Solutions to Exercises

Portfolio Optimization: Theory and Application
Chapter 13 – Index Tracking Portfolios

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

portfoliooptimizationbook.com

Exercise 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.

Solution

We will plot the Hang Seng Index (ticker symbol ^HSI) and the Tracker Fund of Hong Kong (ticker symbol 2800.HK):

```
library(quantmod)

# Get data from Yahoo!Finance
HSI_prices  <- Ad(getSymbols("^HSI", from = "2023-01-01", auto.assign = FALSE))
ETF_prices  <- Ad(getSymbols("2800.HK", from = "2023-01-01", auto.assign = FALSE))
prices <- na.omit(cbind(HSI_prices, ETF_prices))
colnames(prices) <- c("Hang Seng Index (HSI)", "Tracker Fund of Hong Kong")
normalized_prices <- sweep(prices, 2, as.numeric(prices[1,]), "/")
```

```r
library(ggplot2)
library(patchwork)

p_lin <- ggplot(fortify(normalized_prices, melt = TRUE),
                aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 0.8, show.legend = TRUE) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Tracking of Hang Seng Index (on a linear scale)",
       x= NULL, y = "normalized dollars", color = NULL)

p_log <- ggplot(fortify(normalized_prices, melt = TRUE),
                aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 0.8, show.legend = TRUE) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  scale_y_log10() +
  labs(title = "Tracking of Hang Seng Index (on a log-scale)",
       x= NULL, y = "dollars", color = NULL)

p_lin / p_log
```

We can observe that the tracker properly tracks the short movements and in the long term it outperforms the index.

Exercise 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?

Solution

We will compare with the fund Barings Hong Kong China A HKD Inc (with ticker 0P0000PRAY.HK):

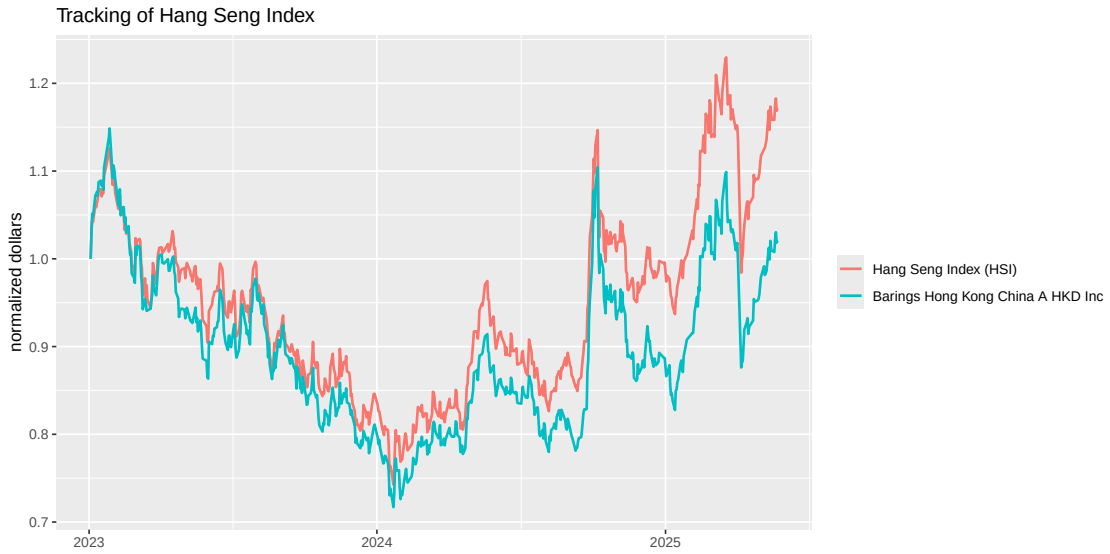
```
library(quantmod)

# Get data from Yahoo!Finance
fund_prices  <- Ad(getSymbols("0P0000PRAY.HK", from = "2023-01-01", auto.assign = FALSE))
prices <- na.omit(cbind(HSI_prices, fund_prices))
colnames(prices) <- c("Hang Seng Index (HSI)", "Barings Hong Kong China A HKD Inc")
normalized_prices <- sweep(prices, 2, as.numeric(prices[1,]), "/")
```

From the plot of the normalized prices, we can already observed that the fund underperforms the index:

```
library(ggplot2)

ggplot(fortify(normalized_prices, melt = TRUE),
       aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 0.8, show.legend = TRUE) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Tracking of Hang Seng Index",
       x= NULL, y = "normalized dollars", color = NULL)
```



We can also calculate the Sharpe ratios:

```r
library(PerformanceAnalytics)

linreturns <- na.omit(prices/lag(prices) - 1)
sharpe_ratio <- SharpeRatio.annualized(linreturns, scale = 252)
print(t(sharpe_ratio), digits = 3)
```

```
                                  Annualized Sharpe Ratio (Rf=0%)
Hang Seng Index (HSI)                                      0.2843
Barings Hong Kong China A HKD Inc                          0.0346
```

This seems to indicate that a fund managed by experts is underperforming the market, hence the market is efficient. Note that the HSI now includes stocks with significant mainland China exposure.

Exercise 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 $\boldsymbol{x}$):

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \|\boldsymbol{x}\|_0 \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{aligned}$$

Finally, solve the following linear program and compare the solution with the previous one:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \|\boldsymbol{x}\|_1 \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{aligned}$$

Solution

First, we generate the data:

```r
# Set seed for reproducibility
set.seed(42)

# Generate an underdetermined system A (5x10) and b (5x1)
# Create a random matrix A
A <- matrix(rnorm(5*10), nrow=5, ncol=10)
n <- ncol(A)

# Create a sparse true solution with only a few non-zero elements
x_true <- rep(0, n)
x_true[c(2, 4, 8)] <- c(1.5, -2.0, 3.0)  # Non-zero elements at positions 2, 4, and 8
pattern_true <- as.logical(intToBits(2 + 8 + 128))[1:n]

# Generate b
b <- A %*% x_true
```

Now, we use a brute force approach to minimize ℓ_0-pseudonorm:

```r
# Function to check if x_candidate is a solution to A x = b
is_solution <- function(x_candidate, A, b, tol = 1e-6) {
  return(all(abs(A %*% x_candidate - b) < tol))
}

# Brute force search to minimize l0-pseudonorm
best_x_l0 <- NULL
best_l0 <- Inf

# Generate all 2^n binary patterns and check (excluding the all-zero pattern)
total_patterns <- 2^n - 1

# Progress tracking
cat("Brute force search over", total_patterns, "patterns...\n")
```

```
Brute force search over 1023 patterns...
```

```
for (i in 1:total_patterns) {
  # Convert integer i to binary pattern
  pattern <- as.logical(intToBits(i))[1:n]

  # Select columns of A corresponding to the pattern
  A_selected <- A[, pattern, drop=FALSE]

  # Solve the reduced system
  x_selected <- tryCatch({
    solve(t(A_selected) %*% A_selected, t(A_selected) %*% b)
  }, error = function(e) NULL)

  if (!is.null(x_selected)) {
    # Create full solution vector
    x_candidate <- rep(0, n)
    x_candidate[pattern] <- x_selected

    # Check if this is a valid solution
    if (is_solution(x_candidate, A, b)) {
      pseudonorm0 <- sum(x_candidate != 0)
      if (pseudonorm0 < best_l0) {
        best_l0 <- pseudonorm0
        best_x_l0 <- x_candidate
        cat("Found better solution with l0-norm:", pseudonorm0, "\n")
      }
    }
  }
}
```

```
Found better solution with l0-norm: 5
Found better solution with l0-norm: 3
```

Finally, we use the approach based on minimizing the ℓ_1-norm via linear programming:

```
library(CVXR)

x <- Variable(n)
problem <- Problem(Minimize(norm1(x)),
                   constraints = list(A %*% x == b))
result <- solve(problem, solver = "SCS")
x_l1 <- as.vector(result$getValue(x))

# Clean up very small values (numerical precision issues)
x_l1[abs(x_l1) < 1e-3*max(abs(x_l1))] <- 0
```

```r
# Compare results
cat("\nOriginal sparse solution:\n")
print(x_true)
cat("Cardinality:", sum(x_true != 0), "\n")
cat("L1 norm:", sum(abs(x_true)), "\n\n")

cat("Cardinality minimization result:\n")
print(best_x_l0)
cat("Cardinality:", sum(abs(best_x_l0) > 1e-3), "\n")
cat("L1 norm:", sum(abs(best_x_l0)), "\n\n")

cat("L1 minimization result (CVXR):\n")
print(x_l1, digits = 3)
cat("Cardinality:", sum(abs(x_l1) > 1e-3), "\n")
cat("L1 norm:", sum(abs(x_l1)), "\n\n")
```

```
Original sparse solution:
 [1]  0.0  1.5  0.0 -2.0  0.0  0.0  0.0  3.0  0.0  0.0
Cardinality: 3
L1 norm: 6.5

Cardinality minimization result:
 [1]  0.0  1.5  0.0 -2.0  0.0  0.0  0.0  3.0  0.0  0.0
Cardinality: 3
L1 norm: 6.5

L1 minimization result (CVXR):
 [1] -3.5409  0.0000 -0.7870 -1.0741  0.0235  0.0000  0.0000  0.0000 -0.0534
[10]  0.0000
Cardinality: 5
L1 norm: 5.478946
```

Exercise 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.

Solution

First, we generate the data:

```
# Set seed for reproducibility
set.seed(42)

# Generate system A (10x5) and b (10x1)
# Create a random matrix A
A <- matrix(rnorm(5*10), nrow=10, ncol=5)
n <- ncol(A)

# Create a sparse true solution with only a few non-zero elements
x_true <- rep(0, n)
x_true[c(2, 4)] <- c(1.5, -2.0)  # Non-zero elements at positions 2 and 4

# Generate b
b <- A %*% x_true
```

a. Brute force approach: Similar to Exercise 13.3.

b. ℓ_1-norm approximation:

```r
library(CVXR)

lambda <- 1

x <- Variable(n)
problem <- Problem(Minimize(sum_squares(A %*% x - b) + lambda * norm1(x)))
result <- solve(problem, solver = "SCS")
x_l1 <- as.vector(result$getValue(x))

# Clean up very small values (numerical precision issues)
x_l1[abs(x_l1) < 1e-3*max(abs(x_l1))] <- 0
```

c. Concave approximation using a general-purpose nonlinear solver:

```r
library(nloptr)

lambda <- 1
epsilon <- 0.01


# Define the objective function
objective_function <- function(x, A, b) {
  sum_sq <- sum((A %*% x - b)^2)
  penalty <- lambda * sum(log(1 + abs(x) / epsilon))
  return(sum_sq + penalty)
}

# Initial point
x0 <- rep(0, n)

# Optimization using nloptr
result <- nloptr(x0 = x0,
                 eval_f = function(x) objective_function(x, A, b),
                 opts = list(algorithm = "NLOPT_LN_NELDERMEAD",
                             maxeval = 1000,
                             xtol_rel = 1.0e-8)
                 )
x_solver <- result$solution

# Clean up very small values (numerical precision issues)
x_solver[abs(x_solver) < 1e-3*max(abs(x_solver))] <- 0
```

d. Concave approximation again, but using the iterative reweighted ℓ_1-norm method:

```r
library(CVXR)

# Initial point
x <- rep(0, n)

# Iterate
lambda <- 1
epsilon <- 0.01
max_iter <- 10
u <- 1
for (k in 1:max_iter) {
  d <- 1/(log(1 + u/epsilon)*(epsilon + abs(x)))

  x <- Variable(n)
  problem <- Problem(Minimize(sum_squares(A %*% x - b) + lambda * t(d) %*% abs(x)))
  result <- solve(problem)
  x <- as.vector(result$getValue(x))
}
x_MM <- x

# Clean up very small values (numerical precision issues)
x_MM[abs(x_MM) < 1e-3*max(abs(x_MM))] <- 0
```

Comparison of results:

```r
# Compare results
cat("\nTrue solution:\n")
print(x_true, digits = 3)

cat("\nl1-norm solution:\n")
print(x_l1, digits = 3)

cat("\nConcave-approx with general solver:\n")
print(x_solver, digits = 3)

cat("\nConcave-approx with iterative reweighted l1-norm:\n")
print(x_MM, digits = 3)
```

```
True solution:
[1]  0.0  1.5  0.0 -2.0  0.0

l1-norm solution:
[1]  0.00  1.46  0.00 -1.93  0.00

Concave-approx with general solver:
```

```
[1]  0.00566  1.62921  0.00000 -2.03044  0.00000

Concave-approx with iterative reweighted l1-norm:
[1]  0.00  1.49  0.00 -1.99  0.00
```

The superiority of the iterative reweighted ℓ_1-norm method is evident.

Exercise 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

$$\boldsymbol{b}_t = \frac{\boldsymbol{n} \odot \boldsymbol{p}_t}{\boldsymbol{n}^\top \boldsymbol{p}_t}.$$

Show that this normalized portfolio can also be expressed as

$$\boldsymbol{b}_t = \frac{\boldsymbol{b}_{t-1} \odot (\mathbf{1} + \boldsymbol{r}_t)}{\boldsymbol{b}_{t-1}^\top (\mathbf{1} + \boldsymbol{r}_t)},$$

where the returns are defined as

$$\boldsymbol{r}_t = \frac{\boldsymbol{p}_t - \boldsymbol{p}_{t-1}}{\boldsymbol{p}_{t-1}} = \frac{\boldsymbol{p}_t}{\boldsymbol{p}_{t-1}} - \mathbf{1}.$$

Solution

Start by writing:

$$\boldsymbol{b}_t = \frac{\boldsymbol{n} \odot \boldsymbol{p}_t}{\boldsymbol{n}^\top \boldsymbol{p}_t},$$

$$\boldsymbol{b}_{t-1} = \frac{\boldsymbol{n} \odot \boldsymbol{p}_{t-1}}{\boldsymbol{n}^\top \boldsymbol{p}_{t-1}},$$

and the price evolution as

$$\boldsymbol{p}_t = \boldsymbol{p}_{t-1} \odot (\mathbf{1} + \boldsymbol{r}_t).$$

Substituting this price relationship into the expression for $\boldsymbol{b}_t$:

$$\begin{aligned}\boldsymbol{b}_t &= \frac{\boldsymbol{n} \odot \boldsymbol{p}_t}{\boldsymbol{n}^\mathsf{T}\boldsymbol{p}_t} \\ &= \frac{\boldsymbol{n} \odot [\boldsymbol{p}_{t-1} \odot (\mathbf{1} + \boldsymbol{r}_t)]}{\boldsymbol{n}^\mathsf{T}[\boldsymbol{p}_{t-1} \odot (\mathbf{1} + \boldsymbol{r}_t)]} \\ &= \frac{(\boldsymbol{n} \odot \boldsymbol{p}_{t-1}) \odot (\mathbf{1} + \boldsymbol{r}_t)}{(\boldsymbol{n} \odot \boldsymbol{p}_{t-1})^\mathsf{T}(\mathbf{1} + \boldsymbol{r}_t)} \\ &= \frac{\boldsymbol{b}_{t-1} \odot (\mathbf{1} + \boldsymbol{r}_t)}{\boldsymbol{b}_{t-1}^\mathsf{T}(\mathbf{1} + \boldsymbol{r}_t)},\end{aligned}$$

where we have used

$$\boldsymbol{n} \odot \boldsymbol{p}_{t-1} = (\boldsymbol{n}^\mathsf{T}\boldsymbol{p}_{t-1})\boldsymbol{b}_{t-1} \propto \boldsymbol{b}_{t-1}.$$

Exercise 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).

Solution

First, download data for the Hang Seng Index (ticker symbol ^HSI) and the Tracker Fund of Hong Kong (ticker symbol 2800.HK):

```
library(quantmod)

# Get data from Yahoo!Finance
HSI_prices  <- Ad(getSymbols("^HSI", from = "2023-01-01", auto.assign = FALSE))
ETF_prices  <- Ad(getSymbols("2800.HK", from = "2023-01-01", auto.assign = FALSE))
prices <- na.omit(cbind(HSI_prices, ETF_prices))
colnames(prices) <- c("Hang Seng Index (HSI)", "Tracker Fund of Hong Kong")
X <- prices/lag(prices)[-1] - 1
HSI_returns <- X[, 1]
ETF_returns <- X[, 2]
```

Compute average error measures:

```r
# l2-norm tracking error
TE <- mean((HSI_returns - ETF_returns)^2)

# downside risk
DR <- mean(pmax(0, HSI_returns - ETF_returns)^2)

# l1-norm tracking error
TE_1 <- mean(abs(HSI_returns - ETF_returns))

# Huberized tracking error
hub <- function(x, delta = 0.0005) {
  ifelse(abs(x) <= delta,
         x^2,
         delta * (2*abs(x) - delta))
}
TE_hub <- mean(hub(HSI_returns - ETF_returns))


# Display
rbind("l2-norm tracking error"   = TE,
      "l1-norm tracking error"   = TE_1,
      "downside risk"            = DR,
      "Huberized tracking error" = TE_hub)
```

```
                                [,1]
l2-norm tracking error   1.917748e-06
l1-norm tracking error   1.011761e-03
downside risk            7.665826e-07
Huberized tracking error 7.875943e-07
```

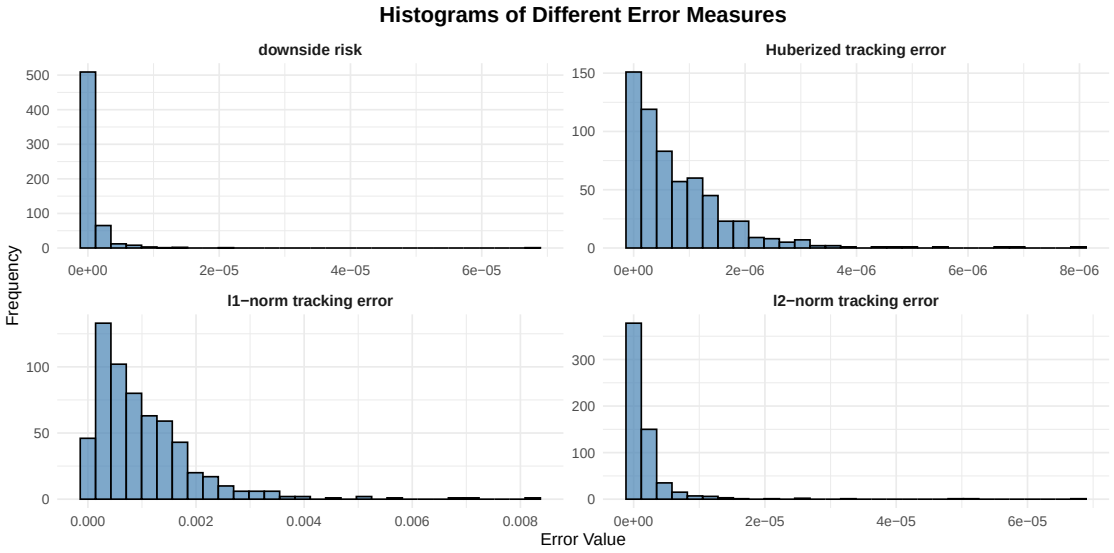
Compute and plot histogram of error measures:

```r
library(ggplot2)
library(dplyr)
library(tidyr)

# Calculate error vectors instead of averages
error_data <- data.frame(
  "l2_error"       = as.numeric((HSI_returns - ETF_returns)^2),
  "l1_error"       = as.numeric(abs(HSI_returns - ETF_returns)),
  "downside_error" = as.numeric(pmax(0, HSI_returns - ETF_returns)^2),
  "huber_error"    = as.numeric(hub(HSI_returns - ETF_returns))
)

# Reshape your error_data to long format
error_long <- error_data %>%
  pivot_longer(everything(),
               names_to = "error_type",
               values_to = "error_value") %>%
  mutate(error_type = case_when(
    error_type == "l2_error" ~ "l2-norm tracking error",
    error_type == "l1_error" ~ "l1-norm tracking error",
    error_type == "downside_error" ~ "downside risk",
    error_type == "huber_error" ~ "Huberized tracking error"
  ))

# Create the faceted histogram plot
ggplot(error_long, aes(x = error_value)) +
  geom_histogram(bins = 30, fill = "steelblue", alpha = 0.7, color = "black") +
  facet_wrap(~ error_type, scales = "free", ncol = 2) +
  labs(title = "Histograms of Different Error Measures",
       x = "Error Value",
       y = "Frequency") +
  theme_minimal() +
  theme(
    strip.text = element_text(size = 10, face = "bold"),
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5)
  )
```



Exercise 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

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \frac{1}{T}\left\|\boldsymbol{r}^{\mathrm{b}}-\boldsymbol{X}\boldsymbol{w}\right\|_2^2 \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w}=1,\ \boldsymbol{w}\geq\mathbf{0}, \\ & \|\boldsymbol{w}\|_0\leq K. \end{array}$$

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.

Solution

First, we get the data:

```
library(xts)
library(pob)   # For data
data(SP500_2015to2020)
T <- nrow(SP500_2015to2020$stocks)
N <- ncol(SP500_2015to2020$stocks)

X <- SP500_2015to2020$stocks/lag(SP500_2015to2020$stocks)[-1] - 1
r <- SP500_2015to2020$index/lag(SP500_2015to2020$index)[-1] - 1
```

a. Solve the problem via a naive two-stage approach: simply select the K active assets with some heuristic and then renormalize so that $\mathbf{1}^\mathsf{T}\boldsymbol{w} = 1$.

```
library(CVXR)

two_stage_naive_tracking <- function(X, r, K = 20) {
  N <- ncol(X)

  # First, get the dense portfolio as the reference
  b <- Variable(N)
  prob <- Problem(Minimize(sum_squares(as.matrix(r) - as.matrix(X) %*% b)),
                  constraints = list(b >= 0, sum(b)==1))
  result <- solve(prob)
  b <- as.vector(result$getValue(b))

  # Then, keep the K strongest
  idx_Klargest <- sort(b, decreasing = TRUE, index.return = TRUE)$ix[1:K]
  w <- rep(0, N)
  w[idx_Klargest] <- b[idx_Klargest]
  w <- w/sum(w)
  return(w)
}

w_2stage_naive <- two_stage_naive_tracking(X, r, K = 20)
```

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.

```r
two_stage_refitted_tracking <- function(X, r, K = 20) {
  N <- ncol(X)

  # First, get the dense portfolio as the reference
  b <- Variable(N)
  prob <- Problem(Minimize(sum_squares(as.matrix(r) - as.matrix(X) %*% b)),
                  constraints = list(b >= 0, sum(b)==1))
  result <- solve(prob)
  b <- as.vector(result$getValue(b))


  # Then, keep the K strongest but reoptimizing weights
  idx_Klargest <- sort(b, decreasing = TRUE, index.return = TRUE)$ix[1:K]
  wK <- Variable(K)
  prob <- Problem(Minimize(sum_squares(as.matrix(r) - as.matrix(X[, idx_Klargest]) %*% wK)),
                  constraints = list(wK >= 0, sum(wK)==1))
  result <- solve(prob)
  w <- rep(0, N)
  w[idx_Klargest] <- as.vector(result$getValue(wK))
  return(w)
}

w_2stage_refitted <- two_stage_refitted_tracking(X, r, K = 20)
```

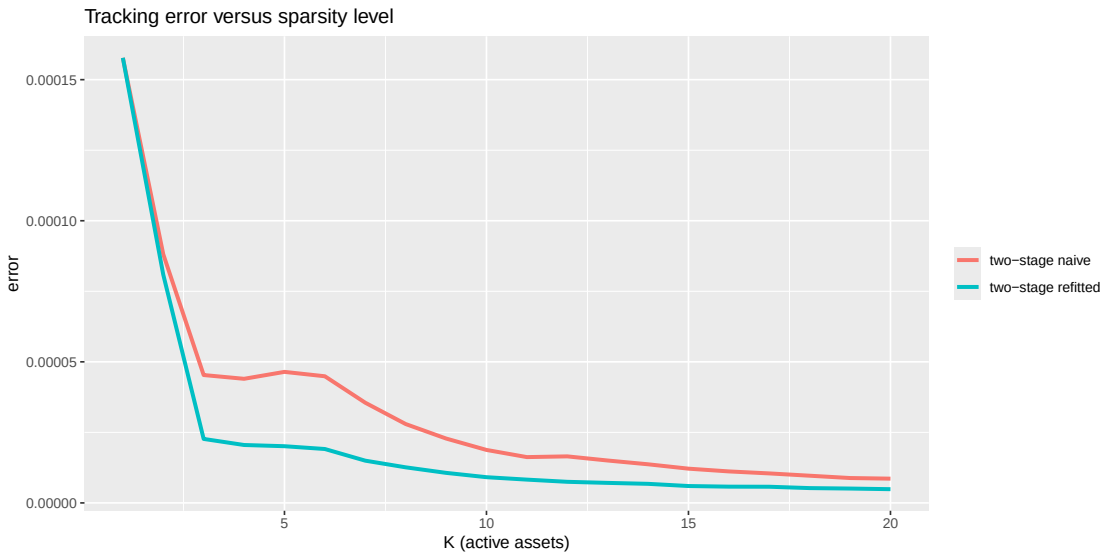
Plot the trade-off curve of regression error vs. sparsity level K for each method:

```r
library(ggplot2)

# Sweeping loop
df <- data.frame()
for (K in 1:20) {
  w_2stage_naive <- two_stage_naive_tracking(X, r, K = K)
  df <- rbind(df, data.frame("method"  = "two-stage naive",
                             "TE"      = mean((r - X %*% w_2stage_naive)^2),
                             "K"       = K))

  w_2stage_refitted <- two_stage_refitted_tracking(X, r, K = K)
  df <- rbind(df, data.frame("method"  = "two-stage refitted",
                             "TE"      = mean((r - X %*% w_2stage_refitted)^2),
                             "K"       = K))
}


# plot
ggplot(df, aes(x = K, y = TE, color = method)) +
  geom_line(linewidth = 1.2) +
  theme(legend.title = element_blank()) +
  labs(title = "Tracking error versus sparsity level",
       x = "K (active assets)", y = "error")
```



Exercise 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).

Solution

First, we get the data:

```
library(xts)
library(pob)   # For data
data(SP500_2015to2020)
T <- nrow(SP500_2015to2020$stocks)
N <- ncol(SP500_2015to2020$stocks)

X <- SP500_2015to2020$stocks/lag(SP500_2015to2020$stocks)[-1] - 1
r <- SP500_2015to2020$index/lag(SP500_2015to2020$index)[-1] - 1
```

a. Approximate the sparsity regularizer with the concave log-function and solve it with a general-purpose nonconvex solver:

```r
library(nloptr)

nonconvex_solver_log_penalty_tracking <- function(X, r, lambda = 1e-6) {
  N <- ncol(X)
  epsilon <- 0.01
  X <- as.matrix(X)
  r <- as.matrix(r)
  result <- nloptr(
    x0 = rep(1/N, N),
    eval_f = function(w) sum((X %*% w - r)^2) + lambda * sum(log(1 + abs(w) / epsilon)),
    eval_g_ineq = function(w) {
                    return(-w)  # w >= 0 constraint (reformulated as -w <= 0)
                  },
    eval_g_eq = function(w) {
                    return(sum(w) - 1)  # sum(w) = 1 constraint
                  },
    opts = list(algorithm = "NLOPT_LN_COBYLA",  # Gradient-free constrained algorithm
                            maxeval = 1000,
                            xtol_rel = 1.0e-8)
    )
  w <- result$solution

  # Clean up very small values (numerical precision issues)
  w[abs(w) < 1e-3*max(abs(w))] <- 0
  w <- w/sum(w)
  return(w)
}

w_log_penalty <- nonconvex_solver_log_penalty_tracking(X, r, lambda = 1e-6)
```

b. Apply the majorization–minimization approach to get the iterative reweighted ℓ_1-norm method. This can be done with a code similar to Exercise 13.4(d). For simplicity, we will resort to the R package sparseIndexTracking.

```r
library(sparseIndexTracking)

w_iterative_reweighted_l1 <- spIndexTrack(X, r, lambda = 1e-6)
```

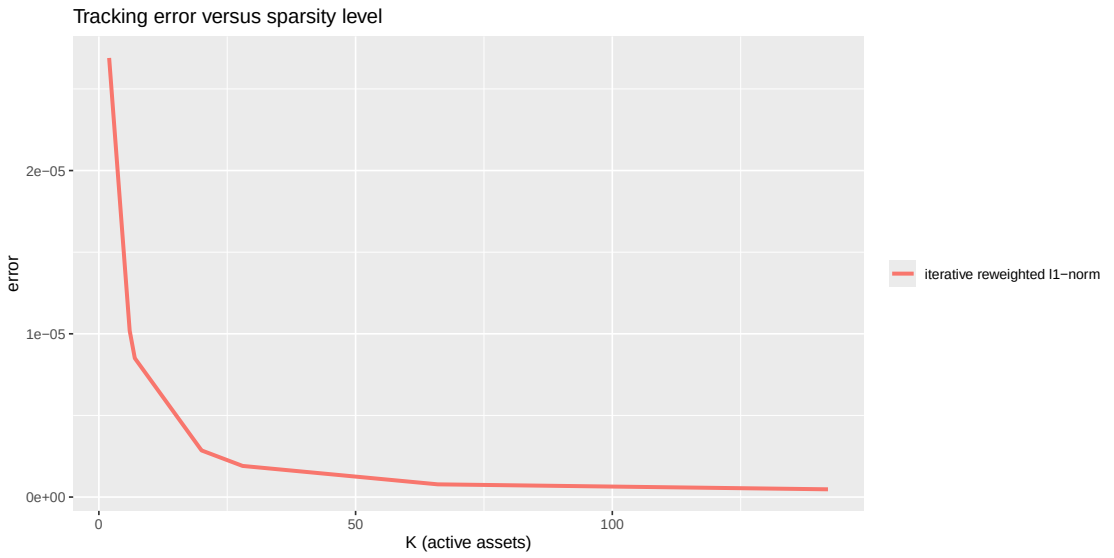
Plot the trade-off curve of regression error vs. sparsity level K for each method:

```r
library(ggplot2)

# Sweeping loop
df <- data.frame()
for (lambda in c(1e-5, 5e-5, 1e-6, 5e-6, 1e-7, 5e-7, 1e-8)) {
  # w_log_penalty <- nonconvex_solver_log_penalty_tracking(X, r, lambda = lambda)
  # df <- rbind(df, data.frame("method"  = "nonconvex solver with log penalty term",
  #                            "TE"      = mean((r - X %*% w_log_penalty)^2),
  #                            "K"       = sum(w_log_penalty > 1e-3)))

  w_iterative_reweighted_l1 <- spIndexTrack(X, r, lambda = lambda)
  df <- rbind(df, data.frame("method"  = "iterative reweighted l1-norm",
                             "TE"      = mean((r - X %*% w_iterative_reweighted_l1)^2),
                             "K"       = sum(w_iterative_reweighted_l1 > 1e-3)))
}

# plot
ggplot(df, aes(x = K, y = TE, color = method)) +
  geom_line(linewidth = 1.2) +
  theme(legend.title = element_blank()) +
  labs(title = "Tracking error versus sparsity level",
       x = "K (active assets)", y = "error")
```



Exercise 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:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \frac{1}{T}\left\|\left(\boldsymbol{r}^{\mathrm{b}}-\boldsymbol{X}\boldsymbol{w}\right)^{+}\right\|_2^2+\lambda\sum_{i=1}^{N}\alpha_i^k|w_i| \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w}=1,\ \boldsymbol{w}\geq\mathbf{0}, \end{array}$$

where

$$\alpha_i^k=\frac{1}{\varepsilon+|w_i^k|}.$$

c. Apply the majorization–minimization approach fully to get the iterative reweighted ℓ_1-norm method that solves sequentially the following convex problem:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \frac{1}{T}\left\|\left(\tilde{\boldsymbol{r}}^{\mathrm{b}}\right)^k-\boldsymbol{X}\boldsymbol{w}\right\|_2^2+\lambda\sum_{i=1}^{N}\alpha_i^k|w_i| \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w}=1,\ \boldsymbol{w}\geq\mathbf{0}, \end{array}$$

where now

$$\left(\tilde{\boldsymbol{r}}^{\mathrm{b}}\right)^k=\boldsymbol{r}^{\mathrm{b}}+\left(\boldsymbol{X}\boldsymbol{w}^k-\boldsymbol{r}^{\mathrm{b}}\right)^{+}.$$

Plot the trade-off curve of regression error vs. sparsity level for each method (by varying the hyper-parameter λ).

Solution

Similar to solution to Exercise 13.8.

Exercise 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 $\boldsymbol{r}^{\mathrm{b}}$ and the assets' return matrix $\boldsymbol{X}$, and formulate the sparse index tracking problem

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \frac{1}{T}\left\|\boldsymbol{r}^{\mathrm{b}}-\boldsymbol{X}\boldsymbol{w}\right\|_2^2+\lambda\|\boldsymbol{w}\|_0 \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w}=1,\ \boldsymbol{w}\geq\mathbf{0}. \end{array}$$

a. Approximate the sparsity regularizer with the ℓ_1-norm:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimize}} & \frac{1}{T}\left\|\boldsymbol{r}^{\mathrm{b}}-\boldsymbol{X}\boldsymbol{w}\right\|_2^2+\lambda\|\boldsymbol{w}\|_1 \\ \text{subject to} & \mathbf{1}^{\mathsf{T}}\boldsymbol{w}=1,\ \boldsymbol{w}\geq\mathbf{0}. \end{array}$$

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.

Solution

First, we get the data:

```
library(xts)
library(pob)   # For data
data(SP500_2015to2020)
T <- nrow(SP500_2015to2020$stocks)
N <- ncol(SP500_2015to2020$stocks)

X <- SP500_2015to2020$stocks/lag(SP500_2015to2020$stocks)[-1] - 1
r <- SP500_2015to2020$index/lag(SP500_2015to2020$index)[-1] - 1
```

a. Approximate the sparsity regularizer with the ℓ_1-norm:

```r
library(CVXR)

l1_norm_approximation_tracking <- function(X, r, lambda = 1e-6) {
  N <- ncol(X)
  T <- nrow(X)
  w <- Variable(N)
  prob <- Problem(Minimize((1/T)*sum_squares(as.matrix(r) - as.matrix(X) %*% w)
                           + lambda*norm1(w)),
                  constraints = list(w >= 0, sum(w)==1))
  result <- solve(prob)
  w <- as.vector(result$getValue(w))
  # Clean up very small values (numerical precision issues)
  w[abs(w) < 1e-3*max(abs(w))] <- 0
  w <- w/sum(w)
  return(w)
}

w_l1_approx <- l1_norm_approximation_tracking(X, r, lambda = 1e-6)
```

In this case, the choice of λ has no effect in the solution since $\|\boldsymbol{w}\|_1 = 1$ due to the constraints.

b. Employ the T-Rex method to automatically choose the active assets with FDR control:

```r
library(TRexSelector)
library(quadprog)

# T-Rex can be used on the cumulative returns as well:
X_cum <- apply(X, 2, function(x) cumsum(x))
r_cum <- cumsum(r)

# Step 1: T-Rex
trex_selected <- TRexSelector::trex(X = tail(as.matrix(X), 120),
                                    y = tail(as.matrix(r), 120),
                                    tFDR = 0.3, method = "trex+DA+NN",
                                    verbose = FALSE)$selected_var
idx_trex <- which(trex_selected == 1)
K <- length(idx_trex)
# Step 2: Refit selected assets
wK <- solve.QP(Dmat = 2 * t(X[, idx_trex]) %*% X[, idx_trex],
               dvec = 2 * t(X[, idx_trex]) %*% r,
               Amat = cbind(rep(1, K), diag(K)), bvec = c(1, rep(0, K)), meq = 1)$solution
w <- rep(0, N)
w[idx_trex] <- wK

# Display optimal sparsity level
cat("Optinal sparsity level =", K)
```

```
Optinal sparsity level = 17
```