}^{*(b)}.$$

14.3.2 Portfolio Resampling

In portfolio design, an optimization problem is formulated based on T observations of the assets' returns, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$, whose solution is supposedly an optimal portfolio $\boldsymbol{w}$. As previously discussed, this solution is very sensitive to the inherent noise in the observed data or in the noise in the estimated parameters used in the portfolio formulation, such as the mean vector $\hat{\boldsymbol{\mu}}$ and covariance matrix $\hat{\boldsymbol{\Sigma}}$.

Fortunately, we can capitalize on the results from the past half-century in statistics; in particular, we can use resampling techniques, such as the bootstrap and bagging, to improve the portfolio design.

The idea of resampling was proposed in the 1990s as a way to assess the accuracy of designed portfolios. The naive approach consists of using the available data to design a series of mean–variance portfolios and then obtaining the efficient frontier, but this is totally unreliable due to the high sensitivity of these portfolios on the data realization (the computed efficient frontier is not realistic and not representative of new data). Instead, resampling allows the computation of a more reliable efficient frontier, called the *resampled efficient frontier*, as well as the identification of statistically equivalent portfolios (Jorion, 1992; Michaud & Michaud, 1998).

In the context of data with temporal structure (as happens with many econometrics time series), apart from block bootstrap methods, a maximum entropy bootstrap has been proposed (Vinod, 2006). This method can be applied not only to time series of returns but even directly to time series of prices, which clearly have a strong temporal structure.

Portfolio Bagging

The technique of aggregating portfolios was considered in 1998 via a bagging procedure (Michaud & Michaud, 1998, 2007, 2008; Scherer, 2002):

1. Resample the original data $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ B times via the bootstrap method and estimate B different versions of the mean vector and covariance matrix: $\hat{\boldsymbol{\mu}}^{*(b)}$ and $\hat{\boldsymbol{\Sigma}}^{*(b)}$ for $b = 1, \ldots, B$.
2. Solve the optimal portfolio $\boldsymbol{w}^{*(b)}$ for each bootstrap sample.

3. Average the portfolios via bagging:

$$\boldsymbol{w}^{\text{bag}} = \frac{1}{B}\sum_{b=1}^{B} \boldsymbol{w}^{*(b)}.$$

This bagging procedure for portfolio aggregation is simple, and the only bottleneck is the increase in computational cost by a factor of the number of bootstraps B compared to the naive approach.

Portfolio Subset Resampling

An attempt to reduce the computational cost of the portfolio bagging procedure is the *subset resampling* technique (Shen & Wang, 2017). The idea is to sample the asset dimension rather than the observation (temporal) dimension, which is the same technique used to develop random forests (Ho, 1998). In more detail, instead of using all the N assets, the method randomly selects a subset, for which a portfolio of reduced dimensionality can be designed, which translates into a reduced computational cost. A rule of thumb is to select subsets of $\lceil N^{0.7} \rceil$ or $\lceil N^{0.8} \rceil$ assets; for example, for $N = 50$ the size of the subsets would be of 16 or 23, respectively. This procedure is repeated a number of times to finally aggregate all the computed portfolios. Note that since the portfolios are of reduced dimensionality, zeros are implicitly used in the elements corresponding to the other dimensions prior to the averaging.

A side benefit from the subset resampling technique, apart from the reduced computational cost, is that the parameters are better estimated because the ratio of observations to dimensionality is automatically increased in a significant way (see Chapter 3 for details on parameter estimation). For example, suppose we have $T = 252$ daily observations; the nominal ratio for $N = 50$ assets would be $T/N \approx 5$, whereas the ratio for subset resampling would be $T/N^{0.7} \approx 16$ or $T/N^{0.8} \approx 11$. Numerical experiments will confirm that this is a good technique in practice.

Portfolio Subset Bagging

Random subset resampling along the asset domain can be straightforwardly combined with the bootstrap along the temporal domain (Shen et al., 2019). In this case, each bootstrap sample only contains a subset of the N assets.

14.3.3 Numerical Experiments

The goal of portfolio resampling is in making the solution more stable and less sensitive to the errors in the parameter estimation, gaining in robustness compared to a naive design.

Sensitivity of Resampled Portfolios

The extreme sensitivity of the naive mean–variance portfolio was shown in Figure 14.1. Then, robust portfolio optimization was shown to be less sensitive in Figure 14.2. We now repeat the same numerical experiment with resampled portfolios to observe their sensitivity.

Figure 14.7 shows the sensitivity of a bagged portfolio with $B = 200$ bootstrap samples over six different realizations of the estimation error in the parameters. Compared to Figure 14.1, it is clear that bagging helps to produce more stable and less sensitive portfolios.

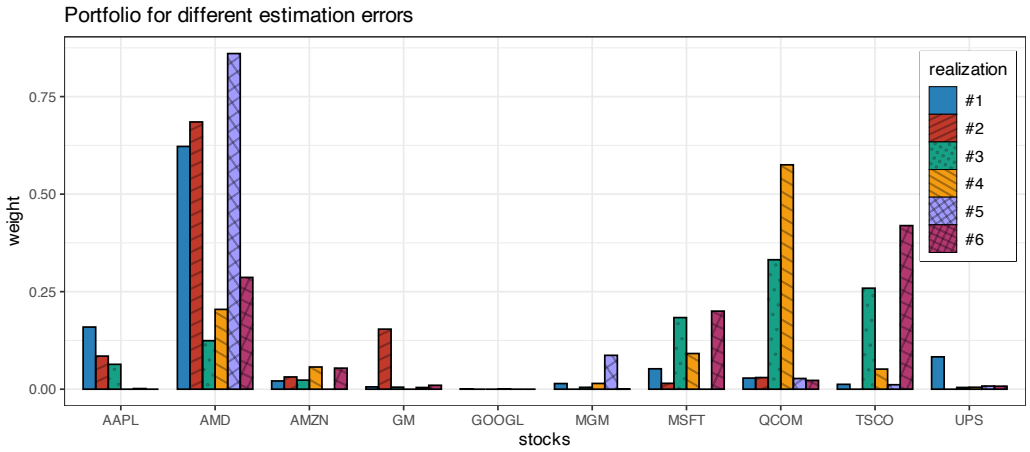


Figure 14.7 Sensitivity of the bagged mean–variance portfolio.

Comparison of Naive vs. Resampled Portfolios

Now that we have empirically observed the improved stability of mean–variance resampled portfolios, we can assess their performance in comparison with the naive design. In particular, we consider resampled portfolios via bagging, subset resampling, and subset bagging. Backtests are conducted for 50 randomly chosen stocks from the S&P 500 during 2017–2020.

Figure 14.8 shows the empirical distribution of the achieved mean–variance objective, as well as the Sharpe ratio, calculated over 1,000 Monte Carlo noisy observations. We can observe that resampled portfolios are more stable and do not suffer from extreme bad realizations (unlike the naive portfolio). However, the naive portfolio can be superior for some realizations.

Figure 14.9 shows the cumulative return and drawdown of the naive and resampled mean–variance portfolios for an illustrative backtest. We can observe how the resampled portfolios seem to be less noisy. However, this is just a single backtest and more exhaustive multiple backtests are necessary.

Finally, multiple backtests are conducted for 200 realizations of 50 randomly chosen stocks from the S&P 500 from random periods during 2015–2020. Figure 14.10 shows the results in terms of the Sharpe ratio and drawdown, confirming that the resampled portfolios are superior to the naive portfolio.

14.4 Summary

An optimal solution to a portfolio optimization problem typically produces unacceptable results in practice, which may seem counterintuitive. Why is that and how can we address it?

- Under ideal conditions, the solution to a portfolio formulation should indeed achieve the desired optimal objective subject to the constraints.
- In reality, however, it may fail miserably. The reason is that the formulation relies on some

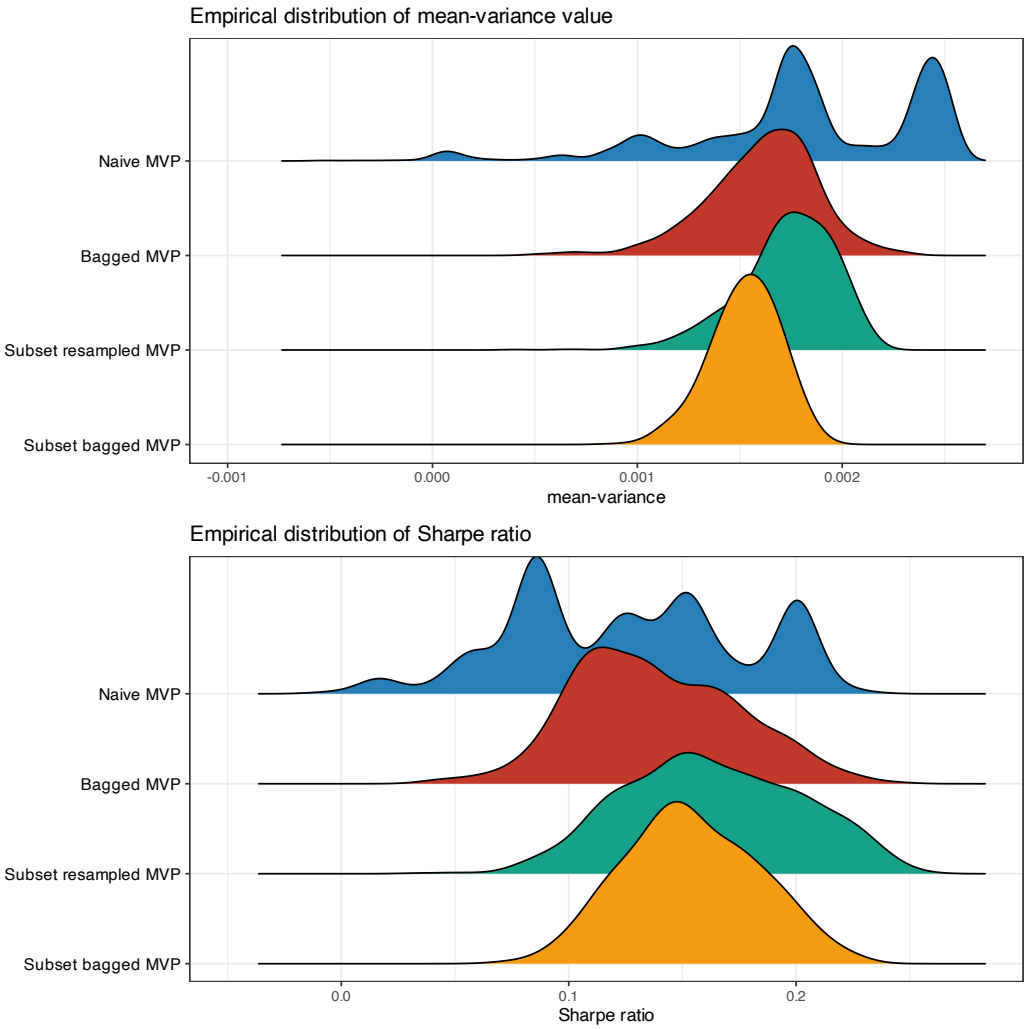


Figure 14.8 Empirical performance distribution of naive vs. resampled mean–variance portfolios.

parameters, such as the mean vector and covariance matrix, but these parameters have to be estimated from noisy and scarce data and will inevitably contain estimation errors.

- The end result of naively ignoring the parameter estimation errors in a portfolio formulation can be catastrophic. For this reason, such portfolio optimization problems have been called "estimation-error maximizers" with solutions that are financially meaningless.

- Some effective approaches to avoid naive solutions include:

  – *Robust portfolios*: Robust optimization is a mature approach in operations research that is able to incorporate the fact that the parameters in the formulation will contain

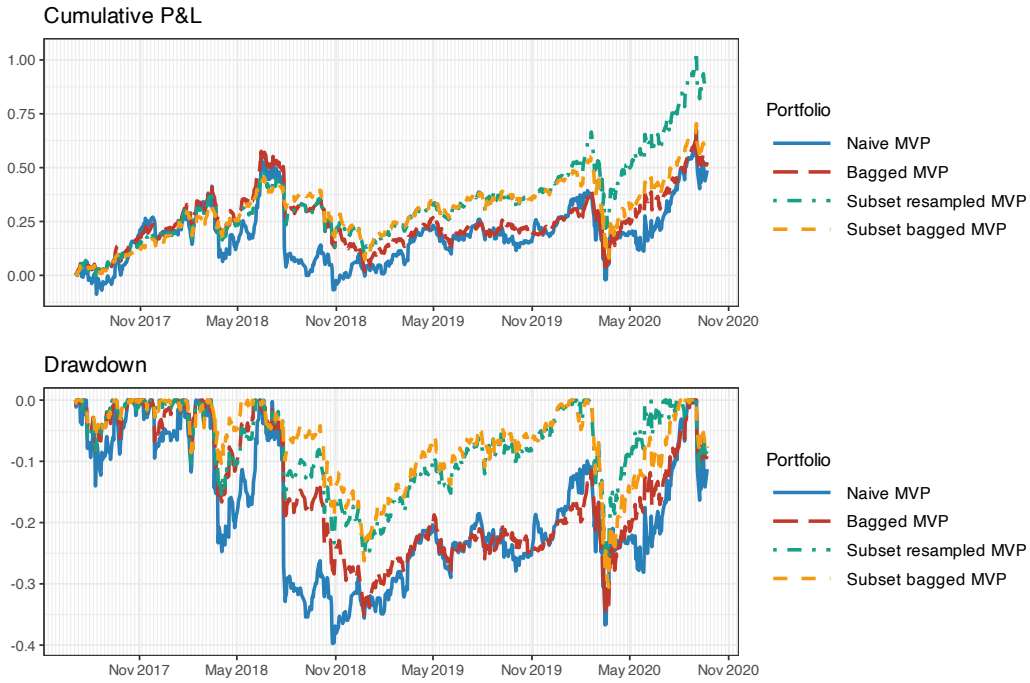


Figure 14.9 Backtest of naive vs. resampled mean–variance portfolios.

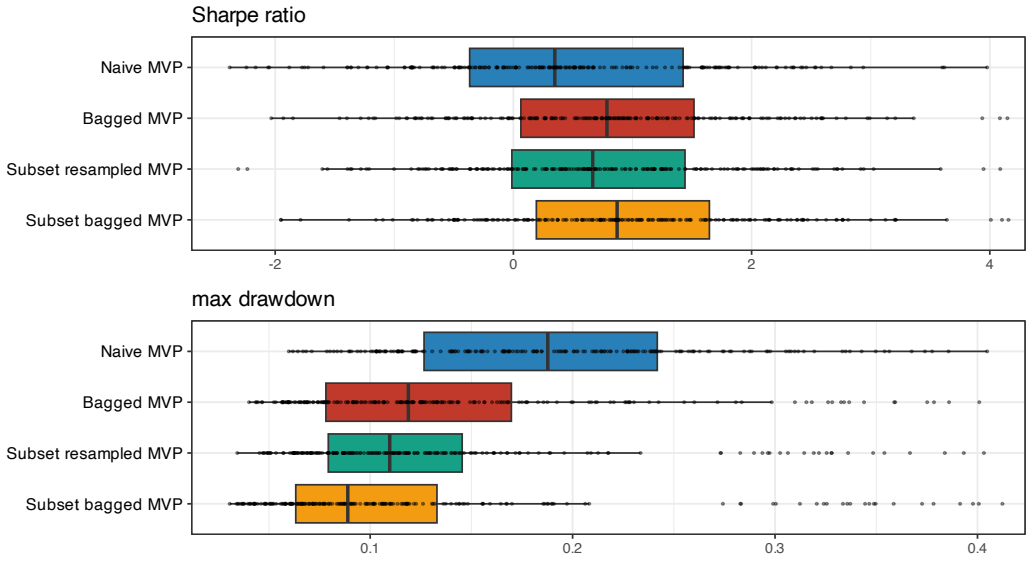


Figure 14.10 Multiple backtests of naive vs. resampled mean–variance portfolios.

some unknown errors (rather than naively assuming no errors). This has been widely developed in the context of portfolio optimization.

– *Resampled portfolios*: Bootstrapping and resampling are mature techniques in statistics that allow the aggregation of multiple naive solutions into a more stable and reliable solution. These techniques can be easily applied to portfolio design.

Exercises

14.1 (Sensitivity of naive portfolios)

a. Choose a portfolio optimization formulation.
b. Collect T observations of the returns of N assets, form the data matrix $\boldsymbol{X} \in \mathbb{R}^{T \times N}$, and estimate the mean vector and covariance matrix.
c. Sample B times new data matrices $\boldsymbol{X}^{(b)} \in \mathbb{R}^{T \times N}$, $b = 1, \ldots, B$, from a Gaussian distribution with the previous mean vector and covariance matrix as true parameters of the distribution.
d. For each data sample, solve the portfolio optimization obtaining the solutions $\boldsymbol{w}^{(b)}$, $b = 1, \ldots, B$.
e. Compare the different portfolios $\boldsymbol{w}^{(b)}$:
   - Use barplots to visually compare the allocations of the different portfolios.
   - Plot a histogram of the objective value achieved by the different portfolios (evaluated under the original mean vector and covariance matrix used to draw the sampled data).

14.2 (Sensitivity of robust worst-case portfolios) Repeat Exercise 14.1 but using a robust worst-case version of the portfolio formulation. Compare its sensitivity with that of the naive portfolio.

14.3 (Sensitivity of resampled portfolios) Repeat Exercise 14.1 but using a bagged version of the portfolio formulation. Compare its sensitivity with that of the naive portfolio.

14.4 (Worst-case mean vector under an ellipsoidal uncertainty set) Consider the ellipsoidal uncertainty set for $\boldsymbol{\mu}$:

$$\mathcal{U}_\mu = \left\{ \boldsymbol{\mu} = \hat{\boldsymbol{\mu}} + \kappa \boldsymbol{S}^{1/2} \boldsymbol{u} \mid \|\boldsymbol{u}\|_2 \leq 1 \right\},$$

where $\boldsymbol{S}^{1/2}$ is the symmetric square-root matrix of the shape $\boldsymbol{S}$ and κ determines the size of the ellipsoid.

Derive the worst-case value of $\boldsymbol{w}^\mathsf{T} \boldsymbol{\mu}$.

14.5 (Worst-case mean vector under a box uncertainty region) Consider the box uncertainty region for $\boldsymbol{\mu}$:

$$\mathcal{U}_\mu = \{ \boldsymbol{\mu} \mid -\boldsymbol{\delta} \leq \boldsymbol{\mu} - \hat{\boldsymbol{\mu}} \leq \boldsymbol{\delta} \},$$

where $\boldsymbol{\delta}$ is the half-width of the box in all dimensions.

Derive the worst-case value of $\boldsymbol{w}^\mathsf{T} \boldsymbol{\mu}$.

14.6 (Worst-case covariance matrix under a data spherical uncertainty region) Consider the spherical uncertainty region for the data matrix $\boldsymbol{X} \in \mathbb{R}^{T \times N}$ (containing T observations of the N assets),

$$\mathcal{U}_{\boldsymbol{X}} = \left\{ \boldsymbol{X} \mid \|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_\mathrm{F} \leq \epsilon \right\}.$$

Derive the worst-case value of $\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}$ under a sample covariance estimation $\hat{\boldsymbol{\Sigma}} = \frac{1}{T} \hat{\boldsymbol{X}}^\top \hat{\boldsymbol{X}}$.

14.7 (Robust worst-case mean–variance portfolio)

a. Formulate a minimax version of the robust worst-case mean–variance portfolio for some choice of uncertainty sets for the mean vector and covariance matrix.
b. Rewrite the problem in convex form by a brute-force sampling of the uncertainty sets and solve with a solver.
c. Rewrite the problem in convex form by properly dealing with the worst-case mean vector and covariance matrix (either deriving the closed form or via Lagrange duality), and solve with a solver.
d. Compare both solutions (try a different number of samples in the brute-force sampling approach).

14.8 (Convexity of robust mean–variance portfolio under ellipsoidal covariance matrix)

a. Formulate in minimax form the robust mean–variance portfolio with robustness in the variance under an ellipsoidal uncertainty set for the covariance matrix.
b. Using the Lagrange dual problem version of the worst-case covariance matrix, rewrite the robust mean–variance portfolio as a regular (not minimax) optimization problem.
c. Is this optimization problem convex? If not, can you rewrite it in convex form as a semidefinite program?

14.9 (Robust worst-case maximum Sharpe ratio portfolio) Write down the following portfolio formulations:

a. Naive formulation of the maximum Sharpe ratio portfolio in convex form.
b. Robust worst-case formulation of the maximum Sharpe ratio portfolio under general uncertainty sets for the mean vector and covariance matrix.
c. Choose some specific uncertainty regions and rewrite the robust worst-case formulation of the maximum Sharpe ratio in convex form.

14.10 (Performance of resampled portfolios)

a. Choose a portfolio optimization formulation.
b. Perform a backtest of
   - the naive portfolio
   - the bagged portfolio
   - the subset resampled portfolio
   - the subset bagged portfolio.
c. Compare the performance and the computational cost.

References

Ben-Tal, A., El Ghaoui, L., & Nemirovski, A. (2009). *Robust Optimization*. Princeton University Press.

Ben-Tal, A., & Nemirovski, A. (2008). Selected topics in robust convex optimization. *Mathematical Programming*, *112*(1), 125–158.

Bertsekas, D. P. (1999). *Nonlinear Programming*. Athena Scientific.

Bertsimas, D., Brown, D. B., & Caramanis, C. (2011). Theory and applications of robust optimization. *SIAM Review*, *53*(3), 464–501.

Best, M. J., & Grauer, R. R. (1991). On the sensitivity of mean–variance-efficient portfolios to changes in asset means: Some analytical and computational results. *Review of Financial Studies*, *4*(2), 315–342.

Birge, J. R., & Louveaux, F. V. (2011). *Introduction to Stochastic Programming*. Springer.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Chopra, V., & Ziemba, W. (1993). The effect of errors in means, variances and covariances on optimal portfolio choice. *Journal of Portfolio Management*, *19*(2), 6–11.

Cornuejols, G., & Tütüncü, R. (2006). *Optimization Methods in Finance*. Cambridge University Press.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, *7*(1), 1–26.

Efron, B., & Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Springer.

El Ghaoui, L., Oks, M., & Oustry, F. (2003). Worst-case value-at-risk and robust portfolio optimization: A conic programming approach. *Operations Research*, *51*(4), 543–556.

Fabozzi, F. J., Kolm, P. N., Pachamanova, D. A., & Focardi, S. M. (2007). *Robust Portfolio Optimization and Management*. John Wiley & Sons.

Feng, Y., & Palomar, D. P. (2016). A signal processing perspective on financial engineering. *Foundations and Trends in Signal Processing, Now Publishers*, *9*(1–2), 1–231.

Goldfarb, D., & Iyengar, G. (2003). Robust portfolio selection problems. *Mathematics of Operations Research*, *28*(1), 1–38.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8), 832–844.

Jorion, P. (1992). Portfolio optimization in practice. *Financial Analysts Journal*, *48*(1), 68–74.

Kleiner, A., Talwalkar, A., Sarkar, P., & Jordan, M. I. (2014). A scalable bootstrap for massive data. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, *76*(4), 795–816.

Lahiri, S. N. (1999). Theoretical comparisons of block bootstrap methods. *The Annals of Statistics*, *27*(1), 386–404.

Lobo, M. S. (2000). *Robust and Convex Optimization With Applications in Finance* [Doctoral dissertation, Stanford University].

Markowitz, H. M. (1952). Portfolio selection. *The Journal of Finance*, *7*(1), 77–91.

Michaud, R. O. (1989). The Markowitz optimization enigma: Is "optimized" optimal? *Financial Analysts Journal*, *45*(1), 31–42.

Michaud, R. O., & Michaud, R. O. (1998). *Efficient Asset Management: A Practical Guide to Stock Portfolio Optimization and Asset Allocation*. Harvard Business School Press.

Michaud, R. O., & Michaud, R. O. (2007). Estimation error and portfolio optimization: A resampling solution. *Journal of Investment and Management*, *6*(1), 8–28.

Michaud, R. O., & Michaud, R. O. (2008). *Efficient Asset Management: A Practical Guide to Stock Portfolio Optimization and Asset Allocation* (2nd ed.). Oxford University Press.

Prekopa, A. (1995). *Stochastic Programming*. Kluwer Academic.

Ruszczynski, A. P., & Shapiro, A. (2003). *Stochastic Programming*. Elsevier.

Scherer, B. (2002). Portfolio resampling: Review and critique. *Financial Analysts Journal*, *58*(6), 98–109.

Shen, W., Wang, B., Pu, J., & Wang, J. (2019). The Kelly growth optimal portfolio with ensemble learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(1), 1134–1141.

Shen, W., & Wang, J. (2017). Portfolio selection via subset resampling. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1), 1517–1523.

Tütüncü, R. H., & Koenig, M. (2004). Robust asset allocation. *Annals of Operations Research*, *132*(1), 157–187.

Vinod, H. D. (2006). Maximum entropy ensembles for time series inference in economics. *Journal of Asian Economics*, *17*, 955–978.

Zhou, R., & Palomar, D. P. (2020). Understanding the quintile portfolio. *IEEE Transactions on Signal Processing*, *68*, 4030–4040.

15

Pairs Trading Portfolios

"A drunk man will find his way home, but a drunk bird may get lost forever."

— Shizuo Kakutani

Pairs trading is a relative-value arbitrage strategy that has been known in the quantitative finance community since the mid-1980s. It seeks to identify two securities whose prices tend to stay together. Upon divergence, the undervalued security is bought long and the overvalued one is sold short, which is typically referred to as a contrarian philosophy (buy when everyone else is selling and vice versa). When the prices revert back to their historical equilibrium, the trade is closed and a profit is realized.

Mathematically, the two assets are combined into a virtual asset with mean reversion, that is, having an historical equilibrium value to which it eventually reverts. A key property of pairs trading is that it exploits the relative mispricings between the two securities while maintaining market neutrality (i.e., not being affected by the market trend). This is in contrast to momentum-based strategies, which precisely try to capture the market trend while treating the fluctuations as undesired noise. The extension of pairs trading to more than two assets is referred to as statistical arbitrage.

In a nutshell, while momentum-based strategies capitalize on the price trend, pairs trading exploits the mean-reverting fluctuations around that trend. This chapter starts with the basic concepts and covers the whole process, from discovering pairs to trading them while tracking the cointegration relationship over time with rolling least squares and Kalman filtering.

15.1 Mean Reversion

Mean reversion is a property of a time series that means that there is a long-term average value around which the series may fluctuate over time but eventually will revert back to (Chan, 2013; Ehrman, 2006; Vidyamurthy, 2004). This property plays a crucial role in pairs trading, where the mean-reverting time series is artificially constructed by combining two (or more) assets. The mean reversion allows the trader to buy at a low price with the expectation that the price will return to the long-term mean over time. When the price reverts back to its

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

historical average, the trader will close the position to realize a profit. Of course, this relies on the assumption that the historical relationship between the assets will persist in the future, which carries some risk; careful monitoring is key.

More generally, there are two types of mean reversion that trading strategies commonly exploit (Chan, 2013):

- *Longitudinal or time series mean reversion*: This occurs when mean reversion takes place along the time axis, and there is a long-term average value. The deviation happens at one point in time in one direction and at another point in time in the opposite direction.
- *Cross-sectional mean reversion*: This type of mean reversion occurs along the asset axis, and there is an average value across assets. Some assets deviate in one direction, while others deviate in the opposite direction (Fabozzi et al., 2010).

Stationarity is a property related to mean reversion, but different. Stationarity refers to the property that the statistics of a time series remain fixed over time. In that sense, a stationary time series can be considered mean reverting (Vidyamurthy, 2004), but not the other way around.

Unit-root stationarity is a specific type of stationarity (Tsay, 2010, 2013). It refers to modeling the time series with an autoregressive (AR) model with no unit roots (this is related to an Ornstein–Uhlenbeck process in continuous time). A time series with a unit root is not stationary and tends to diverge over time. A notable example of unit-root nonstationarity is the random walk model commonly employed for log-prices (see Chapter 3):

$$y_t = \mu + y_{t-1} + \epsilon_t,$$

where y_t denotes the log-price at period t, μ is the drift, and ϵ_t is the residual. Figure 15.1 shows an example of a time series with a unit root that does not return to the mean in a controlled way.[1] On the other hand, in the absence of a unit root, the time series will not diverge and will eventually return to the mean; an example is the AR model of order 1 (AR(1)):

$$y_t = \mu + \rho\, y_{t-1} + \epsilon_t,$$

with $|\rho| < 1$. Figure 15.2 illustrates an AR(1) time series with no unit root ($\rho = 0.2$) and $\mu = 0$, which reverts to the mean in a controlled way.

Even though mean reversion and unit-root stationarity are not equivalent concepts, from a practical standpoint unit-root stationarity is a convenient proxy for mean reversion (Tsay, 2010, 2013). In fact, testing for unit-root stationarity is the de facto approach for determining mean reversion in practice.

Differencing is an operation commonly used to obtain stationarity (Tsay, 2010, 2013). It refers to taking differences between consecutive samples of a time series $y_1, y_2, y_3, \ldots$ to produce $\Delta y_t = y_t - y_{t-1}$. The importance of this operation is that a nonstationary time series, such as a random walk, may become stationary after differencing. This is precisely the case when

[1] Mathematically, it can be shown that a random walk in one dimension and even in two dimensions (e.g., a drunken man walking on a surface) will eventually return to the starting point. Interestingly, this property does not hold in three dimensions (e.g., a drunken bird flying).

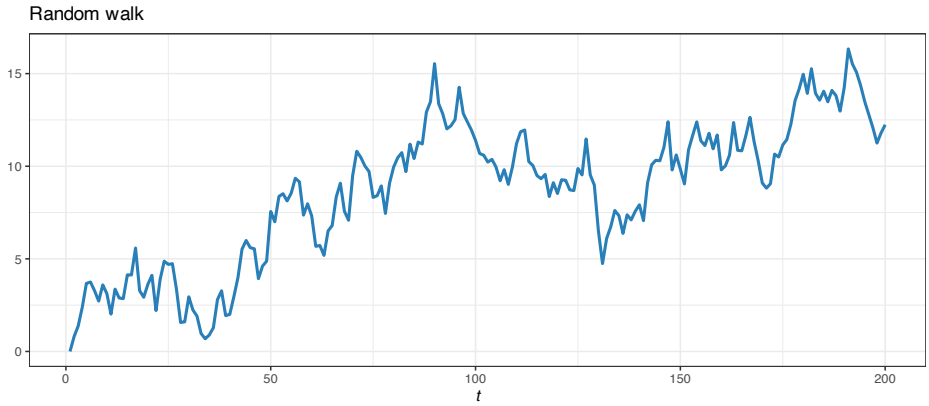


Figure 15.1 Example of a random walk (nonstationary time series with unit root).

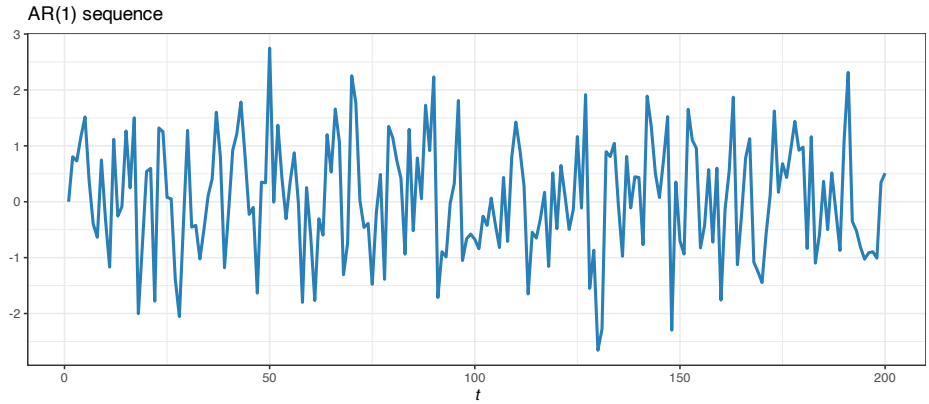


Figure 15.2 Example of a unit-root stationary AR(1) sequence.

differencing the log-prices of an asset to obtain the log-returns (see Chapter 3 for details); we then say that log-prices are integrated of order 1 (higher-order differencing can also be considered).

15.2 Cointegration and Correlation

Mean reversion, as previously described, refers to the tendency of a time series to return to its long-term average value over time. This property allows for a simple trading strategy: buy when below the average and sell when above. While it is virtually impossible to find an asset with controlled and predictable mean reversion, it is much easier to discover pairs of assets with a combined mean reversion property.

15.2.1 Cointegration

Cointegration refers to a property by which two (or more) assets, while not being mean reverting individually, may be mean reverting with respect to each other (Chan, 2013; Ehrman,

2006; Vidyamurthy, 2004). This commonly happens when the series themselves contain stochastic trends (i.e., they are nonstationary) but nevertheless they move closely together over time in a way that their difference remains stable (i.e., stationary). Thus, the concept of cointegration mimics the existence of a long-run equilibrium to which an economic system converges over time.

The intuitive idea is that, while it may be difficult or impossible to predict individual assets, it may be easier to predict their relative behavior. The typical example used to illustrate the concept of cointegration is that of a drunken man wandering the streets (random walk) with a dog (illustrated in Figure 15.3). Both paths of man and dog are nonstationary and difficult to predict, but the distance between them is mean reverting and stationary.



Figure 15.3 Random walk by a drunken man with a dog.[2]

Mathematically, a multivariate time series, $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, \ldots$, is cointegrated if some linear combination becomes integrated of lower order, for example, if $\boldsymbol{y}_t$ is not stationary but the linear combination $\boldsymbol{w}^\mathsf{T}\boldsymbol{y}_t$ is stationary for some weights $\boldsymbol{w}$. In this sense, cointegration can be thought of as a more refined version of a time series being integrated of order 1. To be more specific, suppose the multivariate time series $\boldsymbol{y}_t$ denotes the log-prices of some stocks. Such a time series is nonstationary (random walk) but after differencing we obtain the log-returns, which are stationary. Cointegration provides a more refined version that allows us to obtain a stationary time series without having to difference it. Instead, by taking a linear combination $\boldsymbol{w}^\mathsf{T}\boldsymbol{y}_t$ we might be able to obtain a stationary time series. As covered later, this property has remarkable consequences in terms of trading and it forms the basics of pairs trading.

A simple and common way to model cointegration of two time series is as

$$\begin{aligned} y_{1t} &= \gamma\, x_t + w_{1t}, \\ y_{2t} &= x_t + w_{2t}, \end{aligned} \tag{15.1}$$

[2] Image credit: doodlemachine/DigitalVision Vectors/Getty Images.

where x_t is a stochastic common trend defined as a random walk,

$$x_t = x_{t-1} + w_t,$$

and the terms w_{1t}, w_{2t}, w_t are i.i.d. residual terms, mutually independent, with variances σ_1^2, σ_2^2, and σ^2, respectively. The coefficient γ is the key quantity that determines the cointegration relationship. It is important to note that each of the time series, y_{1t} and y_{2t}, is a random walk plus additional noise, therefore nonstationary. However, since they share a common stochastic trend, a simple linear combination of the two can eliminate this trend. The so-called *spread* is precisely this linear combination without the trend:

$$z_t = y_{1t} - \gamma\, y_{2t} = w_{1t} - \gamma\, w_{2t},$$

which is stationary and mean reverting.

15.2.2 Correlation

Correlation is a basic concept in probability that refers to how "related" two random variables are. We can use this measure for stationary time series but definitely not with nonstationary time series. In fact, when we refer to correlation between two financial assets, we are actually employing this concept on the returns of the assets and not the price values.

Specifically, given two time series of log-prices, y_{1t} and y_{2t}, we can obtain the log-returns as the differences Δy_{1t} and Δy_{2t}. Then the correlation can be safely defined, assuming stationarity, as

$$\rho = \frac{\mathbb{E}\left[(\Delta y_{1t} - \mu_1) \cdot (\Delta y_{2t} - \mu_2)\right]}{\sqrt{\mathrm{Var}(\Delta y_{1t}) \cdot \mathrm{Var}(\Delta y_{2t})}},$$

where μ_1 and μ_2 denote the means of Δy_{1t} and Δy_{2t}, respectively, and the denominator normalizes with respect to the variances of the two variables, $\mathrm{Var}(\Delta y_{1t})$ and $\mathrm{Var}(\Delta y_{1t})$, so that the correlation is bounded as $-1 \leq \rho \leq 1$.

The interpretation of correlation is quite simple: it is high when the two time series co-move (they move simultaneously in the same direction) and it is zero when they move independently.

15.2.3 Correlation vs. Cointegration

At this point, the concepts of correlation and cointegration have been introduced, but their similarity and difference may be unclear and confusing. After all, it seems that they both try to capture the concept of similarity of movements of two time series, so superficially they may seem to be similar concepts. However, they are totally different right from their definition.

As a matter of fact, the correlation of the differences of the two cointegrated time series in the model (15.1) can be analytically derived as

$$\rho = \frac{1}{\sqrt{1 + 2\frac{\sigma_1^2}{\sigma^2}}\sqrt{1 + 2\frac{\sigma_2^2}{\sigma^2}}},$$

which can be made as small as desired by properly choosing the variances of the residual terms

σ_1^2, σ_2^2, and σ^2. That is, we can have two perfectly cointegrated time series with an arbitrarily small correlation, which may be surprising at first. This reveals that cointegration and correlation are two totally different concepts, yet they both attempt to measure the similarity of the movements of two time series. The following examples illustrate this difference.

Example 15.1 (Example of cointegrated time series with low correlation) Consider the common trend model in (15.1) with $\gamma = 1$ and the standard deviations $\sigma = 0.1$ and $\sigma_1 = \sigma_2 = 0.2$. The theoretical correlation is $\rho = 0.111$, whereas the empirical correlation computed with 200 observations is $\rho = 0.034$, and with 2 000 observations $\rho = 0.108$. Figure 15.4 shows the two nonstationary time series, y_{1t} and y_{2t}, the stationary spread, z_t, as well as the scatter plot of the differences Δy_{1t} vs. Δy_{2t}, which does not show any preferred direction as expected for low correlation.

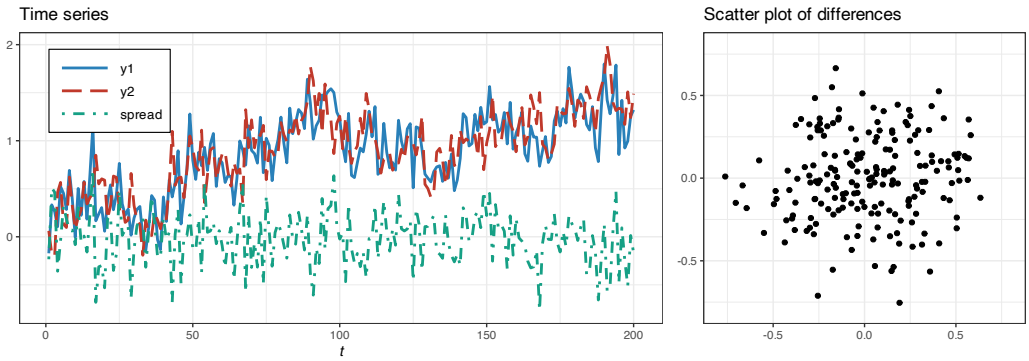


Figure 15.4 Example of cointegrated time series with low correlation.

Example 15.2 (Example of non-cointegrated time series with high correlation) Consider the common trend model in (15.1) with $\gamma = 1$ and the standard deviations $\sigma = 0.3$ and $\sigma_1 = \sigma_2 = 0.05$. In addition, add the linear trend $0.01 \times t$ to the first time series y_{1t}, which will destroy the cointegration between the two time series while not affecting the correlation. In this case, the theoretical correlation is $\rho = 0.947$, whereas the empirical correlation computed with 200 observations is $\rho = 0.952$, and with 2 000 observations $\rho = 0.941$. Figure 15.5 shows the two nonstationary time series, y_{1t} and y_{2t}, the nonstationary spread, z_t, as well as the scatter plot of the differences Δy_{1t} vs. Δy_{2t}, which clearly shows a preferred direction as expected for high correlation.

Thus, both correlation and cointegration attempt to measure the same concept of co-movement of time series, but they do it in very different ways, namely:

- Correlation is high when the two time series co-move (they move simultaneously in the same direction) and zero when they move independently.
- Cointegration is high when the two time series move together and remain close to each other, and nonexistent when they do not stay together.

One way to understand the fundamental difference is in terms of short term vs. long term. Correlation is concerned with the short-term movements, that is, the directional movement

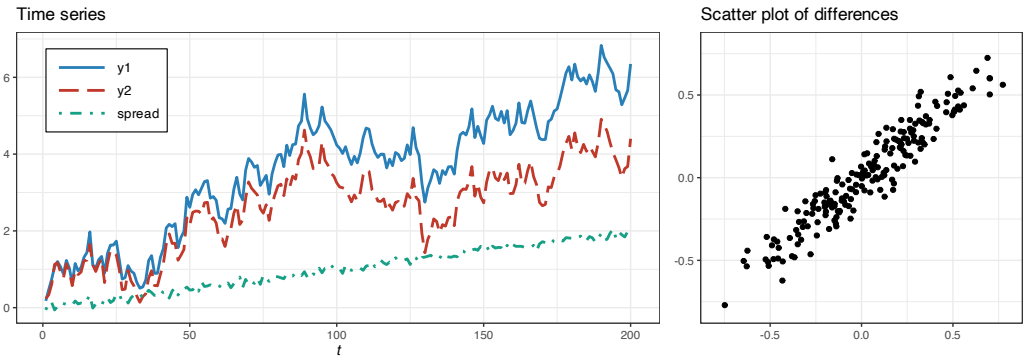


Figure 15.5 Example of non-cointegrated time series with high correlation.

from one period to the next, while ignoring the long-term trends. Cointegration, on the other hand, focuses on the long term, that is, whether the two time series have diverged or not after many periods, while being oblivious to short-term variations.

This short term vs. long term interpretation can be made more precise as follows. Define the difference of a time series y_t over k periods as $r_t(k) = y_t - y_{t-k}$. Our goal is to measure the similarity of two time series y_{1t} and y_{2t} over $t = 0, \ldots, T$:

- Correlation does it via the one-period differences $r_{1t}(1) = \Delta y_{1t}$ and $r_{2t}(1) = \Delta y_{2t}$ over $t = 1, \ldots, T$.
- Cointegration measures the difference between the two time series $y_{1t} - y_{2t}$ (assuming $\gamma = 1$ for simplicity). Equivalently, each time series can be shifted with its initial value and then they can be compared for divergence. Interestingly, these shifted time series are precisely the t-period differences $r_{1t}(t) = y_{1t} - y_{10}$ and $r_{2t}(t) = y_{2t} - y_{20}$ over $t = 1, \ldots, T$.

For pairs trading, it is the cointegration that matters, and not the correlation, because the focus is precisely on the long-term mean reversion property.

15.3 Pairs Trading

Trading an asset with mean reversion is quite simple: buy when it is below its mean value and unwind the position when it recovers to make a profit; similarly, short-sell it when it is above its mean value and unwind the position when it reverts back. Unfortunately, it is virtually impossible to find such a mean-reverting asset directly in a financial market. In fact, if there were such an asset, then many traders would notice and trade it, which would immediately eliminate its profitability.

In practice, one can attempt to discover a cointegrated pair of assets and then create a virtual mean-reverting asset (a spread). By virtue of the mean reversion property of the constructed spread, the common market trend present in the two original assets is nonexistent in the spread, which means that the spread does not follow the market trend and it is market neutral.

Pairs trading is a market-neutral strategy that trades a mean-reverting spread. That is, it

identifies two historically cointegrated financial instruments, such as stocks, and takes long and short positions in the two instruments when their prices deviate from their historical mean relationship, with the expectation that the prices will eventually revert back to the historical equilibrium, allowing the trader to profit from the convergence. Some monographic books on pairs trading include Vidyamurthy (2004), Ehrman (2006), and Chan (2013); see also Feng and Palomar (2016).

Pairs trading was developed in the mid-1980s by a quantitative trading team led by Nunzio Tartaglia at Morgan Stanley, achieving significant success. The team was disbanded in 1989 and the members joined various other trading firms. As a consequence, the initial secrecy of pairs trading was lost and the technique spread over the quant community.

Trading strategies can be broadly classified according to the underlying philosophy as follows:

- *Momentum-based strategies (or directional trading)*: These attempt to capture the market trend while treating the fluctuations as undesired noise (risk).
- *Pairs trading (or statistical arbitrage)*: These strategies are market neutral and try to trade the mean-reverting fluctuations of the relative mispricings between the two securities.

Figure 15.6 displays the breakdown of an asset's prices into the trend component (captured by momentum-based strategies) and the mean-reverting component (captured by pairs trading).

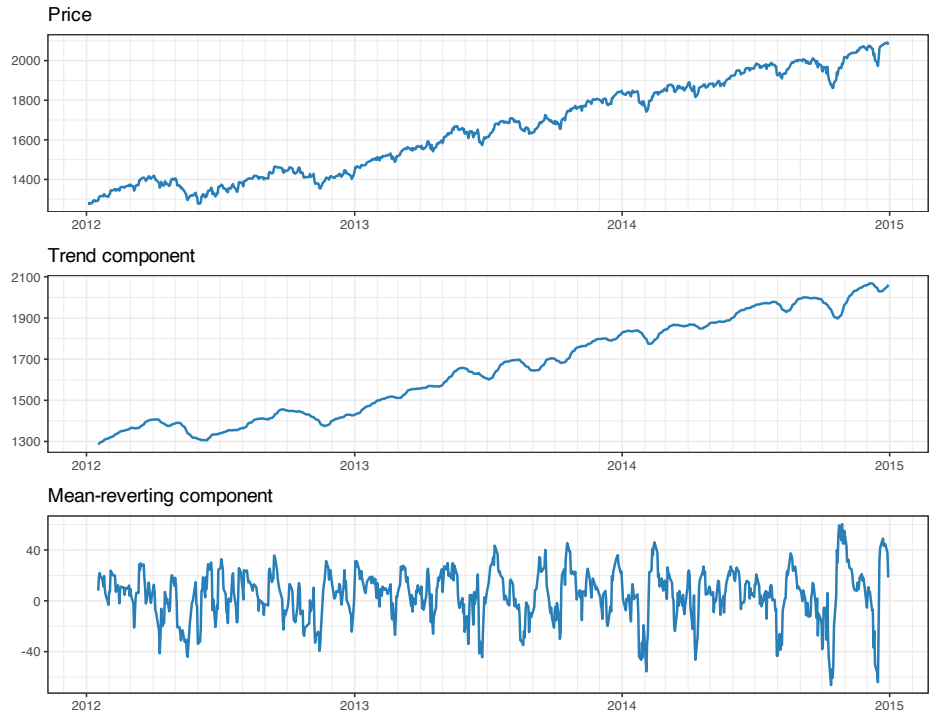


Figure 15.6 Decomposition of asset price into trend component and mean-reverting component.

15.3.1 Spread

The simplest implementation of pairs trading is based on comparing the spread of the two time series y_{1t} and y_{2t} to a threshold s_0. Suppose that the spread

$$z_t = y_{1t} - \gamma\, y_{2t}$$

is mean reverting with mean μ. Then, the idea is to either buy if the spread is low, $z_t < \mu - s_0$, or short-sell if it is high, $z_t > \mu + s_0$, and then unwind the position, for example, when it reverts back to the mean after k periods, leading to a difference of at least $|z_{t+k} - z_t| \geq s_0$. Figure 15.7 illustrates this process of pairs trading by buying and short-selling the spread according based on the threshold $s_0 = 1.5$.

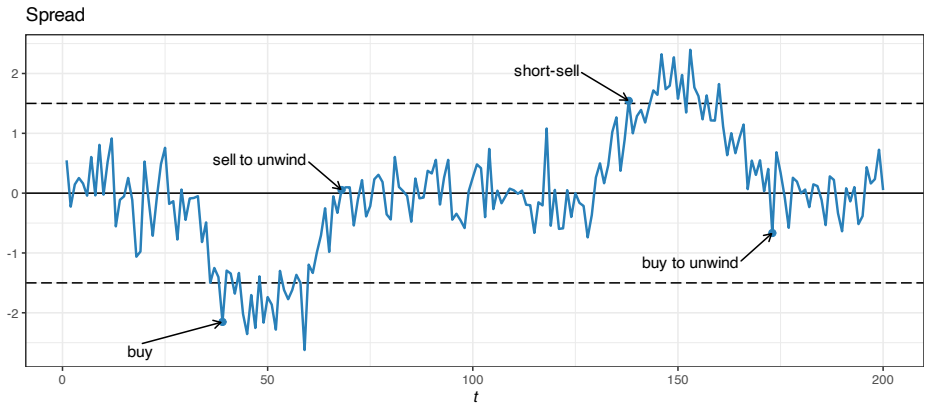


Figure 15.7 Illustration of pairs trading via thresholds on the spread.

15.3.2 Prices vs. Log-Prices

Pairs trading can be implemented in terms of prices or log-prices. This is determined by whether cointegration is exhibited by time series or prices or log-prices. The interpretation is slighly different as explained next.

Suppose first that y_{1t} and y_{2t} represent the prices of the two assets that define the mean-reverting spread $z_t = y_{1t} - \gamma\, y_{2t}$. In this case, the coefficients used in the spread (1 and γ) represent number of shares (to be bought and sold) and the spread has a meaning of price value. Thus, the spread difference corresponds to the profit made during the k periods (ignoring transaction costs):

$$z_{t+k} - z_t = s_0.$$

Suppose now that y_{1t} and y_{2t} represent the log-prices of two assets. In this case, it is convenient to use the portfolio notation (see Chapter 6). Basically, we can define the two-asset portfolio

$$\boldsymbol{w} = \begin{bmatrix} 1 \\ -\gamma \end{bmatrix},$$

where now the coefficients 1 and γ represent normalized dollar values instead of shares, and the spread can be compactly written as

$$z_t = \boldsymbol{w}^\mathsf{T} \boldsymbol{y}_t,$$

where $\boldsymbol{y}_t = \begin{bmatrix} y_{1t} \\ y_{2t} \end{bmatrix}$. With this notation, it becomes evident that the spread difference corresponds (approximately) to the return made during the k periods (ignoring transaction costs):

$$\boldsymbol{w}^\mathsf{T} (\boldsymbol{y}_{t+k} - \boldsymbol{y}_t) = z_{t+k} - z_t = s_0.$$

Note that this is an approximation because the return of the portfolio should be calculated using the linear returns, $\boldsymbol{w}^\mathsf{T} (\exp(\boldsymbol{y}_{t+k} - \boldsymbol{y}_t) - \mathbf{1})$, instead of the log-returns; nevertheless, these two quantities are approximately equal because $\exp(x) - 1 \approx x$ for small x (see Chapter 6 for details). Also, note that this portfolio has leverage $1 + \gamma$, so in practice we may want to normalize it to unit leverage. Moreover, when the hedge ratio is time-varying, as with the rolling least squares and Kalman filtering of Sections 15.5 and 15.6, the spread is itself a function of time and its difference also captures the change in the weights, $\boldsymbol{w}_t^\mathsf{T} \boldsymbol{y}_t - \boldsymbol{w}_{t-1}^\mathsf{T} \boldsymbol{y}_t$, which is not a trading profit; strictly, the P&L over a period should be computed with the weights held during it, that is, those set with the information available at the beginning of the period (the lagged weights $\boldsymbol{w}_{t-1}$ if $\boldsymbol{w}_t$ is computed with data up to time t). The numerical experiments with a time-varying hedge ratio compute the P&L in this way.

Summarizing, depending on whether the original cointegrated time series correspond to prices or log-prices, the threshold s_0 will determine the absolute profit or the return of the trade over these k periods. The choice is dictated by the nature of the cointegration of the time series. It is important to remark that in the case of cointegrated log-prices, the portfolio $\boldsymbol{w}$ has a meaning of normalized dollars, which may require a rebalancing over time (increased transaction costs); for cointegrated prices, the number of shares naturally stays constant over time and does not require rebalancing. This makes cointegrated price series more attractive; unfortunately, in practice, it is more difficult to find cointegrated price series since the noise in prices is less symmetric than that in log-prices, making the potential spreads less stationary.

15.3.3 Is Pairs Trading Profitable?

It is important to note that pairs trading relies on the assumption that the historical relationship between the two instruments will persist in the future. However, this is not always the case and cointegration between financial instruments can change over time due to various factors, such as market conditions, industry trends, or company-specific events. Therefore, pairs trading carries risks, and traders should carefully monitor the relationship between the instruments and use risk management techniques to protect their positions. Some publications show that pairs trading can provide profits (Avellaneda & Lee, 2010; Elliott et al., 2005; Gatev et al., 2006), while others indicate that cointegration relationships may not be preserved over time (Chan, 2013; Clegg, 2014).

The positive results should always be taken with a grain of salt for a number of reasons: backtests may ignore transaction costs, which can exceed the profit made trading the spread

(Chan, 2008); strategies may have yielded profits in the past while their effectiveness may have diminished in more recent times (Chan, 2013); and the practical implementation entails certain technical difficulties, such as the potential lack of liquidity for short-selling, the danger of margin calls (being forced to liquidate positions during inopportune times), the need for higher-frequency trading due to the competition of other traders, and the decimalization of U.S. stock prices, which caused bid–ask spreads to dramatically narrow, affecting the spreads (Chan, 2013). Kalman filtering (covered in Section 15.6) and VECM modeling (overviewed in Section 15.7) can help with some of the modeling shortcomings, but not with those related to costs and implementation.

15.3.4 Design of Pairs Trading

We have covered the basic idea of pairs trading, which relies on cointegrated pairs. The rest of this chapter revolves around the design of pairs trading in detail. In a nutshell, the objective is to trade profitably a mean-reverting spread, which requires: (i) discovering cointegrated pairs (Section 15.4), from simple prescreening to more sophisticated statistical tests, and (ii) designing the trading strategy (Section 15.5), which basically boils down to the choice of the threshold s_0 or other more sophisticated methods.

More advanced topics include the use of Kalman filtering for estimating a time-varying cointegration relationship (Section 15.6) and the extension of pairs trading to more than two assets (Section 15.7).

15.4 Discovering Cointegrated Pairs

The key in pairs trading lies in being able to discover cointegrated pairs. The available methods range from simple heuristics to sophisticated multivariate modeling (Krauss, 2017).

15.4.1 Prescreening

Prescreening is a simple and cheap process by which many pairs can be easily discarded while some potential pairs are selected for further analysis. A common heuristic proxy for cointegration is the normalized price distance (NPD) defined as (Gatev et al., 2006)

$$\text{NPD} \triangleq \sum_{t=1}^{T} (\tilde{p}_{1t} - \tilde{p}_{2t})^2 ,$$

where $\tilde{p}_{1t}$ and $\tilde{p}_{2t}$ are the normalized prices,

$$\tilde{p}_{1t} = p_{1t}/p_{10},$$
$$\tilde{p}_{2t} = p_{2t}/p_{20},$$

with p_{1t} and p_{2t} being the original prices.

A similar distance measure can be defined in terms of log-prices, y_{1t} and y_{2t}, by subtracting the initial value:

$$\tilde{y}_{1t} = y_{1t} - y_{10},$$
$$\tilde{y}_{2t} = y_{2t} - y_{20}.$$

Note that these shifted log-prices correspond to the long-term difference series, that is, the log-returns over long periods described earlier in Section 15.2 and denoted by $r_{1t}(t)$ and $r_{2t}(t)$.

15.4.2 Cointegration Tests

After the initial prescreening process of potential cointegrated pairs of assets, a more thorough analysis has to be performed. This is the job of the *cointegration tests* developed in the statistics literature for decades (Harris, 1995; Tsay, 2010, 2013). In a nutshell, these tests check whether or not a linear combination of the two time series follows a stationary autoregressive model and will be mean reverting. A time series with a unit root is nonstationary and behaves like a random walk. On the other hand, in the absence of unit roots, a time series tends to revert to its long-term mean. Thus, cointegration tests are typically implemented via unit-root stationarity tests.

Mathematically, we want to determine whether there exists a value of γ such that the spread

$$z_t = y_{1t} - \gamma\, y_{2t}$$

is stationary. Note that, in practice, the mean of the spread μ (equilibrium value) is not necessarily zero and γ does not have to be one. In fact, many studies artificially set $\gamma = 1$ to obtain dollar-neutral strategies (Elliott et al., 2005; Gatev et al., 2006; Triantafyllopoulos & Montana, 2011); however, that reduces the number of cointegrated pairs.

One of the simplest and most direct methods to test for cointegration is the Engle–Granger[3] test (Engle & Granger, 1987). It is based on two steps: first, the value of γ is obtained via least squares regression, and then the residual is tested for stationarity.[4] More exactly, the two sequences y_{1t} and y_{2t} are regressed against each other (see Chapter 3 for details on least squares regression),

$$y_{1t} - \gamma\, y_{2t} = \mu + r_t,$$

and the residual r_t is checked for unit-root stationarity or some form of mean reversion.

There are many heuristic ways to measure the strength of the mean reversion of the residual. For example, one can use the mean-crossing rate, that is, the number of times the residual crosses its mean value over a period of time (Vidyamurthy, 2004): the higher the mean crossing rate, the stronger the mean reversion. Another measure is the half-life of the mean reversion (Chan, 2013), which quantifies the time it takes for a time series to return to within half of the distance from the mean after deviating a certain amount from the mean.

More formally, we can use mathematically well-defined statistical tests. A variety of such tests have been proposed over time, with some of the most popular ones being (Banerjee et al., 1993; Harris, 1995; Pfaff, 2008; Tsay, 2010, 2013):

[3] The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 2003 was divided equally between Robert F. Engle III "for methods of analyzing economic time series with time-varying volatility (ARCH)" and Clive W. J. Granger "for methods of analyzing economic time series with common trends (cointegration)."

[4] The R packages `urca` and `egcm` implement a long list of stationarity and cointegration tests (Clegg, 2023; Pfaff et al., 2022).

- Dickey–Fuller (DF)
- augmented Dickey–Fuller (ADF)
- Phillips–Perron (PP)
- Pantula, Gonzales-Farias, and Fuller (PGFF)
- Elliott, Rothenberg, and Stock DF-GLS (ERSD)
- Johansen's trace test (JOT)
- Schmidt and Phillips rho (SPR)

For example, the simplest model for the residual is

$$r_t = \rho\, r_{t-1} + \epsilon_t,$$

where ϵ_t is the innovation term, and stationarity requires no unit root in the autoregressive term, that is, $|\rho| < 1$. The DF test (Dickey & Fuller, 1979) precisely formulates a hypothesis testing problem by defining the null hypothesis as a unit root being present ($\rho = 1$) and the alternative hypothesis as the series being stationary ($|\rho| < 1$). Under these two hypotheses, a small p-value[5] indicates strong stationarity (rejection of the null hypothesis). The model for the residual can be extended to incorporate a constant and a linear trend:

$$r_t = \phi_0 + c\, t + \rho\, r_{t-1} + \epsilon_t.$$

The popular ADF test includes further higher-order autoregressive terms in the model.

15.4.3 Cointegration of More Than Two Time Series

The Engle–Granger cointegration test has some drawbacks: it is designed for two time series (assets) and, even then, the first step in performing the regression of one time series vs. the other is sensitive to the ordering of the variables. The method can be naturally extended to more than two assets (described in Section 15.7), but then the ordering of the variables becomes more critical. An alternative method is Johansen's test (Johansen, 1991, 1995), which is based on a multivariate time series modeling, explored in Section 15.7 (see Chapter 4 for details on time series models).

Specifically, Johansen's test first fits a multivariate VECM time series model for N assets (see (15.6) in Section 15.7), which contains a key $N \times N$ matrix $\mathbf{\Pi}$ characterizing the cointegration. Then, it proceeds to analyze the rank of this matrix $\mathbf{\Pi}$, which precisely reveals the number of different cointegration relationships present.

15.4.4 Are Cointegrated Pairs Persistent?

It may seem that once a cointegrated pair has been discovered and has passed the necessary tests, the job is done and pairs trading will be profitable. Unfortunately, an additional issue to consider is whether this cointegration will be persistent over time or not.

[5] The p-value is the probability of obtaining the observed results under the assumption that the null hypothesis is correct. A small p-value means that there is strong evidence to reject the null hypothesis and accept the alternative hypothesis. Typical thresholds for determining whether a p-value is small enough are in the range 0.01–0.05.

In practice, it is not difficult to find cointegrated pairs during some chosen period of time of historical data, but they can just as easily lose cointegration in the subsequent out-of-sample period (Chan, 2013). The reason for this difficulty is that the fortunes of one company can change very quickly depending on management decisions, the competition, or simply bad news affecting one company and not the other.

In fact, empirical studies have shown evidence that does not support the hypothesis that cointegration is a persistent property (Clegg, 2014). The spread series of pairs are typically affected by a steady stream of permanent shocks that affect the cointegration. To bypass such practical problems, time-varying versions of cointegration can be considered (see Section 15.6 for the use of Kalman filtering) and even relaxed forms of cointegration can also be entertained, such as the concept of partial cointegration that allows the spread to contain a random walk component (Clegg & Krauss, 2018).

15.4.5 Numerical Experiments

We start with synthetic data and then consider some real examples based on stocks, commodities, and exchange-traded funds (ETFs).

Synthetic Data in Example 15.1

Recall Example 15.1, and the corresponding Figure 15.4, where a synthetic cointegrated time series was generated with low correlation. The estimated cointegration relationship via least squares based on $T = 200$ observations is

$$y_{2t} = 0.80\, y_{1t} + 0.20 + r_t,$$
$$r_t = 0.12\, r_{t-1} + \epsilon_t,$$

where the residual r_t has a small autoregressive coefficient of 0.12, indicating no unit root. This can be observed from the plot of the residual in Figure 15.8, with an estimated half-life of 0.33 (strong mean reversion). More quantitatively, Table 15.1 gives the p-values corresponding to several cointegration and residual unit-root tests. All the p-values are below a reasonable threshold of, say, 0.01 and therefore the null hypothesis (existence of a unit root) can be rejected, which means cointegration of the two time series is accepted.

Table 15.1 Cointegration and residual unit-root tests for Example 15.1.

Test	p-value
ADF	0.0081
PP	0.0001
PGFF	0.0001
ERSD	0.0008
JOT	0.0001
SPR	0.0001

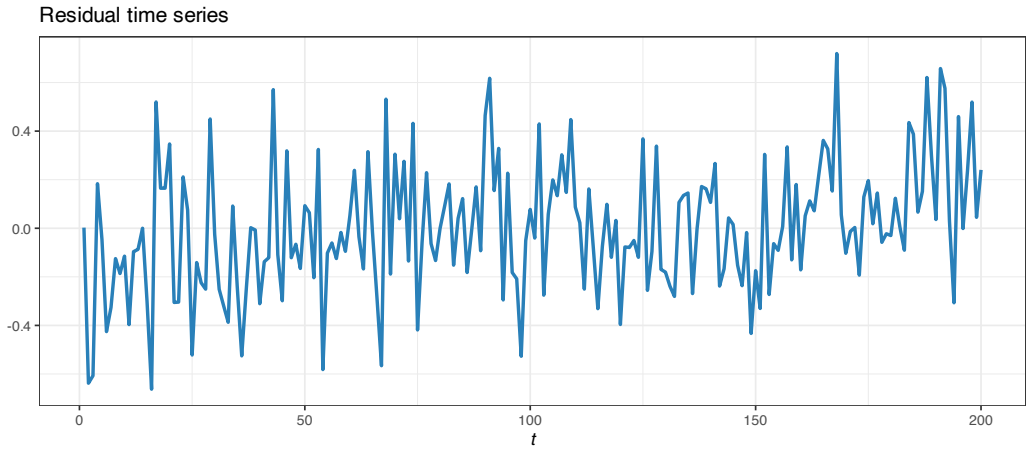


Figure 15.8 Cointegration residual for Example 15.1 with cointegration and low correlation.

Synthetic Data in Example 15.2

Consider now Example 15.2, and the corresponding Figure 15.5, where a synthetic non-cointegrated time series was generated with high correlation. The estimated cointegration relationship via least squares based on $T = 200$ observations is

$$y_{2t} = 0.68y_{1t} + 0.16 + r_t,$$
$$r_t = 0.91r_{t-1} + \epsilon_t,$$

where the residual r_t has a dangerous autoregressive coefficient of 0.91, which is close to 1, suggesting that the existence of a unit root cannot be excluded. This can be corroborated from the residual shown in Figure 15.9, with an estimated half-life of 7.29 (weak mean reversion). Additionally, Table 15.2 gives the p-values corresponding to several cointegration and residual unit-root tests. In this case, all the p-values are much higher than any reasonable threshold of, say, 0.01 and therefore the null hypothesis (existence of a unit root) cannot be rejected, which means we cannot conclude that the two time series are cointegrated.

Table 15.2 Cointegration and residual unit-root tests for Example 15.2.

Test	p-value
ADF	0.4529
PP	0.0608
PGFF	0.0700
ERSD	0.0767
JOT	0.0996
SPR	0.2671

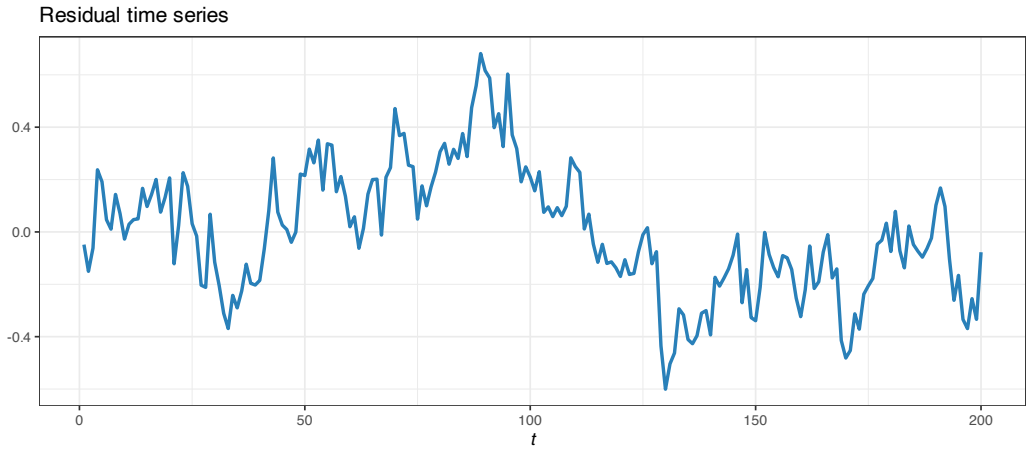


Figure 15.9 Cointegration residual for Example 15.2 with no cointegration and high correlation.

Market Data: EWA and EWC

EWA is an ETF that tracks the performance of the MSCI[6] Australia Index, which includes Australian companies from various sectors such as financials, materials, healthcare, consumer staples, and energy. Similarly, EWC is an ETF that tracks the performance of the MSCI Canada Index. Thus, the EWA and EWC provide exposure to the Australian and Canadian equity markets, respectively, and can be used by investors to gain broad exposure to these countries' economies.

EWA and EWC constitute a popular example in the quant community of cointegrated ETFs (Chan, 2013). The logic is that both the Canadian and Australian economies are commodity based, therefore their stock market performance is likely to be related through natural resources' prices.

The cointegration relationship during 2016–2019 is estimated via least squares. When EWA is regressed against EWC, the resulting hedge ratio is $\gamma = 0.74$; but when EWC is regressed against EWA, we obtain 1.27, which is not exactly the inverse, $1/0.74 \approx 1.35$. If instead we employ Johansen's test, we obtain the more accurate weights of 1 for EWA and -0.80 for EWC.

Figure 15.10 shows the residual of the cointegration relationship (spread), with an estimated half-life of 19 days (not very strong mean reversion). Table 15.3 shows the results for the cointegration tests, with the majority of the tests indicating cointegration at the 1% level (i.e., p-value less than 0.01), albeit two of the tests reject cointegration, so caution should be taken.

[6] Morgan Stanley Capital International (MSCI) is a leading provider of investment decision support tools and services. The company is best known for its global equity indices, which are widely used by investors to benchmark and analyze the performance of equity markets around the world.

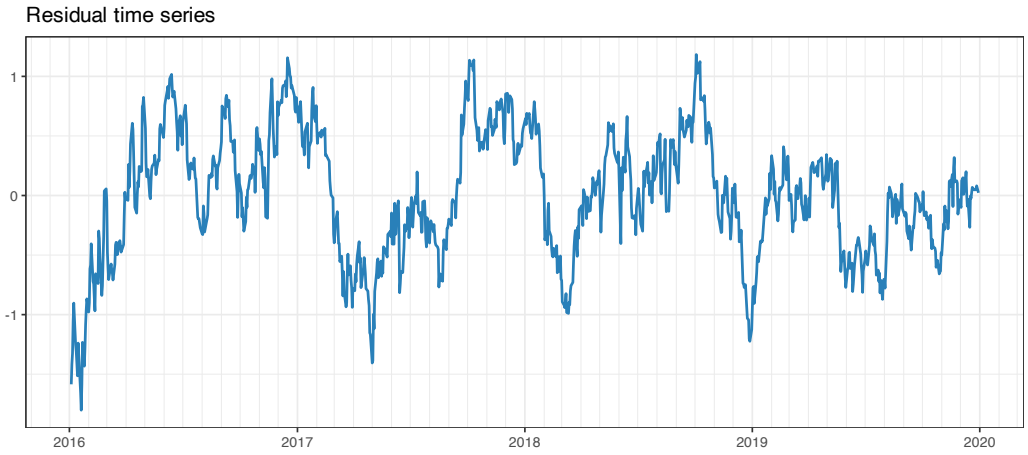


Figure 15.10 Cointegration residual for EWA–EWC.

Table 15.3 Cointegration and residual unit-root tests for EWA–EWC.

Test	p-value
ADF	0.0049
PP	0.0058
PGFF	0.0062
ERSD	0.5310
JOT	0.0069
SPR	0.3840

Market Data: Coca-Cola and Pepsi

The stocks Coca-Cola (with ticker KO) and Pepsi (with ticker PEP) are often mentioned as an example of a pair of securities in the same industry group for which pairs trading might be fruitful. However, as already pointed out in Chan (2008), they do not seem to be cointegrated.

We assess the cointegration relationship during 2017–2019 via least squares. Their returns show a correlation of 0.66, which is statistically significant, but different from cointegration. Figure 15.11 shows the residual of the cointegration relationship (spread), with an estimated half-life of 70 days (not indicative of any cointegration). Table 15.4 shows the results for the cointegration tests, all of which reject the hypothesis of cointegration (all p-values are much larger than 0.01).



Figure 15.11 Cointegration residual for KO–PEP.

Table 15.4 Cointegration and residual unit-root tests for KO–PEP.

Test	p-value
ADF	0.2675
PP	0.1845
PGFF	0.1395
ERSD	0.0484
JOT	0.5627
SPR	0.1982

Market Data: SPY, IVV, and VOO

Standard & Poor's 500 (S&P 500) is one of the world's best-known indices and one of the most commonly used benchmarks for the U.S. stock market. There are a multitude of ETFs that track this index, such as Standard & Poor's Depository Receipts SPY, iShares IVV, and Vanguard's VOO. Given that they all track the same underlying asset, it is likely that these three ETFs will have a strong cointegrating relationship.

In this case, since we want to assess cointegration among more than two time series, namely, SPY, IVV, and VOO, we cannot use the Enger–Granger test. Instead, we have to resort to Johansen's test, which first fits a VECM multivariate model and then proceeds to check sequentially the rank of matrix $\mathbf{\Pi} \in \mathbb{R}^{3\times 3}$, which satisfies $0 \leq r \leq 3$.

Based on the period 2017–2019, Johansen's test produces the following results:

- First, the null hypothesis is $r = 0$ vs. the alternative hypothesis $r > 0$: there is clear evidence to reject the null hypothesis.
- Then, the null hypothesis is $r \leq 1$ vs. the alternative hypothesis $r > 1$: again we have sufficient evidence to reject the null hypothesis.
- Finally, the null hypothesis is $r \leq 2$ vs. the alternative hypothesis $r > 2$: in this case we cannot reject the null hypothesis.

Thus, the conclusion is that the rank is $r = 2$, that is, we can find two different cointegrating relationships, whose residuals are shown in Figure 15.12.

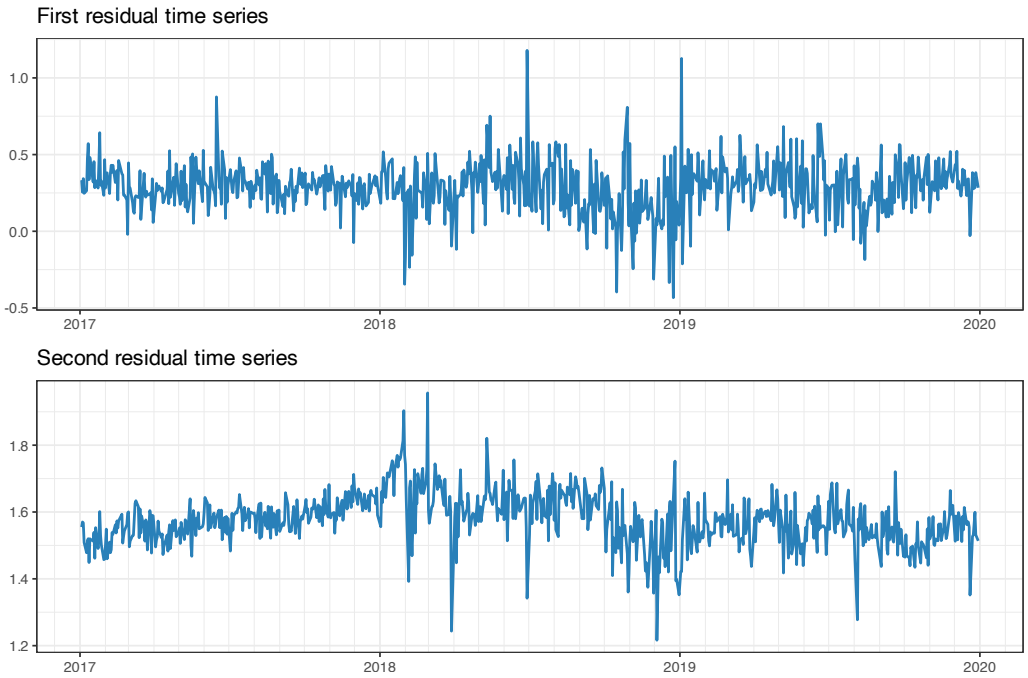


Figure 15.12 Cointegration residuals for SPY–IVV–VOO.

15.5 Trading the Spread

Suppose we have discovered a cointegrated pair of log-price time series, y_{1t} and y_{2t}, and have formed the spread $y_{1t} - \gamma\, y_{2t}$, which is effectively using the two-asset portfolio $\boldsymbol{w} = [1, -\gamma]^\mathsf{T}$ with a leverage of $\|\boldsymbol{w}\|_1 = 1 + \gamma$. In order to make fair comparisons, it is necessary to normalize the leverage to 1. Thus, the portfolio with normalized leverage is

$$\boldsymbol{w} = \frac{1}{1+\gamma} \begin{bmatrix} 1 \\ -\gamma \end{bmatrix}, \tag{15.2}$$

with corresponding normalized spread $z_t = \boldsymbol{w}^\mathsf{T} \boldsymbol{y}_t$. The return of this portfolio at time t (ignoring transaction costs) is given by $\boldsymbol{w}^\mathsf{T}\,(\boldsymbol{y}_t - \boldsymbol{y}_{t-1}) = z_t - z_{t-1}$ (see Chapter 6 for details on portfolio notation). Suppose we enter a position at time t and after k periods the spread reverts to the mean and the position is closed. This would lead to a difference of at least $|z_{t+k} - z_t| \geq s_0$, which is the portfolio return during these k periods.

Trading the spread boils down to deciding when to buy or short-sell the spread, and how much to invest (termed sizing). This is conveniently done by defining a "signal" time series $s_1, s_2, s_3, \ldots$, where s_t denotes the sizing (positive for buying, zero for no position, and negative for short-selling) usually bounded as $-1 \leq s_t \leq 1$ to control the leverage. We will assume that the value of the signal at time t, s_t, has been decided based on information up to

(and including) time t, that is, $\ldots, \boldsymbol{y}_{t-2}, \boldsymbol{y}_{t-1}, \boldsymbol{y}_t$. Thus, the combination of the spread portfolio (15.2) and the signal s_t produces the time-varying portfolio $s_t \times \boldsymbol{w}$, with corresponding return

$$R_t^{\text{portf}} = s_{t-1} \times \boldsymbol{w}^{\mathsf{T}} (\boldsymbol{y}_t - \boldsymbol{y}_{t-1}) = s_{t-1} \times (z_t - z_{t-1}).$$

15.5.1 Trading Strategies

To define a strategy to trade the spread, it suffices to determine a rule for the sizing signal s_t. For this purpose, it is convenient to use a normalized version of the spread, called the standard score or z-score:

$$z_t^{\text{score}} = \frac{z_t - \mathbb{E}[z_t]}{\sqrt{\text{Var}(z_t)}},$$

which has zero mean and unit variance. This z-score cannot be used in a real implementation since it is not causal and suffers from look-ahead bias (when estimating the mean and variance). A naive approach would be to use some training data to determine the mean and standard deviation, and then apply that to the future out-of-sample data. A more sophisticated way is to make the calculation adaptive by implementing it in a rolling fashion, for example via the so-called Bollinger Bands.

Bollinger Bands are a technical trading tool created by John Bollinger in the early 1980s. They arose from the need for adaptive trading bands and the observation that volatility was dynamic. They are computed on a rolling-window basis over some lookback window. In particular, the rolling mean and rolling standard deviation are first computed, from which the upper and lower bands are easily obtained (typically the mean plus/minus one or two standard deviations). In the context of the z-score, the spread can be adaptively normalized with the rolling mean and rolling standard deviation.

We now describe two of the simplest possible strategies for trading a spread, namely, the linear strategy and the thresholded strategy.

- The *linear strategy* is very simple to describe based on the contrarian idea of buying low and selling high (Chan, 2013). As a first attempt, we could define the sizing signal simply as the negative z-score, $s_t = -z_t^{\text{score}}$, to gradually scale-in and scale-out or, even better, including a scaling factor as $s_t = -z_t^{\text{score}}/s_0$, where s_0 denotes the threshold at which the signal is fully leveraged. In practice, to limit the leverage to 1, we can project this value to lie in the interval $[-1, 1]$:

$$s_t = -\left[\frac{z_t^{\text{score}}}{s_0}\right]_{-1}^{+1},$$

  where $[\cdot]_a^b$ clips the argument to a if below that value and to b if above that value.

- The *thresholded strategy* follows similarly the contrarian nature of buying low and selling high, but rather than linear it takes an all-in or all-out sizing based on thresholds (Vidyamurthy, 2004). The simplest implementation is based on comparing the z-score to a threshold s_0: buy when the z-score is below $-s_0$ and short-sell when it is above s_0, while unwinding the position after reverting to the equilibrium value of zero. In terms of the

sizing signal:

$$s_t = \begin{cases} +1 & \text{if } z_t^{\text{score}} < -s_0, \\ 0 & \text{after } z_t^{\text{score}} \text{ reverts to 0}, \\ -1 & \text{if } z_t^{\text{score}} > +s_0. \end{cases}$$

Figures 15.13 and 15.14 illustrate the linear and thresholded strategies, respectively, based on a synthetic spread generated as an AR(1) with an autoregressive coefficient of 0.7. Observe the different nature of the sizing signal: continuous vs. on–off. The thresholds have been chosen arbitrarily as $s_0 = 1$ and should be properly optimized for maximizing the profit (as described in detail in the next section). In practice, the rolling version of the z-score should be used to make it implementable without look-ahead bias, such as based on the Bollinger Bands (Chan, 2013).

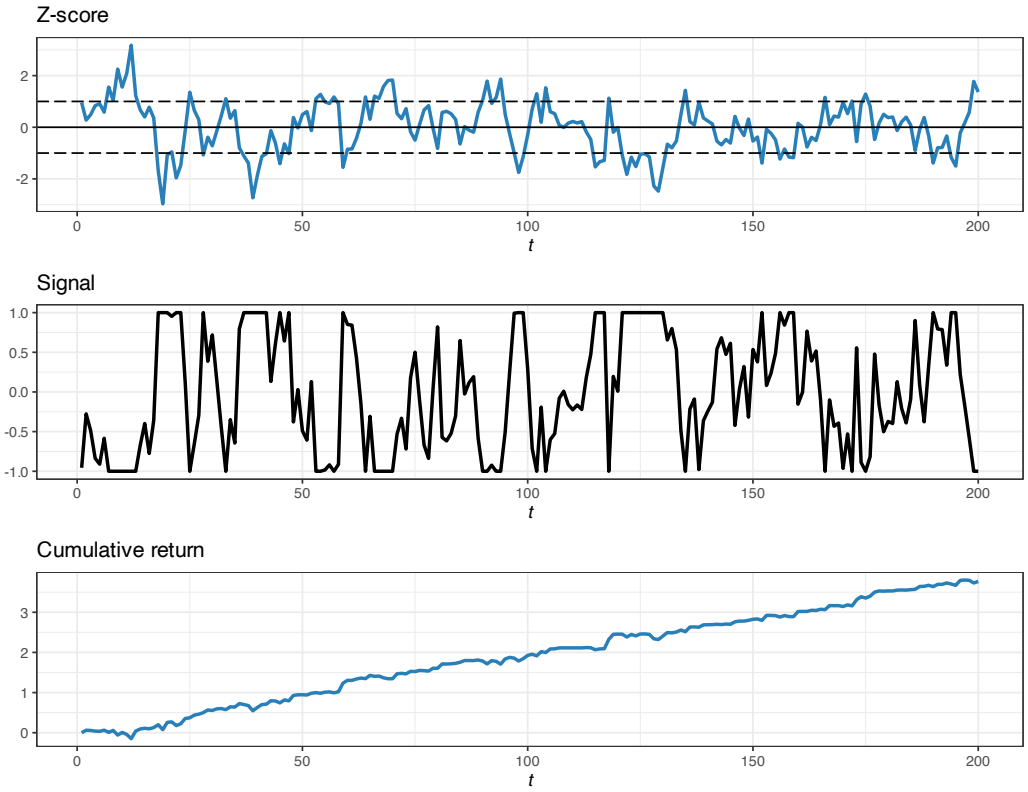


Figure 15.13 Illustration of pairs trading via the linear strategy on the spread.

15.5.2 Optimizing the Threshold

Consider now the simple thresholded strategy that buys when the z-score is below the threshold $-s_0$ and short-sells when it is above s_0, unwinding the position after reverting to the equilibrium value of zero. Note that in terms of the spread, the threshold is $s_0 \times \sigma$, where σ is the standard deviation of the spread.

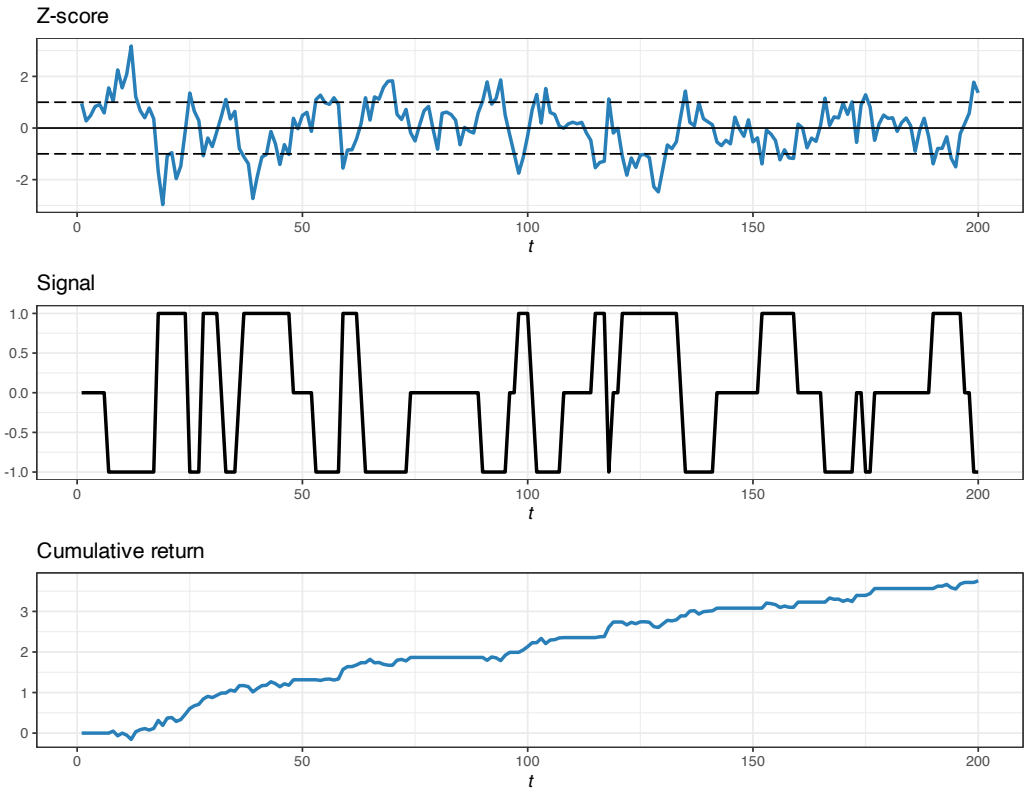


Figure 15.14 Illustration of pairs trading via the thresholded strategy on the spread.

The choice of this threshold is critical as it determines how often the position is closed (cashing a profit) as well as how large that minimum profit is. The total profit equals the number of trades times the profit of each trade. Recall that, when a position is closed after k periods, the meaning of the spread difference $z_{t+k} - z_t$ depends on whether the spread represents log-prices or prices:

- for log-prices, the spread difference denotes the log-return of the profit;
- for prices, the spread difference denotes the absolute profit (to be scaled with the initial budget).

Thus, after N^{trades} successful trades have been executed, the total (uncompounded) profit can be accounted as $N^{\text{trades}} \times \sigma\, s_0$ (the compounded profit could also be used).

We will now obtain the optimum choice of the threshold to maximize the total profit in both parametric and nonparametric approaches.

Parametric Approach

Suppose the z-score follows a standard normal distribution, $z_t^{\text{score}} \sim \mathcal{N}(0, 1)$. Then, the probability that it deviates from zero by s_0 or more is $1 - \Phi(s_0)$, where $\Phi(\cdot)$ is the cumulative distribution function (cdf) of the standard normal distribution. For a time series path of T

periods, the number of tradable events (in one direction) can be approximated by $T \times (1 - \Phi(s_0))$ with a total profit of $T\ (1 - \Phi(s_0)) \times \sigma\ s_0$.

Thus, under this simple parametric model, the optimal threshold can be obtained simply as

$$s_0^\star = \arg\max_{s_0}\ (1 - \Phi(s_0)) \times s_0.$$

Figure 15.15 illustrates the parametric evaluation of the profit vs. the threshold for a synthetic Gaussian spread.

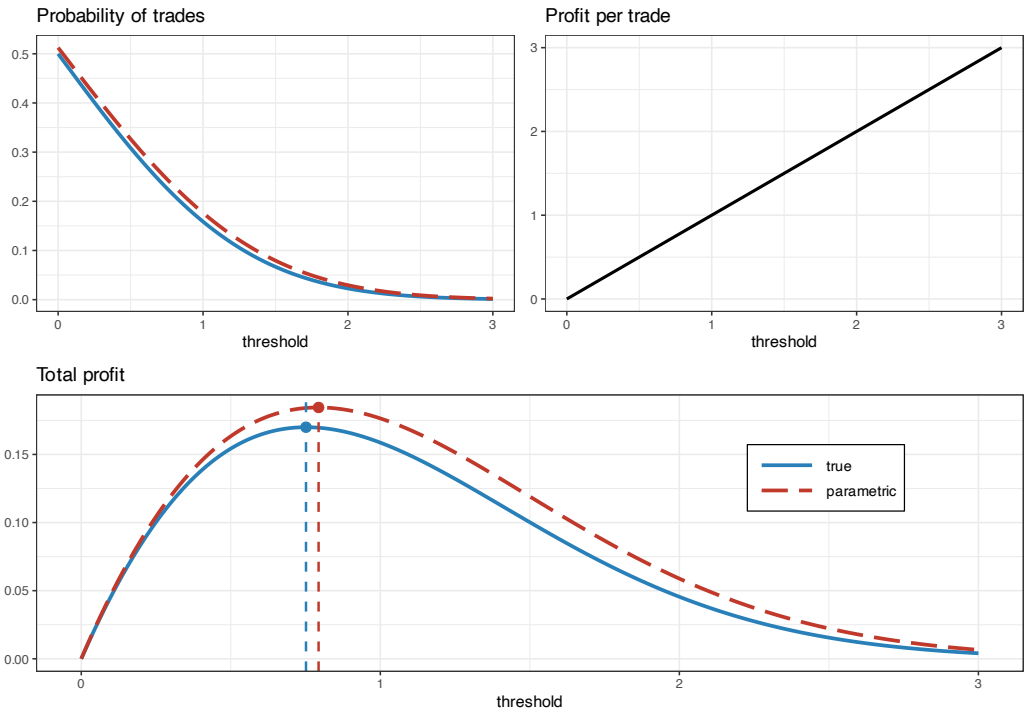


Figure 15.15 Calculation of optimum threshold in pairs trading via a parametric approach.

Nonparametric Data-Driven Approach

An alternative to the parametric approach (based on a probably inaccurate model) is a data-driven approach that does not rely on any model assumption. The idea is to simply use the available data to empirically count the number of tradable events for each possible threshold. Given T observations of the z-score, z_t^{score} for $t = 1, \ldots, T$, and J discretized threshold values, $s_{01}, \ldots, s_{0J}$, we can compute the empirical trading frequency for each threshold s_{0j} (in one direction) as

$$\bar{f}_j = \frac{1}{T} \sum_{t=1}^{T} 1\{z_t^{\text{score}} > s_{0j}\},$$

where $1\{\cdot\}$ is the indicator function that equals 1 if the argument is true and zero otherwise.

Unfortunately, the empirical values $\bar{f}_j$ can be very noisy, which can affect the assessment of the total profit. One way to reduce the noise in the values $\bar{\boldsymbol{f}} = (\bar{f}_1, \ldots, \bar{f}_J)$ is by taking advantage of the fact that the trading frequency should be a smooth function of the threshold. We can obtain a smoothed version by solving the least squares problem

$$\underset{\boldsymbol{f}}{\text{minimize}} \quad \sum_{j=1}^{J} (f_j - \bar{f}_j)^2 + \lambda \sum_{j=1}^{J-1} (f_j - f_{j+1})^2,$$

where the first term measures the difference between the noisy and smoothed values, and the second term enforces smoothness controlled by the hyper-parameter λ. This problem can be rewritten with a compact notation as

$$\underset{\boldsymbol{f}}{\text{minimize}} \quad \|\boldsymbol{f} - \bar{\boldsymbol{f}}\|_2^2 + \lambda \|\boldsymbol{D}\boldsymbol{f}\|_2^2,$$

where $\boldsymbol{D}$ is a difference matrix defined as

$$\boldsymbol{D} = \begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{(J-1)\times J}.$$

Since this problem is a least squares, we can write down the solution in closed form as

$$\boldsymbol{f}^\star = \left(\boldsymbol{I} + \lambda \boldsymbol{D}^\top \boldsymbol{D}\right)^{-1} \bar{\boldsymbol{f}}.$$

Finally, the optimal threshold can be obtained by maximizing the smoothed total profit:

$$s_0^\star = \underset{s_{0j} \in \{s_{01}, s_{02}, \ldots, s_{0J}\}}{\arg\max} \; s_{0j} \times f_j.$$

Figure 15.16 illustrates the nonparametric evaluation of the profit vs. the threshold for a synthetic Gaussian spread (both the original noisy version and the improved smoothed version).

15.5.3 Numerical Experiments

We now execute pairs trading with market data examples during 2013–2022. The first two years are used to estimate the hedge factor γ, which is critical to form the spread. Then the spread is traded over the remaining out-of-sample period. The z-score is computed on a rolling-window basis (as in Bollinger Bands) to make sure it adapts to the market changes over time and stays mean reverting. To trade the spread, the thresholded strategy is employed (the linear strategy can also be used with very similar results). For simplicity, the threshold is simply chosen as $s_0 = 1$; of course it could be optimized but care has to be taken to avoid look-ahead bias and overfitting (see Chapter 8 for the dangers of backtesting and overfitting of hyper-parameters).

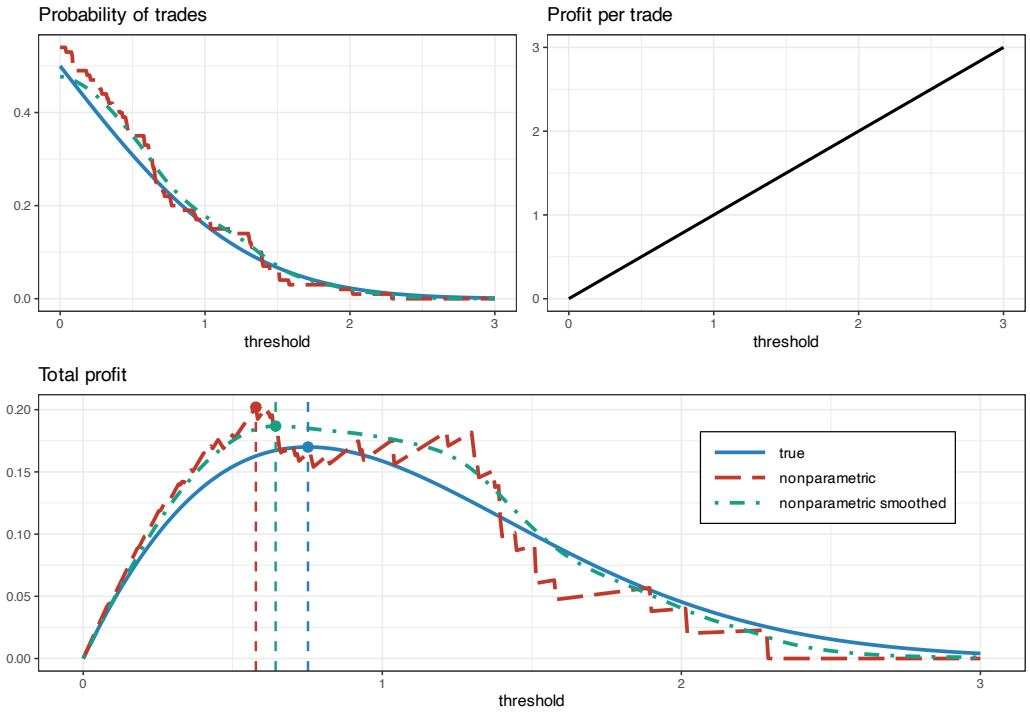


Figure 15.16 Calculation of optimum threshold in pairs trading via a nonparametric approach.

Market Data: EWA and EWC

We start with the two ETFs EWA and EWC, which track the performance of the Australian and Canadian economies, respectively. As previously checked in Section 15.4, the majority of the tests indicate that cointegration is present during most of the period, although in occasions it may be lost. The z-score is computed on a rolling-window basis with a lookback period of six months.

Figure 15.17 shows the spread (with hedge ratio γ estimated via least squares during the first two years), the z-score, the trading signal, and the cumulative return. We can observe that the spread does not show a strong persistent cointegration relationship over the whole period; in practice, the hedge ratio should be adapted over time. The z-score is able to produce a more constant mean-reverting version due to the rolling window adaptation. In any case, any practical implementation of pairs trading would recompute the hedge ratio over time, as assumed in the rest of the experiments via a rolling least squares.

Figure 15.18 shows the results when the hedge ratio γ is computed on a rolling-window basis with a lookback period of two years. We can appreciate that the spread has been improved compared to the previous fixed least squares and looks more mean-reverting. Still, the z-score is able to further improve it and produce a much better mean reverting version. The cumulative return shows a modest improvement thanks to the rolling least squares approach (a final value of about 0.51 compared with 0.48 for the fixed least squares, and a smaller drawdown); the

Kalman filter, explored in Section 15.6, is an alternative way to track the hedge ratio over time.

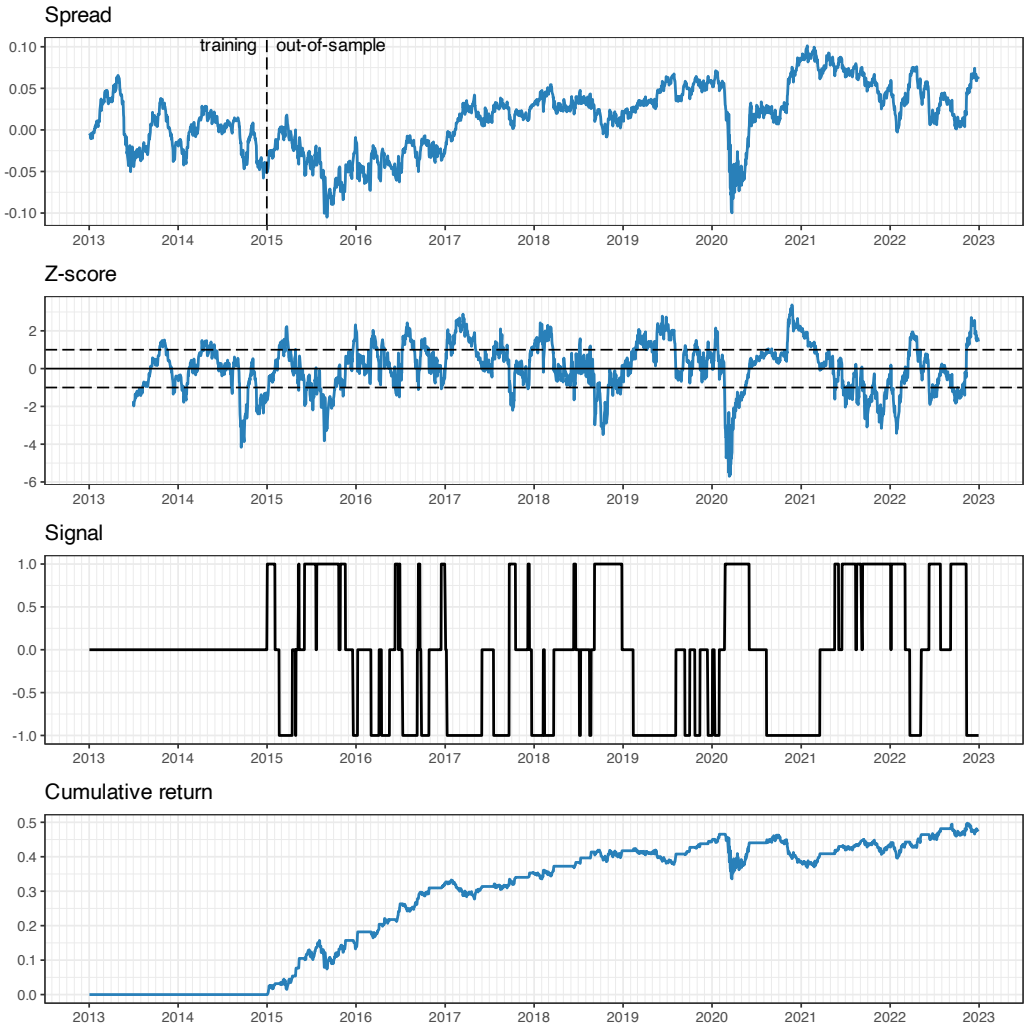


Figure 15.17 Pairs trading on EWA–EWC with six-month rolling z-score and two-year fixed least squares.

Market Data: KO and PEP

We continue with an attempt to execute pairs trading on the stocks Coca-Cola (KO) and Pepsi (PEP), which do not seem to be cointegrated according to the previous experiments in Section 15.4. To be realistic, the hedge ratio γ is estimated via a rolling least squares with a lookback period of two years (this may help in adapting to the changing cointegration relationship). The z-score is computed on a rolling-window basis with a lookback period of six months (as well as one month for a faster adaptation).

Figure 15.19 shows the spread, the z-score, the trading signal, and the cumulative return. We

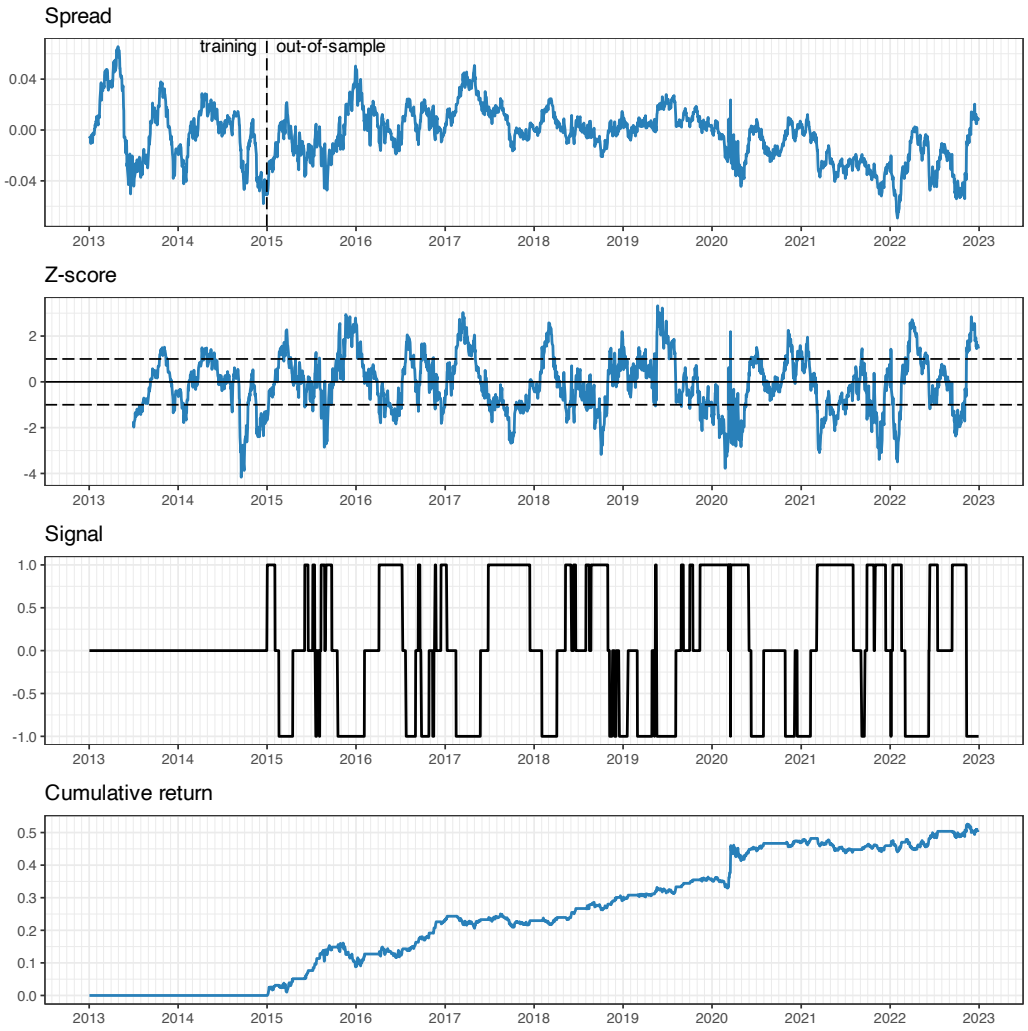


Figure 15.18 Pairs trading on EWA–EWC with six-month rolling z-score and two-year rolling least squares.

can observe that the spread loses the mean reversion property despite the rolling nature of the hedge ratio calculation. The z-score is able to produce a more constant mean-reverting version, but still one can observe jumps indicating loss of cointegration. The cumulative return indicates that trading is not very successful.

Figure 15.20 shows the results when the z-score is calculated with a faster adaptability using a lookback period of one month (as opposed to the previous six months). The z-score looks much better, with a strong mean reversion. This translates into higher-frequency trading (compare the frequency in the signals), but still the cumulative return does not seem to indicate good profitability.

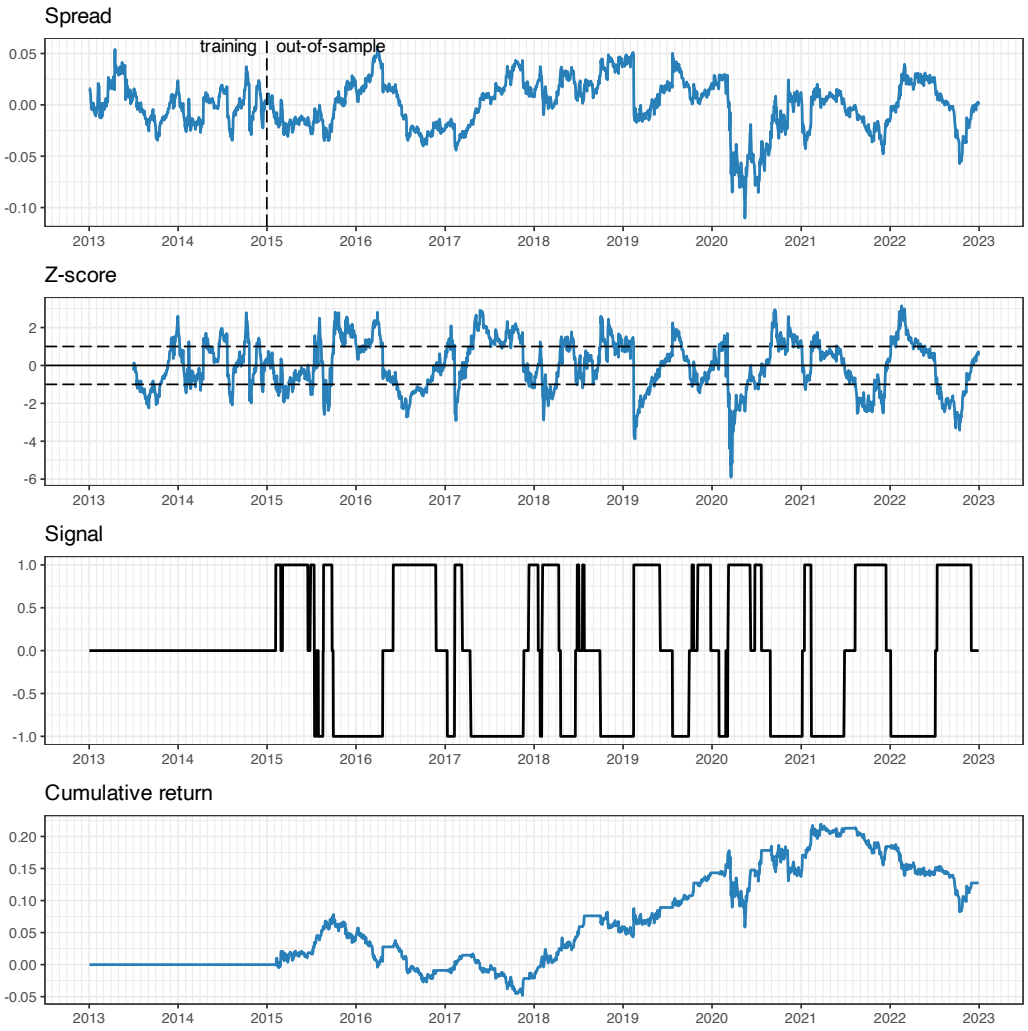


Figure 15.19 Pairs trading on KO–PEP with six-month rolling z-score and two-year rolling least squares.

15.6 Kalman Filtering for Pairs Trading

A key component in pairs trading is the construction of a mean-reverting spread

$$z_t = y_{1t} - \gamma\, y_{2t} = \mu + r_t,$$

where γ is the hedge ratio, μ is the mean, and r_t is the zero-mean residual. Then the trading strategy will properly size the trade depending on the distance of the spread z_t from the equilibrium value μ.

Construction of the spread requires careful estimation of the hedge ratio γ, as well as the mean of the spread μ. The traditional way is to employ least squares regression. In practice, the hedge ratio and the mean will slowly change over time and then the least squares solution

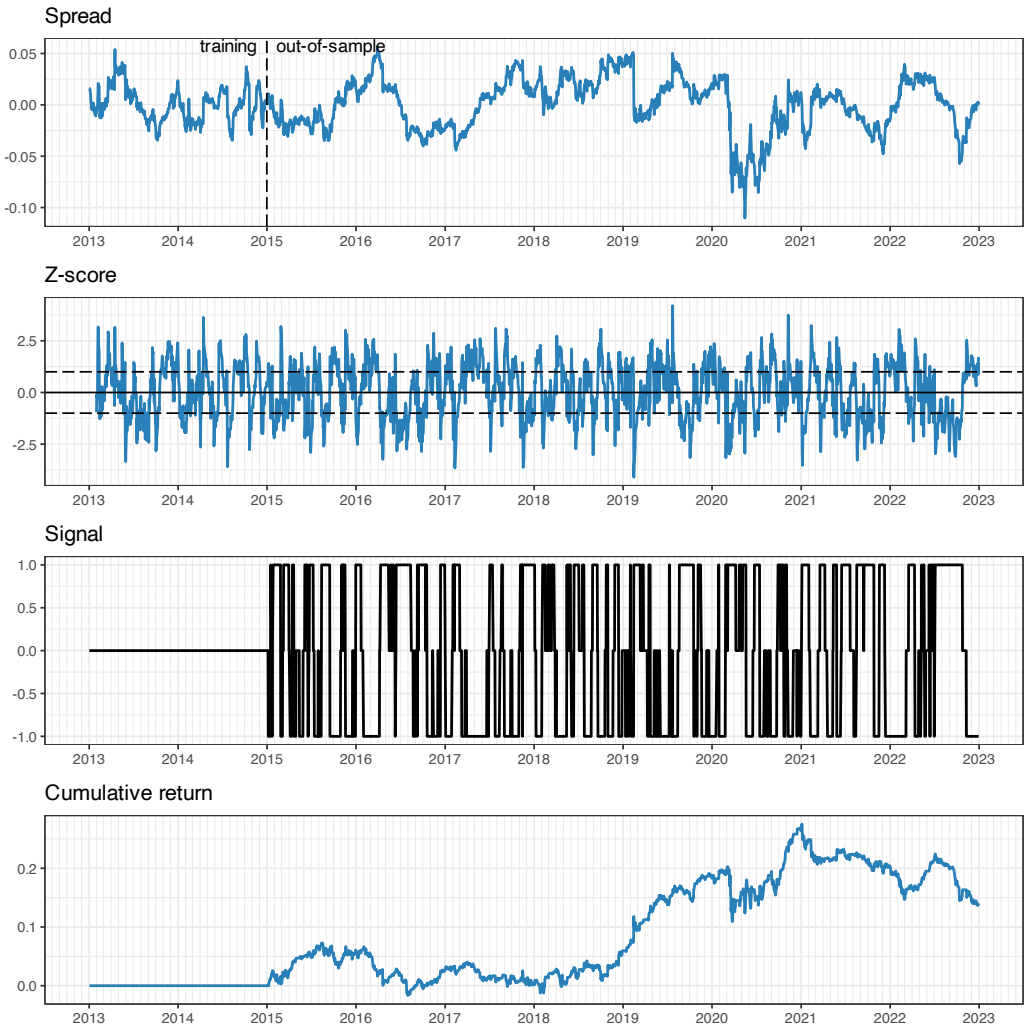


Figure 15.20 Pairs trading on KO–PEP with one-month rolling z-score and two-year rolling least squares.

should be recomputed on a rolling-window basis. A more flexible alternative is to employ a time-varying modeling and estimation technique based on state-space modeling and the Kalman filter (Chan, 2013; Feng & Palomar, 2016).

15.6.1 Spread Modeling via Least Squares

The method of least squares (LS) dates back to 1795 when Gauss used it to study planetary motions. It deals with the linear model $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\epsilon}$ by solving the problem (Kay, 1993; Scharf, 1991)

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2,$$

whose solution gives the least squares estimate $\hat{\boldsymbol{x}} = \left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^\mathsf{T}\boldsymbol{y}$. In addition, the covariance matrix of the estimate $\hat{\boldsymbol{x}}$ is given by $\sigma_\epsilon^2\left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\right)^{-1}$, where σ_ϵ^2 is the variance of the residual $\boldsymbol{\epsilon}$ (in practice, the variance of the estimated residual $\hat{\boldsymbol{\epsilon}} = \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}$ can be used instead).

In our context of spread modeling, we want to fit $y_{1t} \approx \mu + \gamma\, y_{2t}$ based on T observations, so the LS formulation becomes

$$\underset{\mu,\gamma}{\text{minimize}} \quad \|\boldsymbol{y}_1 - (\mu\mathbf{1} + \gamma\,\boldsymbol{y}_2)\|_2^2,$$

where the vectors $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ contain the T observations of the two time series, and $\mathbf{1}$ is the all-ones vector. The solution gives the estimates

$$\begin{bmatrix} \hat{\mu} \\ \hat{\gamma} \end{bmatrix} = \begin{bmatrix} \mathbf{1}^\mathsf{T}\mathbf{1} & \mathbf{1}^\mathsf{T}\boldsymbol{y}_2 \\ \boldsymbol{y}_2^\mathsf{T}\mathbf{1} & \boldsymbol{y}_2^\mathsf{T}\boldsymbol{y}_2 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{1}^\mathsf{T}\boldsymbol{y}_1 \\ \boldsymbol{y}_2^\mathsf{T}\boldsymbol{y}_1 \end{bmatrix}.$$

In practice, it is more convenient to first remove the mean of $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ to produce the centered versions $\bar{\boldsymbol{y}}_1$ and $\bar{\boldsymbol{y}}_2$, then estimate the hedge ratio as

$$\hat{\gamma} = \frac{\bar{\boldsymbol{y}}_2^\mathsf{T}\bar{\boldsymbol{y}}_1}{\bar{\boldsymbol{y}}_2^\mathsf{T}\bar{\boldsymbol{y}}_2},$$

and finally compute the sample mean of $\boldsymbol{y}_1 - \hat{\gamma}\,\boldsymbol{y}_2$:

$$\hat{\mu} = \frac{\mathbf{1}^\mathsf{T}(\boldsymbol{y}_1 - \hat{\gamma}\,\boldsymbol{y}_2)}{T}.$$

The variance of these estimates is given by

$$\begin{aligned} \text{Var}[\hat{\gamma}] &= \tfrac{1}{T}\,\sigma_\epsilon^2/\sigma_2^2, \\ \text{Var}[\hat{\mu}] &= \tfrac{1}{T}\,\sigma_\epsilon^2, \end{aligned}$$

where σ_2^2 is the variance of $\boldsymbol{y}_2$ and σ_ϵ^2 is the variance of the residual $\boldsymbol{\epsilon}$.

It is important to reiterate that in practice the hedge ratio and the mean will slowly change over time, as denoted by γ_t and μ_t, and the least squares solution must be recomputed on a rolling-window basis (based on a lookback window of past samples). Nevertheless, this time-varying case can also be handled with Kalman filtering, as described next.

15.6.2 Primer on the Kalman Filter

State-space modeling provides a unified framework for treating a wide range of problems in time series analysis. It can be thought of as a universal and flexible modeling approach with a very efficient algorithm, the *Kalman filter*. The basic idea is to assume that the evolution of the system over time is driven by a series of unobserved or hidden values, which can only be measured indirectly through observations of the system output. This model can be used for filtering, smoothing, and forecasting. Here we provide a concise summary; more details on state-space modeling and Kalman filtering can be found in Section 4.2 of Chapter 4.

The Kalman filter, which was employed by NASA during the 1960s in the Apollo program, now boasts a vast array of technological applications. It is commonly utilized in the guidance,

navigation, and control of vehicles, including aircraft, spacecraft, and maritime vessels. It has also found applications in time series analysis, signal processing, and econometrics. More recently, it has become a key component in robotic motion planning and control, as well as trajectory optimization.

State-space models and Kalman filtering are mature topics with excellent textbooks available such as the classical references Anderson and Moore (1979) and Durbin and Koopman (2012), which was originally published in 2001. Other textbook references on time series and the Kalman filter include Brockwell and Davis (2002), Shumway and Stoffer (2017), Harvey (1989) and, in particular, for financial time series, Zivot et al. (2004), Tsay (2010), Lütkepohl (2007), and Harvey and Koopman (2009).

Mathematically, the Kalman filter is based on the following linear Gaussian state-space model with discrete time $t = 1, \ldots, T$ (Durbin & Koopman, 2012):

$$\begin{aligned} \boldsymbol{y}_t &= \boldsymbol{Z}_t \boldsymbol{\alpha}_t + \boldsymbol{\epsilon}_t \quad \text{(observation equation)}, \\ \boldsymbol{\alpha}_{t+1} &= \boldsymbol{T}_t \boldsymbol{\alpha}_t + \boldsymbol{\eta}_t \quad \text{(state equation)}, \end{aligned}$$

where $\boldsymbol{y}_t$ denotes the observations over time with observation matrix $\boldsymbol{Z}_t$, $\boldsymbol{\alpha}_t$ represents the unobserved or hidden internal state with state transition matrix $\boldsymbol{T}_t$, and the two noise terms $\boldsymbol{\epsilon}_t$ and $\boldsymbol{\eta}_t$ are Gaussian distributed with zero mean and covariance matrices $\boldsymbol{H}$ and $\boldsymbol{Q}$, respectively, that is, $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{H})$ and $\boldsymbol{\eta}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$. The initial state can be modeled as $\boldsymbol{\alpha}_1 \sim \mathcal{N}(\boldsymbol{a}_1, \boldsymbol{P}_1)$. Mature and efficient software implementations are readily available (Helske, 2017; Holmes et al., 2012; Petris & Petrone, 2011; Tusell, 2011).[7]

The parameters of the state-space model (i.e., $\boldsymbol{Z}_t$, $\boldsymbol{T}_t$, $\boldsymbol{H}$, $\boldsymbol{Q}$, $\boldsymbol{a}_1$, and $\boldsymbol{P}_1$) can be either provided by the user (if known) or inferred from the data with algorithms based on the maximum likelihood method. Again, mature and efficient software implementations are available for this parameter fitting (Holmes et al., 2012).[8]

To be more precise, the Kalman filter is a very efficient algorithm to optimally characterize the distribution of the hidden state at time t, $\boldsymbol{\alpha}_t$, in a causal manner. In particular, $\boldsymbol{\alpha}_{t|t-1}$ and $\boldsymbol{\alpha}_{t|t}$ denote the expected value given the observations up to time $t-1$ and t, respectively. These quantities can be efficiently computed using a "forward pass" algorithm that goes from $t = 1$ to $t = T$ in a recursive way, so that it can operate in real time (Durbin & Koopman, 2012).

15.6.3 Spread Modeling via Kalman

In our context of spread modeling, we want to model $y_{1t} \approx \mu_t + \gamma_t\, y_{2t}$, where now μ_t and γ_t change slowly over time. This can be done conveniently via a state-space modeling by

[7] The Kalman filter is implemented in the R package `KFAS` (Helske, 2017) and the Python package `filterpy`.
[8] The R package `MARSS` implements algorithms for fitting state-space models to time series data (Holmes et al., 2012).

identifying the hidden state as $\boldsymbol{\alpha}_t = (\mu_t, \gamma_t)$, leading to

$$\begin{aligned} y_{1t} &= \begin{bmatrix} 1 & y_{2t} \end{bmatrix} \begin{bmatrix} \mu_t \\ \gamma_t \end{bmatrix} + \epsilon_t, \\ \begin{bmatrix} \mu_{t+1} \\ \gamma_{t+1} \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_t \\ \gamma_t \end{bmatrix} + \begin{bmatrix} \eta_{1t} \\ \eta_{2t} \end{bmatrix}, \end{aligned} \tag{15.3}$$

where all the noise components are independent and distributed as $\epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2)$, $\eta_{1t} \sim \mathcal{N}(0, \sigma_\mu^2)$, and $\eta_{2t} \sim \mathcal{N}(0, \sigma_\gamma^2)$, the state transition matrix is the identity $\boldsymbol{T} = \boldsymbol{I}$, the observation matrix is $\boldsymbol{Z}_t = \begin{bmatrix} 1 & y_{2t} \end{bmatrix}$, and the initial states are $\mu_1 \sim \mathcal{N}\left(\bar{\mu}, \sigma_{\mu,1}^2\right)$ and $\gamma_1 \sim \mathcal{N}\left(\bar{\gamma}, \sigma_{\gamma,1}^2\right)$.

The normalized spread, with leverage one (see (15.2)), can then be obtained as

$$z_t = \frac{1}{1 + \gamma_{t|t-1}} \left(y_{1t} - \gamma_{t|t-1}\, y_{2t} - \mu_{t|t-1} \right),$$

where $\mu_{t|t-1}$ and $\gamma_{t|t-1}$ are the hidden states estimated by the Kalman algorithm.

The model parameters σ_ϵ^2, σ_μ^2, and σ_γ^2 (and the initial states) can be determined by simple heuristics or optimally estimated from the data (more computationally demanding). For example, one can use initial training data of T^{LS} samples to estimate μ and γ via least squares, μ^{LS} and γ^{LS}, obtaining the estimated residual ϵ_t^{LS}. Then the following provide an effective heuristic for the state-space model parameters:

$$\begin{aligned} \sigma_\epsilon^2 &= \text{Var}\left[\epsilon_t^{\text{LS}}\right], \\ \mu_1 &\sim \mathcal{N}\left(\mu^{\text{LS}}, \frac{1}{T^{\text{LS}}} \text{Var}\left[\epsilon_t^{\text{LS}}\right]\right), \\ \gamma_1 &\sim \mathcal{N}\left(\gamma^{\text{LS}}, \frac{1}{T^{\text{LS}}} \frac{\text{Var}\left[\epsilon_t^{\text{LS}}\right]}{\text{Var}\left[y_{2t}\right]}\right), \\ \sigma_\mu^2 &= \alpha \times \text{Var}\left[\epsilon_t^{\text{LS}}\right], \\ \sigma_\gamma^2 &= \alpha \times \frac{\text{Var}\left[\epsilon_t^{\text{LS}}\right]}{\text{Var}\left[y_{2t}\right]}, \end{aligned}$$

where the hyper-parameter α determines the ratio of the variability of the slowly time-varying hidden states to the variability of the spread.

The state-space model of the spread in (15.3) can be extended in different ways to potentially improve the performance. One simple extension involves modeling not only the hedge ratio but also its momentum or velocity. This can be done by expanding the hidden state to $\boldsymbol{\alpha}_t = (\mu_t, \gamma_t, \dot{\gamma}_t)$, which leads to the state-space model

$$\begin{aligned} y_{1t} &= \begin{bmatrix} 1 & y_{2t} & 0 \end{bmatrix} \begin{bmatrix} \mu_t \\ \gamma_t \\ \dot{\gamma}_t \end{bmatrix} + \epsilon_t, \\ \begin{bmatrix} \mu_{t+1} \\ \gamma_{t+1} \\ \dot{\gamma}_{t+1} \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_t \\ \gamma_t \\ \dot{\gamma}_t \end{bmatrix} + \boldsymbol{\eta}_t. \end{aligned} \tag{15.4}$$

This model is meant to make γ_t less noisy; with the parameters used in the numerical experiments below, however, it gave no appreciable improvement over the basic model, so it is not pursued further.

Another extension of the state-space modeling in (15.3) under the concept of partial cointegration is to model the spread with an autoregressive component (Clegg & Krauss, 2018). This can be done by defining the hidden state as $\boldsymbol{\alpha}_t = (\mu_t, \gamma_t, \epsilon_t)$, leading to

$$\begin{aligned} y_{1t} &= \begin{bmatrix} 1 & y_{2t} & 1 \end{bmatrix} \begin{bmatrix} \mu_t \\ \gamma_t \\ \epsilon_t \end{bmatrix}, \\ \begin{bmatrix} \mu_{t+1} \\ \gamma_{t+1} \\ \epsilon_{t+1} \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \rho \end{bmatrix} \begin{bmatrix} \mu_t \\ \gamma_t \\ \epsilon_t \end{bmatrix} + \boldsymbol{\eta}_t, \end{aligned} \tag{15.5}$$

where ρ is a parameter to be estimated satisfying $|\rho| < 1$ (or fixed to some reasonable value like $\rho = 0.9$).

15.6.4 Numerical Experiments

We now repeat the pairs trading experiments with market data during 2013–2022 as in Section 15.5. As before, the z-score is computed on a rolling-window basis with a lookback period of six months and pairs trading is implemented via the thresholded strategy with a threshold of $s_0 = 1$. The difference is that we now employ two different methods to track the hedge ratio over time: (i) rolling least squares with lookback period of two years and (ii) the Kalman filter based on the basic model (15.3) with $\alpha = 5 \times 10^{-5}$. This value of α has been fixed by trial and error (without transaction costs) and could be further optimized. With random-walk states, the Kalman filter can be seen as a recursive least squares with a forgetting mechanism, where α controls how fast the past data is forgotten: a larger α implies a faster tracking and a smaller variance of the spread. In principle, all the parameters in the state-space model can also be learned to better fit the data via maximum likelihood estimation methods, although a better fit does not necessarily translate into a better trading performance.

Market Data: EWA and EWC

We first consider the two ETFs EWA and EWC, which track the performance of the Australian and Canadian economies, respectively. As concluded in Section 15.5, EWA and EWC are cointegrated, and pairs trading was evaluated in Section 15.5 based on least squares. We now experiment with the Kalman filter to see what improvement can be obtained.

Figure 15.21 shows the estimated hedge ratios over time, which should be quite constant since the assets are cointegrated. We can observe that the rolling least squares is very noisy with the value wildly varying between 0.6 and 1.2 (of course a longer lookback period could be used, but then it would not adapt fast enough if the true hedge ratio changed). The Kalman filter, on the other hand, is much more stable with the value between 0.55 and 0.65.

Figure 15.22 presents the spreads resulting from the two methods. It is quite apparent that the spread from the Kalman filter is much more stationary and mean reverting than the one

from the rolling least squares. One important point to notice is that the variance of the spread obtained from the Kalman filter depends on the choice of α: if the spread variance becomes too small, then the profit may totally disappear after taking into account transaction costs, so care has to be taken with this choice.

Finally, Figure 15.23 provides the cumulative returns obtained by the two methods (ignoring transaction costs, and computed with the weights held during each period as discussed in Section 15.3.2). Both strategies are profitable, ending at about 0.51 (rolling least squares) and 0.78 (Kalman). The two curves are, however, close until early 2020, when a large jump takes place (the COVID-19 market turmoil) that the Kalman filter captures much better; most of the difference in the final value comes from that single episode. The Kalman filter tracks the hedge ratio more smoothly and gives a spread with a smaller variance, but it also trades more: its turnover is about 4.5 times that of rolling least squares (0.23 versus 0.05 per day, counting changes of both the signal and the hedge ratio), so transaction costs hurt it more. With a cost of 5 basis points on the traded amount, the final values become 0.56 (Kalman) and 0.46 (rolling least squares), and with 10 basis points the ranking is reversed (0.33 versus 0.41). Again, it is important to reiterate that special care has to be taken in practice with the choice of α so that the spread variance is large enough to provide a profit after transaction costs.

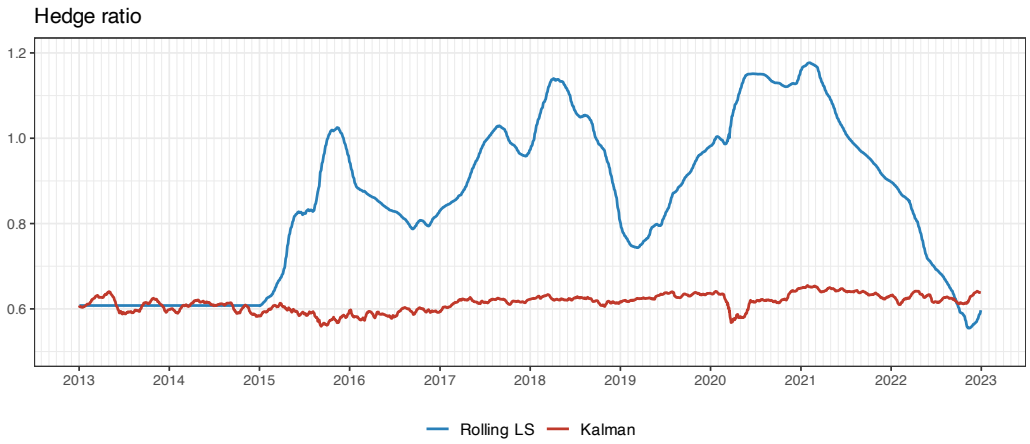


Figure 15.21 Tracking of hedge ratio for pairs trading on EWA–EWC.

Market Data: KO and PEP

Next, we revisit pairs trading with the stocks Coca-Cola (KO) and Pepsi (PEP). As concluded from the cointegration tests in Section 15.4, they do not seem to be cointegrated. In addition, from the trading experiments in Section 15.5, profitability was dubious as indicated in Figures 15.19 and 15.18. We now look at whether the Kalman filter changes the picture.

Figure 15.24 shows the estimated hedge ratios over time. Again, we can observe that rolling least squares is noisy and not very consistent, whereas the Kalman filter is stable while still reacting to the big change that happens in early 2020 (perhaps due to the COVID-19 pandemic).

Figure 15.25 gives the spreads, and one can clearly appreciate a significant difference between

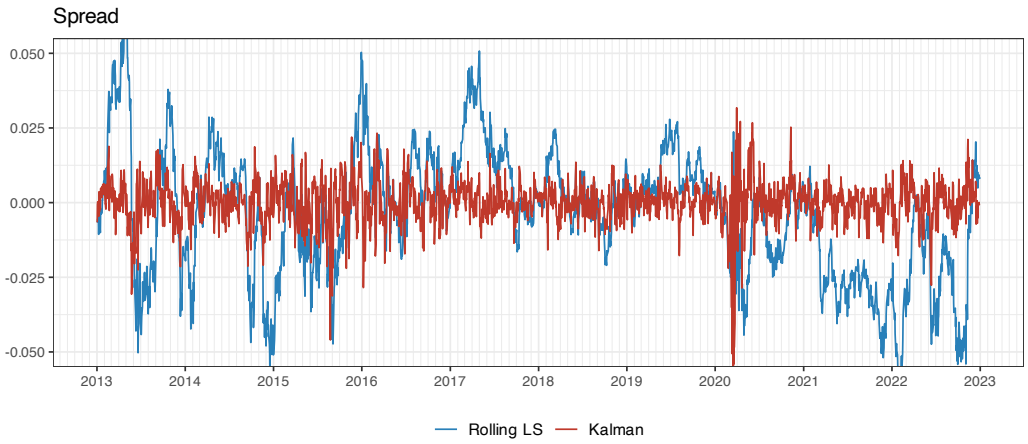


Figure 15.22 Spread for pairs trading on EWA–EWC.

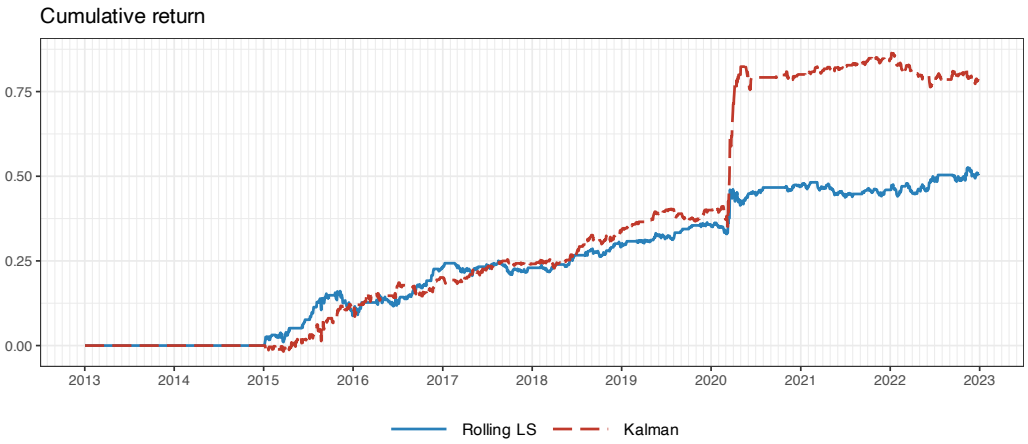


Figure 15.23 Cumulative return for pairs trading on EWA–EWC.

the two methods. Observe early 2020: rolling least squares loses the tracking and cointegration is clearly lost, whereas the Kalman filter is able to track after a momentary loss reflected in the shock on the spread.

Finally, Figure 15.26 provides the cumulative returns. In this case neither method produces a convincing profit: the final values are about 0.13 (rolling least squares) and 0.04 (Kalman), both with sizable drawdowns. Moreover, one cannot forget that transaction costs have not been considered and the Kalman filter trades about five times as much as rolling least squares: with a cost of just 2 basis points on the traded amount, the final value is 0.11 for rolling least squares and already negative (-0.03) for the Kalman filter. This is consistent with KO and PEP not being cointegrated: better tracking of the hedge ratio cannot create a mean-reverting spread where there is none. The conclusion is that Kalman filtering is a useful tool to track a cointegration relationship when it exists, but it cannot create one.

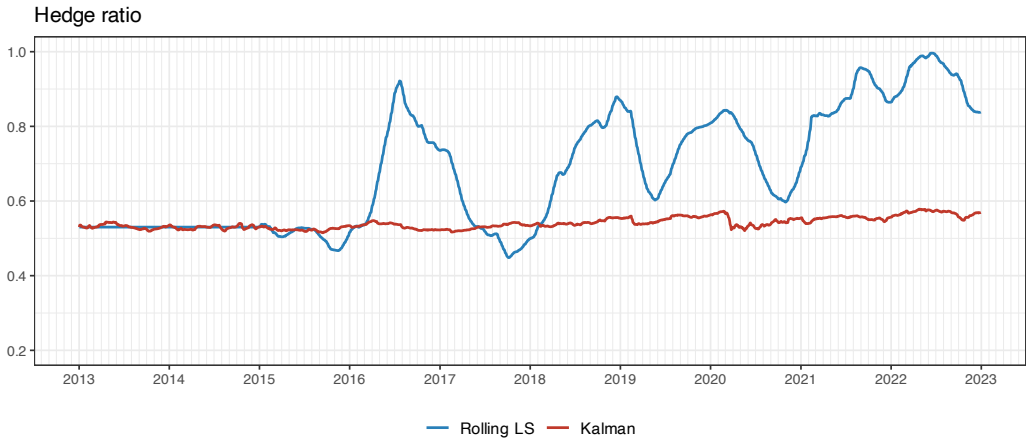


Figure 15.24 Tracking of hedge ratio for pairs trading on KO–PEP.

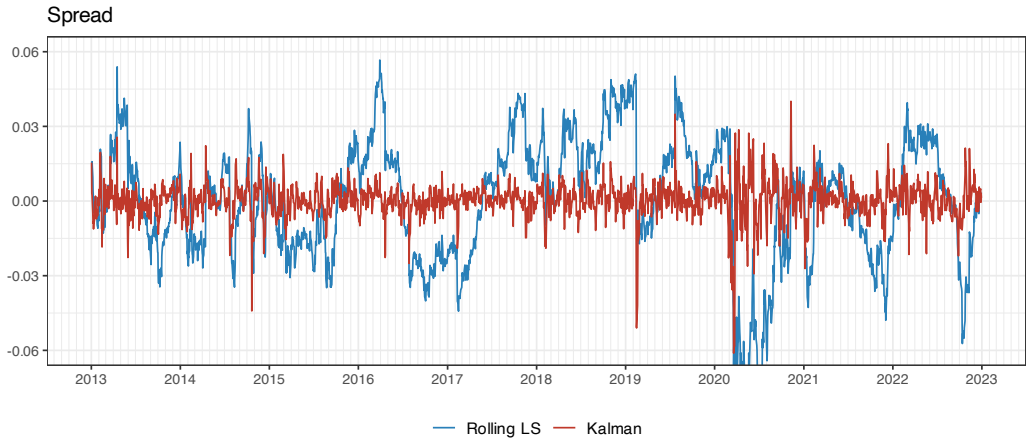


Figure 15.25 Spread for pairs trading on KO–PEP.

15.7 Statistical Arbitrage

Pairs trading focuses on discovering cointegration and tracking the cointegration relationship between pairs of assets. However, the idea can be naturally extended to more than two assets for more flexibility. This is more generally referred to as *statistical arbitrage* or, for short, *StatArb*.

Cointegration for more than two assets follows essentially the same idea: construct a linear combination of multiple time series such that the combination is mean reverting. The difference is that the mathematical modeling to capture the multiple cointegration relationships becomes more involved.

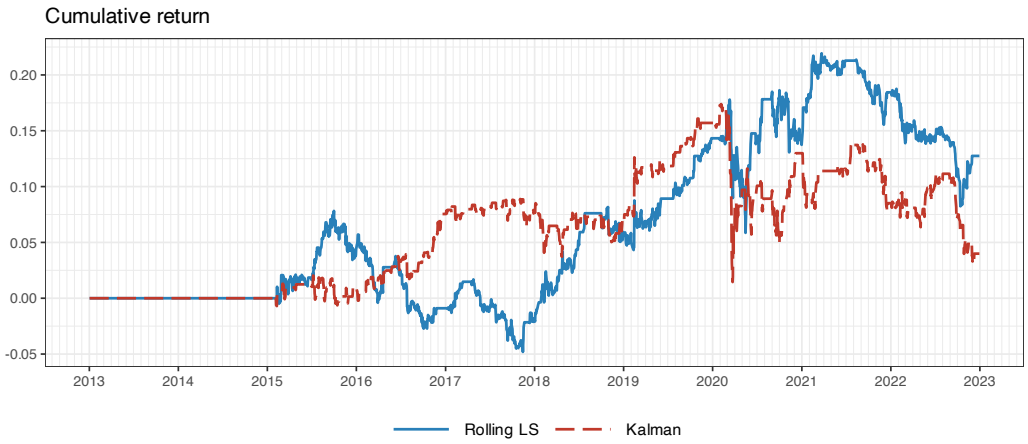


Figure 15.26 Cumulative return for pairs trading on KO–PEP.

15.7.1 Least Squares

Least squares can still be used to determine the cointegration relationship. In the case of $K > 2$ time series, we still need to choose the one to be regressed by the others. Suppose we want to fit $y_{1t} \approx \mu + \sum_{k=2}^{K} \gamma_k \, y_{kt}$ based on T observations. Then, the least squares formulation is

$$\underset{\mu,\boldsymbol{\gamma}}{\text{minimize}} \quad \|\boldsymbol{y}_1 - (\mu \mathbf{1} + \boldsymbol{Y}_2 \boldsymbol{\gamma})\|_2^2 \,,$$

where the vector $\boldsymbol{y}_1$ contains the T observations of the first time series, the matrix $\boldsymbol{Y}_2$ contains the T observations of the remaining $K-1$ time series columnwise, and vector $\boldsymbol{\gamma} \in \mathbb{R}^{K-1}$ contains the $K-1$ hedge ratios. The solution gives the estimates

$$\begin{bmatrix} \hat{\mu} \\ \hat{\boldsymbol{\gamma}} \end{bmatrix} = \begin{bmatrix} \mathbf{1}^\mathsf{T}\mathbf{1} & \mathbf{1}^\mathsf{T}\boldsymbol{Y}_2 \\ \boldsymbol{Y}_2^\mathsf{T}\mathbf{1} & \boldsymbol{Y}_2^\mathsf{T}\boldsymbol{Y}_2 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{1}^\mathsf{T}\boldsymbol{y}_1 \\ \boldsymbol{Y}_2^\mathsf{T}\boldsymbol{y}_1 \end{bmatrix}.$$

In practice, this estimation process has to be performed on a rolling-window basis to adapt to the slow changes over time.

The normalized portfolio (with leverage 1) is

$$\boldsymbol{w} = \frac{1}{1 + \|\boldsymbol{\gamma}\|_1} \begin{bmatrix} 1 \\ -\boldsymbol{\gamma} \end{bmatrix},$$

with corresponding normalized spread $z_t = \boldsymbol{w}^\mathsf{T} \boldsymbol{y}_t$.

It is important to point out that this approach produces a single cointegration relationship, but there may be others that have gone unnoticed. One approach could be to iteratively try to capture more cointegration relationships orthogonal to the previously discovered ones. In addition, this method requires choosing one time series (out of the K possible ones) to be regressed. In practice, the discovery of multiple cointegration relationships is better achieved by the more sophisticated VECM modeling described next.

15.7.2 VECM

A very common model in econometrics for a multivariate time series (typically denoting the log-prices of N assets), $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, \ldots$, is based on taking the first-order difference, $\Delta \boldsymbol{y}_t = \boldsymbol{y}_t - \boldsymbol{y}_{t-1}$, and then using a vector autoregressive (VAR) model of order p:

$$\Delta \boldsymbol{y}_t = \boldsymbol{\phi}_0 + \sum_{i=1}^{p} \boldsymbol{\Phi}_i \Delta \boldsymbol{y}_{t-i} + \boldsymbol{\epsilon}_t,$$

where the parameters of the model are $\boldsymbol{\phi}_0 \in \mathbb{R}^N$, $\boldsymbol{\Phi}_1, \ldots, \boldsymbol{\Phi}_p \in \mathbb{R}^{N \times N}$, and $\boldsymbol{\epsilon}_t$ is the innovation term (see Section 4.3 in Chapter 4 for details). This approach has the feature that due to the differencing it is a stationary model. Unfortunately, this differencing may also destroy some interesting structure in the original data.

The *vector error correction model* (VECM) (Engle & Granger, 1987) was proposed as a way to apply the VAR model directly on the original sequence without differencing, with the potential danger of lack of stationarity. Employing the VAR model on the original series $\boldsymbol{y}_t$ and rewriting it in terms of $\Delta \boldsymbol{y}_t$ leads to the more refined model

$$\Delta \boldsymbol{y}_t = \boldsymbol{\phi}_0 + \boldsymbol{\Pi} \boldsymbol{y}_{t-1} + \sum_{i=1}^{p-1} \tilde{\boldsymbol{\Phi}}_i \Delta \boldsymbol{y}_{t-i} + \boldsymbol{\epsilon}_t, \tag{15.6}$$

where the matrix coefficients $\boldsymbol{\Pi} \in \mathbb{R}^{N \times N}$ and $\tilde{\boldsymbol{\Phi}}_1, \ldots, \tilde{\boldsymbol{\Phi}}_{p-1} \in \mathbb{R}^{N \times N}$ can be related to the $\boldsymbol{\Phi}_i$ used in the previous VAR model. This model includes the term $\boldsymbol{\Pi} \boldsymbol{y}_{t-1}$ that could potentially make the model nonstationary because the time series $\boldsymbol{y}_t$ is nonstationary. However, after a careful inspection of (15.6), it is clear that since the left-hand side, $\Delta \boldsymbol{y}_t$, is stationary, so must be the right-hand side, which implies that $\boldsymbol{\Pi} \boldsymbol{y}_{t-1}$ must be stationary.

As it turns out, the matrix $\boldsymbol{\Pi}$ is of utmost importance in guaranteeing stationarity of the term $\boldsymbol{\Pi} \boldsymbol{y}_t$. In general, this matrix will be of low rank, which means that it can be decomposed as the product of two matrices,

$$\boldsymbol{\Pi} = \boldsymbol{\alpha} \boldsymbol{\beta}^{\mathsf{T}},$$

with $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^{N \times K}$ having K columns, where K is the rank of $\boldsymbol{\Pi}$. This reveals that the nonstationary series $\boldsymbol{y}_t$ (log-prices) becomes stationary after multiplication with $\boldsymbol{\beta}^{\mathsf{T}}$. In other words, the multivariate time series $\boldsymbol{y}_t$ is cointegrated and each column of matrix $\boldsymbol{\beta}$ produces a different cointegration relationship.

To be more precise, three cases can be identified in terms of the rank of $\boldsymbol{\Pi}$:

- $K = N$: this means that $\boldsymbol{y}_t$ is already stationary (rarely the case in practice);
- $K = 0$: this means that $\boldsymbol{y}_t$ is not cointegrated (VECM reduces to a VAR model); and
- $1 < K < N$: this is the interesting case that provides K different cointegration relationships.

Recall that Johansen's test (Johansen, 1991, 1995) described in Section 15.4 precisely tests the value of the rank of the matrix $\boldsymbol{\Pi}$ arising in the VECM time series modeling.

15.7.3 Optimum Mean-Reverting Portfolio

Least squares and VECM modeling can be conveniently employed to obtain cointegration relationships that produce mean-reverting spreads for pairs trading or statistical arbitrage strategies. In fact, VECM provides us with K different cointegration relationships contained in the columns of the matrix $\boldsymbol{\beta} \in \mathbb{R}^{N \times K}$. This means that K different pairs trading strategies could be run in parallel, fully exploiting all these K directions in the N-dimensional space.

Alternatively, an optimization-based approach can be taken to design the portfolio that produces the spread. Since the profit in pairs trading is determined by the product of the number of trades and the threshold, the goal is to maximize the zero-crossing rate (which determines the number of trades) as well as the variance of the spread (which determines the threshold). In practice, several proxies can be used to quantify the zero-crossing rate, producing a variety of problem formulations (Cuturi & d'Aspremont, 2013, 2016; d'Aspremont, 2011).

A combined approach of VECM modeling and the optimization-based approach can also be taken. The K-dimensional subspace defined by the matrix $\boldsymbol{\beta}$ in VECM modeling can be interpreted as defining a *cointegration subspace*. From this perspective, any portfolio lying within this subspace will provide a mean-reverting spread. Then, rather than using all the K dimensions to run K pairs trading strategies in parallel, one could try to further optimize one or more portfolios within that subspace to obtain the best possible spreads (Zhao & Palomar, 2018; Zhao et al., 2019). Overall, this would imply running fewer strategies in parallel but perhaps with stronger mean reversion.

Mathematically, it is convenient to formulate this problem in terms of a portfolio on the K spreads rather than on the N original assets as follows.

1. From the K columns of matrix $\boldsymbol{\beta}$, we get K cointegration relationships:
$$\boldsymbol{\beta}_k \in \mathbb{R}^N, \qquad k = 1, \ldots, K.$$
2. We can then construct K portfolios (normalized with leverage 1):
$$\boldsymbol{w}_k = \frac{1}{\|\boldsymbol{\beta}_k\|_1} \boldsymbol{\beta}_k, \qquad k = 1, \ldots, K.$$
3. From these portfolios, we can compute the K spreads from the original time series $\boldsymbol{y}_t \in \mathbb{R}^N$:
$$z_{kt} = \boldsymbol{w}_k^\mathsf{T} \boldsymbol{y}_t, \qquad k = 1, \ldots, K,$$
or, more compactly,
$$\boldsymbol{z}_t = [\boldsymbol{w}_1 \ldots \boldsymbol{w}_K]^\mathsf{T} \boldsymbol{y}_t \in \mathbb{R}^K.$$
4. At this point, we can conveniently optimize a K-dimensional portfolio $\boldsymbol{w}_z \in \mathbb{R}^K$ on the spreads $\boldsymbol{z}_t$, from which the overall portfolio to be executed on the underlying assets $\boldsymbol{y}_t$ can be recovered as
$$\boldsymbol{w}^{\text{overall}} = [\boldsymbol{w}_1 \ldots \boldsymbol{w}_K] \times \boldsymbol{w}_z.$$

We can now focus on the optimization of the spread portfolio $\boldsymbol{w}_z$ defined on the spreads $\boldsymbol{z}_t$ (Zhao & Palomar, 2018; Zhao et al., 2019). To design the spread portfolio, the goal is to

optimize some proxy of the zero-crossing rate while controlling the spread variance. Defining for convenience the lagged covariance matrices of the spreads as

$$\boldsymbol{M}_i = \mathbb{E}\left[(\boldsymbol{z}_t - \mathbb{E}[\boldsymbol{z}_t])\,(\boldsymbol{z}_{t+i} - \mathbb{E}[\boldsymbol{z}_{t+i}])^{\mathsf{T}}\right], \qquad i = 0, 1, 2, \ldots$$

we can express the variance of the resulting spread as $\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{M}_0\boldsymbol{w}_z$. As for the zero-crossing rate, it is not that straightforward to obtain a convenient expression and several proxies have been proposed in the literature (Cuturi & d'Aspremont, 2013, 2016; Zhao & Palomar, 2018; Zhao et al., 2019):

- *Predictability statistic*: This tries to measure the similarity of a random signal to white noise (small predictability means closer to white noise and vice versa). Mathematically, it is defined as the proportion of the signal variance that is predicted by the autoregressive coefficient in an AR(1) model (Box & Tiao, 1977). Assuming that the spreads follow a vector AR(1) model, $\boldsymbol{z}_t = \boldsymbol{A}\boldsymbol{z}_{t-1} + \boldsymbol{\epsilon}_t$ with the autoregressive matrix coefficient given by $\boldsymbol{A} = \boldsymbol{M}_1^{\mathsf{T}}\boldsymbol{M}_0^{-1}$, it follows that the predictability statistic for the final spread $\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{z}_t$ can be written as

$$\operatorname{pre}(\boldsymbol{w}_z) = \frac{\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{A}\boldsymbol{M}_0\boldsymbol{A}^{\mathsf{T}}\boldsymbol{w}_z}{\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{M}_0\boldsymbol{w}_z} = \frac{\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{M}_1^{\mathsf{T}}\boldsymbol{M}_0^{-1}\boldsymbol{M}_1\boldsymbol{w}_z}{\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{M}_0\boldsymbol{w}_z}.$$

- *Portmanteau statistic*: This also tries to measure the similarity of a random process to white noise. The portmanteau statistic of order p is defined as $\sum_{i=1}^{p}\rho_i^2$, where ρ_i is the signal autocorrelation at the ith lag (Box & Pierce, 1970). Applied to our case, the portmanteau statistic for the final spread $\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{z}_t$ is

$$\operatorname{por}(\boldsymbol{w}_z) = \sum_{i=1}^{p}\left(\frac{\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{M}_i\boldsymbol{w}_z}{\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{M}_0\boldsymbol{w}_z}\right)^2.$$

- *Crossing statistic*: This is defined as the number of zero crossings of a centered stationary process and is given by $\arccos(\rho_1)/\pi$ (Kedem, 1994; Ylvisaker, 1965). Maximizing the number of zero crossings is then equivalent to minimizing ρ_1. For the final spread $\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{z}_t$, the crossing statistic is given by

$$\operatorname{cro}(\boldsymbol{w}_z) = \frac{\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{M}_1\boldsymbol{w}_z}{\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{M}_0\boldsymbol{w}_z}.$$

  The penalized crossing statistic combines $\operatorname{cro}(\boldsymbol{w}_z)$ with $\operatorname{por}(\boldsymbol{w}_z)$ to minimize the high-order autocorrelations (Cuturi & d'Aspremont, 2013, 2016):

$$\operatorname{pcro}(\boldsymbol{w}_z) = \frac{\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{M}_1\boldsymbol{w}_z}{\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{M}_0\boldsymbol{w}_z} + \eta\sum_{i=2}^{p}\left(\frac{\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{M}_i\boldsymbol{w}_z}{\boldsymbol{w}_z^{\mathsf{T}}\boldsymbol{M}_0\boldsymbol{w}_z}\right)^2,$$

  where η is a hyper-parameter that controls the high-order penalization term.

To summarize, we can formulate a mean-reverting portfolio to optimize some zero-crossing

proxy, such as $\text{pcro}(\boldsymbol{w}_z)$, while fixing the spread variance:

$$\begin{array}{ll} \underset{\boldsymbol{w}_z}{\text{minimize}} & \boldsymbol{w}_z^\mathsf{T}\boldsymbol{M}_1\boldsymbol{w}_z + \eta \sum_{i=2}^{p} \left(\boldsymbol{w}_z^\mathsf{T}\boldsymbol{M}_i\boldsymbol{w}_z\right)^2 \\ \text{subject to} & \boldsymbol{w}_z^\mathsf{T}\boldsymbol{M}_0\boldsymbol{w}_z \geq \nu, \\ & \boldsymbol{w}_z \in \mathcal{W}, \end{array} \tag{15.7}$$

where ν, η are hyper-parameters, and $\mathcal{W}$ denotes some portfolio constraints, such as $\|\boldsymbol{w}_z\|_2 = 1$ to avoid numerical issues (Cuturi & d'Aspremont, 2013), a sparsity constraint $\|\boldsymbol{w}_z\|_0 = k$ (Cuturi & d'Aspremont, 2016), a budget / market exposure constraint $\mathbf{1}^\mathsf{T}\boldsymbol{w}_z = 1/0$ (Zhao & Palomar, 2018), a leverage constraint $\|\boldsymbol{w}_z\|_1 = 1$ (Zhao et al., 2019) or, even better in practice, a leverage constraint on the overall portfolio (Zhao et al., 2019):

$$\| [\boldsymbol{w}_1 \ldots \boldsymbol{w}_K] \times \boldsymbol{w}_z \|_1 = 1.$$

15.7.4 Numerical Experiments

Market Data: SPY, IVV, and VOO

To illustrate multiple cointegration relationships via VECM modeling, we consider again three ETFs that track the S&P 500 index, namely, Standard & Poor's Depository Receipts SPY, iShares IVV, and Vanguard's VOO. As previously verified in Section 15.4, Johansen's test indicates two cointegration relationships present, which can be exploited via statistical arbitrage.

Figure 15.27 shows the cumulative returns obtained by executing pairs trading on (i) the first (strongest) spread, (ii) the second (weaker) spread, (iii) the optimized spread according to (15.7), and (iv) both spreads in parallel (allocating half of the budget to each spread). It can be observed that the strongest spread is better than the second spread. The optimized spread does not seem to offer an improvement in this particular case (for a larger dimensionality of the cointegrated subspace it may still offer some benefits). Last, but not least, using both spreads in parallel offers a steadier cumulative return (i.e., better Sharpe ratio) as expected from the diversity gain. Table 15.5 provides the Sharpe ratios of the different approaches for a more quantitative comparison.

Table 15.5 Sharpe ratios for pairs trading on SPY–IVV–VOO: single spreads, both in parallel, and optimized spread.

Spread	Sharpe ratio
Spread #1	6.78
Spread #2	5.39
Optimized spread	6.75
Both spreads	8.37

15.8 Summary

Pairs trading or, more generally, statistical arbitrage refer to market-neutral strategies that arbitrage on the relative value of assets. Some key concepts include:

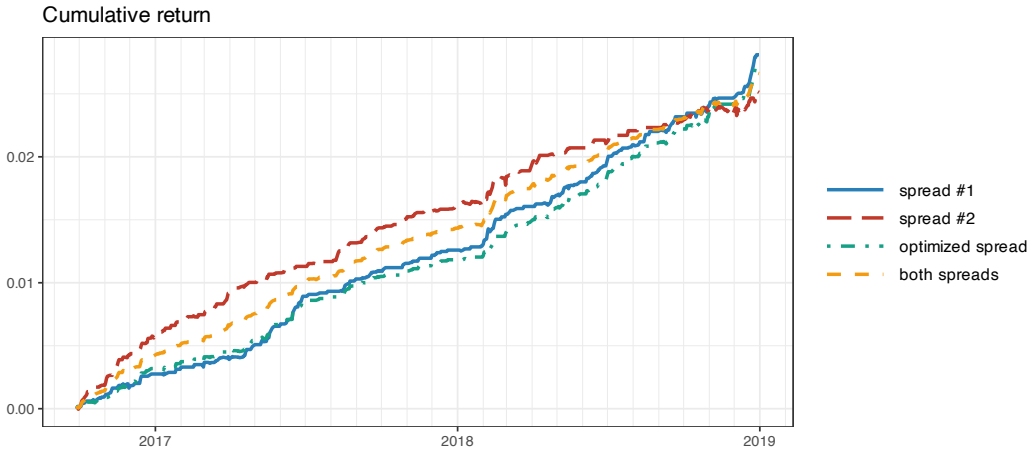


Figure 15.27 Cumulative return for pairs trading on SPY–IVV–VOO: single spreads, both in parallel, and optimized spread.

- *Mean reversion* of a time series means that there is a long-term average value around which the series may fluctuate over time but eventually will revert back to. This allows the contrarian strategy of buying low and selling high (as opposed to a momentum-based strategy, which would buy as the price increases and sell while the price decreases).
- *Cointegration* refers to assets that are not mean reverting themselves, but combined together in the right way become mean reverting.
- *Pairs trading* is a strategy invented in the 1980s that combines two cointegrated assets to generate a synthetic mean-reverting asset. This mean reversion implies that the strategy is not affected by the market trend, that is, it is market neutral. This is in contrast to momentum-based strategies that precisely follow the market trend and exhibit a high market exposure.
- *Implementation* of pairs trading requires:
  - discovering cointegrated assets, for example, via cointegration statistical tests;
  - tracking the cointegration relationship over time, either via rolling least squares or the Kalman filtering; and
  - executing the actual trading, typically with a simple thresholded strategy.
- *Kalman filtering*, originally developed in the 1960s for vehicle guidance and navigation, is a flexible tool to track cointegration over time, although it trades more than rolling least squares and offers no advantage when the pair is not cointegrated. A variety of different state-space models can be formulated to track cointegration and then solved via the Kalman algorithm.
- *Statistical arbitrage* is the generalization of pairs trading to more than two assets. This requires more sophisticated multivariate modeling of the assets (VECM modeling) to discover the cointegration relationships.

Exercises

15.1 (Mean reversion)

a. Generate a random walk and plot it. Is it stationary? Does it revert to the mean?
b. Generate an AR(1) sequence with autoregressive coefficient less than 1 and plot it. Is it stationary? Does it revert to the mean?
c. Change the autoregressive coefficient of the AR(1) model and observe how the strength of the mean reversion changes.

15.2 (Cointegration vs. correlation) Consider the cointegration model of two time series with a common trend:

$$y_{1t} = x_t + w_{1t},$$
$$y_{2t} = x_t + w_{2t},$$

where x_t is a stochastic common trend defined as a random walk,

$$x_t = x_{t-1} + w_t,$$

and the terms w_{1t}, w_{2t}, w_t are i.i.d. residual terms, mutually independent, with variances σ_1^2, σ_2^2, and σ^2, respectively.

Generate realizations of such time series with different values for the residual variances and plot the sequences as well as the scatter plot of the series differences (Δy_{1t} vs. Δy_{2t}). Choose the appropriate values of the variances to obtain cointegrated time series with low correlation as well as non-cointegrated time series with high correlation.

15.3 (Simple pairs trading on AR(1) spread) Generate a synthetic mean-reverting spread with an AR(1) model for the log-prices, implement a simple pairs trading strategy based on thresholds, and plot the cumulative return (ignoring transaction costs).

Note: with a buy position, the portfolio return is the same as that of the spread; with a short position, it is the opposite; and with no position, it is just zero.

15.4 (Discovering cointegrated pairs)

a. Download market data corresponding to several assets (e.g., stocks, commodities, ETFs, or cryptocurrencies).
b. Implement a prescreening approach on different pairs based on normalized prices.
c. Then consider running cointegration tests on the successful pairs from the prescreening phase. In particular, try some of the following tests:
   - DF
   - ADF
   - PP
   - PGFF
   - ERSD
   - JOT
   - SPR
d. Plot the spreads of the successful cointegrated pairs as well as some of the unsuccessful ones for comparison.

15.5 (Pairs trading with least squares)

a. Download market data corresponding to a pair of cointegrated assets (e.g., stocks, commodities, ETFs, or cryptocurrencies).
b. Using an initial window as training data, estimate the hedge ratio γ via least squares.
c. Using that hedge ratio, compute the normalized spread (with leverage 1) in the remaining window as test data, that is, a spread obtained using the normalized portfolio

$$\boldsymbol{w} = \frac{1}{1+\gamma}\begin{bmatrix} 1 \\ -\gamma \end{bmatrix}.$$

d. Trade the normalized spread via the thresholded strategy.
e. Plot the cumulative return ignoring transaction costs.
f. Plot the cumulative return including transaction costs (e.g., as 30–90 bps of the portfolio turnover).

15.6 (Pairs trading with rolling least squares) Repeat Exercise 15.5 but using rolling least squares to track the hedge ratio over time γ_t.

15.7 (Pairs trading with Kalman filtering) Repeat Exercise 15.5 but using the Kalman filter to track the hedge ratio over time γ_t.

15.8 (Statistical arbitrage with more than two assets)

a. Download market data corresponding to $N > 2$ cointegrated assets (e.g., stocks, commodities, ETFs, or cryptocurrencies).
b. Choose a pair of assets and implement pairs trading via least squares.
c. With all the N assets, use VECM to obtain $K > 2$ cointegration relationships and then:
   - implement pairs trading with the strongest direction;
   - implement K parallel pairs trading and combine the result into a final cumulative return plot.
d. Compare and discuss the three implementations: pairs trading on just two assets, pairs trading on the strongest of the K directions, and K parallel pairs trading.

References

Anderson, B. D. O., & Moore, J. B. (1979). *Optimal Filtering*. Prentice Hall.

Avellaneda, M., & Lee, J.-H. (2010). Statistical arbitrage in the US equities market. *Quantitative Finance*, *10*(7), 761–782.

Banerjee, A., Dolado, J. J., Galbraith, J. W., & Hendry, D. F. (1993). *Cointegration, Error Correction, and the Econometric Analysis of Non-Stationary Data*. Oxford University Press.

Box, G. E., & Pierce, D. A. (1970). Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American Statistical Association*, *65*(332), 1509–1526.

Box, G. E., & Tiao, G. C. (1977). A canonical analysis of multiple time series. *Biometrika*, *64*(2), 355–365.

Brockwell, P. J., & Davis, R. A. (2002). *Introduction to Time Series and Forecasting*. Springer.

Chan, E. P. (2008). *Quantitative Trading: How to Build Your Own Algorithmic Trading Business*. John Wiley & Sons.

Chan, E. P. (2013). *Algorithmic Trading: Winning Strategies and Their Rationale*. John Wiley & Sons.

Clegg, M. (2014). On the persistence of cointegration in pairs trading. *SSRN Electronic Journal*. https://dx.doi.org/10.2139/ssrn.2491201

Clegg, M. (2023). *egcm: Engle–Granger Cointegration Models* [R package]. https://CRAN.R-project.org/package=egcm

Clegg, M., & Krauss, C. (2018). Pairs trading with partial cointegration. *Quantitative Finance*, *18*(1), 121–138.

Cuturi, M., & d'Aspremont, A. (2013). Mean reversion with a variance threshold. *Proceedings of the International Conference on Machine Learning (ICML)*, *28*, 271–279.

Cuturi, M., & d'Aspremont, A. (2016). Mean-reverting portfolios: Tradeoffs between sparsity and volatility. In A. N. Akansu, S. R. Kulkarni, & D. M. Malioutov (Eds.), *Financial Signal Processing and Machine Learning* (pp. 23–40). John Wiley & Sons.

d'Aspremont, A. (2011). Identifying small mean-reverting portfolios. *Quantitative Finance*, *11*(3), 351–364.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autore-gressive time series with a unit root. *Journal of the American Statistical Association*, *74*, 427–431.

Durbin, J., & Koopman, S. J. (2012). *Time Series Analysis by State Space Methods* (2nd ed.). Oxford University Press.

Ehrman, D. S. (2006). *The Handbook of Pairs Trading: Strategies using Equities, Options, and Futures*. John Wiley & Sons.

Elliott, R. J., Van Der Hoek, J., & Malcolm, W. P. (2005). Pairs trading. *Quantitative Finance*, *5*(3), 271–276.

Engle, R. F., & Granger, C. W. J. (1987). Co-integration and error correction: Representation, estimation, and testing. *Econometrica: Journal of the Econometric Society*, *55*(2), 251–276.

Fabozzi, F. J., Focardi, S. M., & Kolm, P. N. (2010). *Quantitative Equity Investing: Techniques and Strategies*. John Wiley & Sons.

Feng, Y., & Palomar, D. P. (2016). A signal processing perspective on financial engineering. *Foundations and Trends in Signal Processing, Now Publishers*, *9*(1–2), 1–231.

Gatev, E., Goetzmann, W. N., & Rouwenhorst, K. G. (2006). Pairs trading: Performance of a relative-value arbitrage rule. *Review of Financial Studies*, *19*(3), 797–827.

Harris, R. I. D. (1995). *Using Cointegration Analysis in Econometric Modelling*. Prentice Hall.

Harvey, A. (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

Harvey, A., & Koopman, S. J. (2009). Unobserved components models in economics and finance: The role of the Kalman filter in time series econometrics. *IEEE Control Systems Magazine*, *29*(6), 71–81.

Helske, J. (2017). KFAS: Exponential family state space models in R. *Journal of Statistical Software*, *78*(10), 1–39.

Holmes, E. E., Ward, E. J., & Wills, K. (2012). MARSS: Multivariate autoregressive state-space models for analyzing time-series data. *The R Journal*, *4*(1), 11–19.

Johansen, S. (1991). Estimation and hypothesis testing of cointegration vectors in Gaussian vector autoregressive models. *Econometrica: Journal of the Econometric Society*, *59*(6), 1551–1580.

Johansen, S. (1995). *Likelihood-based Inference in Cointegrated Vector Autoregressive Models*. Oxford University Press.

Kay, S. M. (1993). *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall.

Kedem, B. (1994). *Time Series Analysis by Higher Order Crossings*. IEEE Press.

Krauss, C. (2017). Statistical arbitrage pairs trading strategies: Review and outlook. *Journal of Economic Surveys*, *31*(2), 513–545.

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer.

Petris, G., & Petrone, S. (2011). State space models in R. *Journal of Statistical Software*, *41*(4), 1–25.

Pfaff, B. (2008). *Analysis of Integrated and Cointegrated Time Series With R* (2nd ed.). Springer.

Pfaff, B., Zivot, E., & Stigler, M. (2022). *urca: Unit Root and Cointegration Tests for Time Series Data* [R package]. https://CRAN.R-project.org/package=urca

Scharf, L. L. (1991). *Statistical Signal Processing*. Addison-Wesley.

Shumway, R. H., & Stoffer, D. S. (2017). *Time Series Analysis and Its Applications* (4th ed.). Springer.

Triantafyllopoulos, K., & Montana, G. (2011). Dynamic modeling of mean-reverting spreads for statistical arbitrage. *Computational Management Science*, *8*(1–2), 23–49.

Tsay, R. S. (2010). *Analysis of Financial Time Series* (3rd ed.). John Wiley & Sons.

Tsay, R. S. (2013). *Multivariate Time Series Analysis: With R and Financial Applications*. John Wiley & Sons.

Tusell, F. (2011). Kalman filtering in R. *Journal of Statistical Software*, *39*(2), 1–27.

Vidyamurthy, G. (2004). *Pairs Trading: Quantitative Methods and Analysis*. John Wiley & Sons.

Ylvisaker, N. D. (1965). The expected number of zeros of a stationary Gaussian process. *The Annals of Mathematical Statistics*, *36*(3), 1043–1046.

Zhao, Z., & Palomar, D. P. (2018). Mean-reverting portfolio with budget constraint. *IEEE Transactions on Signal Processing*, *66*(9), 2342–2357.

Zhao, Z., Zhou, R., & Palomar, D. P. (2019). Optimal mean-reverting portfolio with leverage constraint for statistical arbitrage in finance. *IEEE Transactions on Signal Processing*, *67*(7), 1681–1695.

Zivot, E., Wang, J., & Koopman, S. J. (2004). State space modeling in macroeconomics and finance using SsfPack for S+FinMetrics. In A. Harvey, S. J. Koopman, & N. Shephard (Eds.), *State Space and Unobserved Component Models: Theory and Applications* (pp. 284–335). Cambridge University Press.

16

Deep Learning Portfolios

"Robots took the jobs of factory workers. Artificial intelligence will take the jobs of PMs."

— Anonymous

Artificial intelligence (AI) is a very broad term that, roughly speaking, refers to the general ability of computers to emulate human thought and perform tasks in real-world environments. *Machine learning* (ML) is a subset of AI that refers to the technologies and algorithms that enable systems to identify patterns, make decisions, and improve themselves through experience and data. In particular, *neural networks* (NN) are flexible architectures that attempt to emulate the structure of the neurons in the human brain and provide very powerful tools for systems to learn automatically from examples. This is in opposition to the traditional approach in computer science where the machine is already programmed beforehand to perform one specific task, such as the portfolio formulations considered in this book.

More specifically, *deep neural networks*, also broadly referred to as *deep learning* (DL), have ignited a revolution in many domain-specific areas with outstanding performance, putting previous traditional methods to shame in terms of performance. Some areas revolutionized by deep neural networks (in the sense of achieving close-to-human or superhuman performance) include:

- *image recognition*: superhuman performance (this goes from simple recognition of cats to sophisticated cancer detection from X-rays);
- *natural language processing* (NLP): human-level performance;
- *board and video games*: superhuman performance (e.g., chess, Go, and Atari video games);
- *video processing*: close-to-human performance (e.g., real-time video navigation in drones);
- *protein folding*: superhuman performance;
- *self-driving cars*: not yet superhuman or even human-level, but soon to arrive;
- *professional and academic benchmarks*: human-level performance (e.g., passing a simulated bar exam with a score around the top 10% of test takers).

The million-dollar question is whether this revolution will extend to financial systems. To start with, there are many ways in which DL can be used in financial systems. In fact, it has already

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

been successfully employed in some specific problems, such as sentiment analysis of news, credit default detection, or satellite image analysis to detect stock levels or crop production related to companies.

Following the theme of this book, we will investigate how DL can be applied in portfolio design. That is, whether one can design a black box that takes as input some financial data and outputs the portfolio to be used. At the time this book was written, the subject was still developing, and the answer to this question remained unclear. In fact, numerous papers were being published, and various open-source software libraries were becoming available.

16.1 Machine Learning

The approach taken in this book has been based on first modeling the data and then designing a portfolio based on such a model, as illustrated in the big-picture diagram of Figure 1.3 in Chapter 1. In machine learning, the approach is more direct and algorithm based, attempting to *learn* a "black-box" model of reality (Breiman, 2001). A thorough explanation of *statistical learning* (which can be loosely considered equivalent to machine learning) can be found in the textbooks Hastie et al. (2009), James et al. (2013), and Shalev-Shwartz and Ben-David (2014).

Machine learning encompasses a variety of tools for understanding data. The fundamental paradigms of machine learning include:

- *Supervised learning*: The task is to learn a function that maps an input to an output based on example input–output pairs (i.e., providing output labels).
- *Reinforcement learning*: The task is again to learn a mapping function but without explicit input–output examples. Instead, it is based on a reward function that evaluates the performance of the current state and action.
- *Unsupervised learning*: In this case, there are inputs but no supervised output (labels) or reward function. Nevertheless, one can still learn relationships and structure from such data (examples are clustering and the more sophisticated graph learning explored in Chapter 5 with applications for portfolio design in Chapter 12).

16.1.1 Black-Box Modeling

Historically, statistical learning goes back to the introduction of least squares at the beginning of the nineteenth century, what is now called *linear regression*. By the end of the 1970s, many more techniques for learning from data were available, but still they were almost exclusively linear methods, because fitting nonlinear relationships was computationally infeasible at the time. In the mid-1980s, classification and regression trees were introduced as one of the first practical implementations of *nonlinear methods*. Since that time, machine learning has emerged as a new subfield in statistics, focused on supervised and unsupervised modeling and prediction. In recent years, progress in statistical learning has been marked by the increasing availability of powerful and relatively user-friendly software.

The basic idea of ML is to model and learn the input–output relationship or mapping of

some system. Denoting the input or *p features* as $\boldsymbol{x} = (x_1, \ldots, x_p)$ and the output as y, we conjecture that reality can be modeled as the noisy relationship

$$y = f(\boldsymbol{x}) + \epsilon,$$

where f is some unknown function, to be learned, that maps $\boldsymbol{x}$ into y, and ϵ denotes the random noise. This modeling via the function f is what is usually referred to as "black box," in the sense that no attempt is made at understanding the mapping f, it is simply learned.

One of the main reasons to estimate or learn the function f is for prediction, also termed forecasting if the prediction is for a future time: assuming a given input $\boldsymbol{x}$ is available, the output y can be predicted as

$$\hat{y} = f(\boldsymbol{x}).$$

Depending on the nature of the output y, ML systems are classified into

- *regression*: the output is a real-valued number; and
- *classification*: the output can only take discrete values, such as $\{0, 1\}$ or $\{\text{cat}, \text{dog}\}$.

16.1.2 Measuring Performance

In order to learn the function f, we need to be able to evaluate how good a candidate model is. This is conveniently done by defining an error function, also known as a loss (or cost) function. Any appropriate error function for the problem at hand can be used; the two typical choices for error functions are

- *mean squared error* (MSE) for regression:

$$\text{MSE} \triangleq \mathbb{E}\left[(y - \hat{y})^2\right] = \mathbb{E}\left[(y - f(\boldsymbol{x}))^2\right];$$

- *accuracy* (ACC) for classification:[1]

$$\text{ACC} \triangleq \mathbb{E}\left[I\left(y = \hat{y}\right)\right] = \mathbb{E}\left[I\left(y = f(\boldsymbol{x})\right)\right],$$

  where $I\left(y = \hat{y}\right)$ is the indicator function that equals 1 if $y = \hat{y}$ (i.e., correct classification) and zero otherwise (i.e., classification error).

The expectation operator $\mathbb{E}[\cdot]$ is over the distribution of random input–output pairs $(\boldsymbol{x}, y)$. In practice, however, this expectation has to be approximated via the sample mean over some observations. To be specific, in supervised learning, the estimation or learning of the function f is performed based on n observations or data points called *training data*:

$$\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}.$$

The performance or error measure can then be computed using the training data. The hope is that this training error will be close to the true error, but in practice this is not the case since

[1] For classification systems, a large number of performance measures are commonly used, such as accuracy, precision, recall, sensitivity, specificity, miss rate, false discovery rate, among others. All these quantities are easily defined based on the number of true positives, true negatives, false positives, and false negatives. For instance, the accuracy is computed as the sum of true positives and true negatives normalized with the total number of samples.

the learning process can lead to a small training error that is not representative of the actual error, a phenomenon termed *overfitting*. To properly evaluate the performance of a system, new data that has not been used during the learning process has to be employed for testing, termed *test data*, producing the test performance, such as the test MSE or test ACC, which is representative of the true performance. Figure 16.1 illustrates the difference between the training MSE and the test MSE of a system as the complexity of the model f increases (i.e., the number of parameters or degrees of freedom). The divergence between the training MSE and test MSE after some point of complexity implies overfitting: the system is fitting the noise in the training data that is not representative of the test data.

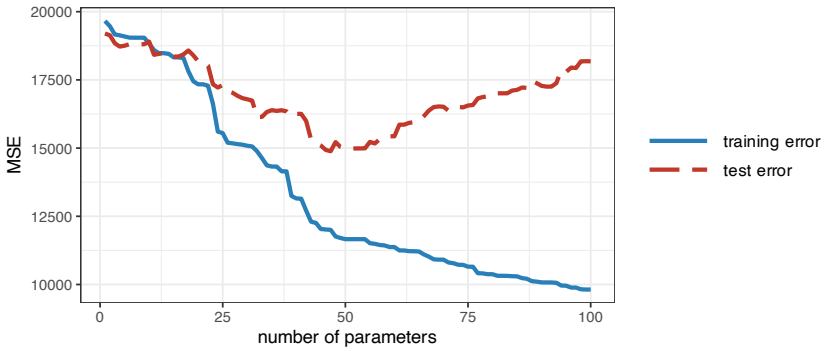


Figure 16.1 Overfitting: training error and test error.

16.1.3 Learning the Model

The black-box model f is learned based on the training data by minimizing the training error or optimizing a performance measure. The specific mechanism by which f is learned depends on the particular black-box model. For example, in artificial neural networks, the training algorithms are variations of stochastic gradient descent (described in Section 16.2).

Supervised learning is used to learn the function f based on input–output pairs (i.e., with labels) in order to minimize the error (e.g., MSE or ACC), as illustrated by the block diagram in Figure 16.2. When explicit labels are not available but the performance of the model can be measured, reinforcement learning can be effectively used, as illustrated by the block diagram in Figure 16.3.

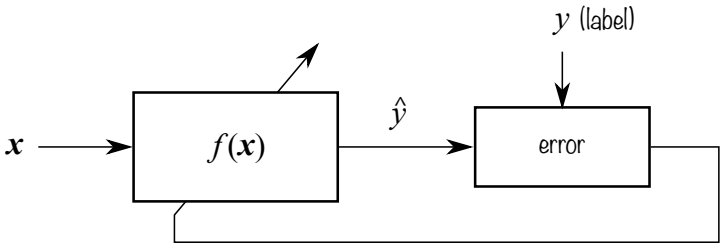


Figure 16.2 Supervised learning in ML via error minimization.

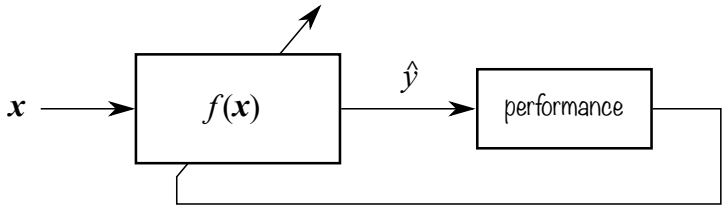


Figure 16.3 Reinforcement learning in ML via performance optimization.

As mentioned before, extreme care has to be taken to avoid overfitting, that is, to avoid fitting the noise in the training data that is not representative of the rest of the data. In practice, this happens when there is too little training data or when the number of parameters (i.e., degrees of freedom) that characterize f is too large, as illustrated in Figure 16.1. To avoid overfitting, two main philosophies have been developed in order to choose an adequate complexity for the model (i.e., the number of degrees of freedom or parameters), termed model assessment (Hastie et al., 2009, Chapter 7):

- *Empirical cross-validation methods*: These simply rely on assessing the performance of a learned model f (estimated from training data) on new data termed *cross-validation data* (note that, once the final model for f has been made, the final performance will be assessed on yet new data termed *test data*).
- *Statistical penalty methods*: To avoid reserving precious data for cross-validation, these methods rely on mathematically derived penalty terms on the degrees of freedom; for example, the Bayesian information criterion (BIC), the minimum description length (MDL), and the Akaike information criterion (AIC).

16.1.4 Types of ML Models

In practice, we cannot handle a totally arbitrary function f in the whole space of possible functions. Instead, we constrain the search to some class of functions f and employ some finite-dimensional parameters, denoted by $\boldsymbol{\theta}$, to conveniently characterize f. For example, linear models correspond to the form $f(\boldsymbol{x}) = \alpha + \boldsymbol{\beta}^{\mathsf{T}}\boldsymbol{x}$, with parameters $\boldsymbol{\theta} = (\alpha, \boldsymbol{\beta})$. Other classes of nonlinear models can adopt more complicated structures, but always with a finite number of parameters.

Over the decades, since the advent of linear regression methods in the 1970s, a plethora of classes of functions have been proposed. The reason it was necessary to introduce so many different statistical learning approaches, rather than just a single best method, is because of the "no free lunch theorem" in statistics: no one method dominates all others over all possible data sets. On a particular data set, one specific method may work best, but some other method may work better on a different data set. Hence it is an important task to decide which method produces the best results for any given set of data. Selecting the best approach can be one of the most challenging parts of performing statistical learning in practice.

Some machine learning methods that have enjoyed success in ML include (Bishop, 2006; Hastie et al., 2009; Shalev-Shwartz & Ben-David, 2014; Vapnik, 1999):

- linear models
- sparse linear models
- decision trees
- k-nearest neighbors
- bagging
- boosting
- random forests (boosting applied to decision trees)
- support vector machines (SVM)
- neural networks, which are the foundation of deep learning.

Interestingly, some of the more complex models, such as random forests and neural networks, are so-called *universal function approximators*, meaning that they are capable of approximating any nonlinear smooth function to any desired accuracy, provided that enough parameters are incorporated.

16.1.5 Applications of ML in Finance

ML can be used in finance in a multitude of ways. In the context of this book, perhaps the two most obvious applications are time series forecasting and portfolio design. However, there are many other aspects where ML can be used, for example, credit risk (Atiya, 2001), sentiment analysis, outlier detection, asset pricing, bet sizing, feature importance, order market execution, big data analysis (López de Prado, 2019), and so on.

López de Prado (2018b) gives a comprehensive treatment of recent machine learning advances in finance, with extensive treatment of the preprocessing and parsing of data from its unstructured form to the appropriate form for standard ML methods to be applied. On a more practical aspect, reasons why most machine learning funds fail are presented in López de Prado (2018a).

In the realm of time series analysis, there exists a large number of publications addressing different aspects. An overview of machine learning techniques for time series forecasting is provided in Ahmed et al. (2010) and Bontempi et al. (2012), while a comparison between support vector machines and neural networks in financial time series is performed in Cao and Tay (2003). Pattern recognition in time series is covered in Esling and Agon (2012) and a comparison of various classifiers for predicting stock market price direction is provided in Ballings et al. (2015).

16.2 Deep Learning

Conventional machine learning methods are limited in their ability to process raw data in the black-box model $f(\cdot)$. For decades, constructing machine-learning systems required careful engineering and considerable domain expertise to transform the raw data into a suitable feature vector $\boldsymbol{x}$ from which the learning subsystem could detect or classify patterns. These are known as *handcrafted features*, in contrast to the *learned features* that a deep architecture can automatically obtain, a process referred to as *representation learning*.

Deep learning, broadly speaking, refers to methods and architectures that can automatically

learn features by means of concatenation of multiple simple – but nonlinear – modules or layers, each of which transforms the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract, level. For example, an image comes in the form of an array of pixel values, and the learned features in the first layer of representation typically represent the presence or absence of edges at particular orientations and locations in the image. The second layer typically detects motifs by spotting particular arrangements of edges. The third layer may assemble motifs into larger combinations that correspond to parts of familiar objects, and subsequent layers would detect objects as combinations of these parts. The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure.

Quoting LeCun et al. (2015):[2]

> Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer.

Deep learning is significantly advancing the solutions to problems that have long challenged the artificial intelligence community. In fact, deep learning has been so successful that it has been referred to with expressions such as "the unreasonable effectiveness of deep learning" and with questions like "Why does deep and cheap learning work so well?" (Lin et al., 2017).

A concise account of deep learning can be found in LeCun et al. (2015), whereas an excellent comprehensive textbook is Goodfellow et al. (2016), and a superb short online introductory book is Nielsen (2015).

16.2.1 Historical Snapshot

Some of the fundamental ingredients of neural networks take us back centuries to 1676 (the chain rule of differential calculus), 1847 (gradient descent), 1951 (stochastic gradient descent), or 1970 (the backpropagation algorithm), to name a few. A detailed historical account of deep learning can be found in Schmidhuber (2015) and Schmidhuber (2022). At the risk of oversimplifying and for illustration purposes, some pivotal historical moments in DL include:

- 1962: Rosenblatt introduces the *multilayer perceptron*;
- 1967: Amari suggests training multilayer perceptrons with many layers via *stochastic gradient descent*;
- 1970: Linnainmaa publishes what is now known as *backpropagation*, the famous algorithm also known as "reverse mode of automatic differentiation" (it would take four decades until it became widely accepted);
- 1974–1980: first major "AI winter" (i.e., period of reduced funding and interest in AI);
- 1987–1993: second major "AI winter";

[2] Yoshua Bengio, Geoffrey Hinton, and Yann LeCun (LeCun et al., 2015) were recipients of the 2018 ACM A. M. Turing Award for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing.

- 1997: LSTM networks introduced (Hochreiter & Schmidhuber, 1997);
- 1998: CNN networks established (LeCun et al., 1998);
- 2010: "AI spring" starts;
- 2012: AlexNet network achieves an error of 15.3% in the ImageNet 2012 Challenge, more than 10.8 percentage points lower than that of the runner up (Krizhevsky et al., 2012);
- 2014: GAN networks established and gained popularity for generating data;
- 2015: AlphaGo by DeepMind beats a professional Go player;
- 2016: Google Translate (originally deployed in 2006) switches to a neural machine translation engine;
- 2017: AlphaZero by DeepMind achieves superhuman level of play in the games of chess, Shogi, and Go;
- 2017: Transformer architecture is proposed based on the self-attention mechanism, which would then become the de facto architecture for most of the subsequent DL systems (Vaswani et al., 2017);
- 2018: GPT-1 (Generative Pre-Trained Transformer) for natural language processing with 117 million parameters, starting a series of advances in the so-called large language models (LLMs);
- 2019: GPT-2 with 1.5 billion parameters;
- 2020: GPT-3 with 175 billion parameters;
- 2022: ChatGPT: a popular chatbot built on GPT-3, astonished the general public, sparking numerous discussions and initiatives centered on AI safety;
- 2023: GPT-4 with ca. 1 trillion parameters (OpenAI, 2023), which allegedly already shows some sparks of artificial general intelligence (AGI) (Bubeck et al., 2023).

As of 2023, the momentum of AI systems based on deep learning is difficult to grasp and it is challenging to keep up with the state of the art. Dozens of startup companies and open-source initiatives are produced by the day, not to mention the astounding number of publications. The pace has become unimaginable and the progress impossible to forecast.

16.2.2 Perceptron and Sigmoid Neuron

The *perceptron*[3] is a function that maps its input vector $\boldsymbol{x}$ to a binary output value according to

$$f(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{w}^\mathsf{T}\boldsymbol{x} + b \geq 0, \\ 0 & \text{otherwise,} \end{cases}$$

where $\boldsymbol{w}$ is a vector of weights and b is the bias. In other words, this function is the composition of the affine function $\boldsymbol{w}^\mathsf{T}\boldsymbol{x}+b$ with the nonlinear binary step function (also called the Heaviside function) $H(z)$ defined as 1 for $z \geq 0$ and 0 otherwise, that is, $f(\boldsymbol{x}) = H(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}+b)$ (alternatively, with the indicator function $I(\cdot)$, we can write $f(\boldsymbol{x}) = I(\boldsymbol{w}^\mathsf{T}\boldsymbol{x} + b \geq 0)$). The weights give different importance to the inputs and the bias is equivalent to having a nonzero activation threshold. This is a minimal approximation of how a biological neuron works.

Sigmoid neurons are similar to perceptrons, but modified so that small changes in their weights

[3] Perceptrons were developed in the 1950s and 1960s by the scientist Frank Rosenblatt, inspired by earlier work by Warren McCulloch and Walter Pitts.

and bias cause only a small change in their output. That is the crucial fact that will allow a network of sigmoid neurons to learn. A sigmoid neuron is defined, similarly to a perceptron, as

$$f(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\mathsf{T}\boldsymbol{x} + b),$$

where σ is the *sigmoid function* defined as $\sigma(z) = 1/(1+e^{-z})$. Interestingly, when z is a large positive number, then $e^{-z} \approx 0$ and $\sigma(z) \approx 1$, and when z is very negative, then $e^{-z} \to \infty$ and $\sigma(z) \approx 0$; this resembles the behavior of the perceptron.

Both the Heaviside function $H(z)$ (also known as a step function) and the sigmoid function $\sigma(z)$ are types of nonlinear activation functions. These nonlinearities are key components in neural networks. Otherwise, the input–output relationship would simply be linear. Figure 16.4 compares the nonlinear activation functions for the perceptron (i.e., the step function $H(z)$) and the sigmoid neuron (i.e., the sigmoid function $\sigma(z)$), as well as the popular ReLU function $\text{ReLU}(z) = \max(0, z)$ to be discussed later.

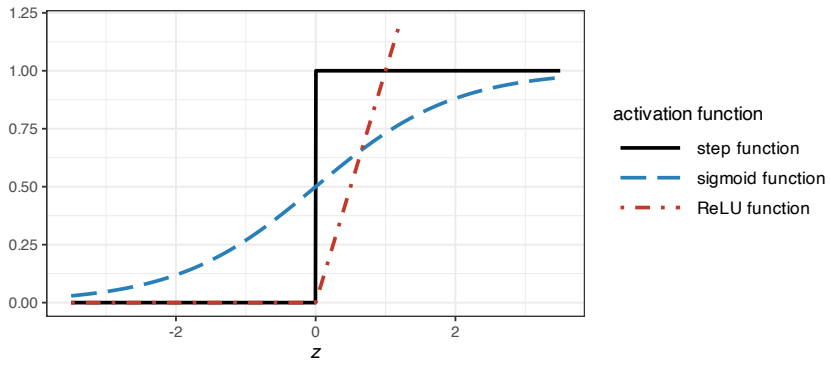


Figure 16.4 Activation functions: step function (for the perceptron), sigmoid function (for the sigmoid neuron), and ReLU (popular in neural networks).

16.2.3 Neural Networks

Perceptrons can be combined in multiple layers, leading to what is referred to as *multilayer perceptron* (MLP). In this way, perceptrons in subsequent layers can make decisions at a more complex and more abstract level than perceptrons in the first layer. In fact, MLPs are universal function approximators (they can approximate arbitrary functions as well as desired).

A *neural network* is simply several layers of neurons of any type (in fact, with some abuse of terminology they are also referred to as MLPs). The leftmost layer is called the *input layer* and it simply contains the input vector $\boldsymbol{x}$ (it is not an operational layer per se). After that come the *hidden layers*. Finally, the rightmost layer is called the *output layer* and contains the output neurons. Figure 16.5 shows a four-layer network with two hidden layers.

The goal of a neural network is to approximate some (possibly vector-valued) function $\boldsymbol{f}$. Neural networks are often referred to as *feedforward neural networks* to emphasize the fact that information flows from the input $\boldsymbol{x}$, through the intermediate layers, to the output $\boldsymbol{y}$,

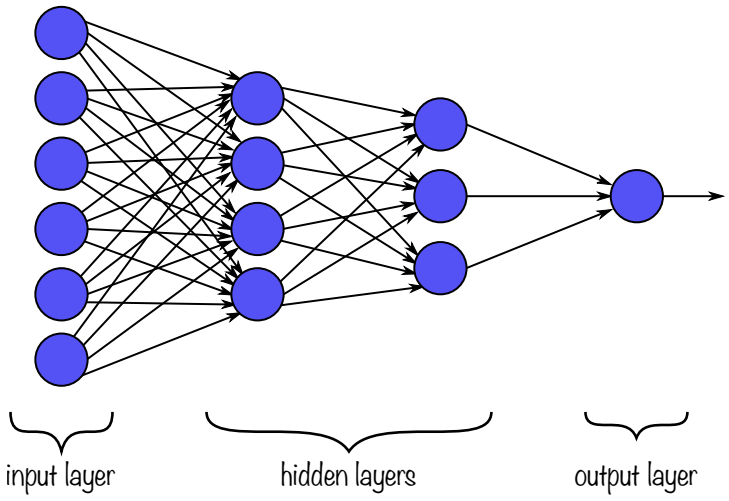


Figure 16.5 Example of a multilayer perceptron with two hidden layers.

without feedback connections in which outputs of the model are fed back into itself. When feedback connections are included, they are called *recurrent neural networks*, as presented later.

When the number of layers, called the depth of the model, is large enough, the network is referred to as *deep*, leading to the so-called *deep neural network*, as well as the mouthful of a name "deep feedforward neural network."

Mathematically, each layer i can be thought of as implementing a vector function $\boldsymbol{f}^{(i)}$, leading to a connected chain of functions (i.e., composition of functions), conveniently written as

$$\boldsymbol{f} = \boldsymbol{f}^{(n)} \circ \cdots \circ \boldsymbol{f}^{(2)} \circ \boldsymbol{f}^{(1)},$$

where $\circ$ denotes function composition. Each hidden layer of the network is typically vector valued, with their dimensionality determining the width of the model. In particular, each layer i produces an intermediate vector $\boldsymbol{h}^{(i)}$ from the previous vector $\boldsymbol{h}^{(i-1)}$ (with $\boldsymbol{h}^{(0)} \triangleq \boldsymbol{x}$) as

$$\boldsymbol{h}^{(i)} = \boldsymbol{f}^{(i)}\left(\boldsymbol{h}^{(i-1)}\right) = \boldsymbol{g}^{(i)}\left(\boldsymbol{W}^{(i)}\boldsymbol{h}^{(i-1)} + \boldsymbol{b}^{(i)}\right),$$

where $\boldsymbol{f}^{(i)}$ is the composition of an affine function with the elementwise nonlinear activation function $\boldsymbol{g}^{(i)}$. That is, each layer implements a function that is simply an affine function composed with a nonlinear activation function, although other operators are also used such as the max-pooling described later. Common elementwise nonlinear activation functions include:

- the *sigmoid function*:

$$\sigma(z) = \frac{1}{1 + e^{-z}};$$

- the *hyperbolic tangent*:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}};$$

- the popular *rectified linear unit* (ReLU) function:

$$\mathrm{ReLU}(z) = \max(0, z),$$

  which typically learns much faster in networks with many layers.

The output layer in classification problems typically employs the so-called *softmax function*,

$$\mathrm{softmax}(\boldsymbol{z}) = \frac{e^{\boldsymbol{z}}}{\mathbf{1}^\mathsf{T} e^{\boldsymbol{z}}}, \tag{16.1}$$

where the exponentiation ensures the outputs are nonnegative and the normalization ensures they sum to one, that is, the output is effectively a probability mass function (in classification problems, each output value denotes the probability of each class). In regression problems, the output layer is typically a simple affine mapping without an activation function, that is, $\boldsymbol{g}(\boldsymbol{z}) = \boldsymbol{z}$.

16.2.4 Learning via Backpropagation

As mentioned earlier in Section 16.1, supervised learning involves developing a black-box model by training the system with data and minimizing an error function. This is achieved by adjusting specific parameters, commonly referred to as weights, which act as "knobs" determining the input–output function, as demonstrated in Figure 16.2. In a typical deep learning system, there may be hundreds of millions (or even billions) of these adjustable weights, and hundreds of millions of labeled examples with which to train the machine.

A conceptually simple way to learn the system is based on the gradient method. To adjust the weight vector $\boldsymbol{w}$, the learning algorithm computes the gradient vector of the error function $\xi(\boldsymbol{w})$, that is, $\partial\xi/\partial\boldsymbol{w}$, which indicates by what amount the error would increase or decrease if the weights were increased by a tiny amount. The weight vector is then adjusted in the opposite direction to the gradient vector to minimize the error or cost function:

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - \kappa \frac{\partial \xi}{\partial \boldsymbol{w}},$$

where κ is the so-called *learning rate*.

The error or cost function is typically defined via a mathematical expectation over the distribution of the possible input–output pairs. In practice, it is not possible to evaluate such an expectation operator and one has to resort to a procedure called *stochastic gradient descent* (SGD). This process involves presenting the input vector for several examples, computing the outputs and errors, determining the average gradient for those examples, and adjusting the weights accordingly. The procedure is repeated for numerous small sets of examples from the training set until the average of the objective function ceases to decrease. It is referred to as stochastic because each small set of examples provides a noisy estimate of the average gradient across all examples. This straightforward process typically identifies a good set of weights faster in comparison to more complex optimization methods. In multilayer architectures, one

can compute gradients using the *backpropagation* procedure, which is nothing more than a practical implementation of the chain rule for derivatives (this method was discovered independently by several different groups during the 1970s and 1980s).

In the late 1990s, neural nets and backpropagation were largely forsaken by the machine learning community and ignored by the computer vision and speech recognition communities. In particular, it was commonly thought that simple gradient descent would get trapped in poor local minima (weight configurations for which no small change would reduce the average error). In practice, however, this is rarely a problem with large networks. Regardless of the initial conditions, the system nearly always reaches solutions of very similar quality. Recent theoretical and empirical results strongly suggest that local minima are not a serious issue in general.

16.2.5 Deep Learning Architectures

Research in DL is extremely vibrant and new architectures are constantly being explored by practitioners and academics. In the following we describe some of the most relevant paradigms.

Fully-Connected Neural Networks

The neural networks previously introduced are actually *fully connected neural networks* in the sense that each neuron takes as inputs all the outputs from the previous layer and combines them with weights. This rapidly results in a significant increase in the number of weights to be trained as illustrated in Figure 16.6, which shows a very simple MLP with just five hidden layers.

To decrease the number of weights, it is necessary to incorporate a meaningful structure into the network, tailored to the specific application being addressed. By reducing the number of weights per layer, we can have many layers to express computationally large models, producing high levels of abstraction, while keeping the number of actual parameters manageable.

Convolutional Neural Networks (CNNs)

One popular example in the history of deep learning is that of *convolutional neural networks* (CNNs), based on the concept of convolution commonly used in signal processing. This network achieved many practical successes during the period when neural networks were out of favor and it has been widely adopted by the computer-vision community. The origins of CNNs go back to the 1970s, but the seminal paper establishing the modern subject of convolutional networks was LeCun et al. (1998), where the architecture "LeNet-5" was proposed consisting of seven layers. Another important achievement was in 2012, with the "AlexNet" architecture (Krizhevsky et al., 2012) that blew existing image classification results out of the water.

The concept of CNNs originated from image processing, where the input is a two-dimensional image, and it makes sense for pixels to be processed only with their nearby pixels, rather than distant ones, as demonstrated in Figure 16.7. Furthermore, the weights are shifted across the image, allowing them to be shared and reused by all neurons in the hidden layer.

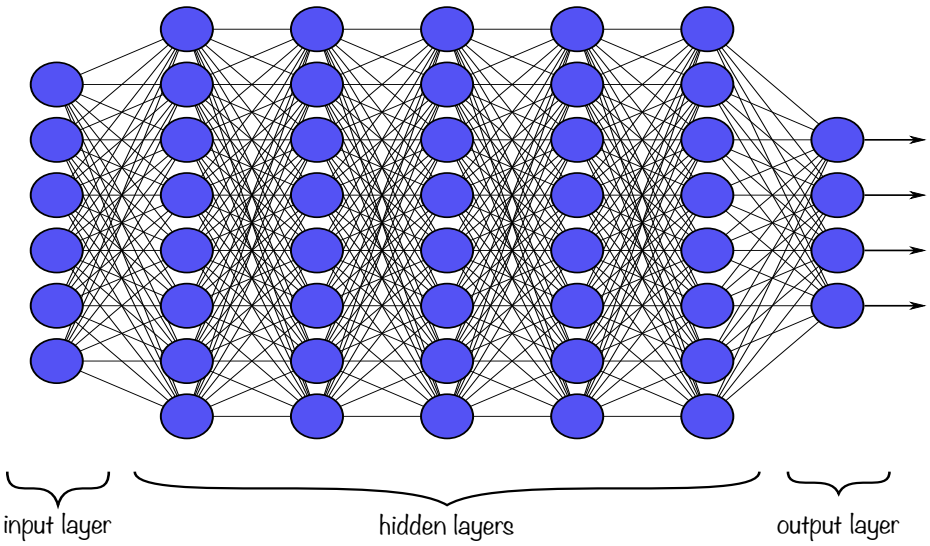


Figure 16.6 Large number of weights in a simple multilayer perceptron with five hidden layers.

This introduces structure in the matrix $\boldsymbol{W}^{(i)}$ found in the affine mapping $\boldsymbol{W}^{(i)}\boldsymbol{h}^{(i-1)} + \boldsymbol{b}^{(i)}$. Specifically, the matrix $\boldsymbol{W}^{(i)}$ will be highly sparse (containing numerous zeros), and the nonzero elements will be repeated multiple times.

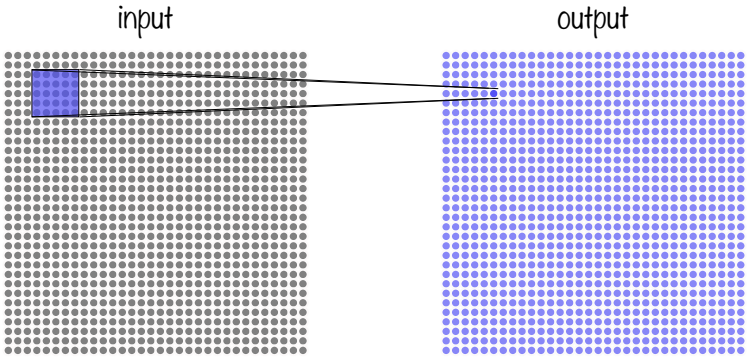


Figure 16.7 CNN filtering layer.

For instance, imagine having a 100×100 image, which corresponds to a 10 000-dimensional input vector. If the first hidden layer has the same size (i.e., the same number of neurons), a fully connected approach would require 10^8 weights. In contrast, a CNN architecture would only need the coefficients of a 5×5, that is, 25 weights plus one for the bias. Of course, we

cannot really do a direct comparison between the number of parameters, since the two models are different in essential ways. But, intuitively, it seems likely that the use of translation invariance by the convolutional layer will significantly reduce the number of parameters needed to achieve performance similar to the fully-connected model. That, in turn, will result in faster training for the convolutional model and, ultimately, will help us build deep networks using convolutional layers.

To reduce network complexity, CNNs use different stride lengths (when shifting the filter) and incorporate pooling layers, such as max-pooling, after convolutional layers to condense feature maps and retain information about the presence of features without precise location details.

An interesting extension of CNNs, which are based on processing neighboring pixels, is that of *graph CNNs* (Scarselli et al., 2009), where the concept of neighborhood is generalized and indicated with a connectivity graph on the input elements (see Chapter 5 for details on graphs).

Recursive Neural Networks (RNNs)

Feedforward neural networks produce an output that solely depends on the current input; they do not have internal memory. *Recursive neural networks* (RNNs), on the other hand, are neural networks with loops in them, allowing information to persist, that is, to have memory.

Figure 16.8 depicts an RNN layer, with an internal loop that allows the implementation of a function of all previous inputs $\boldsymbol{f}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$.

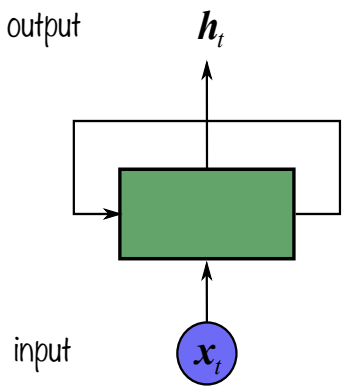


Figure 16.8 RNN layer with a loop.

RNNs are appealing because they have the potential to link past information to current tasks. In instances where the gap between relevant information and its required location is small, RNNs can effectively learn to utilize past data. Nevertheless, when this gap widens significantly, standard RNNs struggle to learn how to connect the information. In theory, RNNs are fully capable of managing long-term dependencies. Yet, in practice, they often struggle to learn them due to the vanishing gradient problem during training. This issue has been addressed by introducing a specific RNN structure called LSTM.

Long Short-Term Memory (LSTM) Networks

Long short-term memory (LSTM) networks, a unique type of RNN capable of learning long-term dependencies, were introduced in Hochreiter and Schmidhuber (1997) and later refined and popularized in subsequent works. They have demonstrated remarkable success in various memory-dependent tasks, including natural language processing and time series analysis.

LSTMs are specifically engineered to tackle the long-term dependency issue. Rather than employing a basic feedback mechanism like vanilla RNNs, they utilize a complex structure composed of four interconnected sub-modules (Hochreiter & Schmidhuber, 1997), resulting in a more effective learning process compared to other RNNs.

Transformers

The *transformer* architecture, introduced in Vaswani et al. (2017), is a groundbreaking neural network design that revolutionized natural language processing tasks. Unlike CNNs and RNNs, transformers rely on self-attention mechanisms to process input sequences simultaneously, rather than sequentially like in RNNs. Transformers have arguably become the de facto universal architecture able to outperform existing architectures in most applications.

CNNs are adept at handling spatial data like images, while RNNs process sequential data but struggle with vanishing and exploding gradient issues. Transformers overcome these limitations by employing self-attention to weigh the importance of input elements, enabling parallel processing, faster training, and improved handling of long-range dependencies, along with position encoding to incorporate positional information.

The relevance of transformers stems from their exceptional performance on a wide range of tasks, such as machine translation, text summarization, and sentiment analysis. They form the basis of state-of-the-art models like GPT (OpenAI, 2023), which have achieved top results on benchmarks and enabled new applications, including conversational AI, automated content generation, and advanced language understanding.

Without going into the details of the architecture, it is worth a brief look at this self-attention mechanism that makes transformers unique. The idea is to present the network with all the inputs at once and let the network decide which parts of the input should influence other parts in an automatic way. Suppose that we have n inputs, each of dimension d, arranged along the columns of the $n \times d$ matrix $\boldsymbol{V}$. The goal is to substitute each row of $\boldsymbol{V}$ by a proper linear weighted combination of all the rows, where the weights have to be calculated so that some inputs can influence other inputs in a precise manner. The way the weights are computed in a transformer is similar to the way "keys" in a database are "queried"; that is, by using a "query" matrix $\boldsymbol{Q}$ and a "key" matrix $\boldsymbol{K}$ (of the same dimension as $\boldsymbol{V}$), we can compute the inner product of the rows of $\boldsymbol{Q}$ and the rows of $\boldsymbol{K}$ to get a similarity matrix $\boldsymbol{Q}\boldsymbol{K}^{\mathsf{T}}$. At this point, we could use this similarity matrix as the weights; however, it is convenient to scale this matrix with the dimension of the keys $\sqrt{d_k}$ and then normalize the rows so that they are nonnegative numbers with a normalized sum via the softmax operator in (16.1). Putting it all together leads to the popular expression for the so-called *scaled dot-product attention* (Vaswani et al.,

2017),

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{d_k}}\right)\boldsymbol{V},$$

which is represented in Figure 16.9. In practice, the three matrices $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are obtained as linear transformations of the inputs. Typically, multiple self-attention mechanisms are used in parallel.

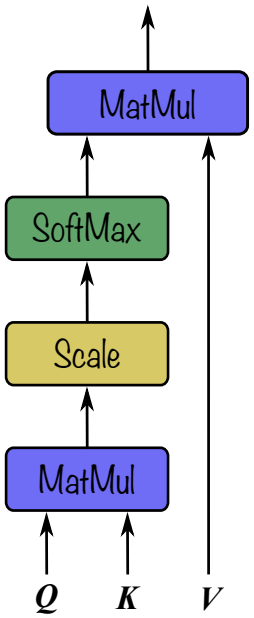


Figure 16.9 Self-attention mechanism (scaled dot-product attention).

Autoencoder Networks

Autoencoder networks are commonly used in DL models to learn data representation via feature extraction and dimensionality reduction (Kramer, 1991). In other words, autoencoder networks perform an unsupervised feature learning process.

The architecture of an autoencoder consists, as usual, of an input layer, one or more hidden layers, and an output layer. More specifically, autoencoder networks have a symmetrical structure separated into the encoder and the decoder, with the same number of nodes in the input and output layers, and a bottleneck at the core called *code* or *latent features*, as illustrated in Figure 16.10. The network is trained so that the output is as close as possible to the input, therefore forcing the central bottleneck to condense the information, performing feature extraction in an unsupervised way.

Generative Adversarial Networks (GANs)

Generative adversarial networks (GANs), developed in Goodfellow et al. (2014), are a type of deep learning architecture that consists of two adversarial neural networks: a generator and

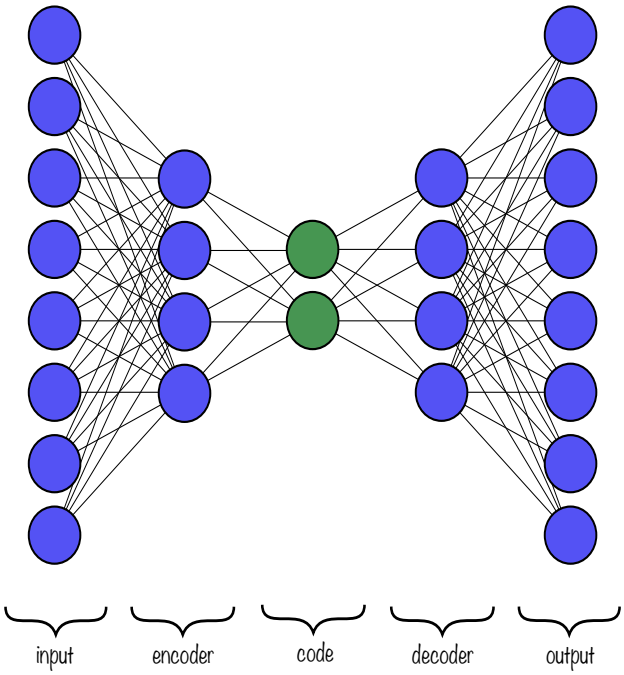


Figure 16.10 Autoencoder structure.

a discriminator. The goal of the generator is to produce realistic data, such as images or text, while the discriminator's role is to distinguish between real and fake data.

The generator and discriminator are trained simultaneously. During training, the generator creates synthetic data and presents it to the discriminator. The discriminator then evaluates whether the data is real or fake and provides feedback to the generator. Based on this feedback, the generator modifies its output to create more realistic data. This process continues until the generator produces data that is indistinguishable from real data, making it difficult for the discriminator to identify which data is real or fake.

GANs have been successfully used in a variety of applications, such as image generation, text-to-image synthesis, and even generating realistic music. In finance, GANs can be employed to generate artificial time series with asset prices for backtesting and stress testing purposes (Takahashi et al., 2019; Yoon et al., 2019).

Diffusion Models

Diffusion models are another type of generative model in deep learning that use a diffusion process to generate samples from a target distribution. They were first introduced in Sohl-Dickstein et al. (2015) but remained behind the curtains for a while and did not gain popularity until the 2020s (Ho et al., 2020; Song & Ermon, 2019).

The idea is very different from the two adversarial networks (the generator and the discriminator) of GANs. Diffusion models iteratively transform an initial noise signal using a series of learnable transformations, such as neural networks, to generate samples that resemble the target distribution. At each iteration step, the model estimates the conditional distribution of the data given the current level of noise.

Both diffusion models and GANs are generative models that can be used to generate high-quality samples from complex distributions. However, diffusion models have some advantages over GANs, such as being more stable during training and not suffering from mode collapse, which is a problem where the generator produces only a small subset of the possible samples. On the other hand, GANs are more flexible and can generate a wider variety of samples, including those that are not present in the training data.

16.2.6 Applications of Deep Learning in Finance

Essentially, all the financial applications discussed in Section 16.1.5 using machine learning can also be addressed with neural networks (which are a type of ML). However, deep learning involves deep neural networks, meaning networks with many layers. With a deep architecture, there are many weights or parameters to learn, requiring a large training dataset. While abundant data in areas involving images, text, or speech is not an issue, it can be problematic in finance. Therefore, it is best to focus on financial applications with access to large datasets.

As previously described, DL has already been successfully employed in many other areas. The financial area is starting to get traction; in fact, the field is wide open and many research opportunities still exist. A comprehensive state-of-the-art snapshot (as of 2020) of the DL models developed for financial applications is provided in Ozbayoglu et al. (2020), where 144 papers are categorized according to their intended subfield in finance and also analyzed based on their DL models.

Some of the areas in finance where DL is currently being researched include financial time series forecasting, algorithmic trading (a.k.a. *algo trading*), risk assessment (e.g., bankruptcy prediction, credit scoring, bond rating, and mortgage risk), fraud detection (e.g., credit card fraud, money laundering, and tax evasion), portfolio management, asset pricing and derivatives markets (options, futures, forward contracts), cryptocurrency and blockchain studies, financial sentiment analysis and behavioral finance, and financial text mining (Ozbayoglu et al., 2020).

Nevertheless, the most widely studied financial application area for DL is forecasting of financial time series, particularly asset price forecasting. Even though some variations exist, the main focus is on predicting the next movement of the underlying asset. More than half of the existing implementations of DL are focused on this direction. Even though there are several subtopics of this general problem, including stock price forecasting, index prediction, forex price prediction, commodity price prediction, bond price forecasting, volatility forecasting, and cryptocurrency price forecasting, the underlying dynamics are the same in all of these applications. The majority of the DL applications for financial time series have appeared quite recently, from 2015 on, as described in the comprehensive survey (as of 2020) Sezer et al. (2020), where 140 papers are classified.

16.3 Deep Learning for Portfolio Design

In the context of portfolio design, deep learning can be used in a variety of ways. Recall that the two main components in portfolio design are data modeling and portfolio optimization. This is depicted in Figure 1.3 (Chapter 1) and reproduced herein for convenience in Figure 16.11.

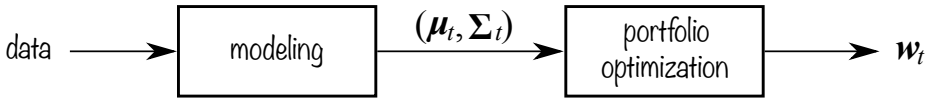


Figure 16.11 Block diagram of data modeling and portfolio optimization.

In light of the block diagram in Figure 16.11, one could envision the usage of DL in at least three ways:

- using DL only in the modeling or time series forecasting component, while keeping the traditional portfolio optimization part;
- using DL only in the portfolio component, while keeping the traditional data modeling part; and
- using DL for both components, what is called *end-to-end* modeling.

We will not consider further the option of using DL only for the optimization part, since that is a well-understood component that does not seem to require DL (in fact, this book has explored a wide variety of different portfolio formulations with efficient algorithms). Thus, we will focus on employing DL either in the forecast component or in the end-to-end system.

Regarding the input data to the DL system, one can use raw time series data, such as price data (e.g., open, high, low, close) and volume, as well as other sources of data derived from technical analysis, fundamental analysis, macroeconomic data, financial statements, news, social media feeds, and investor sentiment analysis. Also, depending on the time horizon, a wide range of options for the frequency of the data may be available, varying from *high-frequency data* and intraday price movements to daily, weekly, or even monthly stock prices.

16.3.1 Challenges

Before we explore the possibilities of DL for portfolio design, it is important to highlight the main challenges faced in this particular area. As already explained, deep neural networks have demonstrated outstanding performance in many domain-specific areas, such as image recognition, natural language processing, board and video games, biomedical applications, self-driving cars, and so on. The million-dollar question is whether this revolution will extend to financial systems.

Since the 2010s, the financial industry and academia have been exploring the potential of DL in various applications, such as financial time series forecasting, algorithmic trading, risk assessment, fraud detection, portfolio management, asset pricing, derivatives markets, cryptocurrency and blockchain studies, financial sentiment analysis, behavioral finance, and financial text mining. The number of research works keeps on increasing every year in an

accelerated fashion, as well as open-source software libraries. However, we are just in the initial years of this new era and it is too early to say whether the success of DL enjoyed in non-financial applications will actually extend to financial systems and, particularly, to portfolio design.

Apart from very specific financial applications that have already enjoyed some success, such as sentiment analysis of news, credit default detection, or satellite image analysis for stock level estimation or crop production, we now focus on the potential of deep neural networks specifically for financial time series modeling and portfolio design. Among the many possible challenges that set these problems apart from other successful applications, the following are definitely worth mentioning:

- *Data scarcity*: Compared to other areas, such as natural language processing (e.g., GPT-3 was trained on a massive dataset of over 570 GB of text data), financial time series are in general extremely scarce (except for high-frequency data). For example, two years of daily stock prices amount to just 504 observations.

- *Low signal-to-noise ratio*: The signal in financial data is extremely weak and totally submerged in noise. For example, an exploratory data analysis on asset returns corrected for the volatility envelope reveals a time series with little temporal structure (see Figures 2.23–2.24 in Chapter 2). This is very different from other applications, for example, an image of a cat typically has a high signal and very small noise (this is not to say that recognizing a cat is easy, but at least the signal-to-noise ratio is large).

- *Data nonstationarity*: Financial time series are clearly nonstationary (see Chapter 2) with a statistical distribution that changes over time (e.g., bull markets, bear markets, side markets). This is in sharp contrast with most other applications where DL has succeeded, in which the distribution remains constant: a cat stays the same, be it yesterday, today, or tomorrow.

- *Data adaptive feedback loop*: Data from financial markets is totally influenced by human and machine decisions based on previous data. As a consequence, there exists a very unique feedback loop mechanism that cannot be ignored. In particular, once a pattern is discovered and a trading strategy is designed, this pattern tends to disappear in future data. Again, this is extremely different from other applications; for example, a cat remains a cat regardless of whether one can detect it in an image.

- *Lack of prior human evidence*: In most areas where DL has been successful, there was obvious prior evidence of human performance that showed that the problem was solvable. For example, humans can easily recognize a cat, translate a sentence from English to Spanish, or drive a car. However, in finance there is no human who can effectively forecast the future performance of companies or trade a portfolio. Simply recall (see Chapter 13) the illustrative and clarifying statement (Malkiel, 1973): "a blindfolded chimpanzee throwing darts at the stock listings can select a portfolio that performs as well as those managed by the experts."

At the risk of oversimplifying, we could make a simple analogy of the problem of financial time series forecasting or portfolio design to that of identifying an octopus in an image, as



Cat Octopus

Figure 16.12 Can you spot the cat? And the octopus? Financial data is more like an octopus.[5]

opposed to the iconic example of identifying a cat. This is exemplified in Figure 16.12. Indeed, this analogy seems to fit the previous list of challenges, namely:

- *Data scarcity*: Arguably there are more images of cats than octopi in the human library of photos.
- *Low signal-to-noise ratio*: Think of an octopus that has camouflaged to look exactly like the background (the octopus creates this noise to blend in) as opposed to a domestic cat that stands out.
- *Data nonstationarity*: Think again of an octopus that changes its camouflage over time to match the background (a cat's appearance is the same today as it was yesterday).[6]
- *Data adaptive feedback loop*: Think once more of an octopus that quickly adapts its camouflage as it is being chased by a predator (a cat is a cat).[7]
- *Lack of prior human evidence*: Humans are good at spotting domestic cats, but the same cannot be said about octopi.

We can finally summarize the previous analogy[8] by saying that "financial data ain't cats, but octopi."

16.3.2 Standard Time Series Forecasting

By far the most common approach to employ DL in portfolio design is by using it in the time series modeling or forecasting component. This area has been intensively explored since 2015, as described in Sezer et al. (2020). LSTM, by its very nature, utilizes the temporal characteristics of any time series signal due to its inherent memory. Thus, LSTM and its

[5] Photo credits – Cat: kelvinjay/E+/Getty Images. Octopus: Andrey Nekrasov/imageBROKER/Getty Images.
[6] Cats, like all living creatures, do evolve, but they do so on an evolutionary time scale of, say, millions of years. So, for practical purposes we can assume them fixed.
[7] Cats, like most animals, have evolved *camouflage* to avoid predators, but cannot adapt it to the changing environment in real time, unlike other species like octopus, squid, and chameleon.
[8] The cat vs. octopus comparison is just an analogy for illustration purposes. This is not to say that DL cannot literally be trained to spot an octopus.

variations initially dominated the financial time series forecasting domain (Fischer & Krauss, 2018). Nevertheless, more recently transformers have been shown to deal with long-term memory more efficiently.

The block diagram in Figure 16.13 illustrates the general process of time series forecasting. Following the supervised learning paradigm in Figure 16.2, the input consists of a lookback of the past k time series values $(\boldsymbol{x}_{t-k}, \ldots, \boldsymbol{x}_{t-1})$, the desired output or label is the next value of the time series $\boldsymbol{x}_t$, and the output produced (i.e., the forecast) is denoted by $\boldsymbol{\mu}_t$. With this, we can define some error measure between $\boldsymbol{\mu}_t$ and $\boldsymbol{x}_t$ to drive the learning process of the deep learning network. Note that the forecast horizon could be chosen further into the future instead of being just the next time index t.

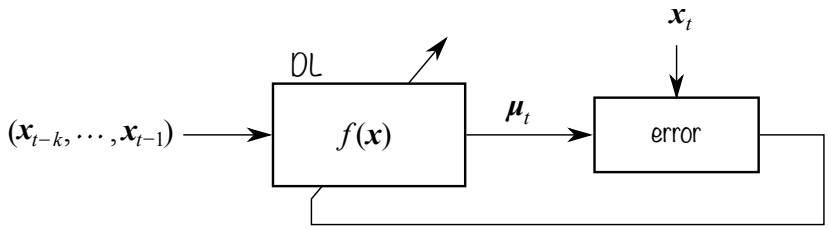


Figure 16.13 Block diagram of standard time series forecasting via DL.

The error measure that drives the learning process can be measured in a variety of ways. In a regression setting, the forecast value is a number or vector of values. We can then define the error vector $\boldsymbol{e}_t = \boldsymbol{\mu}_t - \boldsymbol{x}_t$ and then compute quantities such as the mean square error (MSE), mean absolute error (MAE), median absolute deviation (MAD), mean absolute percentage error (MAPE), and so on. In a classification setting, the forecast is the trend, for example up/down, and typical measures of error are the accuracy (i.e., correct prediction over total predictions), error rate (i.e., wrong predictions over total predictions), cross-entropy, and so on. See Goodfellow et al. (2016) for details.

Mathematically, the DL network implements the function $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-k}, \ldots, \boldsymbol{x}_{t-1})$, with parameters $\boldsymbol{\theta}$, to produce the estimate of $\boldsymbol{x}_t$ as $\boldsymbol{\mu}_t = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-k}, \ldots, \boldsymbol{x}_{t-1})$. The mathematical formulation of a standard time series forecast can be written as the optimization problem

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \mathbb{E}\left[\mathcal{L}\left(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-k}, \ldots, \boldsymbol{x}_{t-1}), \boldsymbol{x}_t\right)\right],$$

where $\mathcal{L}(\cdot, \cdot)$ denotes the loss function or prediction error function to be minimized (e.g., the MSE or cross-entropy).

It is important to point out that this architecture focuses on the time series modeling only, while totally ignoring the subsequent portfolio optimization component, which can also be taken into account as described next.

16.3.3 Portfolio-Based Time Series Forecasting

The previous standard time series model totally ignores the subsequent portfolio optimization component. As a consequence, the performance measure has to be defined in terms of an

error that depends on the forecast $\boldsymbol{\mu}_t$ and the label $\boldsymbol{x}_t$. However, determining the most suitable error definition for the following portfolio optimization step is unclear and the choice is more heuristic.

Alternatively, a more holistic approach is to take into account the portfolio optimization component to measure the overall performance in a meaningful way, so that we do not need to rely on a rather arbitrary error definition.

The block diagram in Figure 16.14 illustrates this process of time series forecasting taking into account the subsequent portfolio optimization block in the training procedure (Bengio, 1997). Following the reinforcement learning paradigm in Figure 16.3, instead of measuring an arbitrary error based on $\boldsymbol{\mu}_t$ and $\boldsymbol{x}_t$ to drive the learning process, the output $\boldsymbol{\mu}_t$ is fed into the subsequent portfolio optimization block to produce the portfolio $\boldsymbol{w}_t$, from which a meaningful measure of performance can be evaluated, such as the Sharpe ratio.

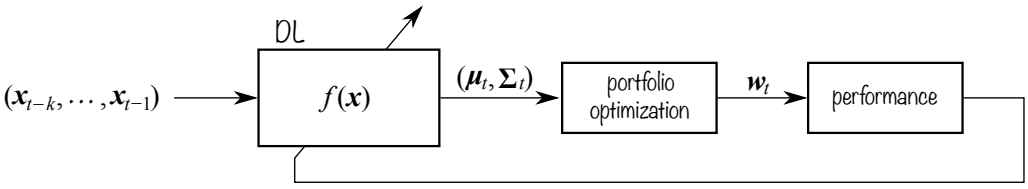


Figure 16.14 Block diagram of portfolio-based time series forecasting via DL.

Mathematically, the DL network implements the function $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-k}, \ldots, \boldsymbol{x}_{t-1})$, with parameters $\boldsymbol{\theta}$, to produce the estimate of $\boldsymbol{x}_t$ as $\boldsymbol{\mu}_t = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-k}, \ldots, \boldsymbol{x}_{t-1})$ (possibly also the corresponding covariance matrix $\boldsymbol{\Sigma}_t$), from which the portfolio $\boldsymbol{w}_t$ will be designed by minimizing some objective function $f_0(\cdot)$ (following any of the portfolio formulation designs covered in this book). The mathematical formulation of a portfolio-based time series forecasting can be written as the optimization problem

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\text{minimize}} \quad & \mathbb{E}\left[\xi\left(\boldsymbol{w}_t, \boldsymbol{x}_t\right)\right] \\ \text{subject to} \quad & \boldsymbol{w}_t = \arg\min_{\boldsymbol{w}} f_0\left(\boldsymbol{w}; \boldsymbol{\mu}_t = \boldsymbol{f}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-k}, \ldots, \boldsymbol{x}_{t-1}\right)\right), \end{aligned}$$

where $\xi(\cdot, \cdot)$ denotes the error function to be minimized that measures the overall system performance (e.g., the negative of the Sharpe ratio). Note that in this approach, the parameters of the DL network $\boldsymbol{\theta}$ are optimized to directly minimize the overall system performance instead of a simple forecasting error, such as the MSE or the cross-entropy. In principle, one may use $f_0 = \xi$, that is, use the same criterion to design the portfolio as used to measure the overall performance; however, there may be reasons to actually use a different criterion.

The difficulty of this architecture is in the learning process. To be more specific, the backpropagation learning algorithm requires the computation of the partial derivatives of the output of each block with respect to its input (to be used in the chain rule for differentiation). If the portfolio optimization block has a closed-form expression, for example $\boldsymbol{w}_t = \boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}_t$, then the partial derivatives are trivially computed. However, if this block is defined in terms of the solution to an optimization problem, then it becomes trickier since one has to be able to compute the partial derivatives of the solution via the Karush–Kuhn–Tucker optimality

conditions of the optimization problem (see Section A.6.4 in Appendix A). Fortunately, recent developments have made this possible and are available in open-source libraries (Amos & Kolter, 2017).

It is important to note that in this architecture, the time series forecast not only produces the forecast vector $\boldsymbol{\mu}_t$ but also a measure of the uncertainty of the forecast in the form of the covariance matrix $\boldsymbol{\Sigma}_t$. This is necessary since the subsequent portfolio optimization component may need both $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$.

16.3.4 End-to-End Portfolio Design

The DL portfolio-based time series forecasting architecture in Figure 16.14 is an improvement over the standard time series forecasting architecture in Figure16.13, because it takes into account the subsequent portfolio optimization block and measures the overall performance using a meaningful performance measure.

However, since DL has proven to be such a powerful universal function approximator in many other areas, we can also consider a bolder architecture commonly termed *end-to-end* design, where the whole process is modeled by a single DL component as illustrated in Figure 16.15.

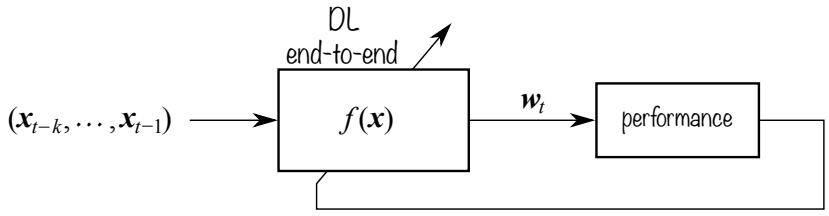


Figure 16.15 Block diagram of end-to-end portfolio design via DL.

Mathematically, the end-to-end DL network implements the function $\boldsymbol{f}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-k}, \ldots, \boldsymbol{x}_{t-1}\right)$, with parameters $\boldsymbol{\theta}$, to directly produce the portfolio $\boldsymbol{w}_t$ (without going through an intermediate forecasting block). The formulation of this end-to-end DL portfolio can be written as

$$\begin{array}{ll} \underset{\boldsymbol{\theta}}{\text{minimize}} & \mathbb{E}\left[\xi\left(\boldsymbol{w}_t, \boldsymbol{x}_t\right)\right] \\ \text{subject to} & \boldsymbol{w}_t = \boldsymbol{f}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-k}, \ldots, \boldsymbol{x}_{t-1}\right), \end{array}$$

where $\xi(\cdot, \cdot)$ denotes the error function to be minimized that measures the overall system performance (e.g., the negative of the Sharpe ratio).

In principle, end-to-end architectures can offer superior performance by optimizing the overall objective function directly. However, they require substantial amounts of training data due to their deep structure and large number of learnable parameters. In financial applications, where data availability is often limited, this data-hungry nature can make end-to-end designs impractical.

High-frequency trading (HFT) presents a notable exception with its abundance of data. However, HFT strategies must account for market impact – where trade execution significantly affects market conditions. Reinforcement learning is particularly well-suited for this challenge,

as it naturally incorporates the feedback loop between trading actions and market responses. A comprehensive overview of research efforts on reinforcement learning methods for quantitative trading is given in Sun et al. (2023).

16.4 Deep Learning Portfolio Case Studies

As previously stated, research and experimentation in using DL for portfolio design (and more broadly in finance) have been flourishing since around 2005. The reality is that we are still in the early stages of this exploration, and it remains uncertain whether the DL revolution will fully merge with financial systems.

There is a continuous and increasing flow of published papers on the application of DL to portfolio design. Generally, the results presented by the authors appear promising; however, one must proceed with caution. As extensively discussed in Chapter 8, numerous dangers and potential pitfalls exist in backtesting portfolios. These naturally extend to backtesting DL architectures for portfolio design; to name a few:

- *Overfitting*: Even when results are obtained from test data not used in the training process, authors may have actually used the test data multiple times while adjusting the deep architectures (adding/removing layers, modifying layer parameters, etc.). Consequently, the results may be overfitted. Authors often do not provide specific details on the final deep networks selected, adding a sense of mystery to the system.
- *Look-ahead bias*: When using high-level DL libraries, there is a possibility of making mistakes by leaking future data during the training process (this could potentially be detected in the testing phase). Even worse, leaking future data in the input (which affects both training and testing) or having incorrect time alignment in performance evaluation could also occur.
- *Ignoring transaction costs*: DL systems typically work with high-frequency data because large amounts of training data are necessary. Ignoring transaction costs in the assessment with frequent rebalancing is entirely misleading and unacceptable. However, if the rebalancing is slow enough, such as weekly, monthly, or quarterly, transaction costs can be initially disregarded as a rough approximation.

An exhaustive overview of papers (as of 2020) of the developed DL models for financial applications can be found in Ozbayoglu et al. (2020) and, in particular, of DL applied to financial time series forecasting in Sezer et al. (2020). In the following, we will look into a few illustrative examples, with the understanding that this is just a snapshot that will become obsolete very quickly as other publications appear.

16.4.1 LSTM for Financial Time Series Forecasting

Fischer and Krauss (2018) constitutes an example of the standard time series forecasting introduced in Section 16.3.2 and illustrated in Figure 16.13.

This is the most prevalent method for applying DL to portfolio design, specifically for time series modeling or forecasting. The authors employ an LSTM network, due to its inherent

memory capabilities, and formulate the problem as binary classification by defining two classes based on whether the return of each asset is larger or smaller than the cross-sectional median return. The network is then trained to minimize the cross-entropy.

In particular, the network for each asset has the following structure:

- input layer: one feature (daily returns) with a lookback of $k = 240$ timesteps (corresponding approximately to one trading year);
- hidden layer: LSTM with 25 hidden neurons (this configuration yields 2 752 parameters, leading to a sensible number of approximately 93 training examples per parameter);
- output layer: fully connected with two neurons (corresponding to the two classes) and a softmax activation function (to obtain the probabilities of the two classes).

Once the DL architecture has been trained, its forecasts can be used to design a portfolio. Specifically, this DL architecture predicts the probability of each asset either outperforming or underperforming the cross-sectional median in period t, using only information available up until time $t - 1$. The assets are then ranked based on the probability of outperforming the median, and a long–short quintile portfolio is subsequently formed (refer to Section 6.4.4 in Chapter 6 for details on quintile portfolios).

The empirical results in Fischer and Krauss (2018), based on daily data of S&P 500 stocks, demonstrate that using LSTM networks for forecasting in conjunction with a quintile portfolio outperforms the benchmarks (i.e., random forest, logistic regression, and a fully connected deep network with three hidden layers). Before transaction costs, the Sharpe ratio is approximately 5.8 (followed by the random forest at 5.0 and the fully connected network at 2.4). After accounting for transaction costs (using 5 bps or 0.05%), the Sharpe ratio decreases to 3.8 (followed by the random forest at 3.4 and the fully connected network at 0.9). For reference, the market had a Sharpe ratio of 0.7. However, while the overall results are positive, they seem to have been much better during the 1993–2009 period and deteriorated between 2010–2015, with profitability fluctuating around zero.

16.4.2 Financial Time Series Forecasting Integrated with Portfolio Optimization

Butler and Kwon (2023) provides an example of the portfolio-based time series forecasting presented in Section 16.3.3 and illustrated in Figure 16.14.

The overall architecture consists of the following two components:

- a simple linear network for forecasting returns; and
- a mean–variance portfolio (MVP) optimization component (refer to Chapter 7 for details on MVP).

As discussed in Section 16.3.3, the partial derivatives of the portfolio solution are essential for the backpropagation learning algorithm. These derivatives are derived in detail in Butler and Kwon (2023).

Numerical experiments were conducted on a universe of 24 global futures markets, using daily returns from 1986 to 2020. The proposed method was compared to the benchmark,

where the forecasting block is trained to minimize the MSE. The results showed a significant improvement in terms of the Sharpe ratio, although transaction costs were not considered.

16.4.3 End-to-End NN-Based Portfolio

Uysal et al. (2024) serves as an example of both the portfolio-based time series forecasting presented in Section 16.3.3, as illustrated in Figure 16.14, and the end-to-end architecture presented in Section 16.3.4, as depicted in Figure 16.15.

The authors propose two schemes: a model-based approach, where the neural network learns intermediate features that are fed into a portfolio optimization block, and a model-free approach, where the neural network directly outputs the portfolio allocation.

The model-free architecture has the following structure:

- input layer: raw features (past $k = 5$ daily returns, past 10-, 20-, and 30-day average returns, and volatilities of each asset);
- hidden layer: fully connected with 32 neurons; and
- output layer: seven neurons (same as the number of assets) with the softmax function to obtain the normalized portfolio allocation.

The model-based architecture has the following structure:

- input layer: same raw features as in the model-free case;
- hidden layers:
  - first a fully connected hidden layer similar to the one in the model-free case,
  - then a second hidden layer with the softmax function to obtain the risk budgeting; and
- output layer: risk-parity portfolio (RPP) optimization block (refer to Chapter 11 for details on RPP).

The empirical results based on daily market data of seven ETFs during 2011–2021 appear promising (although transaction costs were not considered in this analysis). For the model-based case, the Sharpe ratio was around 1.10 to around 1.15, while for the nominal risk-parity portfolio it was around 0.62 to around 0.79, and for the $1/N$ portfolio around 0.41 to around 0.83. However, the performance for the model-free case was not impressive, with a Sharpe ratio around 0.31 to around 0.56. A plausible explanation for this is that the model-free portfolio lacks any structure to guide the allocation, resulting in overfitting. Therefore, the model-based architecture is preferred.

16.4.4 End-to-End DL-Based Portfolio

C. Zhang et al. (2021), which builds on Z. Zhang et al. (2020a), is an example of the end-to-end architecture presented in Section 16.3.4 and illustrated in Figure 16.15.

This end-to-end framework bypasses the traditional forecasting step and eliminates the need for estimating the covariance matrix. It can optimize various objective functions, such as the Sharpe ratio and mean–variance trade-off. A notable aspect of this work is how the authors

design neural layer structures to ensure that the output portfolio satisfies constraints about short selling, cardinality control, maximum positions for individual assets, and leverage.

The architecture is divided into two blocks: the score block (which produces a kind of raw portfolio) and the portfolio block (which enforces the desired constraints).

- The score block takes the current market information as input, for example, a lookback of the previous k returns $(\boldsymbol{x}_{t-k}, \ldots, \boldsymbol{x}_{t-1})$, and outputs the fitness scores for all the assets $\boldsymbol{s}_t$. This block could be interpreted as making a forecast of the assets' performance, similar to the traditional return forecast $\boldsymbol{\mu}_t$, although it is not quite the same. In fact, it is more like a raw version of the portfolio weights $\boldsymbol{w}_t$. The following different architectures are considered:
  - linear model;
  - fully connected network with 64 units;
  - single LSTM layer with 64 units; and
  - CNN with four layers: the first three layers are one-dimensional convolutional layers with filters of size 32, 64, 128 (i.e., producing these numbers of feature maps), and each filter has the same kernel size (3,1), the last layer being a single LSTM with 64 units.
- The portfolio block takes the previous assets' fitness scores $\boldsymbol{s}_t$ as input and enforces the desired structure as follows:
  - for the typical no-shorting normalized weights, this block is simply a softmax layer:

$$\boldsymbol{w}_t = \frac{e^{\boldsymbol{s}_t}}{\mathbf{1}^\top e^{\boldsymbol{s}_t}};$$

  - if shorting is allowed, then the softmax is modified to include the sign as:

$$\boldsymbol{w}_t = \text{sign}(\boldsymbol{s}_t) \times \frac{e^{\boldsymbol{s}_t}}{\mathbf{1}^\top e^{\boldsymbol{s}_t}};$$

  - to control the maximum position u, the authors propose using the generalized sigmoid $\sigma_a(z) = a + 1/(1 + e^{-z})$ (with $a = (1-u)/(Nu-1)$) applied elementwise to the scores $\boldsymbol{s}_t$:

$$\boldsymbol{w}_t = \text{sign}(\boldsymbol{s}_t) \times \frac{\boldsymbol{\sigma}_a(|\boldsymbol{s}_t|)}{\mathbf{1}^\top \boldsymbol{\sigma}_a(|\boldsymbol{s}_t|)};$$

  - for the cardinality constraint (assuming shorting is allowed), the authors propose a layer that implements a long–short quintile portfolio (refer to Section 6.4.4 for a description of the quintile portfolio and C. Zhang et al. (2021) for details on how to implement this layer in a way that it is differentiable, which is required for the learning process); and
  - to enforce the leverage L, one simply scales up with the factor L.

The empirical results in C. Zhang et al. (2021), based on daily data, show that the end-to-end architecture based on a single LSTM layer yields the best results, with a Sharpe ratio of 2.6, while the benchmarks achieved no better than 1.6. However, these results were obtained without considering transaction costs. Unfortunately, when even small transaction costs

of 2 bps (i.e., 0.02%) are factored in, the superior performance disappears, resulting in a performance not significantly different from that of a simple benchmark (e.g., the maximum diversification portfolio). As the authors themselves suggest, further work is needed to account for transaction costs in the learning process, such as by controlling the turnover.

16.4.5 End-to-End Deep Reinforcement Learning Portfolio

Z. Zhang et al. (2020b) offers an example of the deep reinforcement learning. The system is designed to maximize the expected cumulative return, which aims to maximize expected cumulative rewards through an agent's interaction with an uncertain environment. Within this reinforcement learning framework, the system can efficiently map various market situations to trading positions and seamlessly incorporate market frictions, such as commissions, into the reward functions. This allows for the direct optimization of trading performance. To represent the state-space, the authors take into account several features, including past prices, returns over varying time frames, and technical indicators like the MACD[9] and the RSI.[10] The action space is modeled as a simple discrete set ($\{-1, 0, 1\}$ representing short, no holding, and long positions, respectively), and a continuous set that encompasses the entire $[-1, 1]$ interval. The reward function consists of the volatility-adjusted return after accounting for transaction costs. In all models, the authors utilize two-layer LSTM networks with 64 and 32 units.

The authors evaluate their algorithms using 50 highly liquid futures contracts spanning from 2011 to 2019, examining performance variations across various asset classes such as commodities, equity indexes, fixed income, and foreign exchange markets. They contrast their algorithms with traditional time series momentum strategies, demonstrating that their approach surpasses these baseline models by generating positive profits even in the face of substantial transaction costs. The experimental results indicate that the proposed algorithms can effectively track major market trends without altering positions, as well as scale down or maintain positions during consolidation periods.

16.5 Summary

Deep neural networks have successfully demonstrated outstanding performance in many domain-specific areas, such as image recognition, natural language processing, board games, self-driving cars, and so on. The million-dollar question is whether this revolution will extend to financial systems.

Some problems like sentiment analysis of news for trading purposes have clearly benefited from the advances in natural language processing. However, other problems related to financial time series forecasting and portfolio design remain unclear. Among the many challenges that set these problems apart from other successful applications, we can list the following:

- *Data scarcity*: The amount of financial data is generally limited (e.g., two years of daily

[9] The moving average convergence divergence (MACD) is a momentum oscillator primarily used to trade trends.

[10] The relative strength index (RSI) is a momentum indicator used in technical analysis that measures the magnitude of recent price changes to evaluate overbought or oversold conditions in the price of a stock or other asset.

stock prices amount to just 504 observations). Perhaps high-frequency data provides a more promising direction.

- *Low signal-to-noise ratio*: The signal in financial data (capable of generating alpha) is extremely weak and totally submerged in noise.
- *Data nonstationarity*: Financial time series are clearly nonstationary, which makes learning the statistics complicated.
- *Data adaptive feedback loop*: Patterns discovered in financial data and exploited for trading tend to disappear immediately due the feedback loop mechanism.
- *Lack of prior human evidence*: There seems to be no human capable of forecasting the future performance of companies to design a portfolio with a significant alpha. Recall the provocative statement (Malkiel, 1973): "a blindfolded chimpanzee throwing darts at the stock listings can select a portfolio that performs as well as those managed by the experts."

Despite these challenges, the jury is still out on whether the deep learning revolution will fully extend to financial systems. It is still too early to adventure any future prediction.

Exercises

Machine Learning

16.1 (Classification of spam in emails) Build a black-box model that can accurately classify whether an email is spam or not.

- Use a dataset of labeled emails to train and evaluate your model; for example, the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/spambase) or the Enron-Spam dataset (http://nlp.cs.aueb.gr/software_and_datasets/Enron-Spam).
- Experiment with different black-box models, such as logistic regression, decision trees, random forests, and support vector machines, and compare their performance.

16.2 (Regression of housing prices) Build a black-box model that can predict the price of a house based on its features such as square footage, number of bedrooms, and location.

- Use a dataset of labeled houses to train and evaluate your model; for example, the Boston Housing Dataset at Kaggle (www.kaggle.com/code/prasadperera/the-boston-housing-dataset) or the California Housing Prices at Kaggle (www.kaggle.com/datasets/camnugent/california-housing-prices).
- Experiment with different black-box models, such as linear regression, decision trees, random forests, and support vector machines, and compare their performance.

Deep Learning

16.3 (Image classification with CNNs) Use a CNN to classify images from the CIFAR-10 dataset (www.cs.toronto.edu/~kriz/cifar.html), which consists of 60 000 32×32 color images in 10 classes. Experiment with different architectures, such as varying the number of

convolutional layers, pooling layers, and fully connected layers, to see which one performs best.

16.4 (Object detection with Faster R-CNN) Use a Faster R-CNN model (a deep learning model for object detection, developed by Microsoft Research in 2015) to detect objects in images from the COCO dataset (https://cocodataset.org), which consists of over 200 000 labeled images with 80 different object categories. Experiment with different backbone architectures, such as ResNet and VGG, and adjust the hyper-parameters to improve the detection accuracy.

16.5 (Language translation with sequence-to-sequence models) Use a sequence-to-sequence model with attention to translate text from one language to another. Use a dataset such as the Multi30k dataset (https://github.com/multi30k/dataset), which consists of about 30 000 parallel sentences in English, French, and German. Experiment with different encoder and decoder architectures, such as LSTM and transformer, and adjust the hyper-parameters to improve the translation accuracy.

16.6 (GANs for image synthesis) Use a GAN to generate realistic images that resemble a given dataset.

- Use an image dataset, such as the CIFAR-10 dataset (www.cs.toronto.edu/~kriz/cifar.html), which consists of 60 000 32×32 color images in 10 classes, or the MNIST dataset (https://github.com/mbornet-hl/MNIST/tree/master/IMAGES/GROUPS), which consists of a large database of handwritten digits with 60 000 training images and 10 000 testing images.
- Experiment with different GAN architectures, such as DCGAN and WGAN, and adjust the hyper-parameters to improve the image quality.

16.7 (Handwritten Digit Recognition with CNNs) Use a CNN to classify handwritten digits from the MNIST dataset (https://github.com/mbornet-hl/MNIST/tree/master/IMAGES/GROUPS), which consists of 60,000 training images and 10,000 test images of handwritten digits from 0 to 9. Experiment with different architectures, such as varying the number of convolutional layers, pooling layers, and fully connected layers, to see which one performs best.

Machine Learning for Finance

16.8 (Linear regression for stock prices) Implement a linear regression model to predict the stock price of a company based on its historical data. You can use data from financial databases such as Yahoo Finance. Evaluate the performance of the model using metrics such as mean squared error.

16.9 (Decision trees for default prediction) Build a decision tree model to classify whether a loan applicant is likely to default or not based on features such as income, credit score, and loan amount.

- Use some publicly available dataset for classifying loan defaults; for example, the German Credit Dataset or the UCI Credit Approval Dataset, both available at the UCI Machine

Learning Repository (https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data) and https://archive.ics.uci.edu/ml/datasets/credit+approval, respectively).

- Evaluate the performance of the model using metrics such as accuracy, precision, and recall.

16.10 (Random forests for default prediction) Extend the previous decision tree model to a random forest model and compare the performance of both models. Use cross-validation to tune the hyper-parameters of the random forest model and evaluate it using the same metrics as the decision tree model.

16.11 (SVMs for stock direction forecast) Implement an SVM model to predict whether a stock will go up or down based on technical indicators such as moving averages and relative strength index (RSI). Use grid search to find the best hyper-parameters of the model and evaluate it using metrics such as accuracy, precision, and recall.

Deep Learning for Finance

16.12 (Comparison of LSTMs vs. transformers with synthetic data) Generate synthetic data to compare the long-term memory of LSTMs and transformers.

a. Create the inputs as 20-dimensional vectors (containing 20 samples over time) from two possible predefined sequences and the outputs as two possible labels corresponding to the two possible sequences.
b. Add random noise to the inputs for different values of noise variance, leading to different values of signal-to-noise ratio.
c. Train an LSTM network and a transformer network for different signal-to-noise ratios and compare them.
d. Then, repeat the experiment, but now using as inputs 100-dimensional vectors containing the previous 20-dimensional predefined sequences at the end of the vectors (more recent temporal observations) and zeros elsewhere. The networks should learn that only the more recent 20 samples contain the useful pattern, whereas the first 80 samples contain just noise.
e. Repeat the experiment with 100-dimensional input vectors, but now placing the 20-dimensional predefined sequences at the beginning of the vectors (earlier temporal observations). Again, the networks should learn the correct location of the useful 20 samples, but now they happen earlier in time. The transformer architecture should not be affected, whereas the LSTM may tend to "forget" the useful patterns as they happened earlier.
f. Finally, repeat the experiment with bigger dimensions until the difference between LSTMs and transformers becomes clear.

16.13 (Predicting stock prices using deep learning) Develop a deep neural network to predict the future prices of a stock based on historical time series data. In particular, consider the following architectures and compare their performance: LSTM, CNN, and transformer.

16.14 (Portfolio optimization using deep learning) Use deep learning to optimize a portfolio of investments. This could involve using historical time series data to develop a model that

maximizes returns while minimizing risk. You may want to experiment with different loss functions, such as MSE or MAPE, and explore different optimization algorithms, such as SGD or Adam, to train your model. Additionally, you may want to explore the use of techniques such as attention mechanisms or reinforcement learning to further improve the performance of your model.

References

Ahmed, N. K., Atiya, A. F., Gayar, N. E., & El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, *29*(5–6), 594–621.

Amos, B., & Kolter, J. Z. (2017). OptNet: Differentiable optimization as a layer in neural networks. *Proceedings of the International Conference on Machine Learning (ICML)*, *70*, 136–145.

Atiya, A. F. (2001). Bankruptcy prediction for credit risk using neural networks: A survey and new results. *IEEE Transactions on Neural Networks*, *12*(4), 929–935.

Ballings, M., den Poel, D. V., Hespeels, N., & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, *42*(20), 7046–7056.

Bengio, Y. (1997). Using a financial training criterion rather than a prediction criterion. *International Journal of Neural Systems*, *8*(4), 433–443.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bontempi, G., Taieb, S. B., & Borgne, Y.-A. L. (2012). Machine learning strategies for time series forecasting. In M.-A. Aufaure & E. Zimányi (Eds.), *Business Intelligence* (pp. 62–77). Springer.

Breiman, L. (2001). Statistical modelling: The two cultures. *Statistical Science*, *16*(3), 199–231.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T., & Zhang, Y. (2023). Sparks of artificial general intelligence: Early experiments with GPT-4. *Preprint. Available at arXiv*. https://doi.org/10.48550/arXiv.2303.12712

Butler, A., & Kwon, R. H. (2023). Integrating prediction in mean–variance portfolio optimization. *Quantitative Finance*, *23*(3), 429–452.

Cao, L. J., & Tay, F. E. H. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, *14*(6), 1506–1518.

Esling, P., & Agon, C. (2012). Time series data mining. *ACM Computing Surveys*, *45*(1), 1–34.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*(2), 654–669.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *Proceedings of the 27th International Conference on Neural Information Processing Systems (NeurIPS)*, *27*, 2672–2680.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer.

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)*, 6840–6851.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: With Applications in R*. Springer.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, *37*(2), 233–243.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems (NeurIPS)*, *25*, 1097–1105.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lin, H. W., Tegmark, M., & Rolnick, D. (2017). Why does deep and cheap learning work so well? *Journal of Statistical Physics*, *168*, 1223–1247.

López de Prado, M. (2018a). The 10 reasons most machine learning funds fail. *Journal of Portfolio Management*, *44*(6), 120–133.

López de Prado, M. (2018b). *Advances in Financial Machine Learning*. John Wiley & Sons.

López de Prado, M. (2019). Ten applications of financial machine learning. *SSRN Electronic Journal*. https://dx.doi.org/10.2139/ssrn.3365271

Malkiel, B. G. (1973). *A Random Walk Down Wall Street*. W. W. Norton.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press. Available online. http://neuralnetworksanddeeplearning.com

OpenAI. (2023). GPT-4 technical report. *Preprint. Available at arXiv*. https://doi.org/10.48550/arXiv.2303.08774

Ozbayoglu, A. M., Gudelek, M. U., & Sezer, O. B. (2020). Deep learning for financial applications: A survey. *Preprint. Available at arXiv*. https://doi.org/10.48550/arXiv.2002.05786

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117.

Schmidhuber, J. (2022). Annotated history of modern AI and deep learning. *Preprint. Available at arXiv*. https://doi.org/10.48550/arXiv.2212.11279

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing*, *90*, 106181.

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *Proceedings of the International Conference on Machine Learning (ICML)*, *37*, 2256–2265.

Song, Y., & Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)*, 11918–11930.

Sun, S., Wang, R., & An, B. (2023). Reinforcement learning for quantitative trading. *ACM Transactions on Intelligent Systems and Technology*, *13*(3), 1–29.

Takahashi, S., Chen, Y., & Tanaka-Ishii, K. (2019). Modelling financial time-series with generative adversarial networks. *Physica A: Statistical Mechanics and its Applications*, *527*, 1–12.

Uysal, A. S., Li, X., & Mulvey, J. M. (2024). End-to-end risk budgeting portfolio optimization with neural networks. *Annals of Operations Research*, *339*(1–2), 397–426.

Vapnik, V. (1999). *The Nature of Statistical Learning Theory* (2nd ed.). Springer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, 6000–6010.

Yoon, J., Jarrett, D., & van der Schaar, M. (2019). Time-series generative adversarial networks. *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)*, 5508–5518.

Zhang, C., Zhang, Z., Cucuringu, M., & Zohren, S. (2021). A universal end-to-end approach to portfolio optimization via deep learning. *Preprint. Available at arXiv*. https://doi.org/10.48550/arXiv.2111.09170

Zhang, Z., Zohren, S., & Roberts, S. (2020a). Deep learning for portfolio optimization. *The Journal of Financial Data Science*, *2*(4), 8–20.

Zhang, Z., Zohren, S., & Roberts, S. (2020b). Deep reinforcement learning for trading. *The Journal of Financial Data Science*, *4*(1), 1–16.

Appendices

Appendix A

Convex Optimization Theory

Mathematics, rightly viewed, possesses not only truth, but supreme beauty – a beauty cold and austere, like that of sculpture, without appeal to any part of our weaker nature . . .

— Bertrand Russell, *The Study of Mathematics*

Over the past few decades, numerous fundamental and practical advancements have been made in the field of convex optimization theory. These developments have not only found applications in various fields such as engineering, finance, and machine learning, but they have also stimulated the mathematical progression of both the theory and the development of efficient algorithms. Some notable textbook references in this field include Boyd and Vandenberghe (2004), Luenberger and Ye (2021), Bertsekas (1999), Bertsekas et al. (2003), Nemirovski (2000), Ben-Tal and Nemirovski (2001), and Nesterov (2018). Two classic references are Luenberger (1969) and Rockafellar (1970). A range of applications of optimization in engineering can be found in Palomar and Eldar (2009).

Traditionally, there was a common belief that *linear problems* are easier to solve compared to *nonlinear problems*. However, as Rockafellar pointed out in a 1993 survey (Rockafellar, 1993), "the great watershed in optimization isn't between linearity and nonlinearity, but between convexity and nonconvexity." In essence, convex problems can be optimally solved either in closed form – by applying the optimality conditions derived from Lagrange duality – or numerically – using highly efficient algorithms that exhibit polynomial convergence rates (Boyd & Vandenberghe, 2004). Consequently, it is often said that once a problem is formulated as a convex problem, it is essentially solved.

Unfortunately, most practical problems do not exhibit convexity in their initial formulation. However, many of these problems may possess a hidden convexity that practitioners need to uncover to effectively utilize the tools provided by convex optimization theory.

Generally, certain manipulations are necessary to transform a problem into a convex one. The advantage of formulating a problem in convex form is that, even in the absence of a closed-form solution and despite the problem's complexity (which may include hundreds of variables and a nonlinear, nondifferentiable objective function), it can still be solved numerically with high efficiency, both theoretically and practically (Boyd & Vandenberghe,

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

2004). Another appealing aspect of expressing the problem in a convex form is that additional constraints can be easily incorporated, provided they are convex.

This appendix provides a concise introduction to convex optimization theory, drawing on Boyd and Vandenberghe (2004). For more detailed information, the reader is encouraged to consult this comprehensive source.

A.1 Optimization Problems

A *mathematical optimization problem* with arbitrary equality and inequality constraints can always be written in the following standard form (Boyd & Vandenberghe, 2004, Chapter 1):

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimize}} & f_0(\boldsymbol{x}) & \\ \text{subject to} & f_i(\boldsymbol{x}) \leq 0, & i = 1, \ldots, m, \\ & h_i(\boldsymbol{x}) = 0, & i = 1, \ldots, p, \end{array} \tag{A.1}$$

where $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is the *optimization variable*, $f_0 : \mathbb{R}^n \longrightarrow \mathbb{R}$ is the *objective function* or cost function, $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$, $i = 1, \ldots, m$, are the m *inequality constraint functions*, and $h_i : \mathbb{R}^n \longrightarrow \mathbb{R}$, $i = 1, \ldots, p$, are the p *equality constraint functions*. If there are no constraints, we say that the problem is *unconstrained*.

The goal is to find an optimal solution $\boldsymbol{x}^\star$ that minimizes f_0 while satisfying all the constraints.

Convex optimization is currently used in many different areas, including the following:

- circuit design
- filter design
- communication systems (e.g., transceiver design, multi-antenna beamforming design, maximum likelihood detection)
- radar systems
- communication networks (e.g., power control in wireless networks, congestion control on the internet)
- financial engineering (e.g., portfolio design, index tracking)
- model fitting (e.g., in financial data or recommender systems)
- image processing (e.g., deblurring, compressive sensing, blind separation, inpainting)
- robust designs under uncertainty
- sparse regression
- low-rank matrix discovery
- machine learning
- graph learning from data
- biomedical applications (e.g., DNA sequencing, anti-viral vaccine design).

An optimization problem has three basic elements: variables (as opposed to other fixed parameters), constraints, and an objective.

Example A.1 (Device sizing for electronic circuits) In the context of electronic circuit design, the elements in a device sizing optimization problem may be chosen as:

- Variables: widths and lengths of devices.

- Constraints: manufacturing limits, timing requirements, or maximum area.
- Objective: power consumption.

Example A.2 (Portfolio design) In a financial context, the elements in a portfolio optimization problem may be identified as:

- Variables: amounts invested in different assets.
- Constraints: budget, maximum investments per asset, or minimum return.
- Objective: This could be the overall risk or return variance.

A.1.1 Definitions

Domain and Constraints

The *domain* of the optimization problem (A.1) is defined as the set of points for which the objective and all constraint functions are defined, that is,

$$\mathcal{D} = \bigcap_{i=0}^{m} \operatorname{dom} f_i \cap \bigcap_{i=1}^{p} \operatorname{dom} h_i,$$

and it can be interpreted as a set of implicit constraints $\boldsymbol{x} \in \mathcal{D}$, as opposed to the explicit constraints $f_i(\boldsymbol{x}) \leq 0$ and $h_i(\boldsymbol{x}) = 0$ in (A.1).

A problem is *unconstrained* if it has no explicit constraints. For example,

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \log\left(a - \boldsymbol{b}^{\mathsf{T}}\boldsymbol{x}\right)$$

is an unconstrained problem with implicit constraint $a > \boldsymbol{b}^{\mathsf{T}}\boldsymbol{x}$.

Feasibility

A point $\boldsymbol{x} \in \mathcal{D}$ is *feasible* if it satisfies all the constraints, $f_i(\boldsymbol{x}) \leq 0$ and $h_i(\boldsymbol{x}) = 0$, and *infeasible* otherwise. The problem (A.1) is said to be *feasible* if there exists at least one feasible point and *infeasible* otherwise.

The *optimal value* (minimal value) is defined as

$$p^\star = \inf\left\{f_0(\boldsymbol{x}) \mid f_i(\boldsymbol{x}) \leq 0,\ i = 1, \ldots, m,\ h_i(\boldsymbol{x}) = 0,\ i = 1, \ldots, p\right\}.$$

If the problem is feasible, then the optimal value may be achieved at an optimal solution $\boldsymbol{x}^\star$, that is, $f_0(\boldsymbol{x}^\star) = p^\star$, or the problem may be unbounded below, that is, $p^\star = -\infty$. Otherwise, if the problem is infeasible, it is commonly denoted by $p^\star = +\infty$.

A feasible point $\boldsymbol{x}$ is *optimal* if $f_0(\boldsymbol{x}) = p^\star$. In general, there may be more than one optimal point and the set of optimal points is denoted by $\mathcal{X}_{\text{opt}}$. A feasible point $\boldsymbol{x}$ is *locally optimal* if it is optimal within a ball or a local neighborhood.

Example A.3 We illustrate the concepts of optimal value and optimal solution with a few simple unconstrained optimization problems with scalar variable $x \in \mathbb{R}$:

- $f_0(x) = 1/x$, $\operatorname{dom} f_0 = \mathbb{R}_{++}$: In this case $p^\star = 0$, but there is no optimal point since the optimal value cannot be achieved.

- $f_0(x) = -\log x$, $\operatorname{dom} f_0 = \mathbb{R}_{++}$: This function is unbounded below $p^\star = -\infty$.
- $f_0(x) = x^3 - 3x$: This is a nonconvex function with $p^\star = -\infty$ and a local optimum at $x = 1$.

If $\boldsymbol{x}$ is feasible and $f_i(\boldsymbol{x}) = 0$, we say the ith inequality constraint $f_i(\boldsymbol{x}) \leq 0$ is *active* at $\boldsymbol{x}$. If $f_i(\boldsymbol{x}) < 0$, we say the constraint $f_i(\boldsymbol{x}) \leq 0$ is *inactive*. (The equality constraints are active at all feasible points.) We say that a constraint is *redundant* if deleting it does not change the feasible set.

Feasibility Problem

On many occasions, our goal is not necessarily to minimize or maximize any objective, but simply to find a feasible point. This is referred to as a *feasibility problem*:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{find}} & \boldsymbol{x} \\ \text{subject to} & f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m, \\ & h_i(\boldsymbol{x}) = 0, \quad i = 1, \ldots, p. \end{array}$$

In practice, a feasibility problem can be regarded as a special case of a general optimization problem:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & 0 \\ \text{subject to} & f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m, \\ & h_i(\boldsymbol{x}) = 0, \quad i = 1, \ldots, p, \end{array}$$

where $p^\star = 0$ if the constraints are feasible and $p^\star = \infty$ otherwise.

A.1.2 Solving Optimization Problems

General optimization problems are typically very difficult to solve, meaning that either a long computation time is required to find an optimal solution or that a sub-optimal solution is found in a reasonable computation time. Some exceptions include the family of least squares problems, linear programming problems, and convex optimization problems. Nevertheless, general nonconvex problems (also known as *nonlinear problems* with some abuse of terminology) are difficult to solve.

Least Squares

Least squares (LS) problems go back to Gauss in 1795 and are formulated as

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 .$$

Solving LS problems is straightforward with the closed-form solution $\boldsymbol{x}^\star = \left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^\mathsf{T}\boldsymbol{b}$, for which reliable and efficient algorithms exist. Utilizing LS is considered trivial because these problems are easy to identify and solve.

Linear Programming

A linear problem (LP) can be written as

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{c}^\mathsf{T}\boldsymbol{x} \\ \text{subject to} & \boldsymbol{a}_i^\mathsf{T}\boldsymbol{x} \leq b_i, \quad i = 1, \ldots, m. \end{array}$$

LPs do not have closed-form solutions in general, but there are reliable and efficient algorithms and software. An LP is not as easy to recognize as an LS, but one can easily learn a few standard tricks to convert a variety of problems into LPs.

Convex Optimization Problems

An inequality-constrained convex problem is of the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(\boldsymbol{x}) \\ \text{subject to} \quad & f_i(\boldsymbol{x}) \leq b_i, \quad i = 1, \ldots, m, \end{aligned}$$

where all the functions are convex. Convex problems do not have closed-form solutions in general, but there are reliable and efficient algorithms and software. They are often difficult to recognize and there are many tricks for transforming problems into convex form.

Nonconvex Optimization

Nonconvex optimization problems are generally very difficult to solve, although there are some rare exceptions. In general, they require either a long computation time[1] or the compromise of not always finding the optimal solution. This results in two strategies:

- *Local optimization*: This involves fast algorithms, but there's no guarantee of global optimality. It only provides a local solution around the initial point.
- *Global optimization*: While the worst-case complexity increases exponentially with the problem size, this strategy ensures the discovery of a global solution.

Historical Snapshop of Optimization

The theory of optimization, termed convex analysis, was extensively developed in the past period 1900–1970. However, the computational aspect, that is, efficient algorithms, and applications came later.

The first computationally efficient method was the seminal *simplex method* developed in 1947 for LPs by Dantzig. It represents the beginning of an era of algorithmic development. However, despite the simplex method being very efficient in practice, it has a theoretical exponential worst-case complexity. In the 1970s, the ellipsoid method was proposed with a provable polynomial worst-case complexity, although it could be very slow in practice. Later, in 1984, Karmakar proposed a polynomial-time interior-point method for LPs that was not only good in theory but also in practice (Karmarkar, 1984). This development was followed by numerous researchers extending the application of interior-point methods to quadratic programming and linear complementarity problems. In 1994, Nesterov and Nemirovskii further advanced the field by developing the theory of self-concordant functions. This theory facilitated the expansion of algorithms based on the log-barrier function to a wider array of convex problems, notably including semidefinite programming and second-order cone programming (Nesterov & Nemirovski, 1994).

[1] The computational complexity of an algorithm is measured in the number of operations required to obtain a solution (computation time is measured in time units). This complexity is presented as a function of the number of variables n to optimize, so the important thing is how this complexity grows with n. Typically, polynomial complexity is considered acceptable in practice (of course the order of the polynomial is also important), whereas exponential complexity is considered not acceptable in practice as the complexity quickly explodes.

In terms of practical applications, following the development of the simplex method, linear programming has been widely used to model a variety of real-life problems, such as allocation issues, since the 1950s. However, during that period there was little interest in modeling real-life problems as convex problems. It was only after the mid-1990s, with the development of interior-point methods for convex problems, that there was a surge in activity related to modeling applications as convex problems.

A.1.3 Illustrative Example

The following lamp illumination problem is a well-known example that illustrates how an engineering problem can be formulated in different ways, leading to more or less sophisticated solutions.

Suppose we have m lamps illuminating n small flat patches, located in fixed locations. The overall goal is to achieve a desired illumination I_{des} on all patches by controlling the power of the lamps. The intensity I_k at patch k depends linearly on the lamp powers p_j as

$$I_k = \sum_{j=1}^{m} a_{kj} p_j,$$

where the coefficients a_{kj} are given by $a_{kj} = \cos\theta_{kj}/r_{kj}^2$, with θ_{kj} and r_{kj} denoting the angle and distance, respectively, between lamp j and patch k.

Ideally, we would like to achieve perfect illumination, $I_k = I_{\text{des}}$, for all the patches, but this is not feasible in practice, and we need to relax the problem to $I_k \approx I_{\text{des}}$.

There are many different ways to formulate this problem. The main idea is to somehow measure the error in the approximation for each patch, $I_k - I_{\text{des}}$. One possible formulation is based on minimizing the largest of the errors measured in a logarithmic scale (because the eyes perceive intensity on a log-scale):

$$\begin{array}{ll} \underset{I_1,\ldots,I_n,p_1,\ldots,p_m}{\text{minimize}} & \max_k |\log I_k - \log I_{\text{des}}| \\ \text{subject to} & 0 \le p_j \le p_{\max}, \qquad j = 1,\ldots,m, \\ & I_k = \sum_{j=1}^{m} a_{kj} p_j, \qquad k = 1,\ldots,n. \end{array}$$

This problem appears complex, and we will explore various possible approaches to address it.

1. If one does not know anything about optimization, then a heuristic guess can be made, such as using a uniform power $p_j = p$, perhaps trying different values of p.

2. If one knows about least squares, then the problem formulation could be changed to resemble an LS:

$$\begin{array}{ll} \underset{I_1,\ldots,I_n,p_1,\ldots,p_m}{\text{minimize}} & \sum_{k=1}^{n} (I_k - I_{\text{des}})^2 \\ \text{subject to} & I_k = \sum_{j=1}^{m} a_{kj} p_j, \quad k = 1,\ldots,n, \end{array}$$

   and then clip p_j if $p_j > p_{\max}$ or $p_j < 0$ to make it feasible.

3. If one knows about linear programming, then the problem could be changed to resemble an LP (basically by removing the logarithmic function):

$$\begin{array}{ll} \underset{I_1,\ldots,I_n,p_1,\ldots,p_m}{\text{minimize}} & \max_k |I_k - I_{\text{des}}| \\ \text{subject to} & 0 \le p_j \le p_{\max}, \qquad j = 1,\ldots,m, \\ & I_k = \sum_{j=1}^{m} a_{kj} p_j, \qquad k = 1,\ldots,n, \end{array}$$

   which may not look like an LP at first sight, but it is in disguise (as revealed after a few simple manipulations).

4. If one knows about convex optimization, then it turns out that after some smart manipulations, the problem can be equivalently reformulated in convex form as

$$\begin{array}{ll} \underset{I_1,\ldots,I_n,p_1,\ldots,p_m}{\text{minimize}} & \max_k \; h\left(I_k / I_{\text{des}}\right) \\ \text{subject to} & 0 \le p_j \le p_{\max}, \qquad j = 1,\ldots,m, \\ & I_k = \sum_{j=1}^{m} a_{kj} p_j, \qquad k = 1,\ldots,n, \end{array}$$

   where $h(u) = \max\{u, 1/u\}$.

At this point, one can go further and ask whether additional constraints can be added. For example, the constraint "no more than half of total power is in any 10 lamps" looks complicated, but it can actually be written in convex form, so computationally keeps the problem solvable. On the other hand, the constraint "no more than half of the lamps are on" may look simple, but instead is a deadly combinatorial constraint that makes the complexity of its resolution exponential. The moral is that untrained intuition does not always work; one needs to obtain the proper background and develop the right intuition to discern between difficult and easy problems.

A.2 Convex Sets

We now provide a concise introduction to convex sets. For more detailed information, the reader is referred to Chapter 2 in Boyd and Vandenberghe (2004).

A.2.1 Definitions

We start with some basic definitions. The equation describing a line passing through two points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ is $\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}$, where $\theta \in \mathbb{R}$. If θ is constrained to be between 0 and 1, then $\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}$ describes the *line segment* between $\boldsymbol{x}$ and $\boldsymbol{y}$.

Definition A.1 (Convex set) A set $C \in \mathbb{R}^n$ is *convex* if the line segment between any two points in C lies in C, that is, if for any $\boldsymbol{x}, \boldsymbol{y} \in C$ and θ with $0 \le \theta \le 1$, we have

$$\theta\boldsymbol{x} + (1-\theta)\,\boldsymbol{y} \in C. \tag{A.2}$$

A *convex combination* of the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ is of the form $\theta_1\boldsymbol{x}_1 + \theta_2\boldsymbol{x}_2 + \cdots + \theta_k\boldsymbol{x}_k$, where $\theta_1 + \cdots + \theta_k = 1$ and $\theta_i \ge 0$, $i = 1,\ldots,k$. It can be shown that a set is convex if and only if it contains every convex combination of its points.

The *convex hull* of a set C is the set of all convex combinations of points in C. As the name

suggests, the convex hull of C is always convex. In fact, it is the smallest convex set that contains C.

A set C is called a *cone* if for every $\boldsymbol{x} \in C$ and $\theta \geq 0$ we have $\theta \boldsymbol{x} \in C$. A set C is a *convex cone* if it is convex and a cone, that is, for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in C$ and $\theta_1, \theta_2 \geq 0$, we have

$$\theta_1 \boldsymbol{x}_1 + \theta_2 \boldsymbol{x}_2 \in C.$$

A.2.2 Elementary Convex Sets

We describe next some important simple examples of convex sets.

Hyperplanes and Halfspaces

A *hyperplane* is a set of the form

$$\left\{\boldsymbol{x} \mid \boldsymbol{a}^{\mathsf{T}} \boldsymbol{x} = b\right\},$$

where $\boldsymbol{a} \in \mathbb{R}^n$, $b \in \mathbb{R}$. Geometrically, it can be interpreted as the set of points orthogonal to the vector $\boldsymbol{a}$ with an offset by rewriting it as $\left\{\boldsymbol{x} \mid \boldsymbol{a}^{\mathsf{T}}(\boldsymbol{x} - \boldsymbol{x}_0) = 0\right\}$.

A hyperplane divides the space $\mathbb{R}^n$ into two halfspaces. A (closed) *halfspace* is a set of the form

$$\left\{\boldsymbol{x} \mid \boldsymbol{a}^{\mathsf{T}} \boldsymbol{x} \leq b\right\}.$$

Polyhedra

A *polyhedron* is defined as the solution set of a finite number of linear equalities and inequalities:

$$\mathcal{P} = \{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{C}\boldsymbol{x} = \boldsymbol{d}\},$$

where $A \in \mathbb{R}^{m \times n}$, $C \in \mathbb{R}^{p \times n}$, $b \in \mathbb{R}^m$, $d \in \mathbb{R}^p$.

Two important examples of polyhedra are the *unit simplex*, defined as

$$\left\{\boldsymbol{x} \mid \boldsymbol{x} \geq \boldsymbol{0}, \boldsymbol{1}^{\mathsf{T}} \boldsymbol{x} \leq 1\right\},$$

and the *probability simplex*, defined as

$$\left\{\boldsymbol{x} \mid \boldsymbol{x} \geq \boldsymbol{0}, \boldsymbol{1}^{\mathsf{T}} \boldsymbol{x} = 1\right\}.$$

Balls and Ellipsoids

A *Euclidean ball* (or just a ball) with center $\boldsymbol{x}_{\mathrm{c}}$ and radius r has the form

$$\mathcal{B}(\boldsymbol{x}_{\mathrm{c}}, r) = \{\boldsymbol{x} \mid \|\boldsymbol{x} - \boldsymbol{x}_{\mathrm{c}}\|_2 \leq r\} = \left\{\boldsymbol{x} \mid (\boldsymbol{x} - \boldsymbol{x}_{\mathrm{c}})^{\mathsf{T}}(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{c}}) \leq r^2\right\},$$

where $\|\cdot\|_2$ denotes the Euclidean norm or ℓ_2-norm. Another common representation for the Euclidean ball is

$$\mathcal{B}(\boldsymbol{x}_{\mathrm{c}}, r) = \{\boldsymbol{x}_{\mathrm{c}} + r\boldsymbol{u} \mid \|\boldsymbol{u}\|_2 \leq 1\}.$$

A set related to a ball is an *ellipsoid*, defined as

$$\begin{aligned}\mathcal{E}(\boldsymbol{x}_\mathrm{c}, \boldsymbol{P}) &= \left\{\boldsymbol{x} \mid (\boldsymbol{x}-\boldsymbol{x}_\mathrm{c})^\mathsf{T}\boldsymbol{P}^{-1}(\boldsymbol{x}-\boldsymbol{x}_\mathrm{c}) \leq 1\right\} \\ &= \left\{\boldsymbol{x}_\mathrm{c} + \boldsymbol{A}\boldsymbol{u} \mid \|\boldsymbol{u}\|_2 \leq 1\right\},\end{aligned}$$

with $\boldsymbol{P} = \boldsymbol{P}^\mathsf{T} \in \mathbb{R}^{n\times n} \succ \boldsymbol{0}$, that is, $\boldsymbol{P}$ is symmetric and positive definite, and $\boldsymbol{A}$ is the square-root matrix $\boldsymbol{P}^{1/2}$ satisfying $\boldsymbol{A}^\mathsf{T}\boldsymbol{A} = \boldsymbol{P}$. The matrix $\boldsymbol{P}$ (or $\boldsymbol{A}$) determines how far the ellipsoid extends in every direction from the center $\boldsymbol{x}_\mathrm{c}$. A ball is an ellipsoid with the particular choice $\boldsymbol{P} = r^2\boldsymbol{I}$.

Norm Balls and Norm Cones

Suppose $\|\cdot\|$ is any norm on $\mathbb{R}^n$ (not necessarily the Euclidean norm). A *norm ball* with center $\boldsymbol{x}_\mathrm{c}$ and radius r is defined as

$$\mathcal{B}(\boldsymbol{x}_\mathrm{c}, r) = \{\boldsymbol{x} \mid \|\boldsymbol{x}-\boldsymbol{x}_\mathrm{c}\| \leq r\}.$$

A *norm cone* $C \subseteq \mathbb{R}^{n+1}$ is defined as the convex set

$$C = \left\{(\boldsymbol{x}, t) \in \mathbb{R}^{n+1} \mid \|\boldsymbol{x}\| \leq t\right\}.$$

One particular case of interest is the *second-order cone* (a.k.a. *ice-cream cone*):

$$C = \left\{(\boldsymbol{x}, t) \in \mathbb{R}^{n+1} \mid \|\boldsymbol{x}\|_2 \leq t\right\},$$

where the norm is the Euclidean norm $\|\cdot\|_2$.

Positive Semidefinite Cone

The set of symmetric positive semidefinite matrices

$$\mathbb{S}^n_+ = \left\{\boldsymbol{X} \in \mathbb{R}^{n\times n} \mid \boldsymbol{X} = \boldsymbol{X}^\mathsf{T} \succeq \boldsymbol{0}\right\}$$

is a convex cone.

A.2.3 Operations that Preserve Convexity

To establish that a set is convex, one can directly use the definition of convexity in (A.2). However, it can be cumbersome to prove that for any two points in the set the line segment is also contained in the set. A more interesting way to establish convexity in most practical cases is by showing that the set can be obtained from simple convex sets (e.g., hyperplanes, hyperspaces, balls, ellipsoids, cones) by operations that preserve convexity of sets, that is, a calculus of convex sets.

Some simple operations that preserve convexity of sets include: intersection of sets, composition with affine functions, and the perspective function.

Intersection

Convexity is preserved under intersection: if $\mathcal{S}_1$ and $\mathcal{S}_2$ are convex, then $\mathcal{S}_1 \cap \mathcal{S}_1$ is convex. This property extends to the case of intersection of multiple sets (even an infinite number of sets).

One trivial example is a polyhedron, which is the intersection of halfspaces and hyperplanes and therefore convex.

A more sophisticated example is the set

$$\mathcal{S} = \{\boldsymbol{x} \in \mathbb{R}^n \mid |p_{\boldsymbol{x}}(t)| \leq 1 \text{ for } |t| \leq \pi/3\},$$

where $p_{\boldsymbol{x}}(t) = x_1\cos(t) + x_2\cos(2t) + \cdots + x_n\cos(nt)$. Note that this set is an intersection of an infinite number of sets indexed by t.

Affine Composition

A function is *affine* if it has the form $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$, that is, the sum of a linear function and a constant.

Suppose $\mathcal{S} \subseteq \mathbb{R}^n$ is a convex set and $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is an affine function. Then the image of $\mathcal{S}$ under f,

$$f(\mathcal{S}) = \{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{S}\},$$

is convex.

Two trivial examples are *scaling* and *translation*. Another simple example is the *projection* of a convex set onto some of its coordinates: if $\mathcal{S} \subseteq \mathbb{R}^m \times \mathbb{R}^n$ is convex, then

$$\{\boldsymbol{x}_1 \in \mathbb{R}^m \mid (\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathcal{S} \text{ for some } \boldsymbol{x}_2 \in \mathbb{R}^n\}$$

is convex.

A useful example is the affine composition of the norm cone $\left\{(\boldsymbol{x}, t) \in \mathbb{R}^{n+1} \mid \|\boldsymbol{x}\| \leq t\right\}$:

$$\left\{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}\| \leq \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} + d\right\}.$$

Perspective Function

The perspective function scales or normalizes vectors so the last component is one, and then drops the last component.

Mathematically, we define the perspective function $P : \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^n$, with domain $\mathbf{dom}\, P = \mathbb{R}^n \times \mathbb{R}_{++}$, as $P(\boldsymbol{x}, t) = \boldsymbol{x}/t$.

Images and inverse images of convex sets under perspective functions are convex.

A.3 Convex Functions

We now provide a concise overview of convex functions. For more detailed information, the reader is referred to Chapter 3 in Boyd and Vandenberghe (2004).

Definition A.2 (Convex function) A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if the domain, $\operatorname{dom} f$, is a convex set and if for all $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} f$ and $0 \le \theta \le 1$, we have

$$f(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) \le \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y}). \tag{A.3}$$

Geometrically, this inequality means that the line segment between $(\boldsymbol{x}, f(\boldsymbol{x}))$ and $(\boldsymbol{y}, f(\boldsymbol{y}))$, which is the *chord* from $\boldsymbol{x}$ to $\boldsymbol{y}$, lies above the graph of f.

A function f is *strictly convex* if strict inequality holds in (A.3) whenever $\boldsymbol{x} \neq \boldsymbol{y}$ and $0 < \theta < 1$. We say f is *concave* if $-f$ is convex, and *strictly concave* if $-f$ is strictly convex.

For an affine function we always have equality in (A.3), so all affine (and therefore also linear) functions are both convex and concave. Conversely, any function that is convex and concave is affine.

A.3.1 Elementary Convex and Concave Functions

Apart from linear and affine functions, which are both convex and concave, it is good to become familiar with some elementary examples.

Let us start with some basic examples on $\mathbb{R}$:

- *exponential*: e^{ax} is convex on $\mathbb{R}$ for any $a \in \mathbb{R}$;
- *powers*: x^a is convex on $\mathbb{R}_{++}$ when $a \ge 1$ or $a \le 0$ (e.g., x^2), and concave for $0 \le a \le 1$;
- *powers of absolute value*: $|x|^p$ is convex on $\mathbb{R}$ for $p \ge 1$ (e.g., $|x|$);
- *logarithm*: $\log x$ is concave on $\mathbb{R}_{++}$;
- *negative entropy*: $x \log x$ is convex on $\mathbb{R}_{++}$.

Now, some interesting examples on $\mathbb{R}^n$:

- *quadratic function*: $f(\boldsymbol{x}) = \boldsymbol{x}^\mathsf{T}\boldsymbol{P}\boldsymbol{x} + 2\boldsymbol{q}^\mathsf{T}\boldsymbol{x} + r$ is convex on $\mathbb{R}^n$ if and only if $\boldsymbol{P} \succeq \boldsymbol{0}$;
- *norms*: every norm $\|\boldsymbol{x}\|$ is convex on $\mathbb{R}^n$ (e.g., $\|\boldsymbol{x}\|_\infty$, $\|\boldsymbol{x}\|_1$, and $\|\boldsymbol{x}\|_2$);
- *max function*: $f(x) = \max\{x_1, \ldots, x_n\}$ is convex on $\mathbb{R}^n$;
- *quadratic over linear function*: $f(x, y) = x^2/y$ is convex on $\mathbb{R} \times \mathbb{R}_{++}$;
- *geometric mean*: $f(\boldsymbol{x}) = \left(\prod_{i=1}^n x_i\right)^{1/n}$ is concave on $\mathbb{R}^n_{++}$;
- *log-sum-exp function*: $f(\boldsymbol{x}) = \log\left(e^{x_1} + \cdots + e^{x_n}\right)$ is convex on $\mathbb{R}^n$ (it can be used to approximate the function $f(\boldsymbol{x}) = \max\{x_1, \ldots, x_n\}$).

Finally, some examples on $\mathbb{R}^{n\times n}$:

- *log-determinant*: the function $f(\boldsymbol{X}) = \log\det(\boldsymbol{X})$ is concave on $\mathbb{S}^n_{++} = \{\boldsymbol{X} \in \mathbb{R}^{n\times n} \mid \boldsymbol{X} \succ \boldsymbol{0}\}$;
- *maximum eigenvalue*: the function

$$f(\boldsymbol{X}) = \lambda_{\max}(\boldsymbol{X}) \triangleq \sup_{\boldsymbol{y}\neq\boldsymbol{0}} \frac{\boldsymbol{y}^\mathsf{T}\boldsymbol{X}\boldsymbol{y}}{\boldsymbol{y}^\mathsf{T}\boldsymbol{y}}$$

  is convex on $\mathbb{S}^n$.

A.3.2 Epigraph

So far, we have used the adjective "convex" to describe both sets and functions, even though it refers to two distinct properties. Interestingly, these two can be linked, as we will demonstrate next.

The *graph* of a function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as the set

$$\left\{(\boldsymbol{x}, f(\boldsymbol{x})) \in \mathbb{R}^{n+1} \mid \boldsymbol{x} \in \operatorname{dom} f\right\},$$

which is a subset of $\mathbb{R}^{n+1}$.

The *epigraph* of a function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as the set

$$\operatorname{epi} f = \left\{(\boldsymbol{x}, t) \in \mathbb{R}^{n+1} \mid \boldsymbol{x} \in \operatorname{dom} f,\ f(\boldsymbol{x}) \leq t\right\}. \tag{A.4}$$

One way to conceptualize the epigraph is to envision pouring a bucket of water over the function and filling it up indefinitely.

The link between convex sets and convex functions is precisely via the epigraph: A function is convex if and only if its epigraph is a convex set,

$$f \text{ is convex} \quad \Longleftrightarrow \quad \operatorname{epi} f \text{ is convex.}$$

A.3.3 Characterization of Convex Functions

Apart from the definition of convexity, there are several ways to characterize convex functions such as restriction to a line, first-order condition, and second-order conditions.

Restriction of a Convex Function to a Line

A function is convex if and only if it is convex when restricted to any line that intersects its domain. In other words, $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if the function $g : \mathbb{R} \to \mathbb{R}$ defined as

$$g(t) = f(\boldsymbol{x} + t\boldsymbol{v})$$

is convex on its domain $\operatorname{dom} g = \{t \mid \boldsymbol{x} + t\boldsymbol{v} \in \operatorname{dom} f\}$, for any $\boldsymbol{x} \in \operatorname{dom} f$ and $\boldsymbol{v} \in \mathbb{R}^n$.

This property is very useful, since it allows us to check whether a function is convex by restricting it to a line, which is much easier (and can even be plotted in an exploratory analysis).

For example, the proof of the concavity of the log-determinant function $f(\boldsymbol{X}) = \log\det(\boldsymbol{X})$ can be reduced to the concavity of the log function:

$$\begin{aligned} g(t) = \log\det(\boldsymbol{X} + t\boldsymbol{X}) &= \log\det(\boldsymbol{X}) + \log\det\left(\boldsymbol{I} + t\boldsymbol{X}^{-1/2}\boldsymbol{V}\boldsymbol{X}^{-1/2}\right) \\ &= \log\det(\boldsymbol{X}) + \sum_{i=1}^{n} \log(1 + t\lambda_i), \end{aligned}$$

where the λ_i are the eigenvalues of $\boldsymbol{X}^{-1/2}\boldsymbol{V}\boldsymbol{X}^{-1/2}$.

First-Order Condition

For a differentiable function f, the gradient

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f(\boldsymbol{x})}{\partial x_n} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^n$$

exists at each point in $\operatorname{dom} f$, which is open. We can use it to write the first-order Taylor approximation of f near $\boldsymbol{x}$:

$$f(\boldsymbol{y}) \approx f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\mathsf{T}(\boldsymbol{y} - \boldsymbol{x}).$$

Suppose f is differentiable. Then f is convex if and only if $\operatorname{dom} f$ is convex and

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\mathsf{T}(\boldsymbol{y} - \boldsymbol{x}) \tag{A.5}$$

holds for all $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} f$.

Geometrically, the inequality (A.5) states that for a convex function, the first-order Taylor approximation is in fact a *global underestimator* of the function. Conversely, if the first-order Taylor approximation of a function is always a global underestimator of the function, then the function is convex.

The inequality (A.5) shows that from local information about a convex function (i.e., its value and derivative at a point) we can derive global information (i.e., a global underestimator of it). This is a remarkable property of convex functions and it justifies the connection between local optimality and global optimality in convex optimization problems.

Second-Order Condition

For a twice-differentiable function f, the Hessian

$$\nabla^2 f(\boldsymbol{x}) = \left(\frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j} \right)_{ij} \in \mathbb{R}^{n \times n}$$

exists at each point in $\operatorname{dom} f$, which is open. The Hessian can be used to write the second-order Taylor approximation of f near $\boldsymbol{x}$:

$$f(\boldsymbol{y}) \approx f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\mathsf{T}(\boldsymbol{y} - \boldsymbol{x}) + \frac{1}{2}(\boldsymbol{y} - \boldsymbol{x})^\mathsf{T} \nabla^2 f(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x}).$$

Suppose f is twice differentiable. Then f is convex if and only if $\operatorname{dom} f$ is convex and its Hessian is positive semidefinite:

$$\nabla^2 f(\boldsymbol{x}) \succeq \mathbf{0}$$

for all $\boldsymbol{x} \in \operatorname{dom} f$.

For a function on $\mathbb{R}$, this reduces to the simple condition $f''(x) \geq 0$ (and $\operatorname{dom} f$ convex, i.e., an interval), which means that the derivative is nondecreasing. The condition $\nabla^2 f(\boldsymbol{x}) \succeq \mathbf{0}$ can be interpreted geometrically as the requirement that the graph of the function has positive (upward) curvature at $\boldsymbol{x}$.

Similarly, f is concave if and only if $\operatorname{dom} f$ is convex and $\nabla^2 f(\boldsymbol{x}) \preceq \mathbf{0}$ for all $\boldsymbol{x} \in \operatorname{dom} f$.

A.3.4 Operations that Preserve Convexity

Thus far, we have explored four different methods to characterize the convexity of a function: applying the definition directly, restricting the function to a line, using the first-order condition, and employing the second-order condition. However, in most practical scenarios, a more intriguing approach to establish convexity is to demonstrate that the function can be derived from basic convex or concave functions (such as exponentials, powers, and norms) through operations that preserve the convexity of functions. This approach is essentially a calculus of convex functions.

Some simple operations that preserve convexity of functions include: nonnegative weighted sum, composition with affine functions, pointwise maximum, pointwise supremum, certain compositions with nonaffine functions, partial minimization, and perspective.

Nonnegative Weighted Sum

If f_1 and f_2 are both convex functions, then so is their sum $f_1 + f_2$. Also, scaling a function f with a nonnegative number $\alpha \geq 0$ preserves convexity. Combining nonnegative scaling and addition, we get that a nonnegative weighted sum of convex functions, with weights $w_1, \ldots, w_m \geq 0$,

$$f = w_1 f_1 + \cdots + w_m f_m,$$

is convex.

Composition with an Affine Mapping

Suppose $h : \mathbb{R}^m \to \mathbb{R}$, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, and $\boldsymbol{b} \in \mathbb{R}^m$. Define $f : \mathbb{R}^n \to \mathbb{R}$ as the composition of f with the affine mapping $\boldsymbol{Ax} + \boldsymbol{b}$:

$$f(\boldsymbol{x}) = h(\boldsymbol{Ax} + \boldsymbol{b}),$$

with $\operatorname{dom} f = \{\boldsymbol{x} \mid \boldsymbol{Ax} + \boldsymbol{b} \in \operatorname{dom} h\}$. Then, if f is convex, so is g; if f is concave, so is g.

For example, $f(\boldsymbol{x}) = \|\boldsymbol{y} - \boldsymbol{Ax}\|$ is convex and $f(\boldsymbol{X}) = \log\det\left(\boldsymbol{I} + \boldsymbol{HXH}^{\mathsf{T}}\right)$ is concave.

Pointwise Maximum and Supremum

If f_1 and f_2 are convex functions, then their pointwise maximum f, defined as

$$f(\boldsymbol{x}) = \max\left\{f_1(\boldsymbol{x}), f_2(\boldsymbol{x})\right\},$$

with $\operatorname{dom} f = \operatorname{dom} f_1 \cap \operatorname{dom} f_2$, is also convex. This property extends to more than two functions. If $f_1, \ldots, f_m$ are convex, then their pointwise maximum

$$f(\boldsymbol{x}) = \max\left\{f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})\right\}$$

is also convex.

For example, the sum of the r largest components of $\boldsymbol{x} \in \mathbb{R}^n$, $f(\boldsymbol{x}) = x_{[1]} + x_{[2]} + \cdots + x_{[r]}$, where $x_{[i]}$ is the ith largest component of $\boldsymbol{x}$, is convex because it can be written as the pointwise maximum

$$f(\boldsymbol{x}) = \max\left\{x_{i_1} + x_{i_2} + \cdots + x_{i_r} \mid 1 \leq i_1 < i_2 < \cdots < i_r \leq n\right\}.$$

The pointwise maximum property extends to the *pointwise supremum* over an infinite set of convex functions. If for each $\boldsymbol{y} \in \mathcal{Y}$, $f(\boldsymbol{x}, \boldsymbol{y})$ is convex in $\boldsymbol{x}$, then the pointwise supremum g, defined as

$$g(\boldsymbol{x}) = \sup_{\boldsymbol{y} \in \mathcal{Y}} f(\boldsymbol{x}, \boldsymbol{y}),$$

is convex in $\boldsymbol{x}$.

One simple example is distance to farthest point in a set C:

$$f(\boldsymbol{x}) = \sup_{\boldsymbol{y} \in C} \|\boldsymbol{x} - \boldsymbol{y}\|.$$

Another example is the maximum eigenvalue function of a symmetric matrix,

$$\lambda_{\max}(\boldsymbol{X}) = \sup_{\boldsymbol{y} \neq \boldsymbol{0}} \frac{\boldsymbol{y}^T \boldsymbol{X} \boldsymbol{y}}{\boldsymbol{y}^T \boldsymbol{y}}.$$

Composition with Arbitrary Functions

We have observed that the composition of a function with an affine mapping preserves convexity. We will now examine the conditions under which this can be generalized to non-affine mappings.

Suppose $h : \mathbb{R}^m \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$. Their composition $f = h \circ g : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$f(\boldsymbol{x}) = h(g(\boldsymbol{x}))$$

with $\operatorname{dom} f = \{\boldsymbol{x} \in \operatorname{dom} g \mid g(\boldsymbol{x}) \in \operatorname{dom} h\}$.

Let us start with the composition with a scalar mapping $m = 1$ for simplicity, so $h : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}$. The function $f(\boldsymbol{x}) = h(g(\boldsymbol{x}))$ satisfies

$$f \text{ is convex if } \begin{cases} h \text{ is convex nondecreasing and } g \text{ is convex} \\ h \text{ is convex nonincreasing and } g \text{ is concave} \end{cases}$$

and

$$f \text{ is concave if } \begin{cases} h \text{ is concave nondecreasing and } g \text{ is concave} \\ h \text{ is concave nonincreasing and } g \text{ is convex.} \end{cases}$$

For the case $n = 1$, one can easily derive the previous results from the second derivative of the composition function $f = h \circ g$ given by

$$f''(x) = h''(g(x))g'(x)^2 + h'(g(x))g''(x).$$

So, for example, if h is convex and nondecreasing, we have $h''(g(x)) \geq 0$ and $h'(g(x)) \geq 0$, and if g is convex then $g''(x) \geq 0$, which guarantees that $f''(x) \geq 0$ so f is convex.

Here are some examples:

- if g is convex, then $\exp g(x)$ is convex;
- if g is concave and positive, then $\log(g(x))$ is concave;
- if g is concave and positive, then $1/g(x)$ is convex;
- if g is convex and nonnegative, then $g(x)^p$ is convex for $p \geq 1$;
- if g is convex, then $-\log(-g(x))$ is convex on $\{x \mid g(x) < 0\}$.

Now let consider the general case of vector composition:

$$f(\boldsymbol{x}) = h(g(\boldsymbol{x})) = h(g_1(\boldsymbol{x}), \ldots, g_m(\boldsymbol{x}))$$

with $h : \mathbb{R}^m \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$. The rules in this case are generalized to:

$$f \text{ is convex if } \begin{cases} h \text{ is convex, nondecreasing in each argument, and } g_i \text{ are convex} \\ h \text{ is convex, nonincreasing in each argument, and } g_i \text{ are concave} \end{cases}$$

and

$$f \text{ is concave if } \begin{cases} h \text{ is concave, nondecreasing in each argument, and } g_i \text{ are concave} \\ h \text{ is concave, nonincreasing in each argument, and } g_i \text{ are convex.} \end{cases}$$

Partial Minimization

We have seen that the maximum or supremum of an arbitrary family of convex functions is convex. It turns out that some special forms of minimization also yield convex functions.

If $f(\boldsymbol{x}, \boldsymbol{y})$ is convex in $(\boldsymbol{x}, \boldsymbol{y})$ and C is a convex set, then the function

$$g(\boldsymbol{x}) = \inf_{\boldsymbol{y} \in C} f(\boldsymbol{x}, \boldsymbol{y})$$

is convex in $\boldsymbol{x}$. Note that the requirement here is for joint convexity in $(\boldsymbol{x}, \boldsymbol{y})$, unlike the case of supremum where the requirement is for convexity in $\boldsymbol{x}$ for any given $\boldsymbol{y}$.

One simple example is the distance to a set C:

$$f(\boldsymbol{x}) = \inf_{\boldsymbol{y} \in C} \|\boldsymbol{x} - \boldsymbol{y}\|,$$

which is convex if C is convex.

Perspective

Suppose $f : \mathbb{R}^n \to \mathbb{R}$, then the perspective of f is the function $g : \mathbb{R}^{n+1} \to \mathbb{R}$ defined as

$$g(\boldsymbol{x}, t) = t f(\boldsymbol{x}/t),$$

with domain $\operatorname{dom} g = \left\{(\boldsymbol{x}, t) \in \mathbb{R}^{n+1} \mid \boldsymbol{x}/t \in \operatorname{dom} f,\ t > 0\right\}$.

The perspective operation preserves convexity: If f is a convex function, then so is its perspective function g.

For example, since $f(\boldsymbol{x}) = \boldsymbol{x}^\mathsf{T}\boldsymbol{x}$ is convex, then its perspective $g(\boldsymbol{x}, t) = \boldsymbol{x}^\mathsf{T}\boldsymbol{x}/t$ is convex for $t > 0$.

Also, since the negative logarithm $f(x) = -\log x$ is convex, its perspective (known as the relative entropy function) $g(x, t) = t\log t - t\log x$ is also convex on $\mathbb{R}^2_{++}$.

A.3.5 Quasi-convex Functions

The *α-sublevel set* of a function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\mathcal{S}_\alpha = \{\boldsymbol{x} \in \operatorname{dom} f \mid f(\boldsymbol{x}) \leq \alpha\}.$$

Sublevel sets of a convex function are convex for any value of α. The converse is not true: a function can have all its sublevel sets convex, but not be a convex function. For example, $f(x) = -e^x$ is not convex on $\mathbb{R}$ (indeed, it is strictly concave) but all its sublevel sets are convex.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *quasi-convex* if its domain and all its sublevel sets $\mathcal{S}_\alpha$, for all α, are convex. A function f is *quasi-concave* if $-f$ is quasi-convex. A function that is both quasi-convex and quasi-concave is called *quasi-linear*.

For example, for a function on $\mathbb{R}$ to be quasi-convex, each sublevel set must be an interval. Here are some other illustrative examples:

- $\sqrt{|x|}$ is quasi-convex on $\mathbb{R}$;
- ceil $(x) = \inf\{z \in \mathbb{Z} \mid z \geq x\}$ is quasi-linear;
- $\log x$ is quasi-linear on $\mathbb{R}_{++}$;
- $f(x_1, x_2) = x_1 x_2$ is quasi-concave on $\mathbb{R}^2_{++}$;
- the linear-fractional function

$$f(\boldsymbol{x}) = \frac{\boldsymbol{a}^\mathsf{T}\boldsymbol{x} + b}{\boldsymbol{c}^\mathsf{T}\boldsymbol{x} + d}, \qquad \text{dom } f = \left\{\boldsymbol{x} \mid \boldsymbol{c}^\mathsf{T}\boldsymbol{x} + d > 0\right\}$$

  is quasi-linear.

We can always represent the sublevel sets of a quasi-convex function f (which are convex) via inequalities of convex functions:

$$f(\boldsymbol{x}) \leq t \quad \Longleftrightarrow \quad \phi_t(\boldsymbol{x}) \leq 0,$$

where $\phi_t(\boldsymbol{x})$ is a family of convex functions in $\boldsymbol{x}$ (indexed by t).

For example, consider a convex over concave function $f(\boldsymbol{x}) = p(\boldsymbol{x})/q(\boldsymbol{x})$, where $p(\boldsymbol{x}) \geq 0$ and $q(\boldsymbol{x}) > 0$. The function $f(\boldsymbol{x})$ is not convex but it is quasi-convex:

$$f(\boldsymbol{x}) \leq t \quad \Longleftrightarrow \quad p(\boldsymbol{x}) - tq(\boldsymbol{x}) \leq 0,$$

so we can take the convex function $\phi_t(\boldsymbol{x}) = p(\boldsymbol{x}) - tq(\boldsymbol{x})$ for $t \geq 0$.

A.4 Convex Optimization Problems

We will now delve into the basics of convex optimization problems. For more detailed information, readers are referred to Chapter 4 in Boyd and Vandenberghe (2004).

If the objective and inequality constraint functions of the general optimization problem (A.1) are convex and the equality constraint functions are linear (or, more generally, affine), the problem is then a *convex optimization problem* or *convex program*.

We can write a convex optimization problem in standard form as

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & f_0(\boldsymbol{x}) \\ \text{subject to} & f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{array} \tag{A.6}$$

where $f_0,\ f_1, \ldots, f_m$ are convex and the p equality constraints are affine with $\boldsymbol{A} \in \mathbb{R}^{p\times n}$ and $\boldsymbol{b} \in \mathbb{R}^p$.

Convex problems enjoy a rich body of theory and availability of algorithms with desirable convergence properties. However, most problems are not convex when naturally formulated. Reformulating a nonconvex problem in convex form may still be possible, but it is an art and there is no systematic way.

A fundamental property of convex optimization problems is that any locally optimal point is also (globally) optimal.

A.4.1 Optimality Characterization

Suppose f_0 is differentiable. Then, from the first-order characterization of convexity in Section A.3, we have that

$$f_0(\boldsymbol{y}) \geq f_0(\boldsymbol{x}) + \nabla f_0(\boldsymbol{x})^\mathsf{T}(\boldsymbol{y} - \boldsymbol{x})$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} f_0$.

Then, a feasible point $\boldsymbol{x}$ is optimal if and only if

$$\nabla f_0(\boldsymbol{x})^\mathsf{T}(\boldsymbol{y} - \boldsymbol{x}) \geq 0 \text{ for all } y \in \mathcal{X}, \tag{A.7}$$

where $\mathcal{X}$ denotes the feasible set. This is the so-called *minimum principle*. Geometrically, it means that the gradient $\nabla f_0(\boldsymbol{x})$ defines a supporting hyperplane.

The minimum principle in (A.7) may be difficult to manage in most practical cases. One more convenient characterization of optimality, when the feasible set $\mathcal{X}$ is explicitly given in terms of constraint functions, is the so-called *KKT optimality conditions* as elaborated in Section A.6. Two simple illustrative examples are considered next.

Example A.4 (Unconstrained minimization problem) For an unconstrained problem (i.e., $m = p = 0$ with feasible set $\mathcal{X} = \mathbb{R}^n$), the condition (A.7) reduces to the well-known necessary and sufficient condition

$$\nabla f_0(\boldsymbol{x}) = 0$$

for $\boldsymbol{x}$ to be optimal.

Example A.5 (Minimization over the nonnegative orthant) Consider the problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & f_0(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \geq \boldsymbol{0}, \end{array}$$

where the only inequality constraints are nonnegativity constraints on the variables and there are no equality constraints.

The optimality condition (A.7) becomes

$$\boldsymbol{x} \geq \boldsymbol{0}, \quad \nabla f_0(\boldsymbol{x})^\mathsf{T}(\boldsymbol{y} - \boldsymbol{x}) \geq \boldsymbol{0} \text{ for all } \boldsymbol{y} \geq \boldsymbol{0}.$$

The term $\nabla f_0(\boldsymbol{x})^\mathsf{T}\boldsymbol{y}$ is unbounded below on $\boldsymbol{y} \geq \boldsymbol{0}$, unless $\nabla f_0(\boldsymbol{x}) \geq \boldsymbol{0}$. The condition reduces

to $-\nabla f_0(\boldsymbol{x})^\mathsf{T}\boldsymbol{x} \geq \mathbf{0}$, which further becomes $\nabla f_0(\boldsymbol{x})^\mathsf{T}\boldsymbol{x} = \mathbf{0}$, due to $\boldsymbol{x} \geq \mathbf{0}$ and $\nabla f_0(\boldsymbol{x}) \geq \mathbf{0}$, and also elementwise $(\nabla f_0(\boldsymbol{x}))_i\, x_i = 0$. This means that if $x_i > 0$, that is, the ith component is in the interior of the feasible set, then $(\nabla f_0(\boldsymbol{x}))_i = 0$. So we can finally write the optimality conditions as

$$\boldsymbol{x} \geq \mathbf{0}, \quad \nabla f_0(\boldsymbol{x}) \geq \mathbf{0}, \quad (\nabla f_0(\boldsymbol{x}))_i\, x_i = 0, \quad i = 1, \ldots, n.$$

Note that the condition $(\nabla f_0(\boldsymbol{x}))_i\, x_i = 0$ implies that the two conditions $(\nabla f_0(\boldsymbol{x}))_i > 0$ and $x_i > 0$ cannot both be true. This is known as a *complementary* condition, which we will revisit in Section A.6.

A.4.2 Equivalent Reformulations

As previously mentioned, most problems are not convex when naturally formulated. In some cases, fortunately, there is a hidden convexity that can be unveiled by properly reformulating an equivalent problem. However, there is no systematic way to reformulate a problem in convex form: it is rather an art.

Two problems are considered *equivalent* if a solution to one can be easily converted into a solution for the other, and vice versa. A stricter form of this equivalence can be established by requiring a mapping between the two problems for every feasible point, not just for the optimal solutions.

Example A.6 Consider the problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & 1/\left(1 + x^2\right) \\ \text{subject to} & x^2 \geq 1, \end{array}$$

which is nonconvex (both the cost function and the constraint are nonconvex). It can be rewritten in convex form, after the change of variable $y = x^2$, as

$$\begin{array}{ll} \underset{y}{\text{minimize}} & 1/(1 + y) \\ \text{subject to} & y \geq 1, \end{array}$$

and the optimal points x can be recovered from the optimal y as $x = \pm\sqrt{y}$.

Example A.7 This example does not employ a change of variable, but transforms the problematic functions into more convenient ones. Consider the problem

$$\begin{array}{ll} \underset{x_1, x_2}{\text{minimize}} & x_1^2 + x_2^2 \\ \text{subject to} & x_1/\left(1 + x_2^2\right) \leq 0, \\ & \left(x_1 + x_2\right)^2 = 0, \end{array}$$

which is nonconvex (the inequality constraint function is nonconvex and the equality constraint function is not affine). It can be equivalently rewritten as the convex problem

$$\begin{array}{ll} \underset{x_1, x_2}{\text{minimize}} & x_1^2 + x_2^2 \\ \text{subject to} & x_1 \leq 0, \\ & x_1 = -x_2. \end{array}$$

Example A.8 The class of *geometric problems* is a very important example of nonconvex problems that can be reformulated in convex form by a change of variable. This is revisited in Section A.5.

We now explore some of the common tricks to obtain equivalent problems.

Change of Variables

Suppose ϕ is a one-to-one mapping from $\boldsymbol{z}$ to $\boldsymbol{x}$, then we can define $\tilde{f}_i(\boldsymbol{z}) = f_i(\phi(\boldsymbol{z}))$ and problem (A.6) can be rewritten (ignoring equality constraints) as

$$\begin{array}{ll} \underset{\boldsymbol{z}}{\text{minimize}} & \tilde{f}_0(\boldsymbol{z}) \\ \text{subject to} & \tilde{f}_i(\boldsymbol{z}) \leq 0, \quad i = 1, \ldots, m. \end{array}$$

Convexity may or may not be preserved depending on the mapping ϕ (see Section A.3 for details). With equality constraints, the mapping ϕ has to be affine to preserve the convexity of the problem.

Transformation of Objective and Constraint Functions

Suppose ψ_i are strictly increasing functions satisfying $\psi_i(0) = 0$. Then we can define $\tilde{f}_i(\boldsymbol{x}) = \psi_i\left(f_i(\boldsymbol{x})\right)$ and problem (A.6) can be rewritten (ignoring equality constraints) as

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \tilde{f}_0(\boldsymbol{x}) \\ \text{subject to} & \tilde{f}_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m. \end{array}$$

For equality constraints, the mapping has to be affine to preserve the convexity of the problem.

The requirements on the mappings can be relaxed; for example, for the inequality constraints we simply need $\psi_i(u) \leq 0$ if and only if $u \leq 0$.

Slack Variables

One simple transformation of interest is based on the observation that $f_i(\boldsymbol{x}) \leq 0$ if and only if there is an $s_i \geq 0$ that satisfies $f_i(\boldsymbol{x}) + s_i = 0$.

By introducing nonnegative *slack variables* $s_i \geq 0$, we can transform linear (or affine) inequalities $\boldsymbol{a}_i^\top \boldsymbol{x} \leq b_i$ into linear equalities $\boldsymbol{a}_i^\top \boldsymbol{x} + s_i = b_i$.

Eliminating Equality Constraints

Equality constraints in convex problems must be affine, that is, of the form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. From linear algebra, we know that the subspace of points satisfying such affine constraints can be written as $\boldsymbol{x} = \boldsymbol{F}\boldsymbol{z} + \boldsymbol{x}_0$, where $\boldsymbol{x}_0$ is any solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, $\boldsymbol{F}$ is a matrix whose range is the nullspace of $\boldsymbol{A}$, that is, $\boldsymbol{A}\boldsymbol{F} = \boldsymbol{0}$, and $\boldsymbol{z}$ is any vector of appropriate dimensions.

Then, the problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & f_0(\boldsymbol{x}) \\ \text{subject to} & f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array}$$

is equivalent to

$$\begin{array}{ll} \underset{z}{\text{minimize}} & f_0(\boldsymbol{F}\boldsymbol{z}+\boldsymbol{x}_0) \\ \text{subject to} & f_i(\boldsymbol{F}\boldsymbol{z}+\boldsymbol{x}_0) \le 0, \quad i=1,\ldots,m, \end{array}$$

with variable z. Since the composition of a convex function with an affine function is convex, eliminating equality constraints preserves the convexity of a problem.

Introducing Equality Constraints

We can introduce new variables and equality constraints into a convex optimization problem, provided the equality constraints are linear, and the resulting problem will also be convex.

For example, if an objective or constraint function has the form $f_i(\boldsymbol{A}_i\boldsymbol{x}+\boldsymbol{b}_i)$, we can introduce a new variable $\boldsymbol{y}_i$, replace $f_i(\boldsymbol{A}_i\boldsymbol{x}+\boldsymbol{b}_i)$ with $f_i(\boldsymbol{y}_i)$, and add the linear equality constraint $\boldsymbol{y}_i = \boldsymbol{A}_i\boldsymbol{x}+\boldsymbol{b}_i$.

Epigraph Problem Form

The *epigraph form* of the convex problem (A.6) is the problem

$$\begin{array}{ll} \underset{t,\boldsymbol{x}}{\text{minimize}} & t \\ \text{subject to} & f_0(\boldsymbol{x}) - t \le 0, \\ & f_i(\boldsymbol{x}) \le 0, \qquad i=1,\ldots,m, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{array} \tag{A.8}$$

with variables $\boldsymbol{x} \in \mathbb{R}^n$ and $t \in \mathbb{R}$. We can easily see that it is equivalent to the original problem: $(\boldsymbol{x}, t)$ is optimal for (A.8) if and only if $\boldsymbol{x}$ is optimal for (A.6) and $t = f_0(\boldsymbol{x})$.

It is often stated that a linear objective is universal for convex optimization, as any convex optimization problem can be readily transformed into one with a linear objective. This transformation can aid in theoretical analysis and algorithm development.

Minimizing Over Some Variables

We always have

$$\inf_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y}) = \inf_{\boldsymbol{x}} \tilde{f}(\boldsymbol{x}),$$

where $\tilde{f}(\boldsymbol{x}) = \inf_{\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y})$. In addition, if $f(\boldsymbol{x},\boldsymbol{y})$ is jointly convex in $\boldsymbol{x}$ and $\boldsymbol{y}$, that is, it is convex in $(\boldsymbol{x},\boldsymbol{y})$, then $\tilde{f}(\boldsymbol{x})$ is convex.

In plain words, we can always minimize a function by first minimizing over some set of variables, and then minimizing over the remaining ones. Note that this is a *nested minimization*, in the sense that as $\boldsymbol{x}$ changes, then implicitly the $\boldsymbol{y}$ that minimizes $f(\boldsymbol{x},\boldsymbol{y})$ to obtain $\tilde{f}(\boldsymbol{x})$ changes as well. Do not confuse this nested minimization with an alternate minimization method, where one optimizes alternately over $\boldsymbol{x}$ and $\boldsymbol{y}$ until convergence is achieved.

Suppose the variable $\boldsymbol{x} \in \mathbb{R}^n$ is partitioned into two blocks as $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$. Then, the convex problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & f_0(\boldsymbol{x}_1,\boldsymbol{x}_2) \\ \text{subject to} & f_i(\boldsymbol{x}_1) \le 0, \quad i=1,\ldots,m_1, \\ & \tilde{f}_i(\boldsymbol{x}_2) \le 0, \quad i=1,\ldots,m_2, \end{array}$$

in which each of the constraints involves either $\boldsymbol{x}_1$ or $\boldsymbol{x}_2$, is equivalent to the convex problem

$$\begin{array}{ll} \underset{\boldsymbol{x}_1}{\text{minimize}} & \tilde{f}_0(\boldsymbol{x}_1) \\ \text{subject to} & f_i(\boldsymbol{x}_1) \leq 0, \quad i = 1, \ldots, m_1, \end{array}$$

where

$$\tilde{f}_0(\boldsymbol{x}_1) = \inf_{\boldsymbol{z}} \{ f_0(\boldsymbol{x}_1, \boldsymbol{z}) \mid \tilde{f}_i(\boldsymbol{z}) \leq 0, \ i = 1, \ldots, m_2 \}.$$

A.4.3 Approximate Reformulations

In many cases, the formulated optimization problem may still remain nonconvex despite all attempts to use some transformation to unveil any possible hidden convexity. In such situations, one can resort to some kind of approximation to form an *approximated problem*, possibly convex, that is easy to solve:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \tilde{f}_0(\boldsymbol{x}) \\ \text{subject to} & \tilde{f}_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{array}$$

where $\tilde{f}_i(\boldsymbol{x}) \approx f_i(\boldsymbol{x})$.

There are three types of approximations:

1. *Conservative* approximation or *tightened* approximation leading to a *tightened formulation*: $\tilde{f}_i(\boldsymbol{x}) \geq f_i(\boldsymbol{x})$. This approximation defines a feasible set that is a subset of the original feasible set, which guarantees the feasibility of the approximated solution.

2. *Relaxed* approximation leading to a *relaxed formulation* or simply a *relaxation*: $\tilde{f}_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x})$. This approximation defines a feasible set that is a superset of the original feasible set. It does not guarantee the feasibility of the approximated solution and may require an additional step to enforce feasibility.

3. Approximation without any guarantees: $\tilde{f}_i(\boldsymbol{x}) \approx f_i(\boldsymbol{x})$.

Relaxed formulations are very commonly used in practice, often by simply removing some of the constraints (typically the more difficult ones). A notable example of this approach is found in Bengtsson and Ottersten (2001) for multiuser beamforming in wireless communications, where a nonconvex rank-one constraint was removed (thus relaxing the problem). However, it was still proven to be an equivalent reformulation and not a relaxation.

Example A.9 Suppose a problem contains the nonconvex constraint $x^2 = 1$ or, equivalently, $x \in \{\pm 1\}$, which is a nonconvex discrete set. This is typically associated with combinatorial optimization, which has exponential complexity. A relaxation would involve enlarging the feasible set, which could be achieved by using instead the interval $-1 \leq x \leq 1$. On the other hand, a tightening would involve reducing the feasible set, which could be accomplished simply by using $x = 1$. Both approximations, $-1 \leq x \leq 1$ and $x = 1$, are convex and can be easily handled.

A.4.4 Quasi-convex Optimization

A *quasi-convex* optimization problem has the standard form

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & f_0(\boldsymbol{x}) \\ \text{subject to} & f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{array} \tag{A.9}$$

where the inequality constraint functions $f_1, \ldots, f_m$ are convex, and the objective f_0 is quasi-convex, unlike in a convex optimization problem where the objective is convex. For details on quasi-convex functions, see Section A.3.5.

The most significant difference between convex and quasi-convex optimization is that a quasi-convex optimization problem can have locally optimal solutions that are not globally optimal. For instance, this can occur when the function becomes flat before reaching the optimal value.

A general approach to quasi-convex optimization relies on the representation of the sublevel sets of a quasi-convex function via a family of convex inequalities:

$$f(\boldsymbol{x}) \leq t \quad \Longleftrightarrow \quad \phi_t(\boldsymbol{x}) \leq 0,$$

where $\phi_t(\boldsymbol{x})$ is a family of convex functions in $\boldsymbol{x}$ (indexed by t).

Let $p^\star$ denote the optimal value of the quasi-convex optimization problem (A.9). If the (now convex) feasibility problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{find}} & \boldsymbol{x} \\ \text{subject to} & \phi_t(\boldsymbol{x}) \leq 0, \\ & f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \tag{A.10}$$

is feasible, then we have $p^\star \leq t$. Conversely, if it is infeasible, then $p^\star > t$.

This observation can serve as the foundation for a simple algorithm to solve quasi-convex optimization problems termed the *bisection method*. This involves solving a sequence of convex feasibility problems, like the one in (A.10), at each step. Suppose that the original problem (A.9) is feasible and that we start with an interval $[l, u]$ known to contain the optimal value $p^\star$. We can then solve the convex feasibility problem at the interval midpoint $t = (l + u)/2$ to determine whether the optimal value is in the lower or upper half of this interval, and update the interval accordingly. This produces a new interval, which also contains the optimal value, but has half the width of the initial interval; so the length of the interval after k iterations is $2^{-k}(u - l)$, where $(u - l)$ is the length of the initial interval. Therefore, if a tolerance of ϵ is desired in the computation of $p^\star$, the number of iterations is $\lceil \log_2 ((u - l)/\epsilon) \rceil$, where $\lceil \cdot \rceil$ denoted the ceiling rounding operation. The bisection method (a.k.a. *sandwich technique*) is summarized in Algorithm A.1.

Algorithm A.1: Bisection method for the quasi-convex optimization problem in (A.9).

Choose interval $[l, u]$ containing $p^\star$ and tolerance $\epsilon > 0$;
repeat
  $t \leftarrow (l + u)/2$;
  Solve the convex feasibility problem (A.10);
  if feasible **then**
    $u \leftarrow t$ and keep solution $\boldsymbol{x}$;
  else
    $l \leftarrow t$;
  end if
until $u - l \leq \epsilon$;

A.5 Taxonomy of Convex Problems

A convex problem, such as the one in (A.6), can be further classified into different specific types of problems. These can be identified by abbreviations such as LP, QP, QCQP, SOCP, SDP, CP, FP, LFP, and GP, which we will briefly explore next. For more detailed information, readers are referred to Chapter 4 in Boyd and Vandenberghe (2004). In traditional optimization literature, a problem is also referred to as a *program*. Therefore, for example, a linear program is the same as a linear problem.

This classification is beneficial for both theoretical and algorithmic purposes. For instance, solvers are designed to handle specific types of problems (see Section B.1 in Appendix B for more details on solvers).

A.5.1 Linear Programming

When the objective and constraint functions are all affine, a problem is called a *linear program* or *linear problem* (LP):

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{c}^\mathsf{T}\boldsymbol{x} + d \\ \text{subject to} & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h}, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{array}$$

where the parameters $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$, d, $\boldsymbol{G}$, and $\boldsymbol{h}$ are of appropriate size. Linear programs are, of course, convex optimization problems.

The geometric interpretation of an LP can be visualized as a polyhedron on an inclined flat surface, where an optimal solution is always located at a corner of the polyhedron. This observation forms the basis of the popular simplex method, developed by Dantzig in 1947, for solving LPs.

While problems invoving only linear functions are easily recognizable as LPs, some formulations involving ℓ_∞-norm and ℓ_1-norm minimization can also be rewritten as LPs, as shown next.

Example A.10 (ℓ_∞-norm minimization as an LP) The problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \|\boldsymbol{x}\|_\infty \\ \text{subject to} & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h}, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{array}$$

is equivalent to the LP

$$\begin{array}{ll} \underset{t,\boldsymbol{x}}{\text{minimize}} & t \\ \text{subject to} & -t\mathbf{1} \leq \boldsymbol{x} \leq t\mathbf{1}, \\ & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h}, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{array}$$

Example A.11 (ℓ_1-norm minimization as an LP) The problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \|\boldsymbol{x}\|_1 \\ \text{subject to} & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h}, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array}$$

is equivalent to the LP

$$\begin{array}{ll} \underset{\boldsymbol{t},\boldsymbol{x}}{\text{minimize}} & \sum_i t_i \\ \text{subject to} & -\boldsymbol{t} \leq \boldsymbol{x} \leq \boldsymbol{t}, \\ & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h}, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{array}$$

A.5.2 Linear-Fractional Programming

The problem of minimizing a ratio of affine functions over a polyhedron is called a *linear-fractional program* (LFP):

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \left(\boldsymbol{c}^\mathsf{T}\boldsymbol{x} + d\right) / \left(\boldsymbol{e}^\mathsf{T}\boldsymbol{x} + f\right) \\ \text{subject to} & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h}, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \tag{A.11}$$

with $\operatorname{dom} f_0 = \left\{\boldsymbol{x} \mid \boldsymbol{e}^\mathsf{T}\boldsymbol{x} + f > 0\right\}$.

An LFP is not a convex problem, but it is quasi-convex. Therefore, problem (A.11) can be solved via bisection (see Algorithm A.1) by sequentially solving a series of feasibility LPs of the form:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{find}} & \boldsymbol{x} \\ \text{subject to} & t\left(\boldsymbol{e}^\mathsf{T}\boldsymbol{x} + f\right) \geq \boldsymbol{c}^\mathsf{T}\boldsymbol{x} + d, \\ & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h}, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{array}$$

Alternatively, the LFP in (A.11) can be transformed into the following LP via the Charnes–

Cooper transform (Bajalinov, 2003; Charnes & Cooper, 1962):

$$\begin{array}{ll} \underset{\boldsymbol{y},t}{\text{minimize}} & \boldsymbol{c}^\top \boldsymbol{y} + dt \\ \text{subject to} & \boldsymbol{G}\boldsymbol{y} \leq \boldsymbol{h}t, \\ & \boldsymbol{A}\boldsymbol{y} = \boldsymbol{b}t, \\ & \boldsymbol{e}^\top \boldsymbol{y} + ft = 1, \\ & t > 0, \end{array}$$

with variables $\boldsymbol{y}, t$, related to the original variable $\boldsymbol{x}$ as

$$\boldsymbol{y} = \frac{\boldsymbol{x}}{\boldsymbol{e}^\top \boldsymbol{x} + f} \quad \text{and} \quad t = \frac{1}{\boldsymbol{e}^\top \boldsymbol{x} + f}.$$

The original variable can be easily recovered from $\boldsymbol{y}, t$ as $\boldsymbol{x} = \boldsymbol{y}/t$. For details on the Charnes–Cooper transform, the reader is referred to Section B.5.3 in Appendix B.

A.5.3 Quadratic Programming

The convex optimization problem (A.6) is called a *quadratic program* (QP) if the objective function is (convex) quadratic, and the constraint functions are affine:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x} + r \\ \text{subject to} & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h}, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{array}$$

where $\boldsymbol{P} \succeq \boldsymbol{0}$. QPs include LPs as special case when $\boldsymbol{P} = \boldsymbol{0}$.

The geometric interpretation of a QP can be visualized as an elliptical surface intersecting a polyhedron, where the optimal solution does not necessarily coincide with a vertex of the polyhedron.

Example A.12 (Least squares) The LS problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$$

is an unconstrained QP.

Example A.13 (Box-constrained LS) The following regression problem with upper and lower bounds on the variables,

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \\ \text{subject to} & l_i \leq x_i \leq u_i, \quad i = 1, \ldots, n, \end{array}$$

is a QP.

If the objective function in (A.6) as well as the inequality constraints are (convex) quadratic, then the problem is called a *quadratically constrained quadratic program* (QCQP):

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{P}_0 \boldsymbol{x} + \boldsymbol{q}_0^\top \boldsymbol{x} + r_0 \\ \text{subject to} & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{P}_i \boldsymbol{x} + \boldsymbol{q}_i^\top \boldsymbol{x} + r_i \leq 0, \quad i = 1, \ldots, m, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{array}$$

where $\boldsymbol{P}_i \succeq \boldsymbol{0}$. In this case, the feasible region is the intersection of ellipsoids. QCQPs include QPs as a special case when $\boldsymbol{P}_i = \boldsymbol{0}$ for $i = 1, \ldots, m$.

A.5.4 Second-Order Cone Programming

A convex problem that is closely related to quadratic programming is the *second-order cone program* (SOCP):

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{f}^\top \boldsymbol{x} \\ \text{subject to} & \|\boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i\| \leq \boldsymbol{c}_i^\top \boldsymbol{x} + d_i, \quad i = 1, \ldots, m, \\ & \boldsymbol{F}\boldsymbol{x} = \boldsymbol{g}, \end{array}$$

where the constraints $\|\boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i\| \leq \boldsymbol{c}_i^\top \boldsymbol{x} + d_i$ are called *second-order cone* (SOC) constraints since they are affine compositions of the (convex) SOC

$$\left\{(\boldsymbol{x}, t) \in \mathbb{R}^{n+1} \mid \|\boldsymbol{x}\| \leq t\right\}. \tag{A.12}$$

An SOCP reduces to a QCQP when $c_i = 0$ for $i = 1, \ldots, m$ (by squaring both sides of the inequalities). If each $\boldsymbol{A}_i$ is a row-vector (or $\boldsymbol{A}_i = 0$), then an SOCP reduces to an LP.

A comprehensive monograph on SOCPs is Lobo et al. (1998).

Example A.14 (Robust LP) Consider the linear program

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} & \boldsymbol{a}_i^\top \boldsymbol{x} \leq b_i, \quad i = 1, \ldots, m. \end{array}$$

Now suppose there is some uncertainty in the parameters $\boldsymbol{a}_i$ and they are known to lie in given ellipsoids,

$$\mathcal{E}_i = \{\bar{\boldsymbol{a}}_i + \boldsymbol{P}_i \boldsymbol{u} \mid \|\boldsymbol{u}\| \leq 1\},$$

where $\boldsymbol{P}_i \in \mathbb{R}^{n \times n}$. If we require that the constraints be satisfied for all possible values of the parameters $\boldsymbol{a}_i$, we obtain a *robust LP*:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} & \boldsymbol{a}_i^\top \boldsymbol{x} \leq b_i, \text{ for all } \boldsymbol{a}_i \in \mathcal{E}_i, \quad i = 1, \ldots, m. \end{array}$$

The robust constraint $\boldsymbol{a}_i^\top \boldsymbol{x} \leq b_i$ for all $\boldsymbol{a}_i \in \mathcal{E}_i$ can equivalently expressed as

$$\sup\left\{\boldsymbol{a}_i^\top \boldsymbol{x} \mid \boldsymbol{a}_i \in \mathcal{E}_i\right\} = \bar{\boldsymbol{a}}_i^\top \boldsymbol{x} + \left\|\boldsymbol{P}_i^\top \boldsymbol{x}\right\|_2 \leq b_i.$$

Hence, the robust LP can be expressed as the SOCP

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} & \bar{\boldsymbol{a}}_i^\top \boldsymbol{x} + \left\|\boldsymbol{P}_i^\top \boldsymbol{x}\right\|_2 \leq b_i, \quad i = 1, \ldots, m. \end{array}$$

A.5.5 Semidefinite Programming

A more general convex problem than an SOCP is the *semidefinite program* (SDP), formulated as

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} \\ \text{subject to} & x_1\boldsymbol{F}_1 + x_2\boldsymbol{F}_2 + \cdots + x_n\boldsymbol{F}_n + \boldsymbol{G} \preceq \boldsymbol{0}, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{array}$$

which has (convex) *linear matrix inequality* (LMI) constraints of the form

$$x_1\boldsymbol{F}_1 + \cdots + x_n\boldsymbol{F}_n + \boldsymbol{G} \preceq \boldsymbol{0}, \tag{A.13}$$

where $\boldsymbol{F}_1, \ldots, \boldsymbol{F}_n, \boldsymbol{G} \in \mathbb{S}^k$ ($\mathbb{S}^k$ is the set of symmetric $k \times k$ matrices) and $\boldsymbol{A} \succeq \boldsymbol{B}$ means that $\boldsymbol{A} - \boldsymbol{B}$ is positive semidefinite. Note that multiple LMI constraints can always be written as a single one by using block diagonal matrices.

An SDP reduces to an LP when the matrix in the LMI inequality is diagonal, that is, the SDP

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} \\ \text{subject to} & \operatorname{Diag}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \preceq \boldsymbol{0} \end{array}$$

is equivalent to the LP

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}. \end{array}$$

An SDP can also be reduced to an SOCP when the matrix in the LMI has a specific 2×2 block structure. In this case, the SDP

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{f}^{\mathsf{T}}\boldsymbol{x} \\ \text{subject to} & \begin{bmatrix} (\boldsymbol{c}_i^{\mathsf{T}}\boldsymbol{x} + d_i)\,\boldsymbol{I} & \boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i \\ (\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i)^{\mathsf{T}} & \boldsymbol{c}_i^{\mathsf{T}}\boldsymbol{x} + d_i \end{bmatrix} \succeq \boldsymbol{0}, \quad i = 1, \ldots, m, \end{array}$$

is equivalent to the SOCP

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{f}^{\mathsf{T}}\boldsymbol{x} \\ \text{subject to} & \|\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i\| \leq \boldsymbol{c}_i^{\mathsf{T}}\boldsymbol{x} + d_i, \quad i = 1, \ldots, m. \end{array}$$

The equivalence can be shown via the *Schur complement*:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^{\mathsf{T}} & \boldsymbol{C} \end{bmatrix} \succeq \boldsymbol{0} \quad \Longleftrightarrow \quad \boldsymbol{S} = \boldsymbol{C} - \boldsymbol{B}^{\mathsf{T}}\boldsymbol{A}^{-1}\boldsymbol{B} \succeq \boldsymbol{0},$$

where we have tacitly assumed $\boldsymbol{A} \succ \boldsymbol{0}$.

A comprehensive monograph on SDPs is Vandenberghe and Boyd (1996).

Example A.15 (Eigenvalue minimization) The maximum eigenvalue minimization problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \lambda_{\max}\left(\boldsymbol{A}(\boldsymbol{x})\right),$$

where $\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{A}_0 + x_1\boldsymbol{A}_1 + \cdots + x_n\boldsymbol{A}_n$, is equivalent to the SDP

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & t \\ \text{subject to} & \boldsymbol{A}(\boldsymbol{x}) \preceq t\boldsymbol{I}. \end{array}$$

This follows from

$$\lambda_{\max}\left(\boldsymbol{A}(\boldsymbol{x})\right) \leq t \quad \Longleftrightarrow \quad \boldsymbol{A}(\boldsymbol{x}) \preceq t\boldsymbol{I}.$$

A.5.6 Conic Programming

A useful generalization of the standard convex optimization problem (A.6) can be achieved by allowing the inequality constraints to be vector-valued and incorporating generalized inequalities into the constraints:

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimize}} & f_0(\boldsymbol{x}) & \\ \text{subject to} & \boldsymbol{f}_i(\boldsymbol{x}) \preceq_{\mathcal{K}_i} \boldsymbol{0}, & 1 \leq i \leq m, \\ & \boldsymbol{h}_i(\boldsymbol{x}) = \boldsymbol{0}, & 1 \leq i \leq p, \end{array}$$

where the generalized inequalities[2] '$\preceq_{\mathcal{K}_i}$' are defined by the proper cones $\mathcal{K}_i$ (note that $\boldsymbol{a} \preceq_{\mathcal{K}} \boldsymbol{b} \Leftrightarrow \boldsymbol{b} - \boldsymbol{a} \in \mathcal{K}$) and f_i are $\mathcal{K}_i$-convex[3] (see Section A.7 for more details on generalized inequalities).

Among the simplest convex optimization problems with generalized inequalities are *cone programs* (CP), or *conic-form problems*, which have a linear objective and one inequality constraint function (Ben-Tal & Nemirovski, 2001; Boyd & Vandenberghe, 2004):

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} \\ \text{subject to} & \boldsymbol{F}\boldsymbol{x} + \boldsymbol{g} \preceq_{\mathcal{K}} \boldsymbol{0}, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{array} \tag{A.14}$$

CPs particularize nicely to LPs, SOCPs, and SDPs as follows:

- If $\mathcal{K} = \mathbb{R}^n_+$ (nonnegative orthant), the partial ordering $\preceq_{\mathcal{K}}$ is the usual componentwise inequality between vectors and the CP (A.14) reduces to an LP.
- If $\mathcal{K} = C^n$ (second-order cone), $\preceq_{\mathcal{K}}$ corresponds to a constraint of the form (A.12) and the CP (A.14) becomes an SOCP.
- If $\mathcal{K} = \mathbb{S}^n_+$ (positive semidefinite cone), the generalized inequality $\preceq_{\mathcal{K}}$ reduces to the usual matrix inequality as in (A.13) and the CP (A.14) simplifies to an SDP.

[2] A generalized inequality is a partial ordering on $\mathbb{R}^n$ that has many of the properties of the standard ordering on $\mathbb{R}$. A common example is the matrix inequality defined by the cone of positive semidefinite $n \times n$ matrices $\mathbb{S}^n_+$.

[3] A function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^{k_i}$ is $\mathcal{K}_i$-convex if the domain is a convex set and, for all $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} f$ and $\theta \in [0, 1]$, $\boldsymbol{f}\left(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}\right) \preceq_{\mathcal{K}_i} \theta\boldsymbol{f}(\boldsymbol{x}) + (1-\theta)\boldsymbol{f}(\boldsymbol{y})$.

A.5.7 Fractional Programming

Fractional programming is a family of optimization problems that involve ratios. Its origins can be traced back to a 1937 paper on economic expansion (von Neumann, 1937). Since then, it has inspired research in various fields such as economics, management science, optics, information theory, communication systems, graph theory, and computer science.

The simplest form of a *fractional program* (FP) is

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{maximize}} & \dfrac{f(\boldsymbol{x})}{g(\boldsymbol{x})} \\ \text{subject to} & \boldsymbol{x} \in \mathcal{X}, \end{array} \tag{A.15}$$

where $f(\boldsymbol{x}) \geq 0$, $g(\boldsymbol{x}) > 0$, and $\mathcal{X}$ denotes the feasible set. One particular case is the LFP in (A.11), when both f and g are linear functions.

FPs have been widely studied and extended to deal with multiple ratios such as

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{maximize}} & \min\limits_{i} \dfrac{f_i(\boldsymbol{x})}{g_i(\boldsymbol{x})} \\ \text{subject to} & \boldsymbol{x} \in \mathcal{X}. \end{array}$$

FPs are nonconvex problems, which in principle makes them challenging to solve (Stancu-Minasian, 1997). Fortunately, in the case known as *concave–convex FP*, where f is a concave function and g is a convex function, they can be solved relatively easily using different methods. The three main approaches, covered in detail in Section B.5 of Appendix B, are:

- *Bisection method*: Similar to the linear case of LFP, the bisection method (see Algorithm A.1) involves solving a sequence of convex feasibility problems of the form

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{find}} & \boldsymbol{x} \\ \text{subject to} & tg(\boldsymbol{x}) \leq f(\boldsymbol{x}), \\ & \boldsymbol{x} \in \mathcal{X}. \end{array}$$

- *Dinkelbach's transform*: This approach eliminates the fractional objective by solving a sequence of simpler convex problems of the form

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{maximize}} & f(\boldsymbol{x}) - y^k g(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathcal{X}, \end{array} \tag{A.16}$$

  where the weight y^k is updated as $y^k = f(\boldsymbol{x}^k)/g(\boldsymbol{x}^k)$ (see Section B.5.2).

- *Schaible transform*: This is a more general case of the Charnes–Cooper transform (used for LFPs) proposed by Schaible (1974). The original concave–convex FP is thus transformed into an equivalent convex problem,

$$\begin{array}{ll} \underset{\boldsymbol{y},t}{\text{maximize}} & tf\left(\frac{\boldsymbol{y}}{t}\right) \\ \text{subject to} & tg\left(\frac{\boldsymbol{y}}{t}\right) \leq 1, \\ & t > 0, \\ & \boldsymbol{y}/t \in \mathcal{X}, \end{array}$$

with variables $\boldsymbol{y}, t$, related to the original variable $\boldsymbol{x}$ by

$$\boldsymbol{y} = \frac{\boldsymbol{x}}{g(\boldsymbol{x})} \quad \text{and} \quad t = \frac{1}{g(\boldsymbol{x})}.$$

The original variable can be easily recovered from $\boldsymbol{y}, t$ by $\boldsymbol{x} = \boldsymbol{y}/t$ (see Section B.5.3).

A.5.8 Geometric Programming

We will now examine a family of optimization problems that are not naturally convex, but can be converted into convex optimization problems through a change of variables and a transformation of the objective and constraint functions.

A *monomial function*, or simply a *monomial*, is a function $f : \mathbb{R}^n \to \mathbb{R}$ with $\operatorname{dom} f = \mathbb{R}^n_{++}$ defined as

$$f(\boldsymbol{x}) = c x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n},$$

where $c > 0$ and $a_i \in \mathbb{R}$.

A *posynomial function*, or simply a *posynomial*, is a sum of monomials:

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} c_k x_1^{a_{1k}} x_2^{a_{2k}} \cdots x_n^{a_{nk}},$$

where $c_k > 0$.

A *geometric program* (GP) is a (nonconvex) problem of the form

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimize}} & f_0(\boldsymbol{x}) & \\ \text{subject to} & f_i(\boldsymbol{x}) \le 1, & i = 1, \ldots, m, \\ & h_i(\boldsymbol{x}) = 1, & i = 1, \ldots, p, \end{array}$$

where $f_0, \ldots, f_m$ are posynomials and $h_1, \ldots, h_p$ are monomials. The domain of this problem is $\mathcal{D} = \mathbb{R}^n_{++}$, i.e., the constraint $\boldsymbol{x} > \mathbf{0}$ is implicit.

Suppose we apply the change of variables $y_i = \log x_i$ and $b = \log c$ on the monomial $f(\boldsymbol{x}) = c x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}$. Then the log of the monomial is

$$\tilde{f}(\boldsymbol{y}) = \log f(e^{\boldsymbol{x}}) = b + a_1 y_1 + a_2 y_2 + \cdots + a_n y_n = b + \boldsymbol{a}^\mathsf{T} \boldsymbol{y},$$

which is an affine function.

Similarly, for a posynomial, we obtain

$$\tilde{f}(\boldsymbol{y}) = \log \sum_{k=1}^{K} e^{b_k + \boldsymbol{a}_k^\mathsf{T} \boldsymbol{y}},$$

which is the so-called log-sum-exp function, a convex function.

Finally, the resulting transformed GP in convex form is

$$\begin{array}{lll} \underset{\boldsymbol{y}}{\text{minimize}} & \log \sum_{k=1}^{K_0} e^{b_{0k} + \boldsymbol{a}_{0k}^\mathsf{T} \boldsymbol{y}} & \\ \text{subject to} & \log \sum_{k=1}^{K_i} e^{b_{ik} + \boldsymbol{a}_{ik}^\mathsf{T} \boldsymbol{y}} \le 0, & i = 1, \ldots, m, \\ & h_i + \boldsymbol{g}_i^\mathsf{T} \boldsymbol{y} = 0, & i = 1, \ldots, p. \end{array}$$

Some comprehensive monographs on GP include Boyd et al. (2007) and Chiang (2005).

A.6 Lagrange Duality

Lagrange duality theory is a very rich and mature theory that links the original minimization problem (A.1), termed *primal problem*, with a maximization problem, termed *dual problem*. In some occasions, it is simpler to solve the dual problem than the primal one. A fundamental result in duality theory is given by the Karush–Kuhn–Tucker (KKT) optimality conditions that any primal-dual solution must satisfy. By exploring the KKT conditions, it is possible in many cases to obtain a closed-form solution to the original problem.

We next overview the basic results on duality theory, including the KKT conditions. For further details the reader is referred to Chapter 5 in Boyd and Vandenberghe (2004), Bertsekas (1999), and Bertsekas et al. (2003).

A.6.1 Lagrangian

Recall the optimization problem in standard form (A.1) (not necessarily convex):

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimize}} & f_0(\boldsymbol{x}) & \\ \text{subject to} & f_i(\boldsymbol{x}) \leq 0, & i = 1, \ldots, m, \\ & h_i(\boldsymbol{x}) = 0, & i = 1, \ldots, p, \end{array} \tag{A.17}$$

with variable $\boldsymbol{x} \in \mathbb{R}^n$, domain $\mathcal{D}$, and optimal value $p^\star$.

The basic idea in Lagrange duality is to take the constraints in (A.17) into account by augmenting the objective function with a weighted sum of the constraint functions. The *Lagrangian* $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ associated with the problem (A.17) is defined as

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i h_i(\boldsymbol{x}),$$

with $\operatorname{dom} L = \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$. We refer to λ_i and ν_i as the *Lagrange multipliers* associated with the ith inequality constraint $f_i(\boldsymbol{x}) \leq 0$ and with the ith equality constraint $h_i(\boldsymbol{x}) = 0$, respectively. The vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ are then called *dual variables* or *Lagrange multiplier vectors* associated with the problem (A.17).

We define the *Lagrange dual function* (or just *dual function*) $g : \times\mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ as the minimum value of the Lagrangian over $\boldsymbol{x}$ for a given $(\boldsymbol{\lambda}, \boldsymbol{\nu})$:

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x} \in \mathcal{D}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x} \in \mathcal{D}} \left\{ f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i h_i(\boldsymbol{x}) \right\}. \tag{A.18}$$

Note that the infimum in (A.18) is with respect to all $\boldsymbol{x} \in \mathcal{D}$ (not necessarily feasible points). When the Lagrangian is unbounded below in $\boldsymbol{x}$, the dual function takes on the value $-\infty$. Since the dual function is the pointwise infimum of a family of affine functions of $(\boldsymbol{\lambda}, \boldsymbol{\nu})$, it is concave, even when the original problem (A.17) is not convex.

In contrast to the dual variables and dual function, the original optimization variable $\boldsymbol{x}$ is

then called the *primal variable*, and the original objective function $f_0(\boldsymbol{x})$ is referred to as the *primal function*.

The dual function $g(\boldsymbol{\lambda}, \boldsymbol{\nu})$ provides lower bounds on the optimal value $p^\star$ of the problem (A.17). Specifically, for any $\boldsymbol{\lambda} \geq \mathbf{0}$ and any $\boldsymbol{\nu}$ (referred to *dual feasible*) we have

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq p^\star,$$

which holds even if the original problem is not convex.

This important property can be easily verified through the following inequalities: for any feasible $\boldsymbol{x}$,

$$\begin{aligned} f_0(\boldsymbol{x}) &\geq f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i h_i(\boldsymbol{x}) \\ &\geq \inf_{\boldsymbol{z} \in \mathcal{D}} \left\{ f_0(\boldsymbol{z}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{z}) + \sum_{i=1}^{p} \nu_i h_i(\boldsymbol{z}) \right\} \\ &= g(\boldsymbol{\lambda}, \boldsymbol{\nu}), \end{aligned}$$

where we have used the fact that $f_i(\boldsymbol{x}) \leq 0$ and $h_i(\boldsymbol{x}) = 0$ for any feasible $\boldsymbol{x}$.

As a consequence of the lower-bound property, a primal–dual feasible pair $(\boldsymbol{x}, (\boldsymbol{\lambda}, \boldsymbol{\nu}))$ localizes the optimal value of the primal (and dual) problem within an interval:

$$p^\star \in \left[g\left(\boldsymbol{\lambda}, \boldsymbol{\nu}\right), f_0(\boldsymbol{x}) \right]. \tag{A.19}$$

This can be utilized in optimization algorithms to provide non-heuristic stopping criteria.

A.6.2 Lagrange Dual Problem

As we have seen, the Lagrange dual function gives us a lower bound on the optimal value of the optimization problem (A.17): $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq p^\star$. Since the lower bound depends on the choice of $(\boldsymbol{\lambda}, \boldsymbol{\nu})$, a natural question is what would be the tightest lower bound that can be obtained. Precisely, this leads to the *Lagrange dual problem* associated with the original problem (A.17):

$$\begin{aligned} \underset{\boldsymbol{\lambda}, \boldsymbol{\nu}}{\text{maximize}} \quad & g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \\ \text{subject to} \quad & \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned} \tag{A.20}$$

This is a convex optimization problem (irrespective of the convexity of the original problem (A.17)) because the objective to be maximized is concave, and the constraints are convex.

In view of the dual problem, the original problem (A.17) is also called *primal problem*. In the dual problem (A.20), we say the variables $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ are *dual feasible* if $\boldsymbol{\lambda} \geq \mathbf{0}$ and $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) > -\infty$. We refer to $(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star)$ as *dual optimal* if they are optimal for the problem (A.20). The optimal value of the dual problem (A.20) is denoted by $d^\star$ in contraposition to the optimal value of the primal problem $p^\star$.

Example A.16 (Least-norm solution of linear equations) Consider the problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{x}^\mathsf{T}\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{array}$$

Its Lagrangian is

$$L(\boldsymbol{x}, \boldsymbol{\nu}) = \boldsymbol{x}^\mathsf{T}\boldsymbol{x} + \boldsymbol{\nu}^\mathsf{T}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}).$$

To find the dual function, we need to solve an unconstrained minimization of the Lagrangian. We set the gradient equal to zero,

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\nu}) = 2\boldsymbol{x} + \boldsymbol{A}^\mathsf{T}\boldsymbol{\nu} = \boldsymbol{0} \quad \Longrightarrow \quad \boldsymbol{x} = -\frac{1}{2}\boldsymbol{A}^\mathsf{T}\boldsymbol{\nu},$$

and we plug the solution in L to obtain the dual function g:

$$g(\boldsymbol{\nu}) = L\left(-\frac{1}{2}\boldsymbol{A}^\mathsf{T}\boldsymbol{\nu}, \boldsymbol{\nu}\right) = -\frac{1}{4}\boldsymbol{\nu}^\mathsf{T}\boldsymbol{A}\boldsymbol{A}^\mathsf{T}\boldsymbol{\nu} - \boldsymbol{b}^\mathsf{T}\boldsymbol{\nu},$$

which is, as expected, a concave function of $\boldsymbol{\nu}$.

From the lower bound property, we have

$$p^\star \geq -\frac{1}{4}\boldsymbol{\nu}^\mathsf{T}\boldsymbol{A}\boldsymbol{A}^\mathsf{T}\boldsymbol{\nu} - \boldsymbol{b}^\mathsf{T}\boldsymbol{\nu} \quad \text{for all } \boldsymbol{\nu}.$$

Finally, the dual problem is the QP:

$$\underset{\boldsymbol{\nu}}{\text{maximize}} \quad -\tfrac{1}{4}\boldsymbol{\nu}^\mathsf{T}\boldsymbol{A}\boldsymbol{A}^\mathsf{T}\boldsymbol{\nu} - \boldsymbol{b}^\mathsf{T}\boldsymbol{\nu}.$$

Example A.17 (Standard-form LP) Consider the problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{c}^\mathsf{T}\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} \geq \boldsymbol{0}. \end{array}$$

Its Lagrangian is

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) &= \boldsymbol{c}^\mathsf{T}\boldsymbol{x} + \boldsymbol{\nu}^\mathsf{T}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) - \boldsymbol{\lambda}^\mathsf{T}\boldsymbol{x} \\ &= \left(\boldsymbol{c} + \boldsymbol{A}^\mathsf{T}\boldsymbol{\nu} - \boldsymbol{\lambda}\right)^\mathsf{T}\boldsymbol{x} - \boldsymbol{b}^\mathsf{T}\boldsymbol{\nu}. \end{aligned}$$

To find the dual function, note that L is a linear function of $\boldsymbol{x}$ and it is unbounded if the term multiplying $\boldsymbol{x}$ is nonzero. Therefore, we can write the dual function as

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \begin{cases} -\boldsymbol{b}^\mathsf{T}\boldsymbol{\nu} & \boldsymbol{c} + \boldsymbol{A}^\mathsf{T}\boldsymbol{\nu} - \boldsymbol{\lambda} = \boldsymbol{0}, \\ -\infty & \text{otherwise}, \end{cases}$$

which, as expected, is a concave function of $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ since it is linear on an affine domain.

From the lower bound property, we have

$$p^\star \geq -\boldsymbol{b}^\mathsf{T}\boldsymbol{\nu} \quad \text{if } \boldsymbol{c} + \boldsymbol{A}^\mathsf{T}\boldsymbol{\nu} \geq \boldsymbol{0}.$$

Finally, the dual problem is the LP

$$\begin{array}{ll} \underset{\boldsymbol{\nu}}{\text{maximize}} & -\boldsymbol{b}^T\boldsymbol{\nu} \\ \text{subject to} & \boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\nu} \geq \boldsymbol{0}. \end{array}$$

Example A.18 (Two-way partitioning) Consider the problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{x}^T\boldsymbol{W}\boldsymbol{x} \\ \text{subject to} & x_i^2 = 1, \quad i = 1, \ldots, n. \end{array}$$

This is a nonconvex problem because the matrix $\boldsymbol{W}$ is not necessarily positive semidefinite and because of the quadratic equality constraints (the feasible set contains 2^n discrete points).

Its Lagrangian is

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{\nu}) &= \boldsymbol{x}^T\boldsymbol{W}\boldsymbol{x} + \sum_{i=1}^{n} \nu_i \left(x_i^2 - 1\right) \\ &= \boldsymbol{x}^T \left(\boldsymbol{W} + \operatorname{Diag}(\boldsymbol{\nu})\right) \boldsymbol{x} - \boldsymbol{1}^T\boldsymbol{\nu}. \end{aligned}$$

To find the dual function, note that L is a quadratic function of $\boldsymbol{x}$ and it is unbounded if the matrix $\boldsymbol{W} + \operatorname{Diag}(\boldsymbol{\nu})$ has a negative eigenvalue. Therefore, we can write the dual function as

$$g(\boldsymbol{\nu}) = \inf_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\nu}) = \begin{cases} -\boldsymbol{1}^T\boldsymbol{\nu} & \boldsymbol{W} + \operatorname{Diag}(\boldsymbol{\nu}) \succeq \boldsymbol{0}, \\ -\infty & \text{otherwise.} \end{cases}$$

From the lower bound property, we have

$$p^\star \geq -\boldsymbol{1}^T\boldsymbol{\nu} \quad \text{if } \boldsymbol{W} + \operatorname{Diag}(\boldsymbol{\nu}) \succeq \boldsymbol{0}.$$

One particular lower bound is obtained by choosing $\boldsymbol{\nu} = -\lambda_{\min}(\boldsymbol{W})\boldsymbol{1}$:

$$p^\star \geq n\lambda_{\min}(\boldsymbol{W}).$$

Finally, the dual problem is the SDP

$$\begin{array}{ll} \underset{\boldsymbol{\nu}}{\text{maximize}} & -\boldsymbol{1}^T\boldsymbol{\nu} \\ \text{subject to} & \boldsymbol{W} + \operatorname{Diag}(\boldsymbol{\nu}) \succeq \boldsymbol{0}. \end{array}$$

Equivalent formulations of a problem can lead to different dual problems as the following examples illustrate.

Example A.19 (Introducing new variables) Consider the problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 .$$

Since the problem has no constraints, the dual problem does not even make sense since it would be a constant. Instead, we can introduce some dummy variables in the original problem and rewrite it as

$$\begin{array}{ll} \underset{\boldsymbol{x},\boldsymbol{y}}{\text{minimize}} & \|\boldsymbol{y}\|_2 \\ \text{subject to} & \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}. \end{array}$$

Then the dual problem can be derived as

$$\begin{aligned} \underset{\boldsymbol{\nu}}{\text{maximize}} \quad & \boldsymbol{b}^\top \boldsymbol{\nu} \\ \text{subject to} \quad & \boldsymbol{A}^\top \boldsymbol{\nu} = \mathbf{0}, \quad \|\boldsymbol{\nu}\|_2 \le 1. \end{aligned}$$

Example A.20 (Implicit constraints) Consider the following LP with box constraints:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \\ & -\mathbf{1} \le \boldsymbol{x} \le \mathbf{1}. \end{aligned}$$

The dual problem is

$$\begin{aligned} \underset{\boldsymbol{\nu}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2}{\text{maximize}} \quad & -\boldsymbol{b}^\top \boldsymbol{\nu} - \mathbf{1}^\top \boldsymbol{\lambda}_1 - \mathbf{1}^\top \boldsymbol{\lambda}_2 \\ \text{subject to} \quad & \boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\nu} + \boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2 = 0, \\ & \boldsymbol{\lambda}_1 \ge \mathbf{0}, \quad \boldsymbol{\lambda}_2 \ge \mathbf{0}, \end{aligned}$$

which does not give much insight. If, instead, we rewrite the primal problem (after making some explicit constraints implicit) as

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f_0(\boldsymbol{x}) = \begin{cases} \boldsymbol{c}^\top \boldsymbol{x} & -\mathbf{1} \le \boldsymbol{x} \le \mathbf{1}, \\ \infty & \text{otherwise} \end{cases} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{aligned}$$

then the dual becomes way more insightful:

$$\underset{\boldsymbol{\nu}}{\text{maximize}} \quad -\boldsymbol{b}^\top \boldsymbol{\nu} - \left\| \boldsymbol{A}^\top \boldsymbol{\nu} + \boldsymbol{c} \right\|_1 .$$

A.6.3 Weak and Strong Duality

The optimal value $d^\star$ of the Lagrange dual problem (A.20) is, by definition, the tightest lower bound on $p^\star$ that can be obtained from the Lagrange dual function. This property is called the *weak duality*:

$$d^\star \le p^\star, \tag{A.21}$$

which holds even if the original problem is not convex. The difference $\Gamma = p^\star - d^\star$ is called *optimal duality gap* of the original problem (and is always nonnegative).

The bound in (A.21) can be utilized to establish a lower limit on the optimal value of a problem that is challenging to solve, given that the dual problem (A.20) is always convex.

Of particular interest is when equality is attained in (A.21). This is referred to as *strong duality*,

$$d^\star = p^\star, \tag{A.22}$$

and implies that the duality gap is zero. Strong duality is very desirable and may facilitate the resolution of a difficult problem via the dual.

Unfortunately, strong duality does not hold for general optimization problems. However, if the primal problem (A.17) is convex, we usually (but not always) have strong duality. There

are specific conditions under which strong duality holds, and these conditions are referred to as *constraint qualifications*.

Slater's condition is a simple constraint qualification that requires the existence of a strictly feasible point, that is, $\boldsymbol{x} \in \operatorname{relint} \mathcal{D}$ (relative interior of the domain) such that $f_i(\boldsymbol{x}) < 0$, $i = 1, \ldots, m$, and $h_i(\boldsymbol{x}) = 0$, $i = 1, \ldots, p$. Strict feasibility is typically easy to verify in practical problems. Slater's theorem states that strong duality holds if Slater's condition is met and the optimization problem is convex.

Example A.21 (Inequality-form LP) Consider the problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}. \end{array}$$

Its dual problem is also an LP:

$$\begin{array}{ll} \underset{\boldsymbol{\lambda}}{\text{maximize}} & -\boldsymbol{b}^\top \boldsymbol{\lambda} \\ \text{subject to} & \boldsymbol{A}^\top \boldsymbol{\lambda} + \boldsymbol{c} = \boldsymbol{0}, \quad \boldsymbol{\lambda} \geq 0. \end{array}$$

From Slater's condition, strong duality holds if $\boldsymbol{A}\tilde{\boldsymbol{x}} < \boldsymbol{b}$ for some $\tilde{\boldsymbol{x}}$, which may be difficult to verify. Interestingly, in this case, we do not need Slater's condition, as we always have $p^\star = d^\star$ (except when both the primal and dual problems are infeasible).

Example A.22 (Convex QP) Consider the problem (with $\boldsymbol{P} \succeq \boldsymbol{0}$)

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{x}^\top \boldsymbol{P} \boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}. \end{array}$$

Its dual problem is also a QP:

$$\begin{array}{ll} \underset{\boldsymbol{\lambda}}{\text{maximize}} & -\frac{1}{4}\boldsymbol{\lambda}^\top \boldsymbol{A}\boldsymbol{P}^{-1}\boldsymbol{A}^\top \boldsymbol{\lambda} - \boldsymbol{b}^\top \boldsymbol{\lambda} \\ \text{subject to} & \boldsymbol{\lambda} \geq \boldsymbol{0}. \end{array}$$

From Slater's condition, strong duality holds if $\boldsymbol{A}\tilde{\boldsymbol{x}} < \boldsymbol{b}$ for some $\tilde{\boldsymbol{x}}$. However, in this case, we always have $p^\star = d^\star$.

Example A.23 (Nonconvex QP) Consider the problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} + 2\boldsymbol{b}^\top \boldsymbol{x} \\ \text{subject to} & \boldsymbol{x}^\top \boldsymbol{x} \leq 1, \end{array}$$

which is nonconvex in general as $\boldsymbol{A} \not\succeq \boldsymbol{0}$.

Its dual problem can be written as the SDP:

$$\begin{array}{ll} \underset{t,\lambda}{\text{maximize}} & -t - \lambda \\ \text{subject to} & \begin{bmatrix} \boldsymbol{A} + \lambda \boldsymbol{I} & \boldsymbol{b} \\ \boldsymbol{b}^\top & t \end{bmatrix} \succeq 0. \end{array}$$

In this case, strong duality holds even though the original problem is nonconvex (not trivial to show).

A.6.4 Optimality Conditions

In Section A.4, we examined the optimality characterization of a solution through the minimum principle. In the following discussion, we will consider a more explicit characterization that is of great practical interest, as it often enables us to derive solutions to problems.

Complementary Slackness

Suppose that strong duality holds so that the primal and dual optimal values are equal, $d^\star = p^\star$. Let $\boldsymbol{x}^\star$ be a primal optimal point and $(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star)$ be a dual optimal point. Then,

$$\begin{aligned} f_0(\boldsymbol{x}^\star) &= g(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) \\ &= \inf_{\boldsymbol{x}\in\mathcal{D}} \left\{ f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i^\star f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i^\star h_i(\boldsymbol{x}) \right\} \\ &\leq f_0(\boldsymbol{x}^\star) + \sum_{i=1}^{m} \lambda_i^\star f_i(\boldsymbol{x}^\star) + \sum_{i=1}^{p} \nu_i^\star h_i(\boldsymbol{x}^\star) \\ &\leq f_0(\boldsymbol{x}^\star), \end{aligned}$$

where the first line comes from the zero duality gap, the second line is the definition of the dual function, the third line follows trivially from the definition of infimum, and the fourth line results from feasibility (i.e., $\lambda_i^\star \geq 0$, $f_i(\boldsymbol{x}^\star) \leq 0$, and $h_i(\boldsymbol{x}^\star) = 0$).

We then conclude that the two inequalities in this chain must hold with equality. Equality in the first inequality means that $\boldsymbol{x}^\star$ minimizes $L(\boldsymbol{x}, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star)$ over $\boldsymbol{x}$ (note that there may be other minimizers). Equality in the second inequality implies that

$$\sum_{i=1}^{m} \lambda_i^\star f_i(\boldsymbol{x}^\star) = 0,$$

which in turns means (because each term is nonpositive)

$$\lambda_i^\star f_i(\boldsymbol{x}^\star) = 0, \quad i = 1, \ldots, m.$$

These conditions are referred to as *complementary slackness*. They hold for any primal optimal $\boldsymbol{x}^\star$ and dual optimal $(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star)$ (assuming strong duality holds).

The interpretation of complementary slackness provides relevant and practical information:

$$\lambda_i^\star > 0 \quad \Longrightarrow \quad f_i\left(x^\star\right) = 0$$

and

$$f_i\left(x^\star\right) < 0 \quad \Longrightarrow \quad \lambda_i^\star = 0.$$

In simpler terms, if a Lagrange multiplier is active (i.e., $\lambda_i^\star > 0$), it indicates that the constraint is at the boundary of the feasible set and would be violated otherwise. Similarly, if a constraint is strictly feasible (i.e., $f_i(x^\star) < 0$), it implies that it is not active, and the corresponding Lagrange multiplier is not even necessary, hence $\lambda_i^\star = 0$.

KKT Optimality Conditions

We are now ready to write the famous *Karush–Kuhn–Tucker (KKT) optimality conditions* for convex and nonconvex optimization problems. We assume that the functions $f_0, f_1, \ldots, f_m, h_1, \ldots, h_p$ are differentiable.

Let $\boldsymbol{x}^\star$ and $(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star)$ be any primal and dual optimal points with zero duality gap. Since $\boldsymbol{x}^\star$ minimizes $L(\boldsymbol{x}, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star)$ over $\boldsymbol{x}$, it follows that its gradient must vanish, that is,

$$\nabla f_0(\boldsymbol{x}^\star) + \sum_{i=1}^{m} \lambda_i^\star \nabla f_i(\boldsymbol{x}^\star) + \sum_{i=1}^{p} \nu_i^\star \nabla h_i(\boldsymbol{x}^\star) = \mathbf{0}.$$

Thus, for any optimization problem (not necessarily convex), any pair of optimal and dual points, $\boldsymbol{x}^\star$ and $(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star)$, must satisfy the KKT optimality conditions:

$$\begin{aligned}
f_i(\boldsymbol{x}^\star) &\le 0, \quad i = 1, \ldots, m, && \text{(primal feasibility)} \\
h_i(\boldsymbol{x}^\star) &= 0, \quad i = 1, \ldots, p, && \\
\lambda_i^\star &\ge 0, \quad i = 1, \ldots, m, && \text{(dual feasibility)} \\
\lambda_i^\star f_i(\boldsymbol{x}^\star) &= 0, \quad i = 1, \ldots, m, && \text{(complementary slackness)} \\
\nabla f_0(\boldsymbol{x}^\star) + \sum_{i=1}^{m} \lambda_i^\star \nabla f_i(\boldsymbol{x}^\star) &+ \sum_{i=1}^{p} \nu_i^\star \nabla h_i(\boldsymbol{x}^\star) = \mathbf{0}. && \text{(zero Lagrangian gradient)}
\end{aligned} \tag{A.23}$$

When the primal problem is convex, the KKT conditions (A.23) are also sufficient for the points to be primal and dual optimal. To see this, simply note that since the problem is convex, the Lagrangian in $\boldsymbol{x}$ is also convex. A point $\tilde{\boldsymbol{x}}$ that achieves a zero gradient in the Lagrangian implies that it minimizes the Lagrangian, so

$$\begin{aligned}
g(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}}) &= L(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}}) \\
&= f_0(\tilde{\boldsymbol{x}}) + \sum_{i=1}^{m} \tilde{\lambda}_i f_i(\tilde{\boldsymbol{x}}) + \sum_{i=1}^{p} \tilde{\nu}_i h_i(\tilde{\boldsymbol{x}}) \\
&= f_0(\tilde{\boldsymbol{x}}),
\end{aligned}$$

which shows zero duality gap and, therefore, primal and dual optimality. This is summarized in the next statement.

Theorem A.1 (KKT optimality conditions) *Suppose we have an optimization problem with differentiable functions.*

- *For any optimization problem (not necessarily convex) with strong duality, the KKT conditions* (A.23) *are necessary for optimality.*
- *For a convex optimization problem satisfying Slater's condition, strong duality follows and the KKT conditions* (A.23) *are necessary and sufficient for optimality.*

The KKT conditions play a key role in optimization. In some cases, it is possible to characterize analytically the solution in a convenient form. More generally, many algorithms are conceived, or can be interpreted, as methods for solving the KKT conditions.

A.7 Multi-objective Optimization

In Section A.5, we expanded the scope of inequality constraints to include vector values through the use of generalized inequalities, as illustrated in the conic programming (CP) formulation in (A.14). We will now briefly explore the implications of having a vector-valued objective function.

The reader interested in more details is referred to Chapter 2 in Boyd and Vandenberghe (2004) for generalized inequalities and Chapter 4 in Boyd and Vandenberghe (2004) for vector optimization, Pareto optimality, and multi-objective optimization.

A.7.1 Generalized Inequalities

A cone was defined in Section A.2 as a set $\mathcal{K} \subseteq \mathbb{R}^n$ such that for every $\boldsymbol{x} \in \mathcal{K}$ and $\theta \geq 0$ we have $\theta \boldsymbol{x} \in \mathcal{K}$. A cone $\mathcal{K}$ is called a *proper cone* if it is convex, closed, solid (i.e., has a nonempty interior), and pointed (i.e., it contains no line).

A proper cone $\mathcal{K}$ can be used to define a *generalized inequality*, which is a partial ordering on $\mathbb{R}^n$ that has many of the properties of the standard ordering on $\mathbb{R}$:

$$\boldsymbol{x} \preceq_{\mathcal{K}} \boldsymbol{y} \quad \Longleftrightarrow \quad \boldsymbol{y} - \boldsymbol{x} \in \mathcal{K}.$$

We also write $\boldsymbol{x} \preceq_{\mathcal{K}} \boldsymbol{y}$ as $\boldsymbol{y} \succeq_{\mathcal{K}} \boldsymbol{x}$.

Example A.24 (Nonnegative orthant and componentwise inequality) The nonnegative orthant $\mathcal{K} = \mathbb{R}^n_+$ is a proper cone and its associated generalized inequality $\preceq_{\mathcal{K}}$ corresponds to the componentwise inequality between vectors: $\boldsymbol{x} \preceq_{\mathcal{K}} \boldsymbol{y}$ means $x_i \leq y_i,\ i = 1, \ldots, n$.

Example A.25 (Positive semidefinite cone and matrix inequality) The positive semidefinite cone $\mathbb{S}^n_+$ is a proper cone in the set of $n \times n$ symmetric matrices $\mathbb{S}^n$ and its associated generalized inequality $\preceq_{\mathcal{K}}$ is the usual matrix inequality: $\boldsymbol{X} \preceq_{\mathcal{K}} \boldsymbol{Y}$ means $\boldsymbol{Y} - \boldsymbol{X}$ is positive semidefinite (typically written as $\boldsymbol{Y} \succeq \boldsymbol{X}$).

The notation of generalized inequality (i.e., $\preceq_{\mathcal{K}}$) is meant to suggest the analogy to ordinary inequality on $\mathbb{R}$ (i.e., $\leq$). While many properties of ordinary inequality do hold for generalized inequalities, some important ones do not. The most obvious difference is that $\leq$ on $\mathbb{R}$ is a *total ordering*: any two points are comparable, meaning that we can always write $x \leq y$ or $y \leq x$. This property does not hold for other generalized inequalities as they simply define a *partial ordering*. One implication is that concepts like minimum and maximum are more complicated in the context of generalized inequalities.

We can also use generalized inequalities to extend the definition of a convex function from Section A.3 to the vector case. We say a function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^q$ is $\mathcal{K}$-convex if the domain is a convex set and, for all $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} \boldsymbol{f}$ and $\theta \in [0, 1]$, $\boldsymbol{f}(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) \preceq_{\mathcal{K}} \theta \boldsymbol{f}(\boldsymbol{x}) + (1-\theta)\boldsymbol{f}(\boldsymbol{y})$.

A.7.2 Vector Optimization

We denote a general *vector optimization problem* as

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimize}} \ (\text{with respect to } \mathcal{K}) & \boldsymbol{f}_0(\boldsymbol{x}) & \\ \text{subject to} & f_i(\boldsymbol{x}) \leq 0, \quad & 1 \leq i \leq m, \\ & h_i(\boldsymbol{x}) = 0, \quad & 1 \leq i \leq p, \end{array} \tag{A.24}$$

where $\mathcal{K} \in \mathbb{R}^q$ is a proper cone, $\boldsymbol{f}_0 : \mathbb{R}^n \to \mathbb{R}^q$ is the vector-valued objective function, $f_i : \mathbb{R}^n \to \mathbb{R}$ are the inequality constraint functions, and $h_i : \mathbb{R}^n \to \mathbb{R}$ are the equality constraint functions. The only difference between this problem and the standard optimization problem (A.1) is that here the objective function takes values in $\mathbb{R}^q$ and the problem specification includes a proper cone $\mathcal{K}$, which is used to compare objective values.

We say the vector optimization problem (A.24) is a *convex vector optimization problem* if the objective function $\boldsymbol{f}_0$ is $\mathcal{K}$-convex, the inequality constraint functions $f_1, \ldots, f_m$ are convex, and the equality constraint functions $h_1, \ldots, h_p$ are affine (which can then be written as $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$).

The interpretation of problem (A.24) is not straightforward because the general inequality $\preceq_{\mathcal{K}}$, which is used to compare different points based on their objective values $\boldsymbol{f}_0(\boldsymbol{x})$, is not a total ordering but rather a partial ordering. This means we may encounter two points, $\boldsymbol{x}$ and $\boldsymbol{y}$, that are not comparable; that is, neither $\boldsymbol{f}_0(\boldsymbol{x}) \preceq_{\mathcal{K}} \boldsymbol{f}_0(\boldsymbol{y})$ nor $\boldsymbol{f}_0(\boldsymbol{y}) \preceq_{\mathcal{K}} \boldsymbol{f}_0(\boldsymbol{x})$ holds (a situation that cannot occur in the standard optimization problem (A.1)).

A.7.3 Pareto Optimality

Let us start with a special case in which the meaning of the vector optimization problem is clear. Consider the set of *achievable objective values*, that is, the set of objective values of feasible points:

$$O = \{\boldsymbol{f}_0(\boldsymbol{x}) \mid \boldsymbol{x} \text{ is feasible}\}.$$

We say this set has a *minimum element* if there is a feasible $\boldsymbol{x}^\star$ such that $\boldsymbol{f}_0(\boldsymbol{x}^\star) \preceq_{\mathcal{K}} \boldsymbol{f}_0(\boldsymbol{x})$ for all feasible $\boldsymbol{x}$. Then we say that $\boldsymbol{x}^\star$ is *optimal* for the problem (A.24) and refer to $\boldsymbol{f}_0(\boldsymbol{x}^\star)$ as the *optimal value*. Using set notation, a point $\boldsymbol{x}^\star$ is optimal if and only if it is feasible and $O \subseteq \boldsymbol{f}_0(\boldsymbol{x}^\star) + \mathcal{K}$. In this case, $\boldsymbol{x}^\star$ is unambiguously a best choice for $\boldsymbol{x}$ among the feasible points. However, most vector optimization problems do not have an optimal point because there may be other points that are not comparable via the cone $\mathcal{K}$.

We now turn our attention to a more general case, which is common in most vector optimization problems of interest. In this case, the set of achievable objective values, O, does not have a minimum element. Consequently, the problem does not have an optimal point or optimal value. When O lacks a minimum element, we can only discuss the so-called *minimal elements*. These are the best among the objective values that can be compared, although there are other objective values that cannot be compared. The points $\boldsymbol{x}$ that achieve such minimal elements in the set of objective values O are referred to as *Pareto optimal* points. We say that $\boldsymbol{f}_0(\boldsymbol{x})$ is a *Pareto optimal value* for the vector optimization problem (A.24).

Thus, a point $\boldsymbol{x}^{\text{po}}$ is Pareto optimal if it is feasible and, for any other feasible $\boldsymbol{x}$, $\boldsymbol{f}_0(\boldsymbol{x}) \preceq_{\mathcal{K}} \boldsymbol{f}_0(\boldsymbol{x}^{\text{po}})$

implies $\boldsymbol{f}_0(\boldsymbol{x}) = \boldsymbol{f}_0(\boldsymbol{x}^{\text{po}})$. Using set notation, a point $\boldsymbol{x}^{\text{po}}$ is Pareto optimal if and only if it is feasible and $(\boldsymbol{f}_0(\boldsymbol{x}^{\text{po}}) - \mathcal{K}) \cap \mathcal{O} = \boldsymbol{f}_0(\boldsymbol{x}^{\text{po}})$. In words, $\boldsymbol{x}^\star$ cannot be in the cone of points worse than any other point.

A vector optimization problem usually has many Pareto optimal values (and points), and they lie in the boundary of the set of achievable objective values, usually termed *efficient frontier*.

A.7.4 Multi-objective Optimization

When a vector optimization problem is based on the cone $\mathcal{K} = \mathbb{R}^q_+$, that is, the nonnegative orthant, it is called a *multi-objective* or *multi-criterion* optimization problem. The components of the vector objective function $\boldsymbol{f}_0$, denoted by $F_1, \ldots, F_q$, can be interpreted as q different scalar objectives to be minimized. The simplest case of multi-objective optimization is *bi-objective* or *bi-criterion* optimization problems, where we just have two objectives $F_1(\boldsymbol{x})$ and $F_2(\boldsymbol{x})$.

A multi-objective optimization problem is convex if the inequality constraint functions $f_1, \ldots, f_m$ are convex, the equality constraint functions $h_1, \ldots, h_p$ are affine (denoted then as $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$), and the objectives $F_1, \ldots, F_q$ are convex.

For two feasible points $\boldsymbol{x}$ and $\boldsymbol{y}$, $F_i(\boldsymbol{x}) \leq F_i(\boldsymbol{y})$ means that $\boldsymbol{x}$ is at least as good as $\boldsymbol{y}$ and $F_i(\boldsymbol{x}) < F_i(\boldsymbol{y})$ means that $\boldsymbol{x}$ is better than $\boldsymbol{y}$, according to the ith objective. We say that $\boldsymbol{x}$ is better than $\boldsymbol{y}$, or $\boldsymbol{x}$ dominates $\boldsymbol{y}$, if $F_i(\boldsymbol{x}) \leq F_i(\boldsymbol{y})$ for $i = 1, \ldots, q$, and for at least one j, $F_j(\boldsymbol{x}) < F_j(\boldsymbol{y})$. In words, $\boldsymbol{x}$ is better than $\boldsymbol{y}$ if $\boldsymbol{x}$ meets or beats $\boldsymbol{y}$ on all objectives, and beats it in at least one objective.

In a multi-objective optimization problem, for a point to be considered optimal, $\boldsymbol{x}^\star$, it has to be simultaneously optimal for each of the scalar problems:

$$F_i(\boldsymbol{x}^\star) \leq F_i(\boldsymbol{y}), \quad i = 1, \ldots, q.$$

In general, however, this cannot happen unless in the trivial case that the objectives are *noncompeting*, since no compromises have to be made among the objectives. In most practical problems, there is naturally a trade-off among the objectives. Consequently, an optimal solution does not exists and we have to resort to the concept of Pareto optimality. For a point to be considered Pareto optimal, $\boldsymbol{x}^{\text{po}}$, it must be such that no objective can be improved without degrading at least one other objective.

The set of Pareto optimal values for a multi-objective problem is called the *optimal trade-off surface* or, when $q = 2$, the *optimal trade-off curve*. In other contexts, this is also referred to as the *efficient frontier*.

A.7.5 Scalarization

Scalarization is a standard technique for finding Pareto optimal points for a vector optimization problem. Specifically, for a multi-objective optimization problem the scalarized problem is

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimize}} & \sum_{i=1}^{q} \lambda_i F_i(\boldsymbol{x}) & \\ \text{subject to} & f_i(\boldsymbol{x}) \leq 0, & 1 \leq i \leq m, \\ & h_i(\boldsymbol{x}) = 0, & 1 \leq i \leq p, \end{array} \tag{A.25}$$

where the λ_i's are the weights associated to the objectives and, to guarantee Pareto optimality, must satisfy $\lambda_i > 0$ (equivalently, $\boldsymbol{\lambda} > \mathbf{0}$).

An optimal point of the scalarized problem is then Pareto optimal for the multi-objective optimization problem.[4] If some weights are allowed to be zero, an optimal point is only guaranteed to be *weakly* Pareto optimal (meaning that no feasible point strictly improves all objectives simultaneously), unless it is the unique optimal point of the scalarized problem. By choosing different strictly positive weights, we can obtain different Pareto optimal solutions for the multi-objective optimization problem. However, there may be some Pareto optimal points that cannot be obtained via scalarization.

For convex multi-objective optimization problems, the scalarized problem is also convex. Moreover, for every Pareto optimal point, there are weights $\boldsymbol{\lambda} \geq \mathbf{0}$, $\boldsymbol{\lambda} \neq \mathbf{0}$, such that it is optimal in the scalarized problem. In this converse statement, some weights may need to be zero (for example, at the endpoints of the trade-off curve), so requiring $\boldsymbol{\lambda} > \mathbf{0}$ would be too strong.

Example A.26 (Regularized LS) Consider a modified least squares problem where the energy of the solution is also desired to be small. We can then consider a multi-objective optimization problem with two objectives:

- $F_1(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$ is a measure of the regression error;
- $F_2(\boldsymbol{x}) = \|\boldsymbol{x}\|_2^2$ is a measure of the energy of $\boldsymbol{x}$.

The multi-objective optimization problem formulation is

$$\underset{\boldsymbol{x}}{\text{minimize}} \ (\text{with respect to } \mathbb{R}_+^2) \qquad f_0(\boldsymbol{x}) = (F_1(\boldsymbol{x}), F_2(\boldsymbol{x}))$$

and its corresponding scalarization is

$$\underset{\boldsymbol{x}}{\text{minimize}} \qquad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \gamma \|\boldsymbol{x}\|_2^2 \ ,$$

where $\gamma > 0$ is the weight to indicate the preference in the trade-off. The case $\gamma = 0$ gives plain least squares, whose solutions are Pareto optimal only when the least-squares solution is unique. If $\boldsymbol{A}$ has a nontrivial nullspace, there are multiple least-squares solutions and only the minimum-norm one is Pareto optimal.

[4] To see why strictly positive weights guarantee Pareto optimality, suppose that a feasible point $\boldsymbol{y}$ dominates an optimal point $\boldsymbol{x}^\star$ of the scalarized problem. Then $F_i(\boldsymbol{y}) \leq F_i(\boldsymbol{x}^\star)$ for every i, with strict inequality for some j. If $\boldsymbol{\lambda} > \mathbf{0}$, it follows that $\sum_i \lambda_i F_i(\boldsymbol{y}) < \sum_i \lambda_i F_i(\boldsymbol{x}^\star)$, contradicting the optimality of $\boldsymbol{x}^\star$. With some zero weights, this argument no longer holds. For example, with $F_1(\boldsymbol{x}) = x_1^2$, $F_2(\boldsymbol{x}) = x_2^2$, and $\boldsymbol{\lambda} = (1, 0)$, every point $(0, x_2)$ is optimal in the scalarized problem, but only $(0, 0)$ is Pareto optimal.

A.8 Summary

- Optimization has a long history. The theory has been extensively developed over the past century, whereas the evolution of algorithms began in 1947 with the introduction of the simplex method and culminated in the mid-1990s with the advent of the powerful interior-point methods.
- Generally, optimization problems are hard to solve, with an exponential time complexity. However, the class of convex problems enjoys a manageable polynomial time complexity, hence the interest in convex optimization.
- Convex problems are composed of convex functions and convex sets. They enjoy a rich theoretical foundation as well as efficient algorithms. There is a plethora of solvers available in most programming languages that can be used to solve optimization problems numerically.
- Lagrange duality is a beautiful and powerful theory that provides numerous useful theoretical results, including the KKT optimality conditions that can be used to characterize optimal solutions.
- The standard problem formulation can be extended in many ways, such as with multi-objective formulations and robust formulations.

Exercises

A.1 (Concepts on convexity)

a. Define a convex set and provide an example.
b. Define a convex function and provide an example.
c. Explain the concept of convex optimization problems and provide an example.
d. What is the difference between active and inactive constraints in an optimization problem?
e. What is the difference between a locally optimal point and a globally optimal point?
f. Define a feasibility problem and provide an example.
g. Explain the concept of least squares problems and provide an example.
h. Explain the concept of linear programming and provide an example.
i. Explain the concept of nonconvex optimization and provide an example.
j. Explain the difference between a convex and a nonconvex optimization problem.

A.2 (Convexity of sets) Determine the convexity of the following sets:

a. $\mathcal{X} = \{x \in \mathbb{R} \mid x^2 - 3x + 2 \geq 0\}$.
b. $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \max\{x_1, x_2, \ldots, x_n\} \leq 1\}$.
c. $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \alpha \leq \boldsymbol{c}^\top \boldsymbol{x} \leq \beta\}$.
d. $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^2 \mid x_1 \geq 1,\ x_2 \geq 2,\ x_1 x_2 \geq 1\}$.
e. $\mathcal{X} = \{(x, y) \in \mathbb{R}^2 \mid y \geq x^2\}$.
f. $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x} - \boldsymbol{c}\| \leq \boldsymbol{a}^\top \boldsymbol{x} + b\}$.
g. $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n \mid (\boldsymbol{a}^\top \boldsymbol{x} + b)/(\boldsymbol{c}^\top \boldsymbol{x} + d) \geq 1,\ \boldsymbol{c}^\top \boldsymbol{x} + d \geq 1\}$.
h. $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{a}^\top \boldsymbol{x} \geq b \text{ or } \|\boldsymbol{x} - \boldsymbol{c}\| \leq 1\}$.
i. $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x}^\top \boldsymbol{y} \leq 1 \text{ for all } \boldsymbol{y} \in \mathcal{S}\}$, where $\mathcal{S}$ is an arbitrary set.

A.3 (Convexity of functions) Determine the convexity of the following functions:

a. $f(\boldsymbol{x}) = \alpha g(\boldsymbol{x}) + \beta$, where g is a convex function, and α and β are scalars with $\alpha > 0$.
b. $f(\boldsymbol{x}) = \|\boldsymbol{x}\|^p$ with $p \geq 1$.
c. $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$.
d. The difference between the maximum and minimum value of a polynomial on a given interval, as a function of its coefficients:

$$f(\boldsymbol{x}) = \sup_{t \in [0,1]} p_{\boldsymbol{x}}(t) - \inf_{t \in [0,1]} p_{\boldsymbol{x}}(t),$$

where $p_{\boldsymbol{x}}(t) = x_1 + x_2 t + x_3 t^2 + \cdots + x_n t^{n-1}$.
e. $f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{Y}^{-1} \boldsymbol{x}$ (with $\boldsymbol{Y} \succ \boldsymbol{0}$).
f. $f(\boldsymbol{Y}) = \boldsymbol{x}^T \boldsymbol{Y}^{-1} \boldsymbol{x}$ (with $\boldsymbol{Y} \succ \boldsymbol{0}$).
g. $f(\boldsymbol{x}, \boldsymbol{Y}) = \boldsymbol{x}^T \boldsymbol{Y}^{-1} \boldsymbol{x}$ (with $\boldsymbol{Y} \succ \boldsymbol{0}$). Hint: Use the Schur complement.
h. $f(\boldsymbol{x}) = \sqrt{\sqrt{\boldsymbol{a}^T \boldsymbol{x} + b}}$.
i. $f(\boldsymbol{X}) = \log\det(\boldsymbol{X})$ on $\mathbb{S}_{++}^n$.
j. $f(\boldsymbol{X}) = \det(\boldsymbol{X})^{1/n}$ on $\mathbb{S}_{+}^n$.
k. $f(\boldsymbol{X}) = \operatorname{Tr}\left(\boldsymbol{X}^{-1}\right)$ on $\mathbb{S}_{++}^n$.
l. $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{\Sigma} \boldsymbol{x} - \boldsymbol{b}^T \log(\boldsymbol{x})$, where $\boldsymbol{\Sigma} \succ \boldsymbol{0}$ and the log function is applied elementwise.

A.4 (Reformulation of problems)

a. Rewrite the following optimization problem as an LP (assuming $d > \|\boldsymbol{c}\|_1$):

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \dfrac{\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1}{\boldsymbol{c}^T \boldsymbol{x} + d} \\ \text{subject to} & \|\boldsymbol{x}\|_\infty \leq 1. \end{array}$$

b. Rewrite the following optimization problem as an LP:

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \frac{\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1}{1 - \|\boldsymbol{x}\|_\infty}.$$

c. Rewrite the following constraint as an SOC constraint:

$$\left\{(\boldsymbol{x}, y, z) \in \mathbb{R}^{n+2} \mid \|\boldsymbol{x}\|^2 \leq yz, y \geq 0, z \geq 0\right\}.$$

Hint: You may need the equality $yz = \frac{1}{4}\left((y+z)^2 - (y-z)^2\right)$.

d. Rewrite the following problem as an SOCP:

$$\begin{array}{ll} \underset{\boldsymbol{x}, y \geq 0, z \geq 0}{\text{minimize}} & \boldsymbol{a}^T \boldsymbol{x} + \kappa\sqrt{\boldsymbol{x}^T \boldsymbol{\Sigma} \boldsymbol{x}} \\ \text{subject to} & \|\boldsymbol{x}\|^2 \leq yz, \end{array}$$

where $\boldsymbol{\Sigma} \succeq \boldsymbol{0}$.

e. Rewrite the following problem as an SOCP:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} + \boldsymbol{a}^T \boldsymbol{x} \\ \text{subject to} & \boldsymbol{B}\boldsymbol{x} \leq \boldsymbol{b}, \end{array}$$

where $\boldsymbol{A} \succeq \boldsymbol{0}$.

f. Rewrite the following problem as an SDP:

$$\underset{\boldsymbol{X}\succeq\mathbf{0}}{\text{minimize}} \quad \operatorname{Tr}\left((\boldsymbol{I}+\boldsymbol{X})^{-1}\right)+\operatorname{Tr}(\boldsymbol{A}\boldsymbol{X}).$$

A.5 (Concepts on problem resolution)

a. How would you determine if a convex problem is feasible or infeasible?
b. How would you determine if a convex problem has a unique solution or multiple solutions?
c. What are the main ways to solve a convex problem?
d. Given a nonconvex optimization problem, what strategies can be used to find an approximate solution?

A.6 (Linear regression)

a. Consider the line equation $y = \alpha x + \beta$. Choose some values for α and β, and generate 100 noisy pairs $(x_i, y_i),\ i = 1, \ldots, 100$ (i.e., add some random noise to each observation y_i).
b. Formulate a regression problem to fit the 100 data points with a line based on least squares. Plot the true and estimated lines along with the points.
c. Repeat the regression using several other definitions of error in the problem formulation. Plot and compare all the estimated lines.

A.7 (Concepts on Lagrange duality)

a. Define Lagrange duality and explain its significance in convex optimization.
b. Give an example of a problem and its dual.
c. List the KKT conditions and explain their role in convex optimization.
d. Provide an example of a problem with its KKT conditions.
e. Try to find a solution that satisfies the previous KKT conditions. Is this always possible?

A.8 (Solution via KKT conditions) For the following problems, determine the convexity, write the Lagrangian and KKT conditions, and derive a closed-form solution:

a. Risk parity portfolio:

$$\begin{array}{ll}\underset{\boldsymbol{x}\geq\mathbf{0}}{\text{minimize}} & \sqrt{\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{x}} \\ \text{subject to} & \boldsymbol{b}^{\mathsf{T}}\log(\boldsymbol{x})\geq 1,\end{array}$$

where $\boldsymbol{\Sigma}\succ\mathbf{0}$ and the log function is applied elementwise.

b. Projection onto the simplex:

$$\begin{array}{ll}\underset{\boldsymbol{x}}{\text{minimize}} & \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{x}=(\leq)1,\ \boldsymbol{x}\geq\mathbf{0}.\end{array}$$

c. Projection onto a diamond:

$$\begin{array}{ll}\underset{\boldsymbol{x}}{\text{minimize}} & \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 \\ \text{subject to} & \|\boldsymbol{x}\|_1\leq 1.\end{array}$$

A.9 (Dual problems) Find the dual of the following problems:

a. Vanishing maximum eigenvalue problem:

$$\begin{array}{ll} \underset{t,\boldsymbol{X}}{\text{minimize}} & t \\ \text{subject to} & t\boldsymbol{I} \succeq \boldsymbol{X}, \\ & \boldsymbol{X} \succeq \boldsymbol{0}. \end{array}$$

b. Matrix upper bound problem:

$$\begin{array}{ll} \underset{\boldsymbol{X}}{\text{minimize}} & \mathrm{Tr}(\boldsymbol{X}) \\ \text{subject to} & \boldsymbol{X} \succeq \boldsymbol{A}, \\ & \boldsymbol{X} \succeq \boldsymbol{B} \end{array}$$

where $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{S}^n_+$.

c. Log det problem:

$$\begin{array}{ll} \underset{\boldsymbol{X} \succeq \boldsymbol{0}}{\text{minimize}} & \mathrm{Tr}(\boldsymbol{C}\boldsymbol{X}) + \log\det(\boldsymbol{X}^{-1}) \\ \text{subject to} & \boldsymbol{A}_i^\top \boldsymbol{X} \boldsymbol{A}_i \preceq \boldsymbol{B}_i, \qquad i = 1, \ldots, m, \end{array}$$

where $\boldsymbol{C} \in \mathbb{S}^n_+$ and $\boldsymbol{B}_i \in \mathbb{S}^n_{++}$ for $i = 1, \ldots, m,$.

A.10 (Multi-objective optimization)

a. Explain the concept of multi-objective optimization problems.
b. What is the significance of the weights in the scalarization of a multi-objective problem?
c. Provide an example of a bi-objective convex optimization problem and its scalarization.
d. Solve this scalarized bi-objective problem for different values of the weight and plot the optimal trade-off curve.

References

Bajalinov, E. B. (2003). *Linear-Fractional Programming: Theory, Methods, Applications and Software*. Springer.

Bengtsson, M., & Ottersten, B. (2001). Optimal and suboptimal transmit beamforming. In L. C. Godara (Ed.), *Handbook of Antennas in Wireless Communications*. CRC Press.

Ben-Tal, A., & Nemirovski, A. (2001). *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. Society for Industrial and Applied Mathematics (SIAM).

Bertsekas, D. P. (1999). *Nonlinear Programming*. Athena Scientific.

Bertsekas, D. P., Nedić, A., & Ozdaglar, A. E. (2003). *Convex Analysis and Optimization*. Athena Scientific.

Boyd, S., Kim, S. J., Vandenberghe, L., & Hassibi, A. (2007, April). *A Tutorial on Geometric Programming*. Springer.

Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Charnes, A., & Cooper, W. W. (1962). Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, *9*(3–4), 181–186.

Chiang, M. (2005). Geometric programming for communication systems. *Foundations and Trends in Communications and Information Theory*, *2*(1–2), 1–154.

Karmarkar, N. (1984). A new polynomial time algorithm for linear programming. *Combinatorica*, *4*(4), 373–395.

Lobo, M. S., Vandenberghe, L., Boyd, S., & Lebret, H. (1998). Applications of second-order cone programming. *Linear Algebra and Applications*, *284*(1–3), 193–228.

Luenberger, D. G. (1969). *Optimization by Vector Space Methods*. John Wiley & Sons.

Luenberger, D. G., & Ye, Y. (2021). *Linear and Nonlinear Programming* (5th ed.). Springer.

Nemirovski, A. (2000). *Lectures on Modern Convex Optimization* (tech. rep.). Technion - Israel Institute of Technology.

Nesterov, Y. (2018). *Lectures on Convex Optimization* (2nd ed.). Springer.

Nesterov, Y., & Nemirovski, A. (1994). *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics (SIAM).

Palomar, D. P., & Eldar, Y. C. (2009). *Convex Optimization in Signal Processing and Communications*. Cambridge University Press.

Rockafellar, R. T. (1970). *Convex Analysis* (2nd ed.). Princeton Univ. Press.

Rockafellar, R. T. (1993). Lagrange multipliers and optimality. *SIAM Review*, *35*(2), 183–238.

Schaible, S. (1974). Parameter-free convex equivalent and dual programs of fractional programming problems. *Zeitschrift fur Operations Research*, *18*(5), 187–196.

Stancu-Minasian, I. M. (1997). *Fractional Programming: Theory, Methods and Applications*. Kluwer Academic Publishers.

Vandenberghe, L., & Boyd, S. (1996). Semidefinite programming. *SIAM Review*, *38*(1), 49–95.

von Neumann, J. (1937). Über ein ökonomisches Gleichgewichtssystem und eine Verallgemeinerung des Brouwerschen Fixpunktsatzes. *Ergebnisse eines Mathematischen Kolloquiums*, *8*, 73–83.

Appendix B

Optimization Algorithms

Waste no more time arguing what a good man should be. Be one.

— Marcus Aurelius, *Meditations*

Over the past century, there has been a remarkable progression in the development of efficient algorithms designed to solve a broad range of convex optimization problems. In 1947, Dantzig introduced the widely used and efficient *simplex method* for linear programming (LP), despite its theoretical worst-case complexity being exponential. Later, in 1984, Karmarkar presented a groundbreaking paper (Karmarkar, 1984) proposing an *interior-point method* for solving LPs, which offered a worst-case polynomial time complexity. This development was followed by numerous researchers extending the application of interior-point methods to quadratic programming (QP) and linear complementarity problems. In 1994, Nesterov and Nemirovskii further advanced the field by developing the theory of self-concordant functions. This theory facilitated the expansion of algorithms based on the log-barrier function to a wider array of convex problems, notably including semidefinite programming (SDP) and second-order cone programming (SOCP) (Nesterov & Nemirovski, 1994).

In addition to these general-purpose algorithms designed for various classes of convex problems, there exist other highly beneficial techniques and algorithmic frameworks, such as block coordinate descent, majorization–minimization, successive convex approximation, among others. These can be employed to develop customized, straightforward algorithms specifically tailored to certain (potentially nonconvex) problems, often with improved complexity and convergence rates. This chapter will explore a broad array of such practical algorithms.

B.1 Solvers

A *solver*, also referred to as an *optimizer*, is an engine designed to solve specific types of mathematical problems. Most programming languages, including R, Python, MATLAB, Julia, Rust, C, and C++, offer a comprehensive list of available solvers. Each of these solvers is typically capable of handling only a certain category of problems, such as LP, QP, QCQP, SOCP, or SDP. For a detailed classification of optimization problems, refer to Section A.5.

This material has been published as: Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.* Cambridge University Press. This version is free to view and download for personal use only; not for re-distribution, re-sale, or use in derivative works. © Daniel P. Palomar 2025.

B.1.1 Some Popular Solvers

Many of the currently popular solvers were originally developed in a specific programming language, such as Fortran, C, C++, or MATLAB. However, over time, they have been adapted and integrated into a wide range of other programming languages.

For the purpose of illustration, among many others, some popular solvers include:

- GLPK (GNU Linear Programming Kit):[1] intended for large-scale LP including mixed-integer variables, written in C.
- quadprog:[2] very popular open-source QP solver originally written in Fortran by Berwin Turlach in the late 1980s, and now accessible from most other programming languages.
- MOSEK:[3] proprietary LP, QP, SOCP, SDP solver including mixed-integer variables established in 1997 by Erling Andersen (free licence available for academia); specialized in large-scale problems and very fast, robust, and reliable.
- SeDuMi:[4] open-source LP, QP, SOCP, SDP solver originally developed by Sturm in 1999 for MATLAB (Sturm, 1999).
- SDPT3:[5] open-source LP, QP, SOCP, SDP solver originally developed in 1999 for MATLAB (Tütüncü et al., 2003).
- Gurobi:[6] proprietary LP, QP, and SOCP solver including mixed-integer variables (free licence available for academia).
- Embedded COnic Solver (ECOS):[7] SOCP solver originally written in C.
- CPLEX:[8] proprietary LP and QP solver that also handles mixed-integer variables (free licence available for academia).

B.1.2 Complexity of Interior-Point Methods

Internally, solvers can be based on different types of algorithms like the simplex method for LP (Nocedal & Wright, 2006), cutting plane methods (Bertsekas, 1999), interior-point methods (Ben-Tal & Nemirovski, 2001; Boyd & Vandenberghe, 2004; Luenberger & Ye, 2021; Nemirovski, 2000; Nesterov, 2018; Nesterov & Nemirovski, 1994; Nocedal & Wright, 2006), or even more specific algorithms tailored to a more narrow type of problem such as ℓ_2-norm regression with an ℓ_1-norm regularization term (used to promote sparsity) (Tibshirani, 1996).

In general, the complexity of interior-point methods for LP, QP, QCQP, SOCP, and SDP is

[1] GLPK: www.gnu.org/software/glpk
[2] quadprog R version: https://cran.r-project.org/package=quadprog
[3] MOSEK: www.mosek.com
[4] SeDuMi: https://sedumi.ie.lehigh.edu; MATLAB version at https://github.com/sqlp/sedumi
[5] SDPT3: https://blog.nus.edu.sg/mattohkc/softwares/sdpt3, https://github.com/Kim-ChuanToh/SDPT3, https://github.com/sqlp/sdpt3
[6] Gurobi optimizer: www.gurobi.com
[7] ECOS: https://github.com/embotech/ecos
[8] IBM CPLEX optimizer: www.ibm.com/analytics/cplex-optimizer

$O\left(n^3L\right)$,[9] where n is the number of variables and L is the number of accuracy digits of the solution. To further discern the difference in the complexity, we need to take into account the number of constraints, m, and dimensionality of the different cones, k (Nemirovski, 2000):

- LP: complexity of $O\left((m+n)^{3/2}n^2L\right)$, later improved to $O\left(((m+n)n^2+(m+n)^{1.5}n)L\right)$, and even to $O\left(\left(n^3/\log(n)\right)L\right)$ when m and n are of the same order;
- QCQP: complexity of $O\left(\sqrt{m}(m+n)n^2L\right)$;
- SOCP: complexity of $O\left(\sqrt{m+1}\;n(n^2+m+(m+1)k^2)L\right)$; and
- SDP: complexity of $O\left(\sqrt{1+mk}\;n(n^2+nmk^2+mk^3)L\right)$, where the matrices are of dimension $k\times k$.

For example, consider an SOCP where the number of constraints and cone dimension are both of the same order as the number of variables (i.e., $m=O(n)$ and $k=O(n)$), then the complexity order is $O(n^{4.5}L)$. Consider now an SDP where the cone dimension is of the same order as the number of variables (i.e., $k=O(n)$), then the complexity order is to $O(n^4)$; if, in addition, the number of constraints also grows with the number of variables (i.e., $m=O(n)$), then the complexity order becomes $O\left(n^6L\right)$. We can clearly see that the complexity for solving SOCP is much higher than that of LP, QP, and QCQP, and it only gets worse for SDP.

B.1.3 Interface with Solvers

Solvers require that problems be expressed in a standard form; that is, the arguments to be passed to the solvers must be formatted in a very specific way. However, most formulated problems do not immediately present themselves in a standard form and they must be transformed. This process is time-consuming and, more importantly, it is susceptible to human transcription errors, as illustrated by the following examples.

Example B.1 (Norm approximation problem) Consider the problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|,$$

whose solution depends on the choice of the norm and so does the process of conversion to standard form.

- If we choose the Euclidean or ℓ_2-norm, $\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2$, then the problem is just a least squares (LS) with analytic solution $\boldsymbol{x}^\star=(\boldsymbol{A}^\mathsf{T}\boldsymbol{A})^{-1}\boldsymbol{A}^\mathsf{T}\boldsymbol{b}$.
- If we choose the Chebyshev or ℓ_∞-norm, $\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_\infty$, then the problem can be rewritten as the LP

$$\begin{array}{ll}\underset{\boldsymbol{x},t}{\text{minimize}} & t\\ \text{subject to} & -t\mathbf{1}\le\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\le t\mathbf{1}\end{array}$$

[9] The "big O" notation, $O(\cdot)$, measures the order of complexity. To be specific, we say that the complexity is $O\left(g(N)\right)$, as $N\to\infty$, if there exists a positive real number M and N_0 such that the complexity is upper-bounded by $Mg(N)$ for all $N\ge N_0$.

or, equivalently,

$$\begin{array}{ll} \underset{\boldsymbol{x},t}{\text{minimize}} & \begin{bmatrix} \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ t \end{bmatrix} \\ \text{subject to} & \begin{bmatrix} \boldsymbol{A} & -\mathbf{1} \\ -\boldsymbol{A} & -\mathbf{1} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ t \end{bmatrix} \leq \begin{bmatrix} \boldsymbol{b} \\ -\boldsymbol{b} \end{bmatrix}, \end{array}$$

which finally can be written as the MATLAB code

```
xt = linprog( [zeros(n,1); 1],
              [A,-ones(m,1); -A,-ones(m,1)],
              [b; -b] )
x = xt(1:n)
```

- If we choose the Manhattan or ℓ_1-norm, $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1$, then the problem can be rewritten as the LP

$$\begin{array}{ll} \underset{\boldsymbol{x},\boldsymbol{t}}{\text{minimize}} & \mathbf{1}^\top \boldsymbol{t} \\ \text{subject to} & -\boldsymbol{t} \leq \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \leq \boldsymbol{t} \end{array}$$

or, equivalently,

$$\begin{array}{ll} \underset{\boldsymbol{x},t}{\text{minimize}} & \begin{bmatrix} \mathbf{0}^\top & \mathbf{1}^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{t} \end{bmatrix} \\ \text{subject to} & \begin{bmatrix} \boldsymbol{A} & -\boldsymbol{I} \\ -\boldsymbol{A} & -\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{t} \end{bmatrix} \leq \begin{bmatrix} \boldsymbol{b} \\ -\boldsymbol{b} \end{bmatrix}, \end{array}$$

which finally can be written as the MATLAB code

```
xt = linprog( [zeros(n,1); ones(n,1)],
              [A,-eye(m,1); -A,-eye(m,1)],
              [b; -b] )
x = xt(1:n)
```

The previous example was as simple as it can get. We explore now a slightly more complicated problem with linear constraints and observe how quickly the code becomes complicated and prone to errors.

Example B.2 (Euclidean norm approximation problem with linear constraints) Consider the problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 \\ \text{subject to} & \boldsymbol{C}\boldsymbol{x} = \boldsymbol{d}, \\ & \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}. \end{array}$$

This can be expressed as

$$
\begin{aligned}
\underset{\boldsymbol{x},\boldsymbol{y},t,\boldsymbol{s}_l,\boldsymbol{s}_u}{\text{minimize}} \quad & t \\
\text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{y}, \\
& \boldsymbol{C}\boldsymbol{x} = \boldsymbol{d}, \\
& \boldsymbol{x} - \boldsymbol{s}_l = \boldsymbol{l}, \\
& \boldsymbol{x} + \boldsymbol{s}_u = \boldsymbol{u}, \\
& \boldsymbol{s}_l, \boldsymbol{s}_u \geq \boldsymbol{0}, \\
& \|\boldsymbol{y}\|_2 \leq t
\end{aligned}
$$

or, equivalently,

$$
\begin{aligned}
\underset{\boldsymbol{x},\boldsymbol{y},t,\boldsymbol{s}_l,\boldsymbol{s}_u}{\text{minimize}} \quad & \begin{bmatrix} \boldsymbol{0}^\mathsf{T} & \boldsymbol{0}^\mathsf{T} & \boldsymbol{0}^\mathsf{T} & \boldsymbol{0}^\mathsf{T} & 1 \end{bmatrix} \bar{\boldsymbol{x}} \\
\text{subject to} \quad & \begin{bmatrix} \boldsymbol{A} & & & -\boldsymbol{I} \\ \boldsymbol{C} & & & \\ \boldsymbol{I} & -\boldsymbol{I} & & \\ \boldsymbol{I} & & \boldsymbol{I} & \end{bmatrix} \bar{\boldsymbol{x}} \leq \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{d} \\ \boldsymbol{l} \\ \boldsymbol{u} \end{bmatrix}, \\
& \bar{\boldsymbol{x}} \in \boldsymbol{R}^n \times \boldsymbol{R}^n_+ \times \boldsymbol{R}^n_+ \times \boldsymbol{Q}^m,
\end{aligned}
$$

which finally can be written as the MATLAB code

```
AA = [ A, zeros(m,n), zeros(m,n),   -eye(m),    0;
       C, zeros(p,n), zeros(p,n),   zeros(p,n), 0;
       eye(n), -eye(n), zeros(n,n), zeros(n,n), 0;
       eye(n), zeros(n,n), eye(n),  zeros(n,n), 0 ]
bb = [ b; d; l; u ]
cc = [ zeros(3*n + m, 1); 1 ]
K.f = n; K.l = 2*n; K.q = m + 1
xsyz = sedumi( AA, bb, cc, K )
x = xsyz(1:n)
```

B.1.4 Modeling Frameworks

A *modeling framework* simplifies the use of solvers by shielding the user from the specific details of the solver argument formatting. It effectively acts as an interface between the user and the solver. In fact, a modeling framework can typically interface with a variety of solvers that the user can choose depending on the type of problem. A modeling framework is a fantastic tool for rapid prototyping of models while avoiding transcription errors when writing the code. Nevertheless, if high speed is desired, then one should consider calling the solver directly while avoiding the convenient interface provided by a modeling framework.

Arguably, the two most successful examples (both of which are freely available) are YALMIP[10] (Löfberg, 2004) and CVX.[11] CVX was initially released in 2005 for MATLAB and is now available in Python, R, and Julia (Fu et al., 2020; Grant & Boyd, 2008, 2014).

CVX stands for "convex disciplined programming" and it is a convenient and powerful tool

[10] YALMIP: https://yalmip.github.io
[11] CVX: http://cvxr.com; CVX in R: https://cvxr.rbind.io

for the rapid prototyping of models and algorithms incorporating convex optimization (also allows integer constraints). It interfaces with many solvers such as the free solvers SeDuMi and SDPT3, as well as the commercial solvers Gurobi and MOSEK. Internally, CVX knows the elementary convex and concave functions as well as the rules of composition that preserve convexity. This way, it can determine whether the problem is convex or not. CVX is extremely simple to use and it is very convenient for prototyping while avoiding transcript errors.

Example B.3 (Constrained Euclidean norm approximation in CVX) Consider the problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2 \\ \text{subject to} & \boldsymbol{C}\boldsymbol{x}=\boldsymbol{d}, \\ & \boldsymbol{l}\leq\boldsymbol{x}\leq\boldsymbol{u}. \end{array}$$

The corresponding code in different popular languages (e.g., MATLAB, R, and Python) is almost identical and very simple:

- MATLAB:

```
cvx_begin
    variable x(n)
    minimize(norm(A * x - b, 2))
    subject to
        C * x == d
        l <= x
        x <= u
cvx_end
```

- R:

```
x <- Variable(n)
prob <- Problem(Minimize(cvxr_norm(A %*% x - b, 2)),
                list(C %*% x == d,
                     l <= x,
                     x <= u))
solve(prob)
```

- Python:

```
x = cvxpy.Variable(n)
prob = cvxp.Minimize(cvxpy.norm(A @ x - b, 2),
                     [C @ x == d,
                      l <= x,
                      x <= u])
prob.solve()
```

B.2 Gradient Methods

Consider the unconstrained optimization problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad f(\boldsymbol{x}) \tag{B.1}$$

where f is the objective function (assumed to be continuously differentiable).

An iterative method or algorithm produces a sequence of iterates $\boldsymbol{x}^0, \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots$ that may or may not converge to an optimal solution $\boldsymbol{x}^\star$. In an ideal case with f convex, one may expect that, as the iterations proceed with $k \to \infty$, the objective function converges to the optimal value of the problem,

$$f\left(\boldsymbol{x}^k\right) \to p^\star,$$

and the gradient tends to zero,

$$\nabla f\left(\boldsymbol{x}^k\right) \to \mathbf{0}.$$

For details on iterative methods and convergence properties, the reader is referred to the many available textbooks, such as Bertsekas (1999), Boyd and Vandenberghe (2004), Nocedal and Wright (2006), and Beck (2017). The case of nondifferentiable f leads to subgradient methods (Beck, 2017), which are not considered herein.

B.2.1 Descent Methods

Descent methods (a.k.a. *gradient methods*) satisfy the property that $f\left(\boldsymbol{x}^{k+1}\right) < f\left(\boldsymbol{x}^k\right)$ and obtain the iterates as follows:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k,$$

where $\boldsymbol{d}^k$ is the *search direction* and α^k is the *stepsize* at iteration k.

For a sufficiently small step, the descent property implies that $\boldsymbol{d}$ has to satisfy $\nabla f\left(\boldsymbol{x}\right)^\mathsf{T} \boldsymbol{d} < 0$ and that α has to be properly chosen (if α is too large, even with a descent direction, the descent property may be violated).

B.2.2 Line Search

Line search is the procedure by which the stepsize α is chosen. There are several possible methods but the following two are widely used due to their good theoretical convergence properties as well as their practical performance.

- *Exact line search*: Based on solving the univariate optimization problem:

$$\alpha = \underset{\alpha > 0}{\arg\min}\ f(\boldsymbol{x} + \alpha \boldsymbol{d}).$$

- *Backtracking line search* (a.k.a. *Armijo rule*): Starting at $\alpha = 1$, repeat $\alpha \leftarrow \beta\alpha$ until

$$f(\boldsymbol{x} + \alpha \boldsymbol{d}) < f(\boldsymbol{x}) + \sigma\alpha \nabla f(\boldsymbol{x})^\mathsf{T} \boldsymbol{d},$$

  where $\sigma \in (0, 1/2)$ and $\beta \in (0, 1)$ are given parameters. <!—Figure B.1 illustrates the backtracking line search, where backtraking will proceed until $\alpha \leq \alpha_0$.

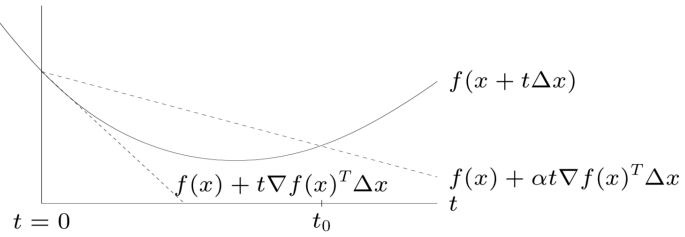


Figure B.1 Backtraking line search.

—>

B.2.3 Gradient Descent Method

The *gradient descent method* (also known as *steepest descent method*) is a descent method where the search direction is chosen as the opposite direction of the gradient:

$$\boldsymbol{d} = -\nabla f(\boldsymbol{x}),$$

which is clearly a descent direction since it satisfies $\nabla f\left(\boldsymbol{x}\right)^{\mathsf{T}} \boldsymbol{d} < 0$.

The gradient descent update is then

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha^k \nabla f\left(\boldsymbol{x}^k\right).$$

As a stopping criterion, the heuristic $\|\nabla f(\boldsymbol{x})\|_2 \leq \epsilon$ is commonly used.

Unfortunately, convergence for the gradient descent method is often slow, so it is rarely used in practice unless there is a need due to the high dimensionality of the problem or a requirement for distributed implementation. The gradient descent method is summarized in Algorithm B.1.

Algorithm B.1: Gradient descent method for the unconstrained problem (B.1).

1: Choose initial point $\boldsymbol{x}^0$;
2: Set $k \leftarrow 0$;
3: **repeat**
4: Compute the negative gradient as descent direction: $\boldsymbol{d}^k = -\nabla f\left(\boldsymbol{x}^k\right)$;
5: Line search: Choose a stepsize $\alpha^k > 0$ via exact or backtracking line search;
6: Obtain next iterate:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha^k \nabla f\left(\boldsymbol{x}^k\right);$$

7: $k \leftarrow k + 1$;
8: **until** convergence;

B.2.4 Newton's Method

Newton's method is a descent method that uses not only the gradient but also the Hessian of f, denoted by $\nabla^2 f(\boldsymbol{x})$, in the search direction:

$$\boldsymbol{d} = -\nabla^2 f(\boldsymbol{x})^{-1} \nabla f(\boldsymbol{x}),$$

which is a descent direction (assuming f is convex). It is tacitly assumed that f is twice continuously differentiable and that the Hessian matrix is positive definite for all $\boldsymbol{x}$.

This direction has the interesting interpretation that $\boldsymbol{x} + \boldsymbol{d}$ minimizes the second-order approximation of $f(\boldsymbol{x})$ around $\boldsymbol{x}$:

$$\hat{f}(\boldsymbol{x} + \boldsymbol{v}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\mathsf{T} \boldsymbol{v} + \frac{1}{2} \boldsymbol{v}^\mathsf{T} \nabla^2 f(\boldsymbol{x}) \boldsymbol{v}.$$

Newton's method update is then

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha^k \nabla^2 f\left(\boldsymbol{x}^k\right)^{-1} \nabla f\left(\boldsymbol{x}^k\right).$$

An interesting quantity in Newton's method is the *Newton decrement*

$$\lambda(\boldsymbol{x}) = (\nabla f(\boldsymbol{x})^\mathsf{T} \nabla^2 f(\boldsymbol{x})^{-1} \nabla f(\boldsymbol{x}))^{1/2}$$

that measures the proximity of $\boldsymbol{x}$ to an optimal point. More specifically, it gives an estimate of $f(\boldsymbol{x}) - p^\star$ as

$$f(\boldsymbol{x}) - \inf_{\boldsymbol{y}} \hat{f}(\boldsymbol{y}) = \frac{1}{2} \lambda(\boldsymbol{x})^2,$$

where $\hat{f}(\boldsymbol{x})$ is the quadratic approximation of $f(\boldsymbol{x})$. Observe that the computational cost of the Newton decrement is negligible given that we already have the Newton direction: $\lambda(\boldsymbol{x})^2 = -\nabla f(\boldsymbol{x})^\mathsf{T} \boldsymbol{d}$.

Newton's method enjoys fast convergence and it is at the heart of most modern solvers. Only when the dimensionality of the problem is very large does it becomes impractical to implement due the computation and storage of the Hessian. For such cases there are approximate versions of Newton's method called quasi-Newton's methods (Nocedal & Wright, 2006). Newton's method is summarized in Algorithm B.2.

B.2.5 Convergence

Ideally, we would like the sequence of a descent method $\{\boldsymbol{x}^k\}$ to converge to a global minimum. Unfortunately, however, this is much to expect, at least when f is not convex, because of the presence of local minima that are not global. Thus, the most we can expect from a descent method is that it converges to a stationary point. Such a point is a global minimum if f is convex, but this need not be so for nonconvex problems.

Descent methods enjoys nice theoretical convergence (Bertsekas, 1999) as summarized in Theorem B.1.

Algorithm B.2: Newton's method for the unconstrained problem (B.1).

1: Choose initial point $\boldsymbol{x}^0$ and tolerance $\epsilon > 0$;
2: Set $k \leftarrow 0$;
3: **repeat**
4: Compute Newton direction and decrement:

$$\boldsymbol{d}^k = -\nabla^2 f(\boldsymbol{x}^k)^{-1} \nabla f(\boldsymbol{x}^k) \quad \text{and} \quad \lambda(\boldsymbol{x}^k)^2 = -\nabla f(\boldsymbol{x}^k)^\mathsf{T} \boldsymbol{d}^k;$$

5: Line search: Choose a stepsize $\alpha^k > 0$ via exact or backtracking line search;
6: Obtain next iterate:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha^k \nabla^2 f\left(\boldsymbol{x}^k\right)^{-1} \nabla f\left(\boldsymbol{x}^k\right);$$

7: $k \leftarrow k + 1$;
8: **until** convergence (i.e., $\lambda(\boldsymbol{x}^k)^2/2 \leq \epsilon$);

Theorem B.1 (Convergence of descent methods) *Suppose $\{\boldsymbol{x}^k\}$ is a sequence generated by a descent method (such as the gradient descent method or Newton's method) and the stepsize α^k is chosen by exact line search or backtracking line search. Then every limit point of $\{\boldsymbol{x}^k\}$ is a stationary point of the problem.*

Other simpler stepsize rules also enjoy theoretical convergence (Bertsekas, 1999):

- *constant stepsize*: $\alpha^k = \alpha$ for sufficiently small α;
- *dimishing stepsize rule*: $\alpha^k \to 0$ with $\sum_{k=0}^{\infty} \alpha^k = \infty$.

Regarding Newton's method, it is worth knowing that its convergence can be divided into two phases:

1. *damped Newton phase*: the convergence is slow; and
2. *quadratically convergent phase*: the convergence is extremely fast.

In practice, the gradient descent method converges slowly, whereas Newton's method enjoys much faster convergence (at the expense of computing the Hessian). Thus, if the problem dimensionality is manageable, Newton's method is preferred. In some applications, however, the dimensionality can be extremely large (such as in deep learning) and computing and storing the Hessian is not a feasible option.

B.3 Projected Gradient Methods

Consider the constrained optimization problem

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{X}, \end{aligned}$$

where f is the objective function (assumed to be continuously differentiable) and $\mathcal{X}$ is a convex set.

If we were to use a descent method,

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k$$

with $\boldsymbol{d}^k$ a descent direction, we would possibly end up with an infeasible point $\boldsymbol{x}^{k+1}$.

Projected gradient methods or *gradient projection methods* address this issue by projecting onto the feasible set after taking the step (Beck, 2017; Bertsekas, 1999):

$$\boldsymbol{x}^{k+1} = \left[\boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k\right]_{\mathcal{X}},$$

where $[\boldsymbol{x}]_{\mathcal{X}}$ denotes projection onto the set $\mathcal{X}$ defined as the solution to $\min_{\boldsymbol{y}} \|\boldsymbol{y} - \boldsymbol{x}\|$ subject to $\boldsymbol{y} \in \mathcal{X}$.

A slightly more general version of the gradient projection method is

$$\begin{aligned}\bar{\boldsymbol{x}}^k &= \left[\boldsymbol{x}^k + s^k \boldsymbol{d}^k\right]_{\mathcal{X}}, \\ \boldsymbol{x}^{k+1} &= \boldsymbol{x}^k + \alpha^k \left(\bar{\boldsymbol{x}}^k - \boldsymbol{x}^k\right),\end{aligned}$$

where $\boldsymbol{d}^k = \bar{\boldsymbol{x}}^k - \boldsymbol{x}^k$ is a feasible direction (because $\bar{\boldsymbol{x}}^k$ is feasible due to the projection), α^k is the stepsize, and s^k is a positive scalar (Bertsekas, 1999). Note that if we choose $\alpha^k = 1$ then the iteration simplifies to the previous expression:

$$\boldsymbol{x}^{k+1} = \left[\boldsymbol{x}^k + s^k \boldsymbol{d}^k\right]_{\mathcal{X}},$$

where now s^k can be viewed as a stepsize. Further note that if $\boldsymbol{x}^k + s^k \boldsymbol{d}^k$ is already feasible, then the gradient projection method reduces to the regular gradient method.

In practice, the gradient projection method only makes sense if the projection is easy to compute.

B.3.1 Projected Gradient Descent Method

The projected gradient descent method simply uses as search direction the negative gradient:

$$\begin{aligned}\bar{\boldsymbol{x}}^k &= \left[\boldsymbol{x}^k - s^k \nabla f\left(\boldsymbol{x}^k\right)\right]_{\mathcal{X}}, \\ \boldsymbol{x}^{k+1} &= \boldsymbol{x}^k + \alpha^k \left(\bar{\boldsymbol{x}}^k - \boldsymbol{x}^k\right).\end{aligned}$$

B.3.2 Constrained Newton's Method

Let us assume that f is twice continuously differentiable and that the Hessian matrix is positive definite for all $\boldsymbol{x} \in \mathcal{X}$.

The *constrained Newton's method* is

$$\begin{aligned}\bar{\boldsymbol{x}}^k &= \arg\min_{\boldsymbol{x} \in \mathcal{X}} \left\{\nabla f\left(\boldsymbol{x}^k\right)^\mathsf{T} \left(\boldsymbol{x} - \boldsymbol{x}^k\right) + \frac{1}{2s^k} \left(\boldsymbol{x} - \boldsymbol{x}^k\right)^\mathsf{T} \nabla^2 f\left(\boldsymbol{x}^k\right) \left(\boldsymbol{x} - \boldsymbol{x}^k\right)\right\}, \\ \boldsymbol{x}^{k+1} &= \boldsymbol{x}^k + \alpha^k \left(\bar{\boldsymbol{x}}^k - \boldsymbol{x}^k\right).\end{aligned}$$

Note that if $s^k = 1$, the quadratic cost above is the second-order Taylor series expansion of f around $\boldsymbol{x}^k$ (apart from a constant term). In particular, if $\alpha^k = 1$ and $s^k = 1$, then $\boldsymbol{x}^{k+1}$ is the

vector that minimizes the second-order Taylor series expansion around $\boldsymbol{x}^k$, just as in the case of Newton's method for unconstrained optimization.

The main difficulty with this method is in solving the quadratic subproblem to find $\bar{\boldsymbol{x}}^k$. This may not be simple even when the constraint set $\mathcal{X}$ has a simple structure. Thus the method typically makes practical sense only for problems of small dimension.

B.3.3 Convergence

The convergence of gradient projection methods can be found in Bertsekas (1999) and it is summarized in Theorem B.2.

Theorem B.2 (Convergence of gradient projection methods) *Suppose $\{\boldsymbol{x}^k\}$ is a sequence generated by a gradient projection method (such as the projected gradient descent method or the constrained Newton's method) and the stepsize α^k is chosen by exact line search or backtracking line search. Then every limit point of $\{\boldsymbol{x}^k\}$ is a stationary point of the problem.*

Other simpler stepsize rules also enjoy theoretical convergence, such as the constant stepsize $\alpha^k = 1$ and $s^k = s$ for sufficiently small s (Bertsekas, 1999).

B.4 Interior-Point Methods

Traditional optimization algorithms based on gradient projection methods may suffer from slow convergence and sensitivity to the algorithm initialization and stepsize selection. The more modern *interior-point methods* (IPM) for convex problems enjoy excellent convergence properties (polynomial convergence) and do not suffer from the usual problems of the traditional methods. Some standard textbooks that cover IPMs in detail include Nesterov and Nemirovski (1994), Ye (1997), Nesterov (2018), Bertsekas (1999), Nemirovski (2000), Ben-Tal and Nemirovski (2001), Boyd and Vandenberghe (2004), Luenberger and Ye (2021), and Nocedal and Wright (2006).

Consider the following convex optimization problem:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & f_0(\boldsymbol{x}) \\ \text{subject to} & f_i(\boldsymbol{x}) \leq 0, \qquad i = 1, \ldots, m, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{array} \tag{B.2}$$

where all f_i are convex and twice continuously differentiable, and $\boldsymbol{A} \in \mathbb{R}^{p \times n}$ is a fat (i.e., more columns than rows), full rank matrix. We assume that the optimal value $p^\star$ is finite and attained, and that the problem is strictly feasible, hence strong duality holds and dual optimum is attained.

B.4.1 Eliminating Equality Constraints

Equality constraints can be conveniently dealt with via Lagrange duality (Boyd & Vandenberghe, 2004). Alternatively, they can be simply eliminated in a pre-processing stage as shown next.

From linear algebra, we know that the possibly infinite solutions to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ can be represented as follows:

$$\{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\} = \{\boldsymbol{F}\boldsymbol{z} + \boldsymbol{x}_0 \mid \boldsymbol{z} \in \mathbb{R}^{n-p}\},$$

where $\boldsymbol{x}_0$ is any particular solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ and the range of $\boldsymbol{F} \in \mathbb{R}^{n\times(n-p)}$ is the nullspace of $\boldsymbol{A} \in \mathbb{R}^{p\times n}$, that is, $\boldsymbol{A}\boldsymbol{F} = \boldsymbol{0}$.

The *reduced* or *eliminated problem* equivalent to problem (B.2) is

$$\begin{aligned} \underset{\boldsymbol{z}}{\text{minimize}} \quad & \tilde{f}_0(\boldsymbol{z}) \triangleq f_0(\boldsymbol{F}\boldsymbol{z} + \boldsymbol{x}_0) \\ \text{subject to} \quad & \tilde{f}_i(\boldsymbol{z}) \triangleq f_i(\boldsymbol{F}\boldsymbol{z} + \boldsymbol{x}_0) \leq 0, \qquad i = 1, \ldots, m, \end{aligned}$$

where each function $\tilde{f}_i$ has gradient and Hessian given by

$$\begin{aligned} \nabla \tilde{f}_i(\boldsymbol{z}) &= \boldsymbol{F}^\mathsf{T} \nabla f_i(\boldsymbol{x}), \\ \nabla^2 \tilde{f}_i(\boldsymbol{z}) &= \boldsymbol{F}^\mathsf{T} \nabla^2 f_i(\boldsymbol{x}) \boldsymbol{F}. \end{aligned}$$

Once the solution $\boldsymbol{z}^\star$ to the reduced problem has been obtained, the solution to the original problem (B.2) can be readily derived as

$$\boldsymbol{x}^\star = \boldsymbol{F}\boldsymbol{z}^\star + \boldsymbol{x}_0.$$

B.4.2 Indicator Function

We can equivalently reformulate problem (B.2) via the *indicator function* as

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f_0(\boldsymbol{x}) + \textstyle\sum_{i=1}^m I_-\left(f_i(\boldsymbol{x})\right) \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{aligned}$$

where the indicator function is defined as

$$I_-(u) = \begin{cases} 0 & \text{if } u \leq 0, \\ \infty & \text{otherwise.} \end{cases}$$

In this form, the inequality constraints have been eliminated at the expense of the indicator function in the objective. Unfortunately, the indicator function is a noncontinuous, nondifferentiable function, which is not practical.

B.4.3 Logarithmic Barrier

In practice, we need to use some smooth approximation of the indicator function. A very popular and convenient choice is the so-called *logarithmic barrier*:

$$I_-(u) \approx -\frac{1}{t}\log(-u),$$

where $t > 0$ is a parameter that controls the approximation and improves as $t \to \infty$. Figure B.2 illustrates the logarithmic barrier for several values of t.

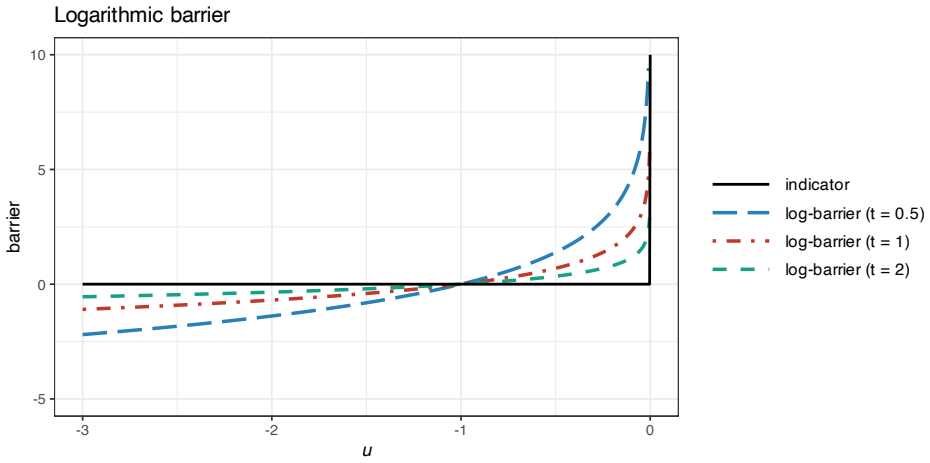


Figure B.2 Logarithmic barrier for several values of the parameter t.

With the logarithmic barrier, we can approximate the problem (B.2) as

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & f_0(\boldsymbol{x}) - \frac{1}{t}\sum_{i=1}^{m} \log\left(-f_i(\boldsymbol{x})\right) \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{array}$$

We can define the overall logarithmic barrier function (excluding the $1/t$ factor) as

$$\phi(\boldsymbol{x}) = -\sum_{i=1}^{m} \log\left(-f_i(\boldsymbol{x})\right),$$

which is convex (from the composition rules in Section A.3) with gradient and Hessian given by

$$\begin{aligned} \nabla\phi(\boldsymbol{x}) &= \sum_{i=1}^{m} \frac{1}{-f_i(\boldsymbol{x})} \nabla f_i(\boldsymbol{x}), \\ \nabla^2\phi(\boldsymbol{x}) &= \sum_{i=1}^{m} \frac{1}{f_i(\boldsymbol{x})^2} \nabla f_i(\boldsymbol{x}) \nabla f_i(\boldsymbol{x})^\mathsf{T} + \sum_{i=1}^{m} \frac{1}{-f_i(\boldsymbol{x})} \nabla^2 f_i(\boldsymbol{x}). \end{aligned}$$

B.4.4 Central Path

For convenience, we multiply the objective function by a positive scalar $t > 0$ and define $\boldsymbol{x}^\star(t)$ as the solution to the equivalent problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & t f_0(\boldsymbol{x}) + \phi(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{array} \tag{B.3}$$

which can be conveniently solved via Newton's method.

The *central path* is defined as the curve $\{\boldsymbol{x}^\star(t) \mid t > 0\}$ (assuming that the solution for each t is unique). Figure B.3 illustrates the central path of an LP.

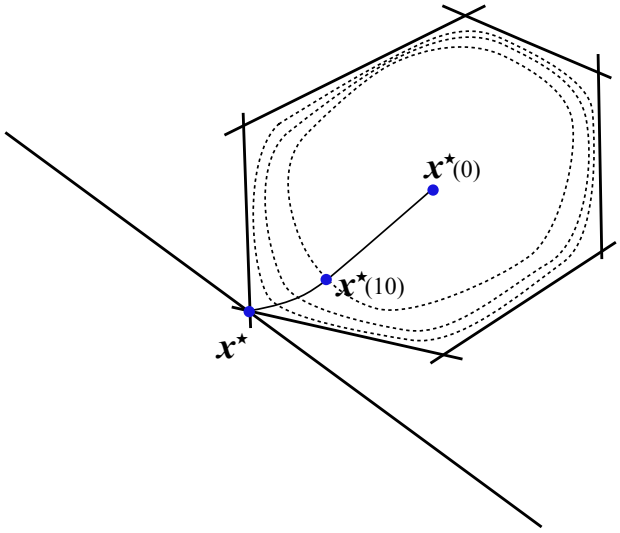


Figure B.3 Example of central path of an LP.

Ignoring the equality constraints for convenience of exposition, a solution to (B.3), $\boldsymbol{x}^\star(t)$, must satisfy the equation

$$t\nabla f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \frac{1}{-f_i(\boldsymbol{x})} \nabla f_i(\boldsymbol{x}) = \boldsymbol{0}.$$

Equivalently, defining $\lambda_i^\star(t) = 1/(-t f_i(\boldsymbol{x}^\star(t)))$, the point $\boldsymbol{x}^\star(t)$ minimizes the Lagrangian

$$L(\boldsymbol{x}; \boldsymbol{\lambda}^\star(t)) = f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i^\star(t) f_i(\boldsymbol{x}).$$

This confirms the intuitive idea that $f_0(\boldsymbol{x}^\star(t)) \to p^\star$ as $t \to \infty$. In particular, from Lagrange duality theory (refer to Section A.6.3 in Appendix A for details),

$$\begin{aligned} p^\star &\geq g\left(\boldsymbol{\lambda}^\star(t)\right) \\ &= L\left(\boldsymbol{x}^\star(t); \boldsymbol{\lambda}^\star(t)\right) \\ &= f_0\left(\boldsymbol{x}^\star(t)\right) - m/t. \end{aligned}$$

In addition, the central path has a nice connection with the Karush–Kuhn–Tucker (KKT) optimality conditions (refer to Section A.6.4 for details). In particular, the point $\boldsymbol{x}^\star(t)$ together with the dual point $\boldsymbol{\lambda}^\star(t)$ satisfy the following:

$$\begin{aligned} f_i(\boldsymbol{x}) &\leq 0, \quad i = 1, \ldots, m, && \text{(primal feasibility)} \\ \lambda_i &\geq 0, \quad i = 1, \ldots, m, && \text{(dual feasibility)} \\ \lambda_i f_i(\boldsymbol{x}) &= -\tfrac{1}{t}, \quad i = 1, \ldots, m, && \text{(approximate complementary slackness)} \\ \nabla f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \nabla f_i(\boldsymbol{x}) &= \boldsymbol{0}. && \text{(zero Lagrangian gradient)} \end{aligned}$$

Clearly, the difference from the KKT conditions of the original constrained problem (see

(A.23) in Appendix A) is that the complementary slackness is now approximately satisfied, with the approximation becoming better as $t \to \infty$.

B.4.5 Barrier Method

Problem (B.3) with the logarithmic barrier is a smooth approximation of the original constrained problem (B.2), and the approximation gets better as $t \to \infty$.

One might be tempted to choose a very large value for t and then apply Newton's method to solve the problem. However, this approach would lead to very slow convergence because the gradients and Hessians would experience extremely large variations near the boundary of the feasible set. Consequently, the convergence of Newton's method would be hindered, failing to reach the phase of quadratic convergence. On the other hand, a small value of t would facilitate better convergence, but the resulting approximation would not be satisfactorily close to the original problem.

Instead of directly choosing one value for t, we may change it over the iterations in an appropriate way, in order to achieve the best of both worlds: fast convergence and an accurate approximation to the original problem. More specifically, at each outer iteration, the value of t is updated and then the solution $\boldsymbol{x}^\star(t)$ is computed (the so-called centering step) with Newton's method starting at the current point $\boldsymbol{x}$ and running over a few inner iterations. Interior-point methods achieve precisely such a trade-off and they get their name because for each value of $t > 0$ the solution $\boldsymbol{x}^\star(t)$ is strictly feasible and lies in the interior of the feasible set.

The *barrier method*, which is one type of primal-based IPM, uses a simple, yet effective, way to update the value of t with the iterations as $t^{k+1} \leftarrow \mu t^k$, where $\mu > 1$ is a parameter and typically $t^0 = 1$. The choice of μ involves a trade-off: large μ means fewer outer iterations, but more inner (Newton) iterations; typical values are $\mu = 10$ to around 20. In addition, the termination criterion can be chosen as $m/t < \epsilon$, guaranteeing $f_0(\boldsymbol{x}) - p^\star \leq \epsilon$. Algorithm B.3 summarizes the implementation of the barrier method; the reader is referred to Boyd and Vandenberghe (2004) for practical discussions on the details.

Algorithm B.3: Barrier method for the constrained problem (B.2).

1: Choose initial point $\boldsymbol{x}^0 \in \mathcal{X}$ stricly feasible, $t^0 > 0$, $\mu > 1$, and tolerance $\epsilon > 0$;
2: Set $k \leftarrow 0$;
3: **repeat**
4: Centering step: compute next iterate $\boldsymbol{x}^{k+1}$ by solving problem (B.3) with $t = t^k$ and initial point $\boldsymbol{x}^k$;
5: Increase t: $t^{k+1} \leftarrow \mu t^k$;
6: $k \leftarrow k + 1$;
7: **until** convergence (i.e., $m/t < \epsilon$);

Example B.4 (Barrier method for LP) Consider the following LP:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}. \end{aligned}$$

We employ Algorithm B.3 with different values of μ to see the effect of this parameter in the total number of Newton iterations.

Figure B.4 shows the convergence for the case of $m = 100$ inequalities and $n = 50$ variables, with $\epsilon = 10^{-6}$ for the duality gap (the centering problem is solved via Newton's method). We can verify that the total number of Newton iterations is not very sensitive to the choice of μ as long as $\mu \geq 10$.

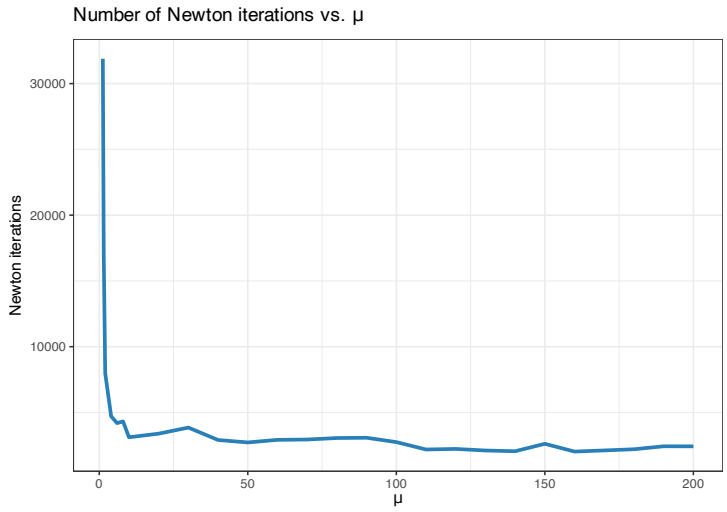


Figure B.4 Convergence of barrier method for an LP for different values of μ.

B.4.6 Convergence

From the termination criterion in Algorithm B.3, the number of outer iterations (or centering steps) required is k such that

$$\frac{m}{\mu^k t^0} \leq \epsilon,$$

that is,

$$\left\lceil \frac{\log\left(m/\left(\epsilon t^0\right)\right)}{\log(\mu)} \right\rceil,$$

where $\lceil \cdot \rceil$ is the ceiling operator. The convergence of each centering step can also be characterized via the convergence for Newton's method. But this approach ignores the specific updates for μ and the good initialization points for each centering step.

For a convergence analysis, the reader is referred to Nesterov and Nemirovski (1994),

Nemirovski (2000), Ben-Tal and Nemirovski (2001), Boyd and Vandenberghe (2004), Nesterov (2018), Luenberger and Ye (2021), and Nocedal and Wright (2006).

B.4.7 Feasibility and Phase I Methods

The barrier method in Algorithm B.3 contains a small hidden detail: the need for a strictly feasible initial point $\boldsymbol{x}^0$ (such that $f_i\left(\boldsymbol{x}^0\right) < 0$). When such a point is not known, the barrier method is preceded by a preliminary stage, called *phase I*, in which a strictly feasible point is computed. Such a point can then be used as a starting point for the barrier method, which is then called *phase II*.

There are several phase I methods that can be used to find a feasible point for problem (B.2) by solving the feasibility problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{find}} & \boldsymbol{x} \\ \text{subject to} & f_i(\boldsymbol{x}) \leq 0, \qquad i = 1, \ldots, m, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{array}$$

However, note that a barrier method cannot be used to solve the feasibility problem directly because it would also require a feasible starting point.

Phase I methods have to be formulated in a convenient way so that a feasible point can be readily constructed and a barrier method can then be used. A simple example relies on solving the convex optimization problem

$$\begin{array}{ll} \underset{\boldsymbol{x},s}{\text{minimize}} & s \\ \text{subject to} & f_i(\boldsymbol{x}) \leq s, \qquad i = 1, \ldots, m, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{array}$$

A strictly feasible point can be easily constructed for this problem: choose any value of $\boldsymbol{x}$ that satisfies the equality constraints, and then choose s satisfying $s > f_i(\boldsymbol{x})$ such as $s = 1.1 \times \max_i\{f_i(\boldsymbol{x})\}$. With such an initial strictly feasible point, the feasibility problem can be solved, obtaining $(\boldsymbol{x}^\star, s^\star)$. If $s^\star < 0$, then $\boldsymbol{x}^\star$ is a strictly feasible point that can be used in the barrier method to solve the original problem (B.2). However, if $s^\star > 0$, it means that no feasible point exists and there is no need to even attempt to solve problem (B.2) since it is infeasible.

B.4.8 Primal-Dual Interior-Point Methods

The barrier method described herein is a primal version of an IPM. However, there are more sophisticated primal–dual IPMs that are more efficient than the primal barrier method when high accuracy is needed. They often exhibit superlinear asymptotic convergence.

The idea is to update the primal and dual variables at each iteration; as a consequence, there is no distinction between inner and outer iterations. In addition, they can start at infeasible points, which alleviates the need for phase I methods.

B.5 Fractional Programming Methods

Consider the *concave–convex fractional program* (FP) (refer to Section A.5 for details)

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{maximize}} & \dfrac{f(\boldsymbol{x})}{g(\boldsymbol{x})} \\ \text{subject to} & \boldsymbol{x} \in \mathcal{X}, \end{array} \tag{B.4}$$

where $f(\boldsymbol{x})$ is a concave function, $g(\boldsymbol{x}) > 0$ is a convex function, and $\mathcal{X}$ denotes the convex feasible set.

Recall that FPs are nonconvex problems, so in principle they are difficult to solve (Stancu-Minasian, 1997). Fortunately, the concave–convex FP in (B.4), albeit still being nonconvex, is a quasi-convex optimization problem and can be conveniently solved. In the following, we briefly look at three practical methods to solve the FP in (B.4), namely, the iterative bisection method, the Dinkelbach method, which is still an iterative method, and the Schaible transform, which allows a direct convex reformulation.

B.5.1 Bisection Method

Problem (B.4) is a quasi-convex problem and can be conveniently solved by solving a sequence of convex feasibility problems (see Section A.4.4 in Appendix A for details) of the form:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{find}} & \boldsymbol{x} \\ \text{subject to} & tg(\boldsymbol{x}) \leq f(\boldsymbol{x}), \\ & \boldsymbol{x} \in \mathcal{X}, \end{array} \tag{B.5}$$

where $t > 0$ is not an optimization variable but a fixed parameter.

The goal is to find the right value of t to be equal to the optimal value of problem (B.4). This can be done iteratively based on the following simple observation: if the feasibility problem (B.5) is infeasible, then t is too large and has to be decreased, otherwise t is probably too small and can be further increased.

The *bisection method*, also known as the *sandwich technique*, performs a series of attempts for the parameter t in a very efficient way by halving an interval known to contain the optimal value of t at each iteration. More specifically, suppose that the original problem (B.4) is feasible and that we start with an interval $[l, u]$ known to contain the optimal value denoted by $t^\star$, we can then solve the convex feasibility problem at the interval midpoint $t = (l + u)/2$, to determine whether the optimal value is in the lower or upper half of this interval, and update the interval accordingly. This produces a new interval, which also contains the optimal value, but has half the width of the initial interval; so the length of the interval after k iterations is $2^{-k}(u - l)$, where $(u - l)$ is the length of the initial interval. Therefore, if a tolerance of ϵ is desired in the computation of $t^\star$, the number of iterations is $\lceil \log_2 ((u - l)/\epsilon) \rceil$, where $\lceil \cdot \rceil$ denotes the ceiling rounding operation. This procedure is summarized in Algorithm B.4 (which is a particular case of the more general Algorithm A.1 in Section A.4).

Algorithm B.4: Bisection method to solve the concave–convex FP in (B.4).

Choose interval $[l, u]$ containing optimal value of problem (B.4) and tolerance $\epsilon > 0$;
repeat
  $t \leftarrow (l + u)/2$;
  Solve the convex feasibility problem (B.5);
  if feasible **then**
    $l \leftarrow t$ and keep solution $\boldsymbol{x}$;
  else
    $u \leftarrow t$;
  end if
until convergence (*i.e.*, $u - l \leq \epsilon$);

B.5.2 Dinkelback Method

The *Dinkelback transform* (Dinkelbach, 1967) reformulates the original concave–convex FP in (B.4) into a sequence of simpler convex problems of the form

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{maximize}} & f(\boldsymbol{x}) - y^k g(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathcal{X}, \end{array} \tag{B.6}$$

where the parameter y^k is sequentially updated as $y^k = f(\boldsymbol{x}^k)/g(\boldsymbol{x}^k)$ with k the iteration index. This is summarized in Algorithm B.5.

Algorithm B.5: Dinkelbach method to solve the concave–convex FP in (B.4).

Choose initial point $\boldsymbol{x}^0$;
Set $k \leftarrow 0$;
repeat
  Set $y^k = f(\boldsymbol{x}^k)/g(\boldsymbol{x}^k)$;
  Solve the convex problem (B.6) and keep current solution as $\boldsymbol{x}^{k+1}$;
  $k \leftarrow k + 1$;
until convergence;

The Dinkelbach method can be shown to converge to the global optimum of the original concave–convex FP in (B.4) by carefully analyzing the increasingness of $\{y^k\}$ and the function $F(y) = \arg\max_{\boldsymbol{x}} f(\boldsymbol{x}) - yg(\boldsymbol{x})$.

B.5.3 Charnes–Cooper–Schaible Transform

We first consider the linear FP case and then move to the more general concave–convex FP.

Charnes–Cooper Transform

One particular case of FP is when both f and g are linear as well as the constraint set, termed *linear FP* (LFP):

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \dfrac{\boldsymbol{c}^\top \boldsymbol{x} + d}{\boldsymbol{e}^\top \boldsymbol{x} + f} \\ \text{subject to} & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h}, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{array} \tag{B.7}$$

with $\operatorname{dom} f_0 = \left\{ \boldsymbol{x} \mid \boldsymbol{e}^\top \boldsymbol{x} + f > 0 \right\}$.

Interestingly, the LFP in (B.7) can be transformed into the following LP via the Charnes–Cooper transform (Charnes & Cooper, 1962):

$$\begin{array}{ll} \underset{\boldsymbol{y},t}{\text{minimize}} & \boldsymbol{c}^\top \boldsymbol{y} + dt \\ \text{subject to} & \boldsymbol{G}\boldsymbol{y} \leq \boldsymbol{h}t, \\ & \boldsymbol{A}\boldsymbol{y} = \boldsymbol{b}t, \\ & \boldsymbol{e}^\top \boldsymbol{y} + ft = 1, \\ & t \geq 0, \end{array} \tag{B.8}$$

from which the original variable $\boldsymbol{x}$ can be easily recovered from $\boldsymbol{y}$ and t as $\boldsymbol{x} = \boldsymbol{y}/t$; see also Bajalinov (2003).

Proof Define two new variables, $\boldsymbol{y}$ and t, related to the original variable $\boldsymbol{x}$ in the LFP in (B.7) by

$$\boldsymbol{y} = \frac{\boldsymbol{x}}{\boldsymbol{e}^\top \boldsymbol{x} + f} \quad \text{and} \quad t = \frac{1}{\boldsymbol{e}^\top \boldsymbol{x} + f}.$$

Obviously, any feasible point $\boldsymbol{x}$ in the original LFP in (B.7) leads to a feasible point $(\boldsymbol{y}, t)$ in the LP in (B.8) with the same objective value. Conversely, any feasible point $(\boldsymbol{y}, t)$ in the LP in (B.8) leads to a feasible point $\boldsymbol{x}$ in the original LFP in (B.7) via $\boldsymbol{x} = \boldsymbol{y}/t$, also with the same objective value:

$$\frac{\boldsymbol{c}^\top \boldsymbol{y} + dt}{1} = \frac{\boldsymbol{c}^\top \boldsymbol{y} + dt}{\boldsymbol{e}^\top \boldsymbol{y} + ft} = \frac{\boldsymbol{c}^\top \boldsymbol{y}/t + d}{\boldsymbol{e}^\top \boldsymbol{y}/t + f} = \frac{\boldsymbol{c}^\top \boldsymbol{x} + d}{\boldsymbol{e}^\top \boldsymbol{x} + f}.$$

□

Schaible Transform

The *Schaible transform* (Schaible, 1974) is a more general case of the Charnes–Cooper transform that rewrites the original *concave–convex FP* in (B.4) into the following convex problem:

$$\begin{array}{ll} \underset{\boldsymbol{y},t}{\text{maximize}} & t f\left(\dfrac{\boldsymbol{y}}{t}\right) \\ \text{subject to} & t g\left(\dfrac{\boldsymbol{y}}{t}\right) \leq 1, \\ & t \geq 0, \\ & \boldsymbol{y}/t \in \mathcal{X}, \end{array} \tag{B.9}$$

from which the original variable $\boldsymbol{x}$ can be easily recovered from $\boldsymbol{y}$ and t as $\boldsymbol{x} = \boldsymbol{y}/t$.

Proof Define two new variables, $\boldsymbol{y}$ and t, related to the original variable $\boldsymbol{x}$ in the FP in (B.4) by

$$\boldsymbol{y} = \frac{\boldsymbol{x}}{g(\boldsymbol{x})} \quad \text{and} \quad t = \frac{1}{g(\boldsymbol{x})}.$$

Any feasible point $\boldsymbol{x}$ in the original FP in (B.4) leads to a feasible point $(\boldsymbol{y}, t)$ in the convex problem (B.9) (in fact, satisfying $tg\left(\boldsymbol{y}/t\right) = 1$) with the same objective value. Conversely, any feasible point $(\boldsymbol{y}, t)$ in (B.9) leads to a feasible point $\boldsymbol{x}$ in the original FP in (B.4) via $\boldsymbol{x} = \boldsymbol{y}/t$, also with the same objective value:

$$tf\left(\frac{\boldsymbol{y}}{t}\right) = \frac{f\left(\boldsymbol{x}\right)}{g\left(\boldsymbol{x}\right)}.$$

□

Observe that if instead of maximizing a concave–convex FP as in (B.4), we consider the minimization of a convex-concave FP, the Schaible transform similarly leads to the following convex problem:

$$\begin{array}{ll} \underset{\boldsymbol{y},t}{\text{minimize}} & tf\left(\dfrac{\boldsymbol{y}}{t}\right) \\ \text{subject to} & tg\left(\dfrac{\boldsymbol{y}}{t}\right) \geq 1, \\ & t \geq 0, \\ & \boldsymbol{y}/t \in \mathcal{X}. \end{array}$$

B.6 Block Coordinate Descent (BCD)

The *block coordinate descent* (BCD) method, also known as the *Gauss–Seidel* or *alternate minimization* method, solves a difficult optimization problem by instead solving a sequence of simpler subproblems. We now give a brief description; for details, the reader is referred to the textbooks Bertsekas (1999), Bertsekas and Tsitsiklis (1997), and Beck (2017).

Consider the following problem where the variable $\boldsymbol{x}$ can be partitioned into n blocks $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ that are separately constrained:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \\ \text{subject to} & \boldsymbol{x}_i \in \mathcal{X}_i, \qquad i = 1, \ldots, n, \end{array} \tag{B.10}$$

where f is the (possibly nonconvex) objective function and each $\mathcal{X}_i$ is a convex set. Instead of attempting to directly obtain a solution $\boldsymbol{x}^\star$ (either a local or global solution), the BCD method will produce a sequence of iterates $\boldsymbol{x}^0, \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots$ that will converge to $\boldsymbol{x}^\star$. Each of these updates is generated by optimizing the problem with respect to each block $\boldsymbol{x}_i$ in a sequential manner, which presumably will be easier to obtain (perhaps even with a closed-form solution).

More specifically, at each outer iteration k, BCD will execute n inner iterations sequentially:

$$\boldsymbol{x}_i^{k+1} = \underset{\boldsymbol{x}_i \in \mathcal{X}_i}{\arg\min}\, f\left(\boldsymbol{x}_1^{k+1}, \ldots, \boldsymbol{x}_{i-1}^{k+1}, \boldsymbol{x}_i, \boldsymbol{x}_{i+1}^{k}, \ldots, \boldsymbol{x}_n^{k}\right), \qquad i = 1, \ldots, n.$$

It is not difficult to see that BCD enjoys monotonicity, that is, $f\left(\boldsymbol{x}^{k+1}\right) \leq f\left(\boldsymbol{x}^{k}\right)$. The BCD method is summarized in Algorithm B.6.

Algorithm B.6: BCD to solve the problem in (B.10).

1: Choose initial point $\boldsymbol{x}^0 = \left(\boldsymbol{x}_1^0, \ldots, \boldsymbol{x}_n^0\right) \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$;
2: Set $k \leftarrow 0$;
3: **repeat**
4: Execute n inner iterations sequentially:

$$\boldsymbol{x}_i^{k+1} = \underset{\boldsymbol{x}_i \in \mathcal{X}_i}{\arg\min}\ f\left(\boldsymbol{x}_1^{k+1}, \ldots, \boldsymbol{x}_{i-1}^{k+1}, \boldsymbol{x}_i, \boldsymbol{x}_{i+1}^{k}, \ldots, \boldsymbol{x}_n^{k}\right), \qquad i = 1, \ldots, n;$$

5: $k \leftarrow k + 1$;
6: **until** convergence;

B.6.1 Convergence

BCD is a very useful framework for deriving simple and practical algorithms. The convergence properties are summarized in Theorem B.3; for details, the reader is referred to Bertsekas and Tsitsiklis (1997) and Bertsekas (1999).

Theorem B.3 (Convergence of BCD) *Suppose that f is (i) continuously differentiable over the convex closed set $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, and (ii) blockwise strictly convex (i.e., in each block variable $\boldsymbol{x}_i$). Then every limit point of the sequence $\{\boldsymbol{x}^k\}$ is a stationary point of the original problem.*

If the convex set $\mathcal{X}$ is compact, that is, closed and bounded, then the blockwise strict convexity can be relaxed to each block optimization having a unique solution.

Theorem B.3 can be extended by further relaxing the blockwise strictly convex condition to any of the following cases (Grippo & Sciandrone, 2000):

- the two-block case $n = 2$;
- the case where f is blockwise strictly quasi-convex with respect to $n - 2$ components; and
- the case where f is pseudo-convex.

B.6.2 Parallel Updates

One may be tempted to execute the n inner iterations in a parallel fashion (as opposed to the sequential update of BCD):

$$\boldsymbol{x}_i^{k+1} = \underset{\boldsymbol{x}_i \in \mathcal{X}_i}{\arg\min}\ f\left(\boldsymbol{x}_1^{k}, \ldots, \boldsymbol{x}_{i-1}^{k}, \boldsymbol{x}_i, \boldsymbol{x}_{i+1}^{k}, \ldots, \boldsymbol{x}_n^{k}\right), \qquad i = 1, \ldots, n.$$

This parallel update is called the *Jacobi* method. Unfortunately, even though the parallel update may be algorithmically attractive, it does not enjoy nice convergence properties. Convergence is guaranteed if the mapping defined by $T(\boldsymbol{x}) = \boldsymbol{x} - \gamma \nabla f(\boldsymbol{x})$ is a contraction for some γ (Bertsekas, 1999).

B.6.3 Illustrative Examples

We now consider a few illustrative examples of applications of BCD (or the Gauss–Seidel method), along with numerical experiments.

It is convenient to start with the introduction of the popular so-called *soft-thresholding operator* (Zibulevsky & Elad, 2010).

Example B.5 (Soft-thresholding operator) Consider the following univariate convex optimization problem:

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|\boldsymbol{a}x - \boldsymbol{b}\|_2^2 + \lambda|x|,$$

with $\lambda \geq 0$.

The solution can be written as

$$x = \frac{1}{\|\boldsymbol{a}\|_2^2}\text{sign}\left(\boldsymbol{a}^\top\boldsymbol{b}\right)\left(|\boldsymbol{a}^\top\boldsymbol{b}| - \lambda\right)^+,$$

where

$$\text{sign}(u) = \begin{cases} +1 & u > 0, \\ 0 & u = 0, \\ -1 & u < 0 \end{cases}$$

is the sign function and $(\cdot)^+ = \max(0, \cdot)$. This can be written more compactly as

$$x = \frac{1}{\|\boldsymbol{a}\|_2^2}\mathcal{S}_\lambda\left(\boldsymbol{a}^\top\boldsymbol{b}\right),$$

where $\mathcal{S}_\lambda(\cdot)$ is the so-called soft-thresholding operator defined as

$$\mathcal{S}_\lambda(u) = \text{sign}(u)(|u| - \lambda)^+ \tag{B.11}$$

and shown in Figure B.5.

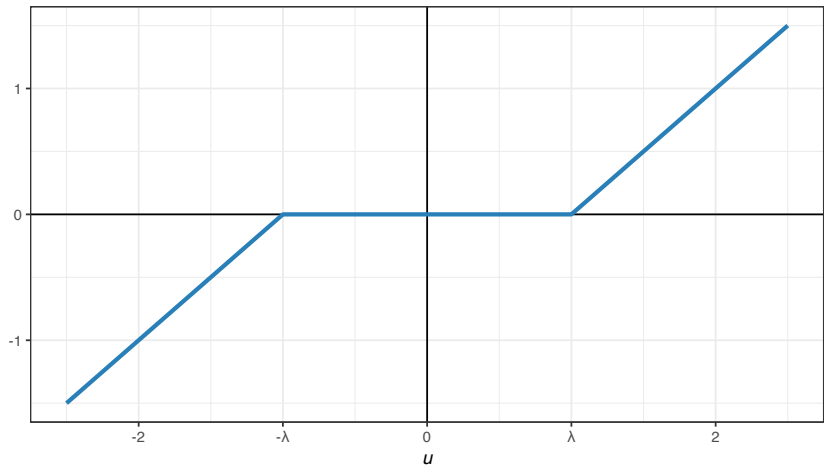


Figure B.5 Soft-thresholding operator.

One useful property of the soft-thresholding operator is the scaling property,

$$\mathcal{S}_\lambda(\kappa \times u) = \kappa \mathcal{S}_{\lambda/\kappa}(u),$$

where $\kappa > 0$ is a positive scaling factor, so we can write

$$x = \frac{1}{\|\boldsymbol{a}\|_2^2} \mathcal{S}_\lambda\left(\boldsymbol{a}^\mathsf{T}\boldsymbol{b}\right) = \mathcal{S}_{\frac{\lambda}{\|\boldsymbol{a}\|_2^2}}\left(\frac{\boldsymbol{a}^\mathsf{T}\boldsymbol{b}}{\|\boldsymbol{a}\|_2^2}\right).$$

Example B.6 (ℓ_2–ℓ_1-norm minimization via BCD) Consider the ℓ_2–ℓ_1-norm minimization problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda\|\boldsymbol{x}\|_1.$$

This problem can be easily solved with a QP solver. However, we will now derive a convenient iterative algorithm via BCD based on the soft-thresholding operator (Zibulevsky & Elad, 2010).

To be specific, we will use BCD by dividing the variable into each constituent element $\boldsymbol{x} = (x_1, \ldots, x_n)$. Therefore, the sequence of problems at each iteration $k = 0, 1, 2, \ldots$ for each element $i = 1, \ldots, n$ is

$$\underset{x_i}{\text{minimize}} \quad \tfrac{1}{2}\left\|\boldsymbol{a}_i x_i - \tilde{\boldsymbol{b}}_i^k\right\|_2^2 + \lambda|x_i|,$$

where $\tilde{\boldsymbol{b}}_i^k \triangleq \boldsymbol{b} - \sum_{j<i} \boldsymbol{a}_j x_j^{k+1} - \sum_{j>i} \boldsymbol{a}_j x_j^k$.

This leads to the following iterative algorithm for $k = 0, 1, 2, \ldots$:

$$x_i^{k+1} = \frac{1}{\|\boldsymbol{a}_i\|_2^2} \mathcal{S}_\lambda\left(\boldsymbol{a}_i^\mathsf{T}\tilde{\boldsymbol{b}}_i^k\right), \qquad i = 1, \ldots, n,$$

where $\mathcal{S}_\lambda(\cdot)$ is the soft-thresholding operator defined in (B.11). Figure B.6 shows the convergence of the BCD iterates.

B.7 Majorization–Minimization (MM)

The *majorization–minimization* (MM) method (or framework) approximates a difficult optimization problem by a sequence of simpler problems. We now give a brief description. For details, the reader is referred to the concise tutorial in Hunter and Lange (2004), the long tutorial with applications in Sun et al. (2017), and the convergence analysis in Razaviyayn et al. (2013).

Suppose the following (difficult) problem is to be solved:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{X}, \end{aligned} \tag{B.12}$$

where f is the (possibly nonconvex) objective function and $\mathcal{X}$ is a (possibly nonconvex) set. Instead of attempting to directly obtain a solution $\boldsymbol{x}^\star$ (either a local or a global solution), the MM method will produce a sequence of iterates $\boldsymbol{x}^0, \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots$ that will converge to $\boldsymbol{x}^\star$.

More specifically, at iteration k, MM approximates the objective function $f(\boldsymbol{x})$ by a *surrogate*

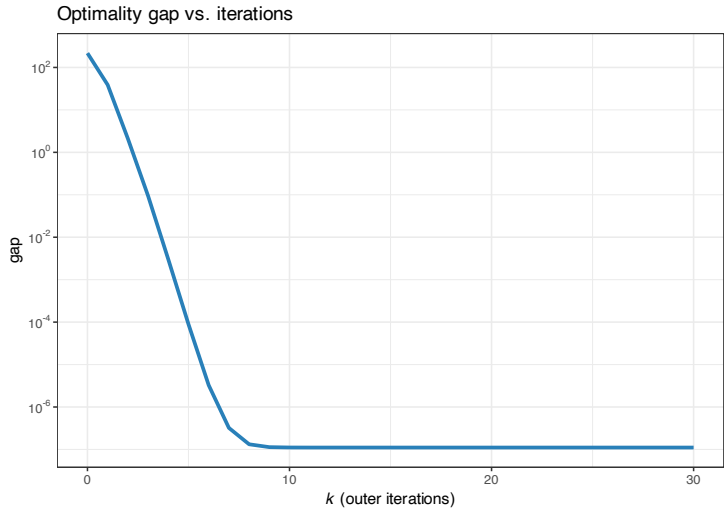


Figure B.6 Convergence of BCD for the ℓ_2–ℓ_1-norm minimization.

function around the current point $\boldsymbol{x}^k$, denoted by $u\left(\boldsymbol{x};\boldsymbol{x}^k\right)$, leading to the sequence of (simpler) problems:

$$\boldsymbol{x}^{k+1} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} u\left(\boldsymbol{x};\boldsymbol{x}^k\right), \qquad k = 0, 1, 2, \ldots$$

This iterative process is illustrated in Figure B.7.

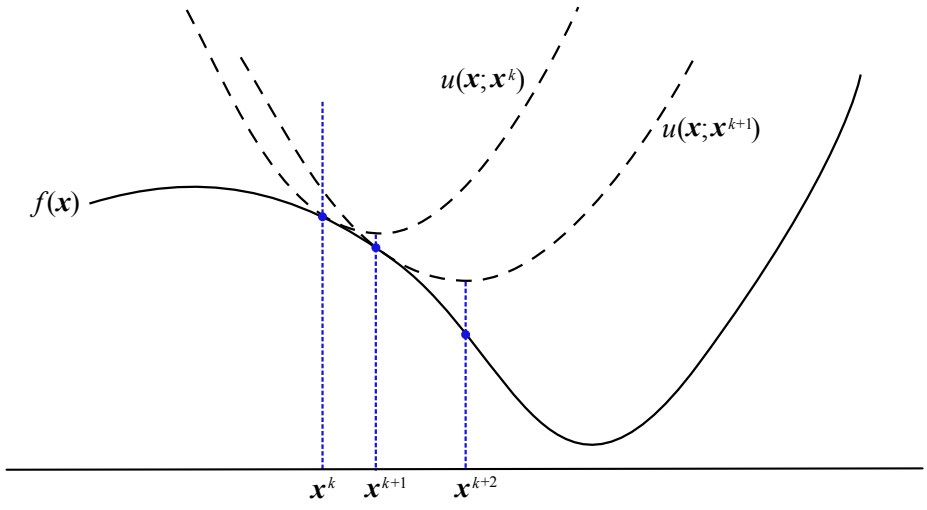


Figure B.7 Illustration of sequence of surrogate problems in MM.

In order to guarantee convergence of the iterates, the surrogate function $u\left(\boldsymbol{x};\boldsymbol{x}^k\right)$ has to satisfy the following technical conditions (Razaviyayn et al., 2013; Sun et al., 2017):

- *upper-bound property*: $u\left(\boldsymbol{x};\boldsymbol{x}^k\right) \geq f\left(\boldsymbol{x}\right)$;

- *touching property*: $u\left(\boldsymbol{x}^k;\boldsymbol{x}^k\right) = f\left(\boldsymbol{x}^k\right)$; and
- *tangent property*: $u\left(\boldsymbol{x};\boldsymbol{x}^k\right)$ must be differentiable with $\nabla u\left(\boldsymbol{x}^k;\boldsymbol{x}^k\right) = \nabla f\left(\boldsymbol{x}^k\right)$, that is, the gradients of u and f must coincide at $\boldsymbol{x}^k$ (not at every $\boldsymbol{x}$).

One immediate consequence of the first two conditions is the monotonicity property of MM, that is, $f\left(\boldsymbol{x}^{k+1}\right) \leq f\left(\boldsymbol{x}^k\right)$, while the third condition is necessary for the convergence of the iterates $\boldsymbol{x}^0, \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots$ – the third condition can be relaxed by requiring continuity and directional derivatives instead (Razaviyayn et al., 2013; Sun et al., 2017). More specifically, under the above technical conditions, if the feasible set $\mathcal{X}$ is convex, then every limit point of the sequence $\{\boldsymbol{x}^k\}$ is a stationary point of the original problem (Razaviyayn et al., 2013).

The surrogate function $u\left(\boldsymbol{x};\boldsymbol{x}^k\right)$ is also referred to as the *majorizer* because it is an upper bound on the original function. The fact that, at each iteration, first the majorizer is constructed and then it is minimized gives the name *majorization–minimization* to the method.

In practice, the difficulty lies in finding the appropriate majorizer $u\left(\boldsymbol{x};\boldsymbol{x}^k\right)$ that satisfies the technical conditions for convergence and, at the same time, leads to a simpler easy-to-solve surrogate problem. The reader is referred to Sun et al. (2017) for techniques and examples on majorizer construction. Once the majorizer has been chosen, the MM description is very simple, as summarized in Algorithm B.7.

Algorithm B.7: MM to solve the general problem in (B.12).

1: Choose initial point $\boldsymbol{x}^0 \in \mathcal{X}$;
2: Set $k \leftarrow 0$;
3: **repeat**
4: Construct majorizer of $f(\boldsymbol{x})$ around current point $\boldsymbol{x}^k$ as $u\left(\boldsymbol{x};\boldsymbol{x}^k\right)$;
5: Obtain next iterate by solving the majorized problem:

$$\boldsymbol{x}^{k+1} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} u\left(\boldsymbol{x};\boldsymbol{x}^k\right);$$

6: $k \leftarrow k+1$;
7: **until** convergence;

B.7.1 Convergence

MM is an extremely versatile framework for deriving practical algorithms. In addition, there are theoretical guarantees for convergence (Razaviyayn et al., 2013; Sun et al., 2017) as summarized in Theorem B.4.

Theorem B.4 (Convergence of MM) *Suppose that the majorizer $u\left(\boldsymbol{x};\boldsymbol{x}^k\right)$ satisfies the previous technical conditions and the feasible set $\mathcal{X}$ is convex. Then, every limit point of the sequence $\{\boldsymbol{x}^k\}$ is a stationary point of the original problem.*

For nonconvex $\mathcal{X}$, convergence must be evaluated on a case-by-case basis – for examples, refer to Song et al. (2015), Sun et al. (2017), Kumar et al. (2019), and Kumar et al. (2020).

B.7.2 Accelerated MM

Unfortunately, in some situations MM may require many iterations to convergence. This may happen if the surrogate function $u\left(\boldsymbol{x};\boldsymbol{x}^k\right)$ is not tight enough due to the strict global upper-bound requirement. For that reason, it is necessary to resort to some acceleration techniques.

One popular quasi-Newton acceleration technique that works well in practice is the so-called SQUAREM (Varadhan & Roland, 2008). Denote the standard MM update as

$$\boldsymbol{x}^{k+1} = \mathrm{MM}(\boldsymbol{x}^k),$$

where

$$\mathrm{MM}(\boldsymbol{x}^k) \triangleq \arg\min_{\boldsymbol{x}\in\mathcal{X}} u\left(\boldsymbol{x};\boldsymbol{x}^k\right).$$

Accelerated MM takes two standard MM steps and combines them in a sophisticated way, with a final third step to guarantee feasibility. The details are as follows:

$$\begin{aligned}
\text{difference first update:} \quad & \boldsymbol{r}^k = R(\boldsymbol{x}^k) \triangleq \mathrm{MM}(\boldsymbol{x}^k) - \boldsymbol{x}^k;\\
\text{difference of differences:} \quad & \boldsymbol{v}^k = R(\mathrm{MM}(\boldsymbol{x}^k)) - R(\boldsymbol{x}^k);\\
\text{stepsize:} \quad & \alpha^k = -\max\left(1, \|\boldsymbol{r}^k\|_2/\|\boldsymbol{v}^k\|_2\right);\\
\text{actual step taken:} \quad & \boldsymbol{y}^k = \boldsymbol{x}^k - \alpha^k\boldsymbol{r}^k;\\
\text{final update on actual step:} \quad & \boldsymbol{x}^{k+1} = \mathrm{MM}(\boldsymbol{y}^k).
\end{aligned}$$

The last step, $\boldsymbol{x}^{k+1} = \mathrm{MM}(\boldsymbol{y}^k)$, can be eliminated (to avoid having to solve the majorized problem a third time) if we can make sure $\boldsymbol{y}^k$ is feasible, becoming $\boldsymbol{x}^{k+1} = \boldsymbol{y}^k$.

B.7.3 Illustrative Examples

We now examine a few illustrative examples of application of MM, along with numerical experiments.

Example B.7 (Nonnegative LS via MM) Consider the nonnegative least-squares problem

$$\underset{\boldsymbol{x}\geq\boldsymbol{0}}{\text{minimize}} \quad \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2,$$

where $\boldsymbol{b}\in\mathbb{R}_+^m$ has nonnegative elements and $\boldsymbol{A}\in\mathbb{R}_{++}^{m\times n}$ has positive elements.

Due to the nonnegativity constraints, we cannot use the conventional LS closed-form solution $\boldsymbol{x}^\star = (\boldsymbol{A}^\mathsf{T}\boldsymbol{A})^{-1}\boldsymbol{A}^\mathsf{T}\boldsymbol{b}$. Since the problem is a QP, we can then resort to using a QP solver. More interestingly, we can develop a simple iterative algorithm based on MM.

The critical step in MM is to find the appropriate majorizer. In this case, one convenient majorizer of the function $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2$ is

$$u\left(\boldsymbol{x};\boldsymbol{x}^k\right) = f\left(\boldsymbol{x}^k\right) + \nabla f\left(\boldsymbol{x}^k\right)^\mathsf{T}\left(\boldsymbol{x}-\boldsymbol{x}^k\right) + \frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{x}^k\right)^\mathsf{T}\boldsymbol{\Phi}\left(\boldsymbol{x}^k\right)\left(\boldsymbol{x}-\boldsymbol{x}^k\right),$$

where $\nabla f\left(\boldsymbol{x}^k\right) = \boldsymbol{A}^\mathsf{T}\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{A}^\mathsf{T}\boldsymbol{b}$ and $\boldsymbol{\Phi}\left(\boldsymbol{x}^k\right) = \mathrm{Diag}\left(\frac{\left[\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\boldsymbol{x}^k\right]_1}{x_1^k},\ldots,\frac{\left[\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\boldsymbol{x}^k\right]_n}{x_n^k}\right)$. It can be

verified that $u\left(\boldsymbol{x};\boldsymbol{x}^k\right)$ is a valid majorizer because (i) $u\left(\boldsymbol{x};\boldsymbol{x}^k\right) \geq f\left(\boldsymbol{x}\right)$ (which can be proved using Jensen's inequality), (ii) $u\left(\boldsymbol{x}^k;\boldsymbol{x}^k\right) = f\left(\boldsymbol{x}^k\right)$, and (iii) $\nabla u\left(\boldsymbol{x}^k;\boldsymbol{x}^k\right) = \nabla f\left(\boldsymbol{x}^k\right)$.

Therefore, the sequence of majorized problems to be solved for $k = 0, 1, 2, \ldots$ is

$$\underset{\boldsymbol{x}\geq\boldsymbol{0}}{\text{minimize}} \quad \nabla f\left(\boldsymbol{x}^k\right)^{\mathsf{T}}\boldsymbol{x} + \tfrac{1}{2}\left(\boldsymbol{x}-\boldsymbol{x}^k\right)^{\mathsf{T}}\boldsymbol{\Phi}\left(\boldsymbol{x}^k\right)\left(\boldsymbol{x}-\boldsymbol{x}^k\right),$$

with solution $\boldsymbol{x} = \boldsymbol{x}^k - \boldsymbol{\Phi}\left(\boldsymbol{x}^k\right)^{-1}\nabla f\left(\boldsymbol{x}^k\right)$.

This finally leads to the following MM iterative algorithm:

$$\boldsymbol{x}^{k+1} = \boldsymbol{c}^k \odot \boldsymbol{x}^k, \qquad k = 0, 1, 2, \ldots$$

where $c_i^k = \frac{[\boldsymbol{A}^{\mathsf{T}}\boldsymbol{b}]_i}{[\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{x}^k]_i}$ and $\odot$ denotes elementwise product. Figure B.8 shows the convergence of the MM iterates.

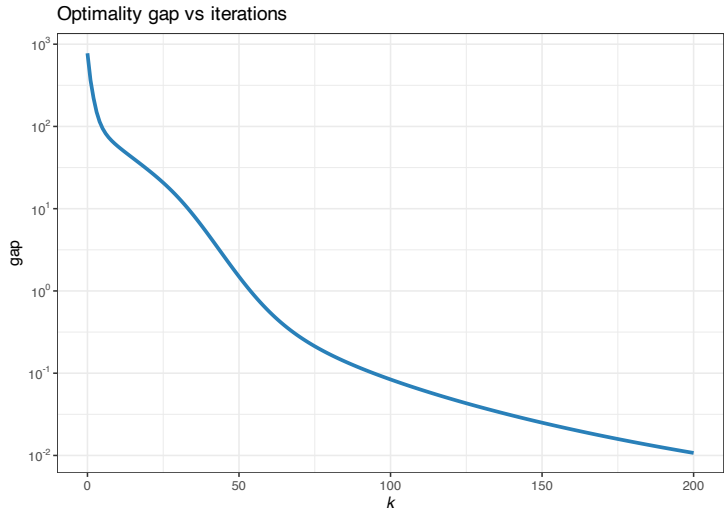


Figure B.8 Convergence of MM for the nonnegative LS.

Example B.8 (Reweighted LS for ℓ_1-norm minimization via MM) Consider the ℓ_1-norm minimization problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_1.$$

If instead we had the ℓ_2-norm in the formulation, then we would have the conventional LS solution $\boldsymbol{x}^\star = (\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{b}$. In any case, this problem is an LP and can be solved with an LP solver. However, we can develop a simple iterative algorithm based on MM that leverages the closed-form LS solution.

The critical step in MM is to find the appropriate majorizer. In this case, since we would like to use the LS solution, we want the majorizer to have the form of an LS problem, that is, we want to go from $\|\cdot\|_1$ to $\|\cdot\|_2^2$. This can be achieved if we manage to majorize $|t|$ in terms of t^2. Indeed, a quadratic majorizer for $f(t) = |t|$ (for simplicitly we assume $t \neq 0$) is (Sun et al.,

2017)

$$u(t;t^k) = \frac{1}{2|t^k|}\left(t^2 + (t^k)^2\right).$$

Consequently, a majorizer of the function $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1$ is

$$u\left(\boldsymbol{x};\boldsymbol{x}^k\right) = \sum_{i=1}^{n} \frac{1}{2\left|[\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b}]_i\right|}\left([\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}]_i^2 - [\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b}]_i^2\right).$$

Therefore, the sequence of majorized problems to be solved for $k = 0, 1, 2, \ldots$ is

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \left\|(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \odot \boldsymbol{c}^k\right\|_2^2,$$

where $c_i^k = \sqrt{\frac{1}{2\left|[\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b}]_i\right|}}$.

This finally leads to the following MM iterative algorithm:

$$\boldsymbol{x}^{k+1} = (\boldsymbol{A}^\mathsf{T}\mathrm{Diag}^2(\boldsymbol{c}^k)\boldsymbol{A})^{-1}\boldsymbol{A}^\mathsf{T}\mathrm{Diag}^2(\boldsymbol{c}^k)\boldsymbol{b}, \qquad k = 0, 1, 2, \ldots$$

Figure B.9 shows the convergence of the MM iterates.

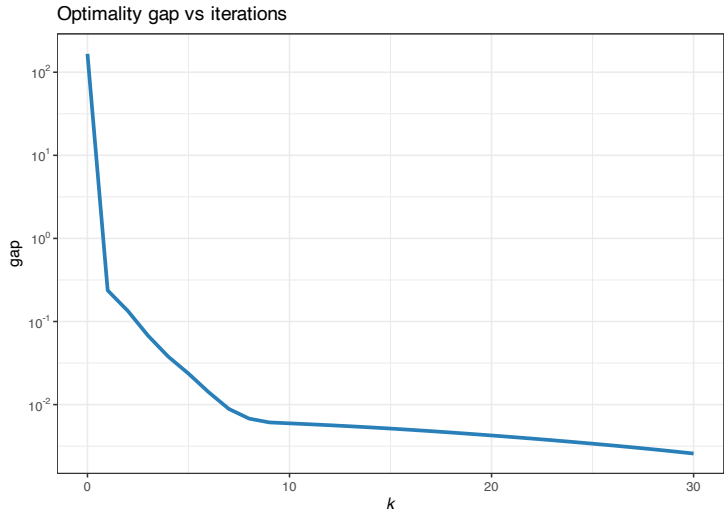


Figure B.9 Convergence of MM for the ℓ_1-norm minimization (reweighted LS).

Example B.9 (ℓ_2–ℓ_1-norm minimization via MM) Consider the ℓ_2–ℓ_1-norm minimization problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda\|\boldsymbol{x}\|_1.$$

This problem can be conveniently solved via BCD (see Section B.6), via SCA (see later Section B.8), or simply by calling a QP solver. Interestingly, we can develop a simple iterative algorithm based on MM that leverages the element-by-element closed-form solution (Zibulevsky & Elad, 2010).

In this case, one possible majorizer of the objective function $f(\boldsymbol{x})$ is

$$u\left(\boldsymbol{x};\boldsymbol{x}^k\right) = \frac{\kappa}{2}\|\boldsymbol{x}-\bar{\boldsymbol{x}}^k\|_2^2 + \lambda\|\boldsymbol{x}\|_1 + \text{constant},$$

where $\bar{\boldsymbol{x}}^k = \boldsymbol{x}^k - \frac{1}{\kappa}\boldsymbol{A}^\mathsf{T}\left(\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b}\right)$. The way to verify that it is indeed a majorizer of $f(\boldsymbol{x})$ is by writing it as

$$u\left(\boldsymbol{x};\boldsymbol{x}^k\right) = f(\boldsymbol{x}) + \text{dist}\left(\boldsymbol{x},\boldsymbol{x}^k\right),$$

where $\text{dist}\left(\boldsymbol{x},\boldsymbol{x}^k\right) = \frac{\kappa}{2}\|\boldsymbol{x}-\boldsymbol{x}^k\|_2^2 - \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{A}\boldsymbol{x}^k\|_2^2$ is a distance measure with $\kappa > \lambda_{\max}\left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\right)$.

Therefore, the sequence of majorized problems to be solved for $k = 0, 1, 2, \ldots$ is

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \frac{\kappa}{2}\left\|\boldsymbol{x}-\bar{\boldsymbol{x}}^k\right\|_2^2 + \lambda\|\boldsymbol{x}\|_1,$$

which decouples into each component of $\boldsymbol{x}$ with closed-form solution given by the soft-thresholding operator (see Section B.6).

This finally leads to the following MM iterative algorithm:

$$\boldsymbol{x}^{k+1} = \mathcal{S}_{\lambda/\kappa}\left(\bar{\boldsymbol{x}}^k\right), \qquad k = 0, 1, 2, \ldots,$$

where $\mathcal{S}_{\lambda/\kappa}(\cdot)$ is the soft-thresholding operator defined in (B.11). Figure B.10 shows the convergence of the MM iterates.

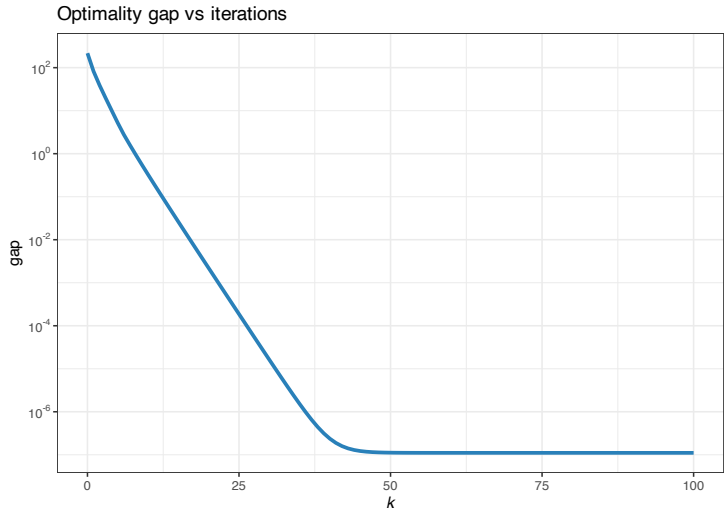


Figure B.10 Convergence of MM for the ℓ_2–ℓ_1-norm minimization.

B.7.4 Block MM

Block MM is a combination of BCD and MM. Suppose that not only is the original problem (B.12) difficult to solve directly, but also a direct application of MM is still too difficult. If the

variables can be partitioned into n blocks $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ that are separately constrained,

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \\ \text{subject to} & \boldsymbol{x}_i \in \mathcal{X}_i, \qquad i = 1, \ldots, n, \end{array}$$

then we can successfully combine BCD and MM. The idea is to solve block by block as in BCD (see Section B.6) but majorizing each block $f(\boldsymbol{x}_i)$ with $u\left(\boldsymbol{x}_i; \boldsymbol{x}^k\right)$ (Razaviyayn et al., 2013; Sun et al., 2017).

B.8 Successive Convex Approximation (SCA)

The *successive convex approximation* (SCA) method (or framework) approximates a difficult optimization problem by a sequence of simpler convex problems. We now give a brief description. For details, the reader is referred to the original paper, Scutari et al. (2014), and the comprehensive book chapter, Scutari and Sun (2018).

Suppose the following (difficult) problem is to be solved:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathcal{X}, \end{array} \tag{B.13}$$

where f is the (possibly nonconvex) objective function and $\mathcal{X}$ is a convex set. Nonconvex sets can also be accommodated at the expense of significantly complicating the method (Scutari & Sun, 2018). Instead of attempting to directly obtain a solution $\boldsymbol{x}^\star$ (either a local or a global solution), the SCA method will produce a sequence of iterates $\boldsymbol{x}^0, \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots$ that will converge to $\boldsymbol{x}^\star$.

More specifically, at iteration k, the SCA approximates the objective function $f(\boldsymbol{x})$ by a surrogate function around the current point $\boldsymbol{x}^k$, denoted by $\tilde{f}\left(\boldsymbol{x}; \boldsymbol{x}^k\right)$. This surrogate function need not be an upper bound like in MM; in this sense, it is more relaxed. At this point, one may be tempted to solve the sequence of (simpler) problems:

$$\boldsymbol{x}^{k+1} = \underset{\boldsymbol{x} \in \mathcal{X}}{\arg\min}\ \tilde{f}\left(\boldsymbol{x}; \boldsymbol{x}^k\right), \qquad k = 0, 1, 2, \ldots$$

Unfortunately, the previous sequence of updates may not converge and a smoothing step is necessary to introduce some memory in the process, which will avoid undesired oscillations. Thus, the correct sequence of problems for the SCA method is

$$\left.\begin{array}{l} \hat{\boldsymbol{x}}^{k+1} = \underset{\boldsymbol{x} \in \mathcal{X}}{\arg\min}\ \tilde{f}\left(\boldsymbol{x}; \boldsymbol{x}^k\right) \\ \boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \gamma^k \left(\hat{\boldsymbol{x}}^{k+1} - \boldsymbol{x}^k\right) \end{array}\right\} \qquad k = 0, 1, 2, \ldots,$$

where $\{\gamma^k\}$ is a properly designed sequence with $\gamma^k \in (0, 1]$ (Scutari et al., 2014). This iterative process is illustrated in Figure B.11; observe that the surrogate function is not an upper bound on the original function.

In order to guarantee convergence of the iterates, the surrogate function $\tilde{f}\left(\boldsymbol{x}; \boldsymbol{x}^k\right)$ has to satisfy the following technical conditions (Scutari et al., 2014):

- $\tilde{f}\left(\boldsymbol{x}; \boldsymbol{x}^k\right)$ must be strongly convex on the feasible set $\mathcal{X}$; and

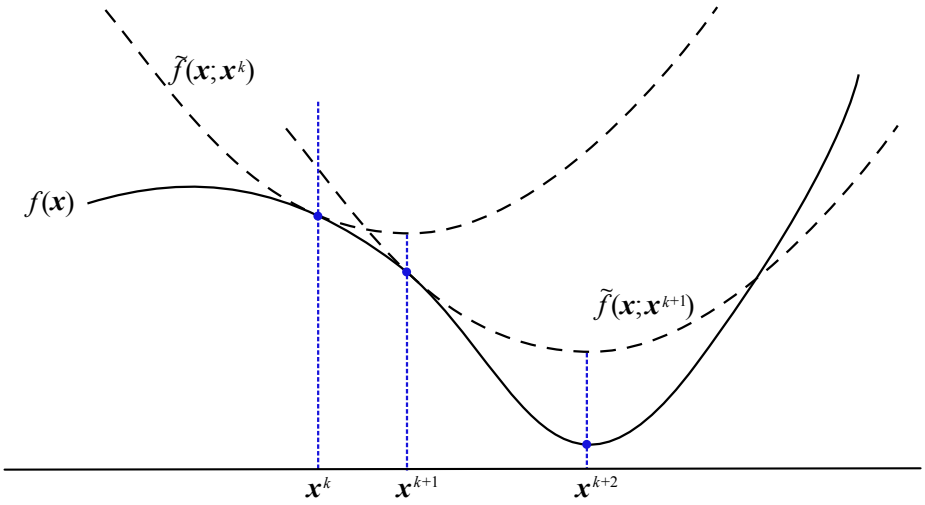


Figure B.11 Illustration of sequence of surrogate problems in SCA.

- $\tilde{f}\left(\boldsymbol{x};\boldsymbol{x}^k\right)$ must be differentiable with $\nabla\tilde{f}\left(\boldsymbol{x}^k;\boldsymbol{x}^k\right) = \nabla f\left(\boldsymbol{x}^k\right)$, that is, the gradients of $\tilde{f}$ and f must coincide at $\boldsymbol{x}^k$ (not at every $\boldsymbol{x}$).

The sequence $\{\gamma^k\}$ can be chosen according to different stepsize rules (Scutari & Sun, 2018; Scutari et al., 2014):

- *Bounded stepsize*: The values γ^k are sufficiently small (difficult to use in practice since the values have to be smaller than some bound which is generally unknown or difficult to compute).
- *Backtracking line search*: This is effective in terms of iterations, but performing the line search requires evaluating the objective function multiple times per iteration, resulting in more costly iterations.
- *Diminishing stepsize*: A good practical choice is a sequence satisfying $\sum_{k=1}^{\infty}\gamma^k = +\infty$ and $\sum_{k=1}^{\infty}(\gamma^k)^2 < +\infty$. Two very effective examples of diminishing stepsize rules are:

$$\gamma^{k+1} = \gamma^k\left(1-\epsilon\gamma^k\right), \quad k = 0,1,\ldots, \quad \gamma^0 < 1/\epsilon,$$

  where $\epsilon \in (0,1)$, and

$$\gamma^{k+1} = \frac{\gamma^k + \alpha^k}{1+\beta^k}, \quad k = 0,1,\ldots, \quad \gamma^0 = 1,$$

  where α^k and β^k satisfy $0 \le \alpha^k \le \beta^k$ and $\alpha^k/\beta^k \to 0$ as $k \to \infty$ while $\sum_k \alpha^k/\beta^k = \infty$. Examples of such α^k and β^k are: $\alpha^k = \alpha$ or $\alpha^k = \log(k)^\alpha$, and $\beta^k = \beta k$ or $\beta^k = \beta\sqrt{k}$, where α and β are given constants satisfying $\alpha \in (0,1)$, $\beta \in (0,1)$, and $\alpha \le \beta$.

SCA takes its name from the fact that the surrogate function is convex by construction. Differently from MM, constructing a convex surrogate function is not difficult. This facilitates the use of SCA in many applications. SCA can be described very simply, as summarized in Algorithm B.8.

Algorithm B.8: SCA to solve the general problem in (B.13).

1: Choose initial point $\boldsymbol{x}^0 \in \mathcal{X}$ and sequence $\{\gamma^k\}$;
2: Set $k \leftarrow 0$;
3: **repeat**
4: Construct surrogate of $f(\boldsymbol{x})$ around current point $\boldsymbol{x}^k$ as $\tilde{f}\left(\boldsymbol{x};\boldsymbol{x}^k\right)$;
5: Obtain intermediate point by solving the surrogate convex problem:

$$\hat{\boldsymbol{x}}^{k+1} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \tilde{f}\left(\boldsymbol{x};\boldsymbol{x}^k\right);$$

6: Obtain next iterate by averaging the intermediate point with the previous one:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \gamma^k\left(\hat{\boldsymbol{x}}^{k+1} - \boldsymbol{x}^k\right);$$

7: $k \leftarrow k+1$;
8: **until** convergence;

B.8.1 Gradient Descent Method as SCA

Consider the unconstrained problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad f(\boldsymbol{x}).$$

If we now employ SCA with the surrogate function

$$\tilde{f}\left(\boldsymbol{x};\boldsymbol{x}^k\right) = f\left(\boldsymbol{x}^k\right) + \nabla f\left(\boldsymbol{x}^k\right)^{\mathsf{T}}\left(\boldsymbol{x}-\boldsymbol{x}^k\right) + \frac{1}{2\alpha^k}\|\boldsymbol{x}-\boldsymbol{x}^k\|^2,$$

then to minimize the convex function $\tilde{f}\left(\boldsymbol{x};\boldsymbol{x}^k\right)$ we just need to set its gradient to zero, leading to the solution $\boldsymbol{x} = \boldsymbol{x}^k - \alpha^k\nabla f\left(\boldsymbol{x}^k\right)$. Thus, the iterations are

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha^k\nabla f\left(\boldsymbol{x}^k\right), \qquad k = 0, 1, 2, \ldots,$$

which coincides with the gradient descent method.

B.8.2 Newton's Method as SCA

If we now include the second-order information (i.e., the Hessian of f) in the surrogate function,

$$\tilde{f}\left(\boldsymbol{x};\boldsymbol{x}^k\right) = f\left(\boldsymbol{x}^k\right) + \nabla f\left(\boldsymbol{x}^k\right)^{\mathsf{T}}\left(\boldsymbol{x}-\boldsymbol{x}^k\right) + \frac{1}{2\alpha^k}\left(\boldsymbol{x}-\boldsymbol{x}^k\right)^{\mathsf{T}}\nabla^2 f\left(\boldsymbol{x}^k\right)\left(\boldsymbol{x}-\boldsymbol{x}^k\right),$$

then the resulting iterations are

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha^k\nabla^2 f\left(\boldsymbol{x}^k\right)^{-1}\nabla f\left(\boldsymbol{x}^k\right), \qquad k = 0, 1, 2, \ldots,$$

which coincides with Newton's method.

B.8.3 Parallel SCA

While SCA applied to problem (B.13) can be instrumental in devising efficient algorithms, its true potential is realized when the variables can be partitioned into n blocks $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ that are separately constrained:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \\ \text{subject to} & \boldsymbol{x}_i \in \mathcal{X}_i, \qquad i = 1, \ldots, n. \end{array}$$

In this case, SCA is able to update each of the variables in a parallel fashion (unlike BCD or block MM where the updates are sequential) by using the surrogate functions $\tilde{f}_i\left(\boldsymbol{x}_i; \boldsymbol{x}^k\right)$ (Scutari et al., 2014):

$$\left.\begin{array}{l} \hat{\boldsymbol{x}}_i^{k+1} = \underset{\boldsymbol{x}_i \in \mathcal{X}_i}{\arg\min}\ \tilde{f}_i\left(\boldsymbol{x}_i; \boldsymbol{x}^k\right) \\ \boldsymbol{x}_i^{k+1} = \boldsymbol{x}_i^k + \gamma^k\left(\hat{\boldsymbol{x}}_i^{k+1} - \boldsymbol{x}_i^k\right) \end{array}\right\} \qquad i = 1, \ldots, n, \quad k = 0, 1, 2, \ldots$$

B.8.4 Convergence

SCA is an extremely versatile framework for deriving practical algorithms. In addition, it enjoys nice theoretical convergence as derived in Scutari et al. (2014) and summarized in Theorem B.5.

Theorem B.5 (Convergence of SCA) *Suppose that the surrogate function $\tilde{f}\left(\boldsymbol{x}; \boldsymbol{x}^k\right)$ (or each $\tilde{f}_i\left(\boldsymbol{x}_i; \boldsymbol{x}^k\right)$ in the parallel version) satisfies the previous technical conditions and $\{\gamma^k\}$ is chosen according to bounded stepsize, diminishing rule, or backtracking line search. Then the sequence $\{\boldsymbol{x}^k\}$ converges to a stationary point of the original problem.*

B.8.5 Illustrative Examples

We now describe a few illustrative examples of applications of SCA, along with numerical experiments.

Example B.10 (ℓ_2–ℓ_1-norm minimization via SCA) Consider the ℓ_2–ℓ_1-norm minimization problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda\|\boldsymbol{x}\|_1.$$

This problem can be conveniently solved via BCD (see Section B.6), via MM (see Section B.7), or simply by calling a QP solver. We will now develop a simple iterative algorithm based on SCA that leverages the element-by-element closed-form solution.

In this case, we can use parallel SCA by partitioning the variable $\boldsymbol{x}$ into each element $(x_1, \ldots, x_n)$ and employing the surrogate functions

$$\tilde{f}\left(\boldsymbol{x}_i; \boldsymbol{x}^k\right) = \frac{1}{2}\left\|\boldsymbol{a}_i x_i - \tilde{\boldsymbol{b}}_i^k\right\|_2^2 + \lambda|x_i| + \frac{\tau}{2}\left(x_i - x_i^k\right)^2,$$

where $\tilde{\boldsymbol{b}}_i^k = \boldsymbol{b} - \sum_{j \neq i} \boldsymbol{a}_j x_j^k$.

Therefore, the sequence of surrogate problems to be solved for $k = 0, 1, 2, \ldots$ is

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\left\|\boldsymbol{a}_i x_i - \tilde{\boldsymbol{b}}_i^k\right\|_2^2 + \lambda|x_i| + \tau\left(x_i - x_i^k\right)^2, \qquad i = 1, \ldots, n,$$

with the solution given by the soft-thresholding operator (see Section B.6).

This finally leads to the following SCA iterative algorithm:

$$\left.\begin{aligned} \hat{x}_i^{k+1} &= \frac{1}{\tau + \|\boldsymbol{a}_i\|^2}\mathcal{S}_\lambda\left(\boldsymbol{a}_i^T\tilde{\boldsymbol{b}}_i^k + \tau x_i^k\right) \\ x_i^{k+1} &= x_i^k + \gamma^k\left(\hat{x}_i^{k+1} - x_i^k\right) \end{aligned}\right\} \qquad i = 1, \ldots, n, \quad k = 0, 1, 2, \ldots,$$

where $\mathcal{S}_\lambda(\cdot)$ is the soft-thresholding operator defined in (B.11). Figure B.12 shows the convergence of the SCA iterates.

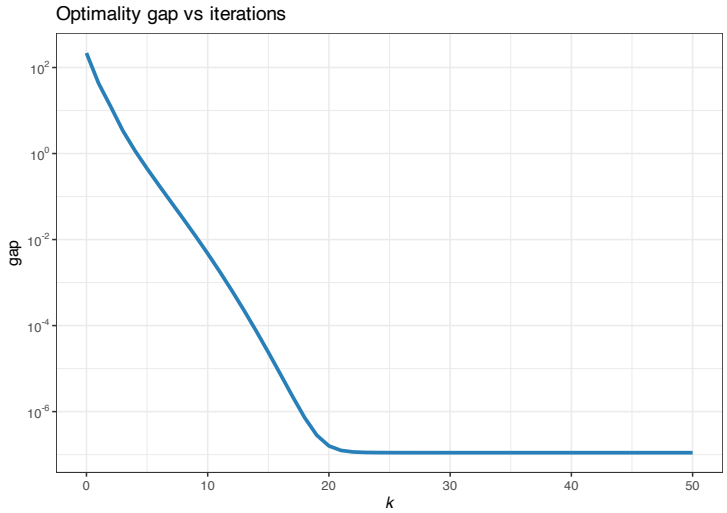


Figure B.12 Convergence of SCA for the ℓ_2–ℓ_1-norm minimization.

Example B.11 (Dictionary learning) Consider the dictionary learning problem:

$$\begin{aligned} \underset{\boldsymbol{D},\boldsymbol{X}}{\text{minimize}} \quad & \tfrac{1}{2}\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 + \lambda\|\boldsymbol{X}\|_1 \\ \text{subject to} \quad & \|[\boldsymbol{D}]_{:,i}\| \le 1, \qquad i = 1, \ldots, m, \end{aligned}$$

where $\|\boldsymbol{D}\|_F$ denotes the matrix Frobenius norm of $\boldsymbol{D}$ (i.e., the ℓ_2-norm of the vectorized form of the matrix or the sum of the squares of all the elements) and $\|\boldsymbol{X}\|_1$ denotes the elementwise ℓ_1-norm of $\boldsymbol{X}$ (i.e., the ℓ_1-norm of the vectorized form of the matrix).

Matrix $\boldsymbol{D}$ is the so-called *dictionary* and it is a fat matrix that contains a dictionary along the columns that can explain the columns of the data matrix $\boldsymbol{Y}$. Matrix $\boldsymbol{X}$ selects a few columns of the dictionary and, hence, has to be sparse (enforced by the regularizer $\|\boldsymbol{X}\|_1$).

This problem is not jointly convex in $(\boldsymbol{D}, \boldsymbol{X})$, but it is *bi-convex*: for a fixed $\boldsymbol{D}$, it is convex in $\boldsymbol{X}$ and, for a fixed $\boldsymbol{X}$, it is convex in $\boldsymbol{D}$. One way to deal with this problem is via BCD (see Section B.6), which would update $\boldsymbol{D}$ and $\boldsymbol{X}$ sequentially. An alternative way is via SCA, which allows a parallel update of $\boldsymbol{D}$ and $\boldsymbol{X}$.

For the SCA approach we can use the following two surrogate functions for $\boldsymbol{D}$ and $\boldsymbol{X}$, respectively:

$$\tilde{f}_1\left(\boldsymbol{D}; \boldsymbol{X}^k\right) = \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}^k\|_F^2,$$
$$\tilde{f}_2\left(\boldsymbol{X}; \boldsymbol{D}^k\right) = \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{D}^k\boldsymbol{X}\|_F^2.$$

This leads to the following two convex problems:

- A normalized LS problem:

$$\begin{aligned} \underset{\boldsymbol{D}}{\text{minimize}} \quad & \tfrac{1}{2}\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}^k\|_F^2 \\ \text{subject to} \quad & \|[\boldsymbol{D}]_{:,i}\| \le 1, \qquad i = 1, \ldots, m. \end{aligned}$$

- A matrix version of the $\ell_2 - \ell_1$-norm problem:

$$\underset{\boldsymbol{X}}{\text{minimize}} \quad \tfrac{1}{2}\|\boldsymbol{Y} - \boldsymbol{D}^k\boldsymbol{X}\|_F^2 + \lambda\|\boldsymbol{X}\|_1,$$

  which can be further decomposed into a set of vectorized $\ell_2 - \ell_1$-norm problems for each column of $\boldsymbol{X}$.

B.8.6 MM vs. SCA

The MM and SCA algorithmic frameworks appear very similar: both are based on solving a sequence of surrogate problems. The difference is in the details, as we compare next.

- *Surrogate function*: MM requires the surrogate function to be a global upper bound (which can be difficult to derive and too restrictive in some cases), albeit not necessarily convex. On the other hand, SCA relaxes this upper-bound condition in exchange for the requirement of strong convexity. Figure B.13 illustrates the potential benefit of relaxing the upper-bound requirement.

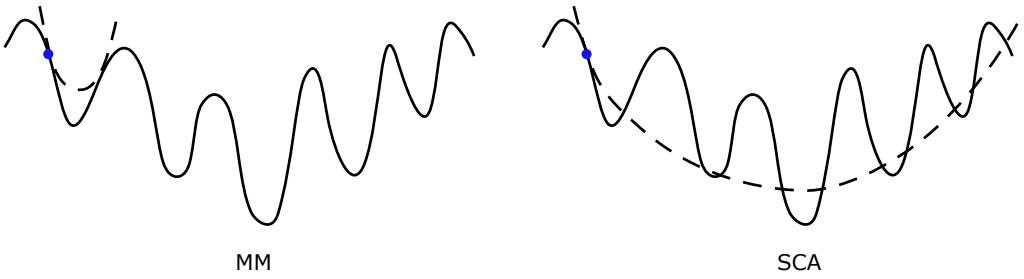


Figure B.13 Illustration of MM vs. SCA: relaxing the upper-bound requirement.

- *Constraint set*: In principle, both SCA and MM require the feasible set $\mathcal{X}$ to be convex. However, the convergence of MM can be easily extended to nonconvex $\mathcal{X}$ on a case-by-case basis – refer to the examples in Song et al. (2015), Sun et al. (2017), Kumar et al. (2019), and Kumar et al. (2020). SCA cannot deal with a nonconvex $\mathcal{X}$ directly, although some extensions allow for a successive convexification of $\mathcal{X}$ at the expense of a much more complex algorithm (Scutari & Sun, 2018).

- *Schedule of updates*: Both MM and SCA can deal with a separable variable $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$. However, block MM requires a sequential update (Razaviyayn et al., 2013; Sun et al., 2017), whereas SCA naturally implements a parallel update, which is more amenable for distributed implementations.

B.9 Alternating Direction Method of Multipliers (ADMM)

The *alternating direction method of multipliers* (ADMM) is a practical algorithm that resembles BCD (see Section B.6), but is able to cope with coupled block variables in the constraints (recall that BCD requires each of the variable blocks to be independently constrained). For details, the reader is referred to Boyd et al. (2010) and Beck (2017).

Consider the convex optimization problem

$$\begin{array}{ll} \underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c}, \end{array} \tag{B.14}$$

where the variables $\boldsymbol{x}$ and $\boldsymbol{z}$ are coupled via the constraint $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c}$. We are going to explore techniques to try to decouple the variables in order to derive simpler practical algorithms rather than attempting to solve (B.14) directly.

A first attempt to decouple the $\boldsymbol{x}$ and $\boldsymbol{z}$ in (B.14) is via the dual problem (see Section A.6 in Appendix A). More exactly, the *dual ascent method* updates the dual variable $\boldsymbol{y}$ via a gradient method and, at each iteration, the Lagrangian is solved for the given $\boldsymbol{y}$:

$$\underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} \quad L(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{y}) \triangleq f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{y}^{\mathsf{T}} \left(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}\right),$$

which now clearly decouples into two separate problems over $\boldsymbol{x}$ and $\boldsymbol{z}$. This decoupling leads to the *dual decomposition method*, developed in the early 1960s, where the primal variables $\boldsymbol{x}$ and $\boldsymbol{z}$ are optimized in parallel:

$$\left.\begin{array}{l} \boldsymbol{x}^{k+1} = \arg\min\limits_{\boldsymbol{x}} \; f(\boldsymbol{x}) + (\boldsymbol{y}^k)^{\mathsf{T}} \boldsymbol{A}\boldsymbol{x} \\ \boldsymbol{z}^{k+1} = \arg\min\limits_{\boldsymbol{z}} \; g(\boldsymbol{z}) + (\boldsymbol{y}^k)^{\mathsf{T}} \boldsymbol{B}\boldsymbol{z} \\ \boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \alpha^k \left(\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}\right) \end{array}\right\} \quad k = 0, 1, 2, \ldots,$$

where α^k is the stepsize. This method is able to decouple the primal variables; however, it requires many technical assumptions to work and it is often slow.

A second attempt to solve problem (B.14) with fewer technical assumptions and with a faster algorithm is via the *method of multipliers*, developed in the late 1960s, which substitutes the Lagrangian by the *augmented Lagrangian*:

$$L_\rho(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{y}) \triangleq f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{y}^{\mathsf{T}} \left(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}\right) + \frac{\rho}{2} \|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}\|_2^2.$$

The algorithm is then

$$\left.\begin{array}{l} (\boldsymbol{x}^{k+1}, \boldsymbol{z}^{k+1}) = \arg\min\limits_{\boldsymbol{x},\boldsymbol{z}} \; L_\rho(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{y}^k) \\ \boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \rho \left(\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}\right) \end{array}\right\} \quad k = 0, 1, 2, \ldots,$$

where the penalty parameter ρ now serves as the (constant) stepsize. This method converges under much more relaxed conditions. Unfortunately, it cannot achieve decoupling in $\boldsymbol{x}$ and $\boldsymbol{z}$ because of the term $\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}\|_2^2$ in the augmented Lagrangian.

A third and final attempt is ADMM, proposed in the mid-1970s, which combines the good features of both the dual decomposition method and the method of multipliers. The idea is to minimize the augmented Lagrangian like in the method of multipliers, but instead of performing a joint minimization over $(\boldsymbol{x}, \boldsymbol{z})$, a single round of the BCD method is performed (see Section B.6). This update follows an alternating or sequential fashion, which accounts for the name "alternating direction," and leads to the ADMM iterations:

$$\left.\begin{aligned} \boldsymbol{x}^{k+1} &= \arg\min_{\boldsymbol{x}} \; L_\rho(\boldsymbol{x}, \boldsymbol{z}^k; \boldsymbol{y}^k) \\ \boldsymbol{z}^{k+1} &= \arg\min_{\boldsymbol{z}} \; L_\rho(\boldsymbol{x}^{k+1}, \boldsymbol{z}; \boldsymbol{y}^k) \\ \boldsymbol{y}^{k+1} &= \boldsymbol{y}^k + \rho\left(\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}\right) \end{aligned}\right\} \qquad k = 0, 1, 2, \ldots$$

This method successfully decouples the primal variables $\boldsymbol{x}$ and $\boldsymbol{z}$, while enjoying faster convergence with few technical conditions required for convergence.

For convenience, it is common to express the ADMM iterative updates in terms of the scaled dual variable $\boldsymbol{u}^k = \boldsymbol{y}^k/\rho$ as used in Algorithm B.9.

Algorithm B.9: ADMM to solve the problem in (B.14).

1: Choose initial point $\left(\boldsymbol{x}^0, \boldsymbol{z}^0\right)$ and ρ;
2: Set $k \leftarrow 0$;
3: **repeat**
4: Iterate primal and dual variables:

$$\begin{aligned} \boldsymbol{x}^{k+1} &= \arg\min_{\boldsymbol{x}} \; f(\boldsymbol{x}) + \frac{\rho}{2}\left\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z}^k - \boldsymbol{c} + \boldsymbol{u}^k\right\|_2^2, \\ \boldsymbol{z}^{k+1} &= \arg\min_{\boldsymbol{z}} \; g(\boldsymbol{z}) + \frac{\rho}{2}\left\|\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c} + \boldsymbol{u}^k\right\|_2^2, \\ \boldsymbol{u}^{k+1} &= \boldsymbol{u}^k + \left(\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}\right); \end{aligned}$$

5: $k \leftarrow k + 1$;
6: **until** convergence;

B.9.1 Convergence

Many convergence results for ADMM are discussed in the literature ((Boyd et al., 2010), and references therein). Theorem B.6 summarizes the most basic, but still very general, convergence result.

Theorem B.6 (Convergence of ADMM) *Suppose that (i) the functions $f(\boldsymbol{x})$ and $g(\boldsymbol{z})$ are convex and both the $\boldsymbol{x}$-update and the $\boldsymbol{z}$-update are solvable, and (ii) the Lagrangian has a saddle point. Then, we have:*

- residual convergence: $\boldsymbol{A}\boldsymbol{x}^k + \boldsymbol{B}\boldsymbol{z}^k - \boldsymbol{c} \to \boldsymbol{0}$ *as* $k \to \infty$*, that is, the iterates approach feasibility;*
- objective convergence: $f(\boldsymbol{x}) + g(\boldsymbol{z}) \to p^\star$ *as* $k \to \infty$*, that is, the objective function of the iterates approaches the optimal value; and*
- dual variable convergence: $\boldsymbol{y}^k \to \boldsymbol{y}^\star$ *as* $k \to \infty$.

Note that $\{\boldsymbol{x}^k\}$ and $\{\boldsymbol{z}^k\}$ need not converge to optimal values, although such results can be shown under additional assumptions.

In practice, ADMM can be slow to converge to high accuracy. However, it is often the case that ADMM converges to modest accuracy within a few tens of iterations (depending on the specific application), which is often sufficient for many practical applications. This behavior is reminiscent of the gradient method (after all, the update of the dual variable is also a first-order method like gradient descent) and also different from the fast convergence of Newton's method.

B.9.2 Illustrative Examples

We now give a few illustrative examples of applications of ADMM, along with numerical experiments.

Example B.12 (Constrained convex optimization) Consider the generic convex optimization problem

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathcal{X}, \end{array}$$

where f is convex and $\mathcal{X}$ is a convex set.

We could naturally use an algorithm for solving constrained optimization problems. Interestingly, we can use ADMM to transform this problem into an unconstrained one. More specifically, we can employ ADMM by defining g to be the indicator function of the feasible set $\mathcal{X}$,

$$g(\boldsymbol{x}) \triangleq \begin{cases} 0 & \boldsymbol{x} \in \mathcal{X}, \\ +\infty & \text{otherwise}, \end{cases}$$

and then formulating the equivalent problem

$$\begin{array}{ll} \underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ \text{subject to} & \boldsymbol{x} - \boldsymbol{z} = \boldsymbol{0}. \end{array}$$

This produces the following ADMM algorithm:

$$\left.\begin{array}{l} \boldsymbol{x}^{k+1} = \arg\min\limits_{\boldsymbol{x}} \; f(\boldsymbol{x}) + \dfrac{\rho}{2}\left\|\boldsymbol{x} - \boldsymbol{z}^k + \boldsymbol{u}^k\right\|_2^2 \\ \boldsymbol{z}^{k+1} = \left[\boldsymbol{x}^{k+1} + \boldsymbol{u}^k\right]_{\mathcal{X}} \\ \boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \left(\boldsymbol{x}^{k+1} - \boldsymbol{z}^{k+1}\right) \end{array}\right\} \quad k = 0, 1, 2, \ldots,$$

where $[\cdot]_{\mathcal{X}}$ denotes projection on the set $\mathcal{X}$.

Example B.13 (ℓ_2–ℓ_1-norm minimization via ADMM) Consider the ℓ_2–ℓ_1-norm minimization problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2+\lambda\|\boldsymbol{x}\|_1.$$

This problem can be conveniently solved via BCD (see Section B.6), MM (see Section B.7), SCA (see Section B.8), or simply by calling a QP solver. We will now develop a simple iterative algorithm based on ADMM that leverages the element-by-element closed-form solution.

We start by reformulating the problem as

$$\begin{array}{ll}\underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} & \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2+\lambda\|\boldsymbol{z}\|_1\\ \text{subject to} & \boldsymbol{x}-\boldsymbol{z}=\boldsymbol{0}.\end{array}$$

The $\boldsymbol{x}$-update, given $\boldsymbol{z}$ and the scaled dual variable $\boldsymbol{u}$, is the solution to

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2+\tfrac{\rho}{2}\|\boldsymbol{x}-\boldsymbol{z}+\boldsymbol{u}\|_2^2,$$

given by $\boldsymbol{x}=\left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}+\rho\boldsymbol{I}\right)^{-1}\left(\boldsymbol{A}^\mathsf{T}\boldsymbol{b}+\rho(\boldsymbol{z}-\boldsymbol{u})\right)$, whereas the $\boldsymbol{z}$-update, given $\boldsymbol{x}$ and $\boldsymbol{u}$, is the solution to

$$\underset{\boldsymbol{z}}{\text{minimize}} \quad \tfrac{\rho}{2}\|\boldsymbol{x}-\boldsymbol{z}+\boldsymbol{u}\|_2^2+\lambda\|\boldsymbol{z}\|_1$$

given by $\boldsymbol{z}=\mathcal{S}_{\lambda/\rho}\left(\boldsymbol{x}+\boldsymbol{u}\right)$, where $\mathcal{S}_{\lambda/\rho}(\cdot)$ is the soft-thresholding operator defined in (B.11).

Thus, the ADMM is finally given by the iterates

$$\left.\begin{array}{l}\boldsymbol{x}^{k+1}=\left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}+\rho\boldsymbol{I}\right)^{-1}\left(\boldsymbol{A}^\mathsf{T}\boldsymbol{b}+\rho\left(\boldsymbol{z}^k-\boldsymbol{u}^k\right)\right)\\ \boldsymbol{z}^{k+1}=\mathcal{S}_{\lambda/\rho}\left(\boldsymbol{x}^{k+1}+\boldsymbol{u}^k\right)\\ \boldsymbol{u}^{k+1}=\boldsymbol{u}^k+\left(\boldsymbol{x}^{k+1}-\boldsymbol{z}^{k+1}\right)\end{array}\right\} \qquad k=0,1,2,\ldots$$

Figure B.14 shows the convergence of the ADMM iterates.

B.10 Numerical Comparison

We will now compare the different algorithms with the ℓ_2–ℓ_1-norm minimization example (Zibulevsky & Elad, 2010):

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2+\lambda\|\boldsymbol{x}\|_1.$$

To facilitate comparison, we express the iterates of the various algorithms in a uniform manner, utilizing the soft-thresholding operator $\mathcal{S}$ as defined in (B.11).

- BCD (a.k.a. Gauss–Seidel) iterates:

$$\boldsymbol{x}^{k+1}=\mathcal{S}_{\frac{\lambda}{\operatorname{diag}(\boldsymbol{A}^\mathsf{T}\boldsymbol{A})}}\left(\boldsymbol{x}^k-\frac{\boldsymbol{A}^\mathsf{T}\left(\boldsymbol{A}\boldsymbol{x}^{(k,i)}-\boldsymbol{b}\right)}{\operatorname{diag}\left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\right)}\right), \qquad i=1,\ldots,n, \quad k=0,1,2,\ldots,$$

  where $\boldsymbol{x}^{(k,i)}\triangleq\left(x_1^{k+1},\ldots,x_{i-1}^{k+1},x_i^k,\ldots,x_n^k\right)$.

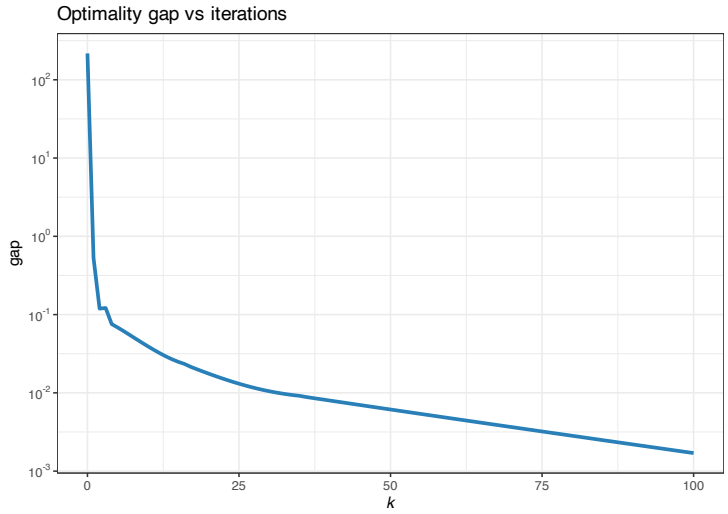


Figure B.14 Convergence of ADMM for the ℓ_2–ℓ_1-norm minimization.

- Parallel version of BCD (a.k.a. Jacobi) iterates:

$$\boldsymbol{x}^{k+1} = \mathcal{S}_{\frac{\lambda}{\text{diag}(\boldsymbol{A}^\mathsf{T}\boldsymbol{A})}}\left(\boldsymbol{x}^k - \frac{\boldsymbol{A}^\mathsf{T}\left(\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b}\right)}{\text{diag}\left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\right)}\right), \qquad i = 1, \ldots, n, \quad k = 0, 1, 2, \ldots$$

- MM iterates:

$$\boldsymbol{x}^{k+1} = \mathcal{S}_{\frac{\lambda}{\kappa}}\left(\boldsymbol{x}^k - \frac{1}{\kappa}\boldsymbol{A}^\mathsf{T}\left(\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b}\right)\right), \qquad k = 0, 1, 2, \ldots$$

- Accelerated MM iterates:

$$\left.\begin{aligned} \boldsymbol{r}^k &= R(\boldsymbol{x}^k) \triangleq \text{MM}(\boldsymbol{x}^k) - \boldsymbol{x}^k \\ \boldsymbol{v}^k &= R(\text{MM}(\boldsymbol{x}^k)) - R(\boldsymbol{x}^k) \\ \alpha^k &= -\max\left(1, \|\boldsymbol{r}^k\|_2 / \|\boldsymbol{v}^k\|_2\right) \\ \boldsymbol{y}^k &= \boldsymbol{x}^k - \alpha^k \boldsymbol{r}^k \\ \boldsymbol{x}^{k+1} &= \text{MM}(\boldsymbol{y}^k) \end{aligned}\right\} \qquad k = 0, 1, 2, \ldots$$

- SCA iterates:

$$\left.\begin{aligned} \hat{\boldsymbol{x}}^{k+1} &= \mathcal{S}_{\frac{\lambda}{\tau + \text{diag}(\boldsymbol{A}^\mathsf{T}\boldsymbol{A})}}\left(\boldsymbol{x}^k - \frac{\boldsymbol{A}^\mathsf{T}\left(\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b}\right)}{\tau + \text{diag}\left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\right)}\right) \\ \boldsymbol{x}^{k+1} &= \gamma^k \hat{\boldsymbol{x}}^{k+1} + \left(1 - \gamma^k\right)\boldsymbol{x}^k \end{aligned}\right\} \qquad k = 0, 1, 2, \ldots$$

- ADMM iterates:

$$\left.\begin{aligned} \boldsymbol{x}^{k+1} &= \left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A} + \rho\boldsymbol{I}\right)^{-1}\left(\boldsymbol{A}^\mathsf{T}\boldsymbol{b} + \rho\left(\boldsymbol{z}^k - \boldsymbol{u}^k\right)\right) \\ \boldsymbol{z}^{k+1} &= \mathcal{S}_{\lambda/\rho}\left(\boldsymbol{x}^{k+1} + \boldsymbol{u}^k\right) \\ \boldsymbol{u}^{k+1} &= \boldsymbol{u}^k + \left(\boldsymbol{x}^{k+1} - \boldsymbol{z}^{k+1}\right) \end{aligned}\right\} \qquad k = 0, 1, 2, \ldots$$

Observe that BCD updates each element sequentially, whereas all the other methods update all the elements simultaneously (this translates into an unacceptable increase in the computational cost for BCD).

The Jacobi update is the parallel version of BCD, although in principle it is not guaranteed to converge. Interestingly, the Jacobi update looks strikingly similar to the SCA update, except that SCA includes τ and the smoothing step, which are precisely the necessary ingredients to guarantee convergence.

A hidden but critical detail in the MM method is the need to compute the largest eigenvalue of $\boldsymbol{A}^\mathsf{T}\boldsymbol{A}$ to be able to choose the parameter $\kappa > \lambda_{\max}\left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\right)$, which results in an upfront increase in computation. In addition, such a value of κ is a very conservative upper bound used in the update of all the elements of $\boldsymbol{x}$. In SCA, this common conservative value of κ is replaced by the vector $\text{diag}\left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\right)$, which is better tailored to each element of $\boldsymbol{x}$ and results in faster convergence.

Figure B.15 compares the convergence of all these methods in terms of iterations, as well as CPU time, for the resolution of the ℓ_2–ℓ_1-norm minimization problem with $m = 500$ linear equations and $n = 100$ variables. Note that each outer iteration of BCD involves a sequential update of each element, which results in an extremely high CPU time and the curve lies outside the range of the plot. Observe that ADMM converges with a much lower accuracy than the other methods.

B.11 Summary

- A broad spectrum of algorithms is available to solve optimization problems. The specific selection depends on the problem's requirements and the user's technical expertise.
- *Solvers* are accessible for most types of convex and nonconvex formulations across most programming languages. Most users will simply invoke a solver or, for greater ease, utilize a modeling framework to define the problem more conveniently. The framework will then call an appropriate solver.
- Inside a solver, various methods can be employed, such as the gradient descent method, Newton's method, and interior-point methods, among others. However, most users do not need to delve into these details.
- More advanced users may seek to develop improved algorithms tailored to their specific problem. These can be faster, more efficient, simpler to implement, or capable of handling larger problems. This advanced approach demands more effort and knowledge from the user, such as knowledge of the Dinkelbach method or the Charnes–Cooper–Schaible transform for fractional problems.
- Complex problems can be addressed using iterative algorithmic frameworks. These frameworks aim to break down the original, intricate problem into more manageable sub-problems, although this approach necessitates iterations. Some of the most popular and widely used methods include:
  - bisection

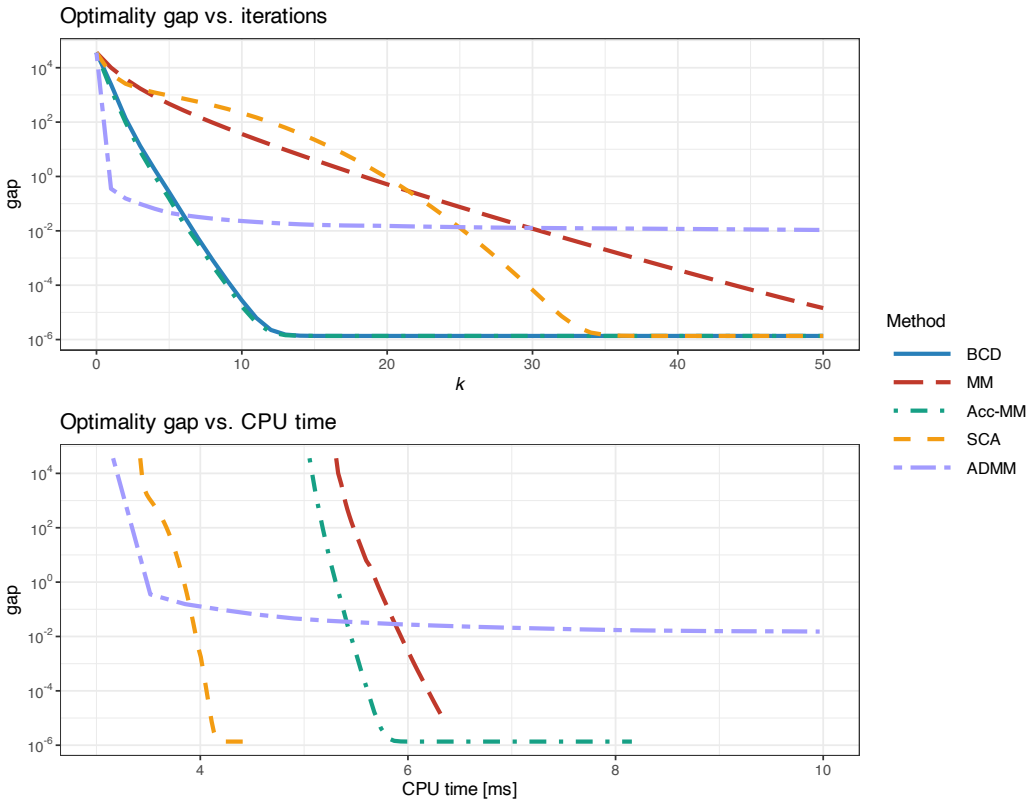


Figure B.15 Comparison of different iterative methods for the ℓ_2–ℓ_1-norm minimization.

- block coordinate descent (BCD)
- majorization–minimization (MM)
- successive convex approximation (SCA)
- alternating direction method of multipliers (ADMM).

Exercises

B.1 (Euclidean norm approximation)

a. Randomly generate the parameters $\boldsymbol{A} \in \mathbb{R}^{10\times 5}$ and $\boldsymbol{b} \in \mathbb{R}^{10}$.
b. Formulate a regression problem to approximate $\boldsymbol{Ax} \approx \boldsymbol{b}$ based on the ℓ_2-norm.
c. Solve it directly with the least squares closed-form solution.
d. Solve it using a modeling framework (e.g., CVX).
e. Solve it invoking a QP solver.

B.2 (Manhattan norm approximation)

a. Randomly generate the parameters $\boldsymbol{A} \in \mathbb{R}^{10\times 5}$ and $\boldsymbol{b} \in \mathbb{R}^{10}$.
b. Formulate a regression problem to approximate $\boldsymbol{Ax} \approx \boldsymbol{b}$ based on the ℓ_1-norm.

c. Solve it using a modeling framework (e.g., CVX).
d. Rewrite it as an LP and solve it invoking an LP solver.

B.3 (Chebyshev norm approximation)

a. Randomly generate the parameters $\boldsymbol{A} \in \mathbb{R}^{10\times 5}$ and $\boldsymbol{b} \in \mathbb{R}^{10}$.
b. Formulate a regression problem to approximate $\boldsymbol{Ax} \approx \boldsymbol{b}$ based on the ℓ_∞-norm.
c. Solve it using a modeling framework (e.g., CVX).
d. Rewrite it as an LP and solve it invoking an LP solver.

B.4 (Solving an LP) Consider the following LP:

$$\begin{array}{ll} \underset{x_1,x_2}{\text{maximize}} & 3x_1 + x_2 \\ \text{subject to} & x_1 + 2x_2 \leq 4, \\ & 4x_1 + 2x_2 \leq 12, \\ & x_1, x_2 \geq 0. \end{array}$$

a. Solve it using a modeling framework (e.g., CVX).
b. Solve it by directly invoking an LP solver.
c. Solve it by invoking a general-purpose nonlinear solver.
d. Implement the projected gradient method to solve the problem.
e. Implement the constrained Newton's method to solve the problem.
f. Implement the log-barrier interior-point method to solve the problem (use (1,1) as the initial point).
g. Compare all the solutions and the computation time.

B.5 (Central path) Formulate the log-barrier problem corresponding to the LP in Exercise B.4 and plot the central path as the parameter t varies.

B.6 (Phase I method) Design a phase I method to find a feasible point for the LP in Exercise B.4, which can then be used as the starting point for the barrier method.

B.7 (Dual problem) Formulate the dual problem corresponding to the LP in Exercise B.4 and solve it using a solver of your choice.

B.8 (KKT conditions) Write down the Karush–Kuhn–Tucker (KKT) conditions for the LP in Exercise B.4 and discuss their role in determining the optimality of a solution.

B.9 (Solving a QP) Consider the following QP:

$$\begin{array}{ll} \underset{x_1,x_2}{\text{minimize}} & x_1^2 + x_2^2 \\ \text{subject to} & x_1 + x_2 = 1, \\ & x_1 \geq 0, x_2 \geq 0. \end{array}$$

a. Solve it using a modeling framework (e.g., CVX).
b. Solve it by directly invoking a QP solver.
c. Solve it by invoking a general-purpose nonlinear solver.
d. Implement the projected gradient method to solve the problem.
e. Implement the constrained Newton's method to solve the problem.
f. Implement the log-barrier interior-point method to solve the problem (use (0.5,0.5) as the initial point).

g. Compare all the solutions and the computation time.

B.10 (Fractional programming) Consider the following fractional program:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{maximize}} & \dfrac{\boldsymbol{w}^\top \mathbf{1}}{\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}} \\ \text{subject to} & \mathbf{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \end{array}$$

where $\boldsymbol{\Sigma} \succ \mathbf{0}$.

a. Solve it with a general-purpose nonlinear solver.
b. Solve it via bisection.
c. Solve it via the Dinkelbach method as a sequence of SOCPs.
d. Develop a modified algorithm that solves the problem as a sequence of QPs instead.
e. Solve it via the Schaible transform method.
f. Reformulate the problem as a minimization and then solve it via the Schaible transform method.
g. Compare all the previous approaches in terms of the accuracy of the solution and the computation time.

B.11 (Soft-thresholding operator) Consider the following convex optimization problem:

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|\boldsymbol{a}x - \boldsymbol{b}\|_2^2 + \lambda |x|,$$

with $\lambda \geq 0$. Derive the solution and show that it can be written as

$$x = \frac{1}{\|\boldsymbol{a}\|_2^2} \mathcal{S}_\lambda \left(\boldsymbol{a}^\top \boldsymbol{b} \right),$$

where $\mathcal{S}_\lambda(\cdot)$ is the so-called soft-thresholding operator defined as

$$\mathcal{S}_\lambda(u) = \operatorname{sign}(u)(|u| - \lambda)^+,$$

with $\operatorname{sign}(\cdot)$ denoting the sign function and $(\cdot)^+ = \max(0, \cdot)$.

B.12 (ℓ_2–ℓ_1-norm minimization) Consider the following ℓ_2–ℓ_1-norm minimization problem (with $\boldsymbol{A} \in \mathbb{R}^{10 \times 5}$ and $\boldsymbol{b} \in \mathbb{R}^{10}$ randomly generated):

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda \|\boldsymbol{x}\|_1.$$

a. Solve it using a modeling framework (e.g., CVX).
b. Rewrite the problem as a QP and solve it by invoking a QP solver.
c. Solve it with an ad hoc LASSO solver.

B.13 (BCD for ℓ_2–ℓ_1-norm minimization) Solve the ℓ_2–ℓ_1-norm minimization problem in Exercise B.12 via BCD. Plot the convergence vs. iterations and CPU time.

B.14 (MM for ℓ_2–ℓ_1-norm minimization) Solve the ℓ_2–ℓ_1-norm minimization problem in Exercise B.12 via MM and its accelerated version. Plot the convergence vs. iterations and CPU time.

B.15 (SCA for ℓ_2–ℓ_1-norm minimization) Solve the ℓ_2–ℓ_1-norm minimization problem in Exercise B.12 via SCA. Plot the convergence vs. iterations and CPU time.

B.16 (ADMM for ℓ_2–ℓ_1-norm minimization) Solve the ℓ_2–ℓ_1-norm minimization problem in Exercise B.12 via ADMM. Plot the convergence vs. iterations and CPU time.

References

Bajalinov, E. B. (2003). *Linear-Fractional Programming: Theory, Methods, Applications and Software*. Springer.

Beck, A. (2017). *First-Order Methods in Optimization*. Society for Industrial and Applied Mathematics (SIAM).

Ben-Tal, A., & Nemirovski, A. (2001). *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. Society for Industrial and Applied Mathematics (SIAM).

Bertsekas, D. P. (1999). *Nonlinear Programming*. Athena Scientific.

Bertsekas, D. P., & Tsitsiklis, J. N. (1997). *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning, Now Publishers*, *3*(1), 1–122.

Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Charnes, A., & Cooper, W. W. (1962). Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, *9*(3–4), 181–186.

Dinkelbach, W. (1967). On nonlinear fractional programming. *Management Science*, *133*(7), 492–498.

Fu, A., Narasimhan, B., & Boyd, S. (2020). CVXR: An R package for disciplined convex optimization. *Journal of Statistical Software*, *94*(14), 1–34.

Grant, M., & Boyd, S. (2008). Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, & H. Kimura (Eds.), *Recent Advances in Learning and Control* (pp. 95–110). Springer.

Grant, M., & Boyd, S. (2014). *CVX: Matlab Software for Disciplined Convex Programming*. http://cvxr.com/cvx

Grippo, L., & Sciandrone, M. (2000). On the convergence of the block nonlinear Gauss–Seidel method under convex constraints. *Operations Research Letters*, *26*(3), 127–136.

Hunter, D. R., & Lange, K. (2004). A tutorial on MM algorithms. *The American Statistician*, *58*, 30–37.

Karmarkar, N. (1984). A new polynomial time algorithm for linear programming. *Combinatorica*, *4*(4), 373–395.

Kumar, S., Ying, J., Cardoso, J. V. M., & Palomar, D. P. (2019). Structured graph learning via Laplacian spectral constraints. *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)*, 11651–11663.

Kumar, S., Ying, J., Cardoso, J. V. M., & Palomar, D. P. (2020). A unified framework for structured graph learning via spectral constraints. *Journal of Machine Learning Research (JMLR)*, *21*(22), 1–60.

Löfberg, J. (2004). YALMIP: A toolbox for modeling and optimization in MATLAB. *Proceedings of the CACSD Conference*.

Luenberger, D. G., & Ye, Y. (2021). *Linear and Nonlinear Programming* (5th ed.). Springer.

Nemirovski, A. (2000). *Lectures on Modern Convex Optimization* (tech. rep.). Technion - Israel Institute of Technology.

Nesterov, Y. (2018). *Lectures on Convex Optimization* (2nd ed.). Springer.

Nesterov, Y., & Nemirovski, A. (1994). *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics (SIAM).

Nocedal, J., & Wright, S. J. (2006). *Numerical Optimization*. Springer.

Razaviyayn, M., Hong, M., & Luo, Z. (2013). A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, *23*(2), 1126–1153.

Schaible, S. (1974). Parameter-free convex equivalent and dual programs of fractional programming problems. *Zeitschrift fur Operations Research*, *18*(5), 187–196.

Scutari, G., Facchinei, F., Song, P., Palomar, D. P., & Pang, J.-S. (2014). Decomposition by partial linearization: Parallel optimization of multi-agent systems. *IEEE Transactions on Signal Processing*, *62*(3), 641–656.

Scutari, G., & Sun, Y. (2018). Parallel and distributed successive convex approximation methods for big-data optimization. In F. Facchinei & J. S. Pang (Eds.), *Multi-agent Optimization* (pp. 141–308). Springer.

Song, J., Babu, P., & Palomar, D. P. (2015). Sparse generalized eigenvalue problem via smooth optimization. *IEEE Transactions on Signal Processing*, *63*(7), 1627–1642.

Stancu-Minasian, I. M. (1997). *Fractional Programming: Theory, Methods and Applications*. Kluwer Academic Publishers.

Sturm, J. F. (1999). Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, *11*(1-4), 625–653.

Sun, Y., Babu, P., & Palomar, D. P. (2017). Majorization–minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, *65*(3), 794–816.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, *58*(1), 267–288.

Tütüncü, R. H., Toh, K. C., & Todd, M. J. (2003). Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming. Series B*, *95*, 189–217.

Varadhan, R., & Roland, C. (2008). Simple and globally convergent methods for accelerating the convergence of any EM algorithm. *Scandinavian Journal of Statistics*, *35*(2), 335–353.

Ye, Y. (1997). *Interior Point Algorithms: Theory and Analysis*. Wiley & Sons.

Zibulevsky, M., & Elad, M. (2010). L1–L2 optimization in signal and image processing. *IEEE Signal Processing Magazine*, *27*(3), 76–88.

Index

algorithms
  alternate minimization, 560
  alternating direction method of multipliers (ADMM), 577
  bisection method, 513
  block coordinate descent (BCD), 302, 560
  FP via bisection, 557
  FP via Dinkelbach, 558
  Gauss–Seidel method, 560
  gradient descent method, 546
  heavy-tailed ML fixed-point via MM, 52
  index tracking via MM, 362
  interior-point method (barrier method), 554
  majorization–minimization (MM), 255, 303, 353, 563, 565
  MSRP via bisection, 184
  MSRP via Dinkelbach, 185
  MVSK portfolio via MM, 256
  MVSK portfolio via SCA, 254
  Newton's method, 301, 547
  portfolio bagging, 395
  portfolio resampling, 395
  projected gradient descent method, 549
  RPP via alternate linearization, 314
  RPP via BCD, 302
  RPP via MM, 303
  RPP via Newton's method, 301
  RPP via SCA, 303, 313, 314
  sparse regression via MM, 354
  successive convex approximation (SCA), 253, 303, 313, 571
  universal mean–variance portfolios, 195
alternating direction method of multipliers (ADMM), *see also* algorithms
AR model, 88
ARCH model, 97
ARIMA model, 89
ARMA model, 88
asset structure, 33
autocorrelation function (ACF), 30
autoregressive conditional heteroskedasticity (ARCH), *see* ARCH

backtest, 205
  best practices, 218
  cross-validation backtest, 223
  dangers
    p-hacking, 216
    data snooping bias, 210
    ignoring shorting cost, 214
    ignoring transaction costs, 212
    look-ahead bias, 208
    outliers, 212
    overfitting, 210, 215
    storytelling bias, 209
    survivorship bias, 207
  multiple randomized backtests, 224
  stress tests, 229
  synthetic data, 226
  vanilla backtest, 220
  walk-forward backtest, 222
Black–Litterman model, 67

capital asset pricing model (CAPM), *see* CAPM
CAPM, 62
CCC model, 103
cointegration, 407, 415
  tests, 416
conditional models, 80
constant conditional correlation (CCC), *see* CCC
correlation, 409
covariance matrix estimators
  M-estimators, 53
  Black–Litterman estimator, 69
  factor model estimator, 64
  Gaussian estimator, 47
  heavy-tailed estimator, 51
  PCA estimator, 66
  sample covariance matrix, 40
  shrinkage estimator, 60
  Tyler's estimator, 54
cumulative P&L, 147
cumulative return, 147

DCC model, 103, 104
deep learning (DL), 458
  autoencoder, 468
  backpropagation, 463
  challenges in portfolio design, 471
  convolutional neural network (CNN), 464
  diffusion models, 469
  finance, 470, 473, 474, 476, 477
  generative adversarial network (GAN), 468

history, 459
long short-term memory (LSTM), 467
multilayer perceptron, 461, 464
neural network, 461
portfolio design, 471, 474, 476, 477
recursive neural networks (RNN), 466
transformer, 467
dendrogram, 324
dynamic conditional correlation (DCC), *see* DCC
efficient-market hypothesis (EMH), 29, 39, 40, 79, 348
EMA, *see* EWMA
EWMA, 87
multivariate volatility model, 103
volatility envelope, 97
expected utility theory, 189
exponential smoothing methods, 91
exponentially weighted moving average (EWMA), *see* EWMA
factor models, 62
false discovery rate (FDR), 369
index tracking, 370
financial index, 348
GARCH model, 98, 99
multivariate, 103, 104
generalized autoregressive conditional heteroskedasticity (GARCH), *see* GARCH
graphical model networks, *see* graphs, graphical model networks
graphs, 111
k-component GMRF graph, 125, 128
k-component graph, 124, 132, 323
k-component graph approximation, 125, 128
adjacency matrix, 113
bipartite GMRF graph, 126
bipartite graph, 125
bipartite graph approximation, 126
clustered graph, 124, 132
connectivity matrix, 113
degree matrix, 114
dynamic graph, 133
financial graphs, 134, 323
Gaussian Markov random field (GMRF), 132
GLASSO, 132
GMRF, 132
graph spectral properties, 123
graphical model networks, 119
correlation network, 120
Gaussian Markov random field (GMRF), 120
GMRF, 122
graphical LASSO (GLASSO), 120
Laplacian-structured GLASSO, 120
partial correlation network, 120
sparse GMRF, 121, 122
heavy-tailed graph, 130–132, 323
heavy-tailed Markov random field (MRF) graph, 130–132
heavy-tailed Markow random field (MRF) graph, 323
Laplacian matrix, 114
low-rank GMRF Laplacian, 125, 128
low-rank Laplacian approximation, 125, 128
low-rank Laplacian matrix, 124, 132
node degree, 119, 128
sparse GMRF graph, 125, 126, 128, 132
sparse graph, 121, 122, 132
structured graphs, 122
time-varying graph, 133
heavy tails, 24
hierarchical clustering, 324
hierarchical clustering portfolios, 327, *see also* portfolios
high-order moments, 238
L-moments, 243
portfolio moments, 238
portfolio parametric moments, 242
structured moments, 240
i.i.d. model, 39, 80
index tracking, 349, 355, 359, 361
industries, 127
Kalman, 81, 434, 435
Kalman filtering, 83, 435
Kalman forecasting, 84, 435
Kalman smoothing, 84
mean models, 91, 95
state-space model, 81, 82
observation equation, 82
state equation, 82
volatility models, 100
kurtosis, 24
least squares (LS), 43
MA, 85
volatility envelope, 96
machine learning (ML), 454
finance, 458
learning, 456
models, 457
majorization–minimization (MM), 44, 52, 121, 130, 303, *see also* algorithms
maximum likelihood (ML), 46
Gaussian estimators, 47
heavy-tailed estimators, 50
mean estimators
M-estimators, 53
Black–Litterman estimator, 69
factor model estimator, 64
Gaussian estimator, 47
heavy-tailed estimator, 51
James–Stein's shrinkage estimator, 59
median, 43
sample mean, 40, 43
spatial median, 44
mean reversion, 405
moving average (MA), *see* MA

net asset value (NAV), 145
Nobel prize
  Clive W. J. Granger, 416
  Eugene F. Fama, 40, 79
  Harry Markowitz, 175
  Robert F. Engle, 97, 416
  Robert J. Shiller, 40, 79
non-Gaussianity, 24
optimization, 492
  Charnes–Cooper transform, 559
  complementary slackness, 528
  conic program (CP), 519
  convex functions, 500
  convex optimization problems, 507
  convex sets, 497
  fractional program (FP), 520
  generalized inequalities, 530
  geometric program (GP), 521
  interior-point methods, 540
  Karush–Kuhn–Tucker (KKT) optimality conditions, 529
  Lagrange dual problem, 523
  Lagrange duality, 522, 526
  Lagrangian, 522
  linear program (LP), 514
  linear-fractional program (LFP), 515
  modeling framework, 543
  multi-objective optimization problems, 530, 532
  optimization problems, 492
  Pareto optimality, 531
  quadratic program (QP), 516
  quasi-convex functions, 506
  quasi-convex optimization problems, 513
  Schaible transform, 185, 559
  second-order cone program (SOCP), 517
  semidefinite program (SDP), 518
  solvers, 539
  types of convex problems, 514
  vector optimization problems, 531
pairs trading, 411, 440
PCA, 66
portfolio constraints, 152
  capital budget, 152
  cardinality, 153
  diversification, 154
  dollar neutral, 154
  holding, 153
  leverage, 154
  long-only, 152
  margin, 155
  market neutral, 154
  no-shorting, 152
  turnover, 153
portfolio performance measures, 155
  Calmar ratio, 163
  conditional value-at-risk (CVaR), 161, 270–272
  downside deviation, 159
  downside risk, 159, 269, 272
  drawdown, 161, 272
  entropic value-at-risk (EVaR), 270
  expected return, 155
  expected shortfall (ES), 161, 270
  expected tail loss (ETL), 270
  gain–loss ratio (GLR), 160
  information ratio (IR), 159
  lower partial moment (LPM), 159, 269, 272
  semi-deviation, 159, 269
  semi-variance, 159, 269
  Sharpe ratio (SR), 158
  Sortino ratio, 160
  Sterling ratio, 163
  value-at-risk (VaR), 160, 270
  volatility, 156
  volatility-adjusted returns, 157
portfolio rebalancing, *see* portfolios
portfolio risk measures, *see* portfolio performance measures
portfolios
  $1/N$ portfolio, 164
  buy and hold (B&H) portfolio, 163
  constraints, *see* portfolio constraints
  deep learning (DL) portfolios, 477
  equal risk portfolio (ERP), 168
  equally weighted portfolio (EWP), 164
  global maximum return portfolio (GMRP), 163
  global minimum variance portfolio (GMVP), 167
  hierarchical clustering based, 327
    cluster-based waterfall portfolio, 328
    hierarchical $1/N$ portfolio, 328
    hierarchical equal risk contribution (HERC) portfolio, 335
    hierarchical risk parity (HRP) portfolio, 332
  index tracking, 359, 361
  inverse volatility portfolio (IVolP), 168
  Kelly criterion portfolio, 187
  maximum decorrelation portfolio (MDecP), 170
  maximum expected utility portfolio, 189
  maximum Sharpe ratio portfolio (MSRP), 184
  mean–Ave-DD portfolio, 283
  mean–CVaR portfolio, 277
  mean–CVaR-DD portfolio, 283
  mean–downside risk portfolio, 274
  mean–EVaR portfolio, 278
  mean–LPM portfolio, 274
  mean–Max-DD portfolio, 282
  mean–semi-variance portfolio, 274, 276
  mean–variance portfolio (MVP), 176
  mean–variance–skewness–kurtosis (MVSK) portfolio, 247
  mean–worst-case risk portfolio, 279
  most diversified portfolio (MDivP), 169
  MVSK portfolio tilting, 249
  performance measures, *see* portfolio performance measures
  polynomial goal programming MVSK portfolio, 249

portfolio bagging, 395
portfolio rebalancing, 150
portfolio resampling, 395
quintile portfolio, 165, 166
risk measures, *see* portfolio performance measures
risk parity portfolio (RPP), 168, 295, 297, 299, 310
robust portfolios, 166, 384, 385, 388
transaction costs, 148
principal component analysis (PCA), *see* PCA
Python packages
CVXPY, 13
filterpy, 82, 435
PyMC, 100
PyPortfolioOpt, 13, 179
Riskfolio-Lib, 13, 179
riskparity.py, 295
scipy, 325
skfolio, 13, 179
statsmodels, 89
R packages
CVXR, 13
egcm, 416
fGarch, 98
finbipartite, 126
fingraph, 130, 323
fitHeavyTail, 51
fPortfolio, 13, 179
highOrderPortfolios, 256
KFAS, 82, 435
MARSS, 82, 435
riskParityPortfolio, 295
rugarch, 89, 98
sparseGraph, 121
sparseIndexTracking, 362
spectralGraphTopology, 121, 125
stochvol, 100
TRexSelector, 370
urca, 416
random walk, 40
resampling methods, 391
bagging, 395
bootstrap, 393, 394
jackknife, 393
portfolio bagging, 395
portfolio resampling, 395
return
cumulative return, 147
returns, 20
risk contributions, 293
risk diversification, 292
risk measures
conditional value-at-risk (CVaR), 270–272
downside risk, 269, 272
drawdown, 272
entropic value-at-risk (EVaR), 270
expected shortfall (ES), 270
expected tail loss (ETL), 270
lower partial moment (LPM), 269, 272
semi-deviation, 269
semi-variance, 269
value-at-risk (VaR), 270
risk parity, 292
risk parity portfolio (RPP), 295
general nonconvex formulations, 310
naive diagonal formulation, 297
vanilla convex formulations, 299
robust estimators, 52, 366
robust optimization, 380
stochastic optimization, 382
worst-case robust optimization, 383
robust portfolios, 385, 388
sample covariance matrix, *see* covariance matrix estimator, sample covariance matrix
sample estimators, 40
sample mean, *see* mean estimator, sample mean
seasonality decomposition, 91
sectors, 127
shrinkage estimators, 58
James–Stein's shrinkage mean estimator, 59
shrinkage covariance matrix estimator, 60
skewness, 24
sparse regression, 351
index tracking, 359
spread, 413, 423, 432
state-space model, *see* Kalman, state-space model
statistical arbitrage, 411, 440
stochastic volatility (SV), 99
AR(1) state-space model, 101, 104
multivariate, 104
random walk state-space model, 101, 104
stress tests, 229
structural time series models, 91
stylized facts, 19, 36
successive convex approximation (SCA), 303, 313, *see also* algorithms
SV, *see* stochastic volatility (SV)
temporal correlation, 29
transaction costs, 148
Tyler's estimator, 54
variance modeling, 95
VARMA model, 94
VECM model, 94, 442
volatility clustering, 29, 30
volatility envelope, 96
volatility modeling, 95
volatility smile, 91, 98