Solutions to Exercises

Portfolio Optimization: Theory and Application
Chapter 6 – Portfolio Basics

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

portfoliooptimizationbook.com

Exercise 6.1: Effect of rebalancing

a. Download market data corresponding to N assets (e.g., stocks or cryptocurrencies) during a period with T observations, $\boldsymbol{r}_1, \dots, \boldsymbol{r}_T \in \mathbb{R}^N$.
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.

Solution

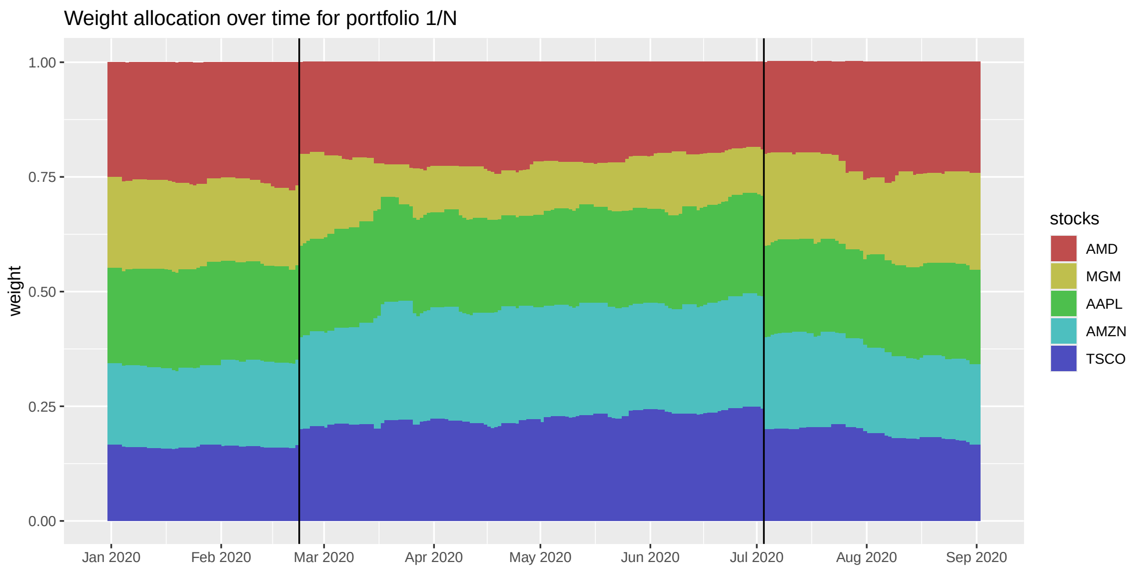
a. Market data corresponding to N stocks:

```r
library(xts)
library(pob)       # Market data used in the book

# Use data from package pob
data(SP500_2015to2020)
stock_prices <- SP500_2015to2020$stocks["2019-10::",
                                        c("AMD", "MGM", "AAPL", "AMZN", "TSCO")]
```

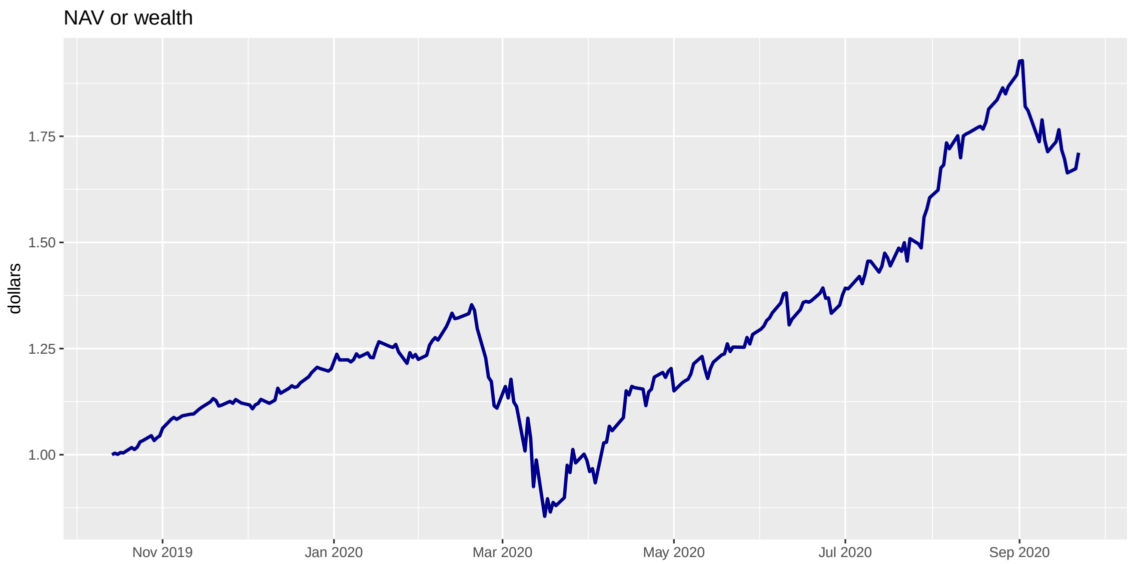
b. Backtest the $1/N$ portfolio with the R package `portfolioBacktest`:

```r
library(portfolioBacktest)

EWP <- function(data, ...) {
  N <- ncol(data[[1]])
  return(rep(1/N, N))
}

bt_5stocks <- portfolioBacktest(
  portfolio_funs = list("1/N" = EWP),
  dataset_list = list(list("prices" = stock_prices)), price_name = "prices",
  lookback = 10, optimize_every = 90, rebalance_every = 90,
  cost = list(buy = 90e-4, sell = 90e-4)
  )
```

```r
library(PerformanceAnalytics)
library(ggplot2)
library(reshape2)

# Plot weights over time
bt_5stocks$`1/N`$data1$w_bop["2020-01::2020-08", ] |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, fill = Series)) +
  geom_bar(stat = "identity", width = 4.0) +
  labs(fill = "stocks") +  scale_fill_manual(values = rainbow6equal) +
  scale_x_date(date_breaks = "1 month", date_labels = "%b %Y") +
  labs(title = "Weight allocation over time for portfolio 1/N", x = NULL, y = "weight") +
  geom_vline(xintercept = as.Date("2020-02-23"), color = "black") +
  geom_vline(xintercept = as.Date("2020-07-03"), color = "black")
```




```
# Plot NAV over time
bt_5stocks$`1/N`$data1$wealth |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value)) +
  geom_line(linewidth = 1, color = "darkblue") +
  scale_x_date(date_breaks = "2 months", date_labels = "%b %Y") +
  labs(title = "NAV or wealth", x = NULL, y = "dollars")
```



Exercise 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:

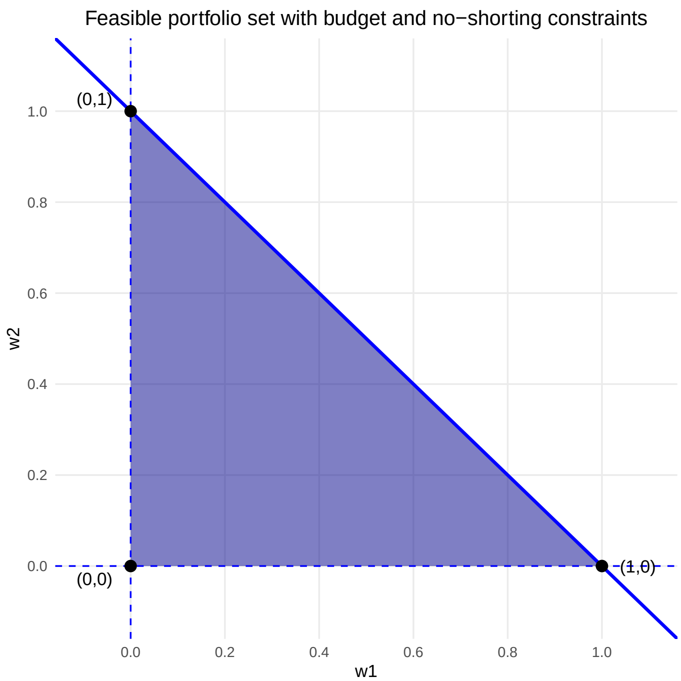
$$\|\boldsymbol{w}\|_1 \leq 1.$$

Solution

a. Budget and no-shorting constraints:

$$w_1 + w_2 \leq 1, \quad w_1 \geq 0, \quad w_2 \geq 0.$$

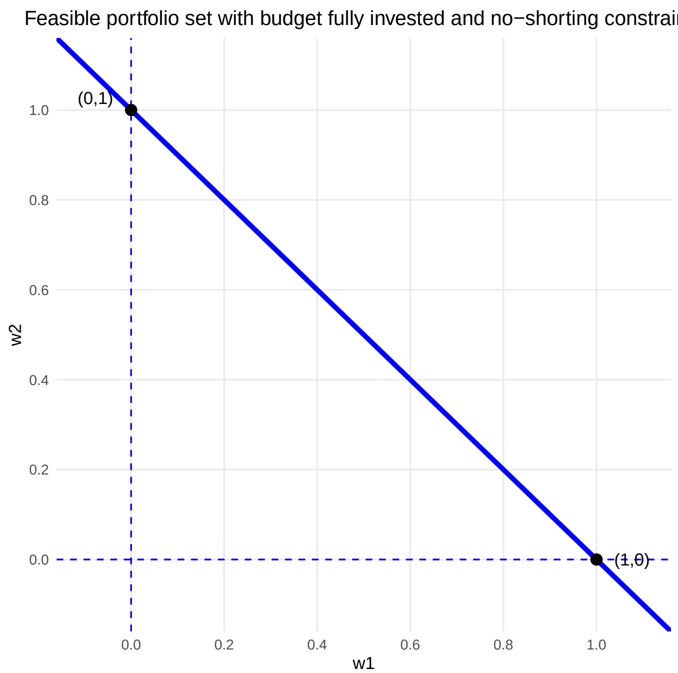
```r
library(ggplot2)

ggplot() +
  # Add the feasible region as a polygon
  geom_polygon(data = data.frame(w1 = c(0, 1, 0), w2 = c(0, 0, 1)),
               aes(x = w1, y = w2), fill = "darkblue", alpha = 0.5) +

  # Add the budget constraint line
  geom_abline(intercept = 1, slope = -1, color = "blue", linewidth = 1) +

  # Add the no-shorting constraint lines (axes)
  geom_vline(xintercept = 0, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = 0, color = "blue", linetype = "dashed") +

  # Add labels for the vertices
  geom_point(data = data.frame(w1 = c(0, 1, 0), w2 = c(0, 0, 1)),
             aes(x = w1, y = w2), size = 3) +
  geom_text(data = data.frame(
    w1 = c(0, 1, 0), w2 = c(0, 0, 1), label = c("(0,0)", "(1,0)", "(0,1)")
  ), aes(x = w1, y = w2, label = label),
  hjust = c(1.5, -0.5, 1.5), vjust = c(1.5, 0.5, -0.5)) +

  # Add a title and axis labels
  labs(title = "Feasible portfolio set with budget and no-shorting constraints",
       x = "w1", y = "w2") +

  # Set axis limits and add a theme
  scale_x_continuous(limits = c(-0.1, 1.1), breaks = seq(0, 1, 0.2)) +
  scale_y_continuous(limits = c(-0.1, 1.1), breaks = seq(0, 1, 0.2)) +
  theme_minimal() +
  theme(panel.grid.minor = element_blank(),
        plot.title = element_text(hjust = 0.5))
```



b. Budget fully invested and no-shorting constraints:

$$w_1 + w_2 = 1, \quad w_1 \geq 0, \quad w_2 \geq 0.$$

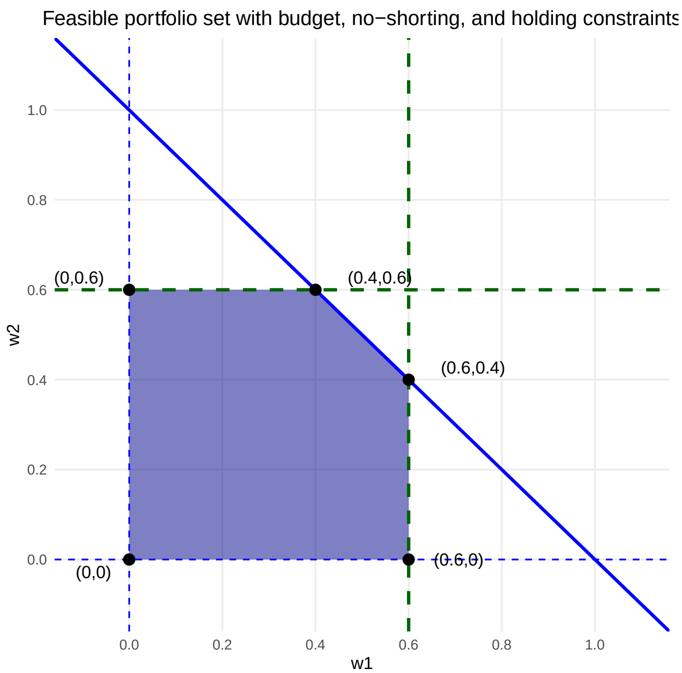
```r
ggplot() +
  # Add the budget constraint line (feasible region) using geom_abline
  geom_abline(intercept = 1, slope = -1, color = "blue", linewidth = 1.5) +

  # Add the no-shorting constraint lines (axes)
  geom_vline(xintercept = 0, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = 0, color = "blue", linetype = "dashed") +

  # Add labels for the endpoints
  geom_point(data = data.frame(w1 = c(0, 1), w2 = c(1, 0)),
             aes(x = w1, y = w2), size = 3) +
  geom_text(data = data.frame(
    w1 = c(0, 1), w2 = c(1, 0), label = c("(0,1)", "(1,0)")
  ), aes(x = w1, y = w2, label = label),
  hjust = c(1.5, -0.5), vjust = c(-0.5, 0.5)) +

  # Add a title and axis labels
  labs(title = "Feasible portfolio set with budget fully invested and no-shorting constraints",
       x = "w1", y = "w2") +

  # Set axis limits and add a theme
  scale_x_continuous(limits = c(-0.1, 1.1), breaks = seq(0, 1, 0.2)) +
  scale_y_continuous(limits = c(-0.1, 1.1), breaks = seq(0, 1, 0.2)) +
  theme_minimal() +
  theme(panel.grid.minor = element_blank(),
        plot.title = element_text(hjust = 0.5))
```



c. Budget, no-shorting, and holding constraints:

$$w_1 + w_2 \leq 1, \quad 0 \leq w_1 \leq 0.6, \quad 0 \leq w_2 \leq 0.6.$$

```r
ggplot() +
  # Add the feasible region as a polygon
  geom_polygon(data = data.frame(
    w1 = c(0, 0.6, 0.6, 0.4, 0),
    w2 = c(0, 0, 0.4, 0.6, 0.6)
  ), aes(x = w1, y = w2), fill = "darkblue", alpha = 0.5) +

  # Add the budget constraint line
  geom_abline(intercept = 1, slope = -1, color = "blue", linewidth = 1) +

  # Add the holding constraint lines
  geom_vline(xintercept = 0.6, color = "darkgreen", linewidth = 1, linetype = "dashed") +
  geom_hline(yintercept = 0.6, color = "darkgreen", linewidth = 1, linetype = "dashed") +

  # Add the no-shorting constraint lines (axes)
  geom_vline(xintercept = 0, color = "blue", linetype = "dashed") +
  geom_hline(yintercept = 0, color = "blue", linetype = "dashed") +

  # Add labels for key points
  geom_point(data = data.frame(
    w1 = c(0, 0.6, 0.6, 0.4, 0),
    w2 = c(0, 0, 0.4, 0.6, 0.6)
  ), aes(x = w1, y = w2), size = 3) +
  geom_text(data = data.frame(
    w1 = c(0, 0.6, 0.6, 0.4, 0),
    w2 = c(0, 0, 0.4, 0.6, 0.6),
    label = c("(0,0)", "(0.6,0)", "(0.6,0.4)", "(0.4,0.6)", "(0,0.6)")
  ), aes(x = w1, y = w2, label = label),
  hjust = c(1.5, -0.5, -0.5, -0.5, 1.5),
  vjust = c(1.5, 0.5, -0.5, -0.5, -0.5)) +

  # Add a title and axis labels
  labs(
    title = "Feasible portfolio set with budget, no-shorting, and holding constraints",
    x = "w1", y = "w2"
  ) +

  # Set axis limits and add a theme
  scale_x_continuous(limits = c(-0.1, 1.1), breaks = seq(0, 1, 0.2)) +
  scale_y_continuous(limits = c(-0.1, 1.1), breaks = seq(0, 1, 0.2)) +
  theme_minimal() +
  theme(
    panel.grid.minor = element_blank(),
    plot.title = element_text(hjust = 0.5)
  )
```

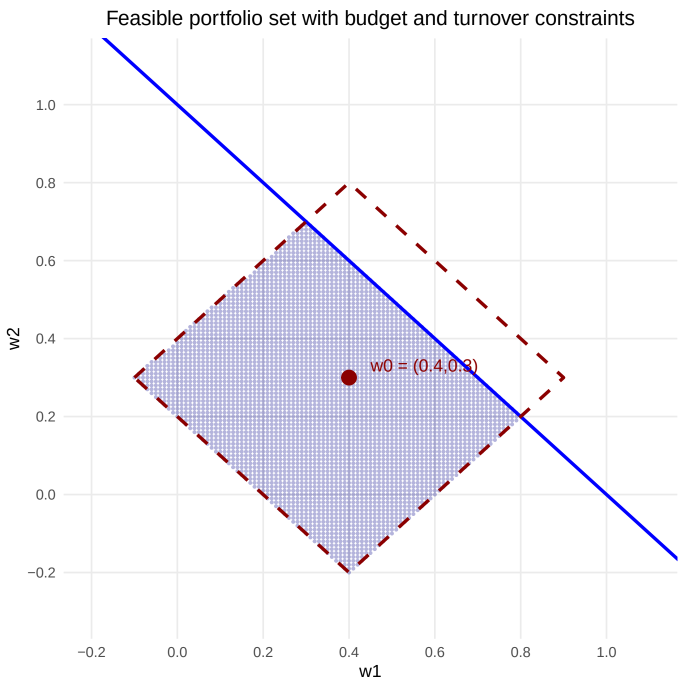


d. Budget and turnover constraints:

$$w_1 + w_2 \leq 1, \quad |w_1 - w_{0,1}| + |w_2 - w_{0,2}| \leq 0.5,$$

with $\boldsymbol{w}_0 = (0.4, 0.3)$.

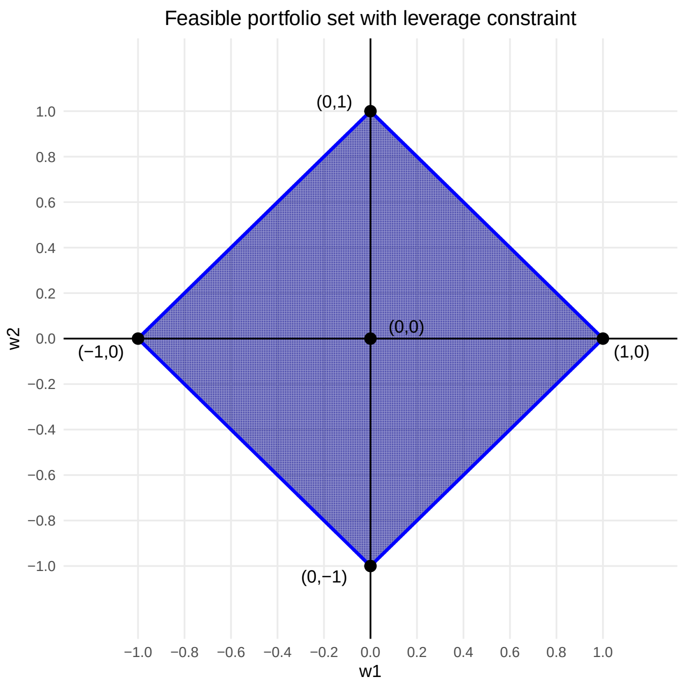
```r
# Define the initial portfolio
w0 <- c(0.4, 0.3)

# Generate points for the feasible region (intersection of constraints)
grid_points <- expand.grid(
  w1 = seq(-0.2, 1.1, by = 0.01),
  w2 = seq(-0.3, 1.1, by = 0.01)
)

# Filter points that satisfy all constraints
feasible_points <- subset(grid_points,
                         w1 + w2 <= 1 &
                         abs(w1 - w0[1]) + abs(w2 - w0[2]) <= 0.5)

# Create the plot
ggplot() +
  # Add the feasible region as points
  geom_point(data = feasible_points, aes(x = w1, y = w2),
             color = "darkblue", alpha = 0.3, size = 0.5) +

  # Add the budget constraint line
  geom_abline(intercept = 1, slope = -1, color = "blue", linewidth = 1) +

  # Add the turnover constraint boundary (diamond)
  geom_polygon(data = data.frame(
    w1 = c(w0[1] - 0.5, w0[1], w0[1] + 0.5, w0[1]),
    w2 = c(w0[2], w0[2] + 0.5, w0[2], w0[2] - 0.5)
  ), aes(x = w1, y = w2), fill = NA, color = "darkred",
  linetype = "dashed", linewidth = 1) +

  # Mark the initial portfolio
  geom_point(aes(x = w0[1], y = w0[2]), size = 4, color = "darkred") +
  geom_text(aes(x = w0[1], y = w0[2],
              label = paste0("w0 = (", w0[1], ",", w0[2], ")")),
          hjust = -0.2, vjust = -0.5, color = "darkred") +

  # Add a title and axis labels
  labs(title = "Feasible portfolio set with budget and turnover constraints",
     x = "w1", y = "w2") +

  # Set axis limits and add a theme
  scale_x_continuous(limits = c(-0.2, 1.1), breaks = seq(-0.2, 1, 0.2)) +
  scale_y_continuous(limits = c(-0.3, 1.1), breaks = seq(-0.2, 1, 0.2)) +
  theme_minimal() +
  theme(panel.grid.minor = element_blank(),
      plot.title = element_text(hjust = 0.5))
```



e. Leverage constraint:

$$|w_1| + |w_2| \leq 1.$$

```r
# Generate points for the feasible region
grid_points <- expand.grid(
  w1 = seq(-1.1, 1.1, by = 0.01),
  w2 = seq(-1.1, 1.1, by = 0.01)
)

# Filter points that satisfy the leverage constraint
feasible_points <- subset(grid_points, abs(w1) + abs(w2) <= 1)

# Key points on the boundary
key_points <- data.frame(
  w1 = c(0, 1, 0, -1, 0),
  w2 = c(0, 0, 1, 0, -1),
  label = c("(0,0)", "(1,0)", "(0,1)", "(-1,0)", "(0,-1)"),
  hjust = c(-0.5, -0.3, 1.5, 1.3, 1.5),
  vjust = c(-0.5, 1.5, -0.3, 1.5, 1.3)
)

# Create the plot
ggplot() +
  # Add the feasible region as points
  geom_point(data = feasible_points, aes(x = w1, y = w2),
             color = "darkblue", alpha = 0.3, size = 0.5) +

  # Add the leverage constraint boundary (diamond)
  geom_polygon(data = data.frame(
    w1 = c(1, 0, -1, 0),
    w2 = c(0, 1, 0, -1)
  ), aes(x = w1, y = w2), fill = NA, color = "blue", linewidth = 1) +

  # Mark key points and add labels
  geom_point(data = key_points, aes(x = w1, y = w2), size = 3) +
  geom_text(data = key_points,
            aes(x = w1, y = w2, label = label, hjust = hjust, vjust = vjust)) +

  # Add coordinate axes
  geom_hline(yintercept = 0, color = "black", linewidth = 0.5) +
  geom_vline(xintercept = 0, color = "black", linewidth = 0.5) +

  # Add a title and axis labels
  labs(title = "Feasible portfolio set with leverage constraint",
       x = "w1", y = "w2") +

  # Set axis limits and add a theme
  scale_x_continuous(limits = c(-1.2, 1.2), breaks = seq(-1, 1, 0.2)) +
  scale_y_continuous(limits = c(-1.2, 1.2), breaks = seq(-1, 1, 0.2)) +
  theme_minimal() +
  theme(panel.grid.minor = element_blank(),
        plot.title = element_text(hjust = 0.5))
```



Exercise 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.

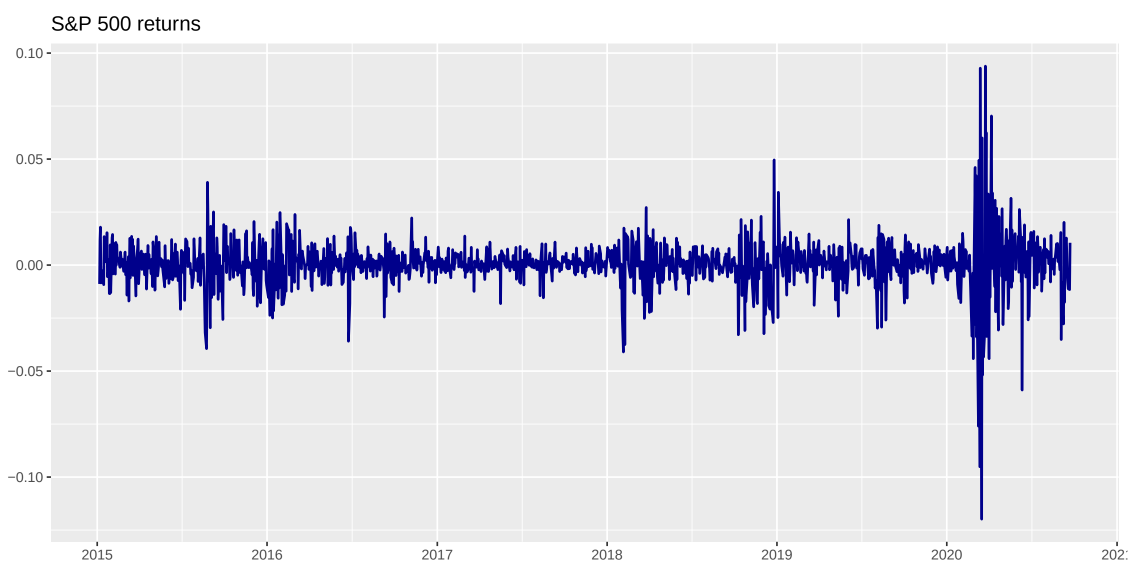
Solution

a. Market data corresponding to the S&P 500 index:

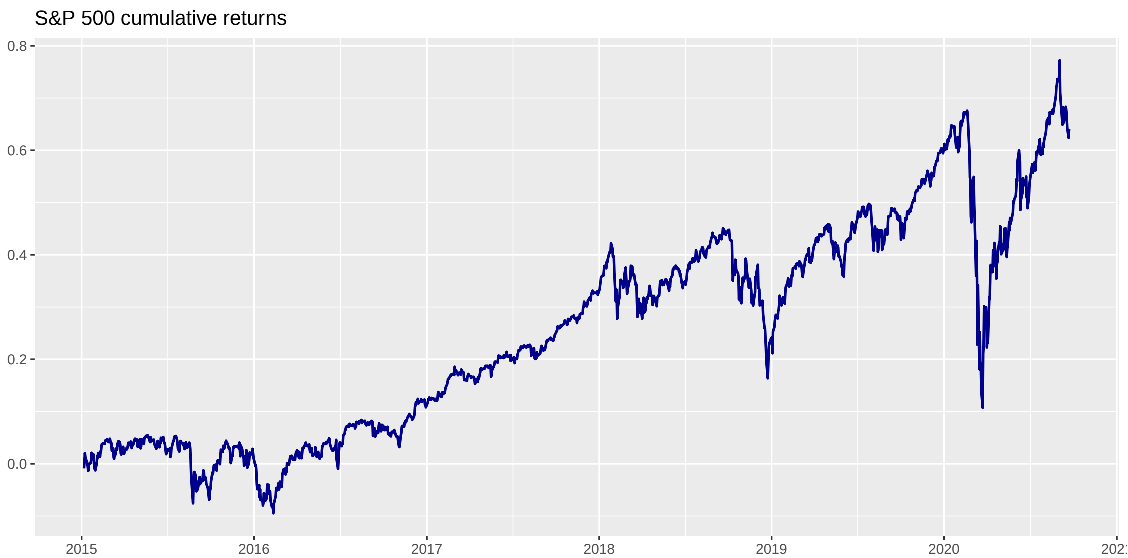
```r
library(xts)
library(pob)       # Market data used in the book

# Use data from package pob
data(SP500_2015to2020)
SP500_price <- SP500_2015to2020$index
SP500_logreturns <- diff(log(SP500_price))[-1]
SP500_linreturns <- exp(SP500_logreturns) - 1
SP500_cumreturns <- cumprod(1 + SP500_linreturns) - 1
```

b. Plot the returns and cumulative returns over time:

```r
library(ggplot2)

ggplot(fortify(SP500_linreturns, melt = TRUE), aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "S&P 500 returns", x = NULL, y = NULL)
```




```r
ggplot(fortify(SP500_cumreturns, melt = TRUE), aes(x = Index, y = Value)) +
  geom_line(linewidth = 0.8, color = "darkblue") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "S&P 500 cumulative returns", x = NULL, y = NULL)
```



c. Calculate the annualized expected return with arithmetic and geometric compounding.

```
library(PerformanceAnalytics)

scale <- 252
n <- nrow(SP500_linreturns)

# Explicit calculation
arithmetic_return <- mean(SP500_linreturns) * scale
geometric_return <- prod(1 + SP500_linreturns)^(scale/n) - 1

# Using the package PerformanceAnalytics
arithmetic_return_pkg <- Return.annualized(SP500_linreturns, scale = scale, geometric = FALSE)
geometric_return_pkg <- Return.annualized(SP500_linreturns, scale = scale)
```

```
# Print results
cat("Arithmetic Annualized Return:", arithmetic_return, "\n")
cat("Arithmetic Annualized Return (using package):", arithmetic_return_pkg, "\n")
cat("Geometric Annualized Return:", geometric_return, "\n")
cat("Geometric Annualized Return (using package):", geometric_return_pkg, "\n")
```

```
Arithmetic Annualized Return: 0.1043978
Arithmetic Annualized Return (using package): 0.1043978
Geometric Annualized Return: 0.0906001
Geometric Annualized Return (using package): 0.0906001
```

d. Calculate the annualized volatility.

```r
library(PerformanceAnalytics)

# Explicit calculation
daily_volatility <- sd(SP500_linreturns)
annualized_volatility <- daily_volatility * sqrt(scale)

# Using the package PerformanceAnalytics
annualized_volatility_pkg <- StdDev.annualized(SP500_linreturns, scale = scale)
```

```r
# Print results
cat("Annualized Volatility:", annualized_volatility, "\n")
cat("Annualized Volatility (using package):", annualized_volatility_pkg, "\n")
```

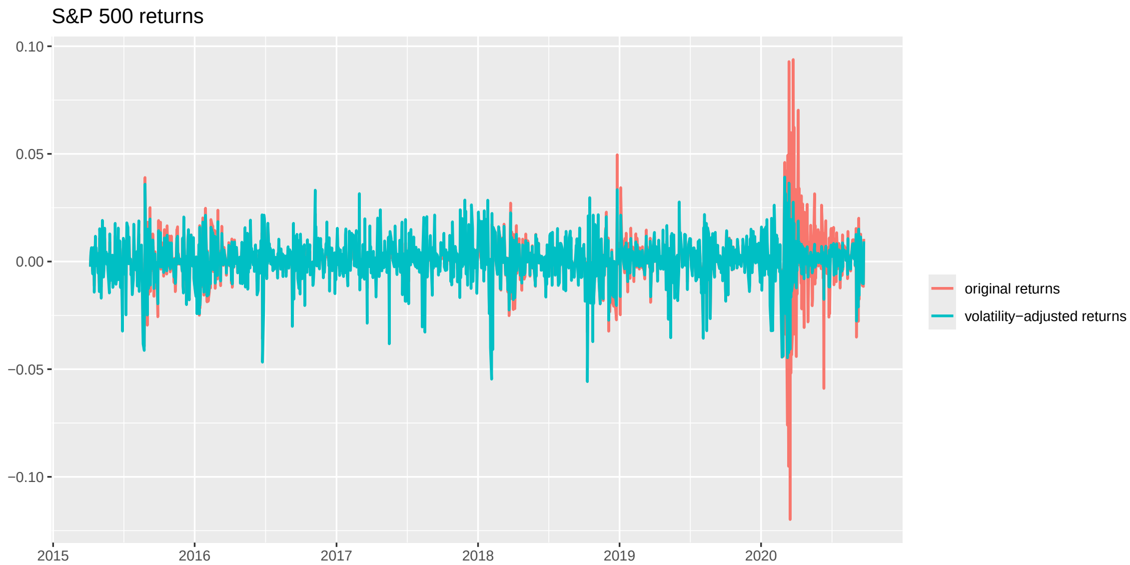
```
Annualized Volatility: 0.1873672
Annualized Volatility (using package): 0.1873672
```

e. Plot the volatility-adjusted returns and cumulative returns over time.

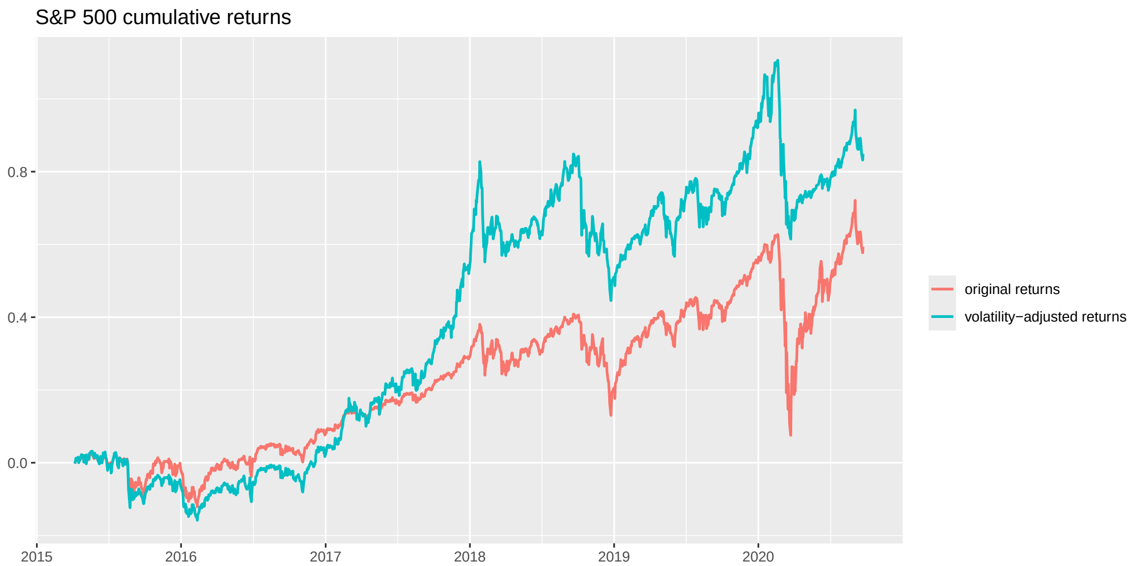
In a short period, we can just scale the returns to satisfy the desired target volatility. In a long period, it is better to do it on a rolling-window basis:

```r
library(zoo)  # To use the rollapply function

target_vol <- 0.01  # target volatility
window_size <- 252/4  # rolling window size (252 is one year, 252/4 one quarter)

# Calculate rolling standard deviation
rolling_sd <- rollapply(SP500_linreturns, width = window_size,
                        FUN = sd, by = 1, align = 'right', fill = NA)

# Adjust returns by target volatility on rolling basis
SP500_linreturns_voladj <- target_vol * SP500_linreturns / rolling_sd

# Compute cumulative returns
all_returns <- cbind(SP500_linreturns, SP500_linreturns_voladj)
all_returns <- na.omit(all_returns)
colnames(all_returns) <- c("original returns", "volatility-adjusted returns")
all_cumreturns <- cumprod(1 + all_returns) - 1
```

```r
# Plots
ggplot(fortify(all_returns, melt = TRUE), aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 0.8) +
  labs(color = "") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "S&P 500 returns", x = NULL, y = NULL)
```




```
ggplot(fortify(all_cumreturns, melt = TRUE), aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 0.8) +
  labs(color = "") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "S&P 500 cumulative returns", x = NULL, y = NULL)
```



f. Calculate the annualized Sharpe ratio with arithmetic and geometric compounding.

```r
library(PerformanceAnalytics)

rf_daily <- 0.03 / 252  # Daily risk-free rate: 3% annual risk-free rate
scale <- 252  # For daily returns
n <- length(SP500_linreturns)


#
# Explicit calculation
#

# Annualized arithmetic Sharpe ratio
annualized_arithmetic_excess_return <- mean(SP500_linreturns - rf_daily) * scale
returns_sd <- sd(SP500_linreturns)
arithmetic_sharpe <- annualized_arithmetic_excess_return / (returns_sd * sqrt(scale))

# Annualized geometric Sharpe ratio
annualized_geometric_excess_return <- prod(1 + (SP500_linreturns - rf_daily))^(scale/n) - 1
geometric_sharpe <- annualized_geometric_excess_return / (returns_sd * sqrt(scale))
```

```r
#
# Calculation using the package PerformanceAnalytics
#
arithmetic_sharpe_pkg <- SharpeRatio.annualized(SP500_linreturns,
                                                Rf = rf_daily,
                                                scale = 252,
                                                geometric = FALSE)
geometric_sharpe_pkg <- SharpeRatio.annualized(SP500_linreturns,
                                               Rf = rf_daily,
                                               scale = 252,
                                               geometric = TRUE)
```

```r
# Print results
cat("Arithmetic Annualized Sharpe Ratio:", arithmetic_sharpe, "\n")
cat("Arithmetic Annualized Sharpe Ratio (using package):", arithmetic_sharpe_pkg, "\n")
cat("Geometric Annualized Sharpe Ratio:", geometric_sharpe, "\n")
cat("Geometric Annualized Sharpe Ratio (using package):", geometric_sharpe_pkg, "\n")
```

```
Arithmetic Annualized Sharpe Ratio: 0.3970694
Arithmetic Annualized Sharpe Ratio (using package): 0.3970694
Geometric Annualized Sharpe Ratio: 0.3115529
Geometric Annualized Sharpe Ratio (using package): 0.3115529
```

g. Calculate the annualized semi-deviation and Sortino ratio.

```r
library(PerformanceAnalytics)

MAR <- 0  # Minimum Acceptable Return
scale <- 252  # For daily returns
n <- length(SP500_linreturns)


#
# Explicit calculation
#
downside_returns <- SP500_linreturns[SP500_linreturns < MAR]
semi_dev <- sqrt(sum((downside_returns)^2) / n)
annualized_semi_dev <- semi_dev * sqrt(scale)
annualized_sortino <- mean(SP500_linreturns) * scale / annualized_semi_dev
```

```r
#
# Calculation using the package PerformanceAnalytics
#
semi_dev_pkg <- DownsideDeviation(SP500_linreturns, MAR = MAR)
annualized_semi_dev_pkg <- semi_dev_pkg * sqrt(scale)
annualized_sortino_pkg <- SortinoRatio(SP500_linreturns, MAR = MAR) * sqrt(scale)
```

```r
# Print results
cat("Annualized Semi-Deviation:", annualized_semi_dev, "\n")
cat("Annualized Semi-Deviation (using package):", annualized_semi_dev_pkg, "\n")
cat("Annualized Sortino Ratio:", annualized_sortino, "\n")
cat("Annualized Sortino Ratio (using package):", annualized_sortino_pkg, "\n")
```

```
Annualized Semi-Deviation: 0.1363358
Annualized Semi-Deviation (using package): 0.1363358
Annualized Sortino Ratio: 0.7657401
Annualized Sortino Ratio (using package): 0.7657401
```

h. Calculate the VaR and CVaR.

```r
# Define confidence level (typically 95% or 99%)
alpha <- 0.95

# Calculate VaR (assuming normal distribution)
var_normal <- qnorm(1-alpha, mean = mean(SP500_linreturns), sd = sd(SP500_linreturns))

# Calculate VaR using historical method
var_historical <- quantile(SP500_linreturns, 1-alpha)

# Calculate CVaR (Expected Shortfall)
returns_below_var <- SP500_linreturns[SP500_linreturns <= var_historical]
cvar_historical <- mean(returns_below_var)
```

```r
# Print results
cat("VaR (Gaussian Method, 95%):", var_normal, "\n")
cat("VaR (Historical Method, 95%):", var_historical, "\n")
cat("CVaR (Historical Method, 95%):", cvar_historical, "\n")
```

```
VaR (Gaussian Method, 95%): -0.01899997
VaR (Historical Method, 95%): -0.01697063
CVaR (Historical Method, 95%): -0.02977491
```

```r
#
# Calculation using the package PerformanceAnalytics
#
library(PerformanceAnalytics)

# Calculate VaR using different methods
var_gaussian_pkg <- VaR(SP500_linreturns, p = alpha, method = "gaussian")
var_historical_pkg <- VaR(SP500_linreturns, p = alpha, method = "historical")

# Calculate CVaR (Expected Shortfall)
cvar_gaussian_pkg <- ES(SP500_linreturns, p = alpha, method = "gaussian")
cvar_historical_pkg <- ES(SP500_linreturns, p = alpha, method = "historical")
```

```r
# Print results
cat("VaR (Gaussian Method, 95%):", var_gaussian_pkg, "\n")
cat("VaR (Historical Method, 95%):", var_historical_pkg, "\n")
#cat("CVaR (Gaussian Method, 95%):", cvar_gaussian_pkg, "\n")
cat("CVaR (Historical Method, 95%):", cvar_historical_pkg, "\n")
```

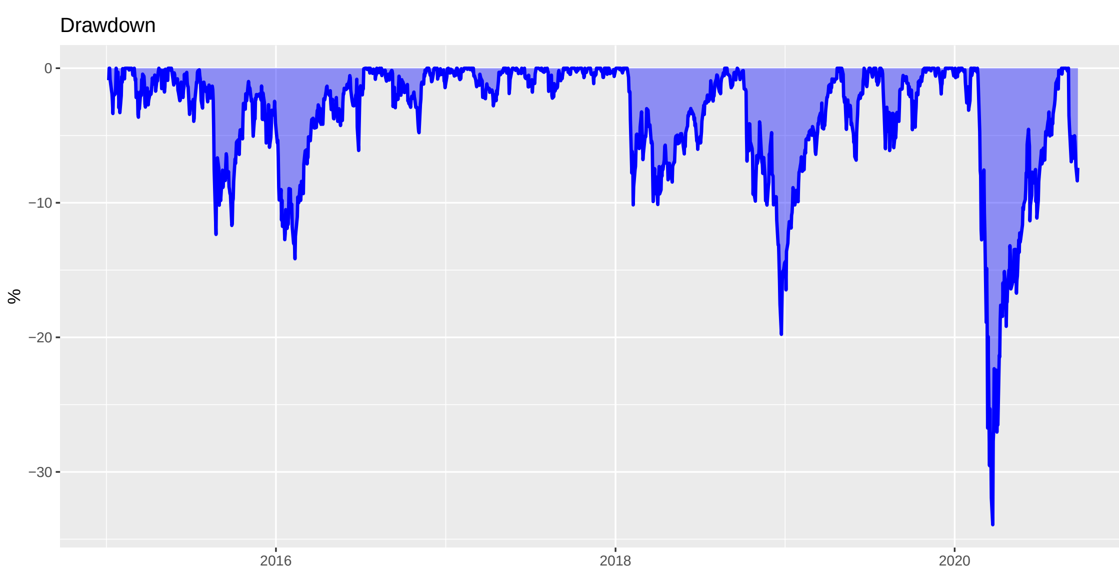
```
VaR (Gaussian Method, 95%): -0.01899322
VaR (Historical Method, 95%): -0.01697063
CVaR (Historical Method, 95%): -0.02977491
```

i. Plot the drawdown over time.

```r
library(PerformanceAnalytics)

# Compute NAV and drawdown
SP500_nav <- cumprod(1 + SP500_linreturns)
hwm <- cummax(c(1, SP500_nav))[-1]
SP500_drawdown <- 100 * (SP500_nav - hwm) / hwm
SP500_drawdown_pkg <- 100 * Drawdowns(SP500_linreturns)

# Sanity check
max(abs(SP500_drawdown - SP500_drawdown_pkg))
```

```
[1] 7.105427e-15
```

```r
# Plot
SP500_drawdown |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series, fill = Series)) +
  geom_area(position = "identity", show.legend = FALSE, linewidth = 1, alpha = 0.4) +
  scale_color_manual(values = "blue") +
  scale_fill_manual(values = "blue") +
  scale_x_date(date_breaks = "2 year", date_labels = "%Y") +
  labs(title = "Drawdown", x = NULL, y = "%")
```



Exercise 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.

Solution

a. Market data corresponding to N stocks:

```
library(xts)
library(pob)       # Market data used in the book

# Use data from package pob
data(SP500_2015to2020)
stock_prices <- SP500_2015to2020$stocks[
  "2019::",
  c("AAPL", "AMZN", "AMD", "GM", "GOOGL", "MGM", "MSFT", "QCOM", "TSCO", "UPS")
  ]
```

b. Using 70% of the data, compute the $1/N$ portfolio and quintile portfolios using different ranking mechanisms:

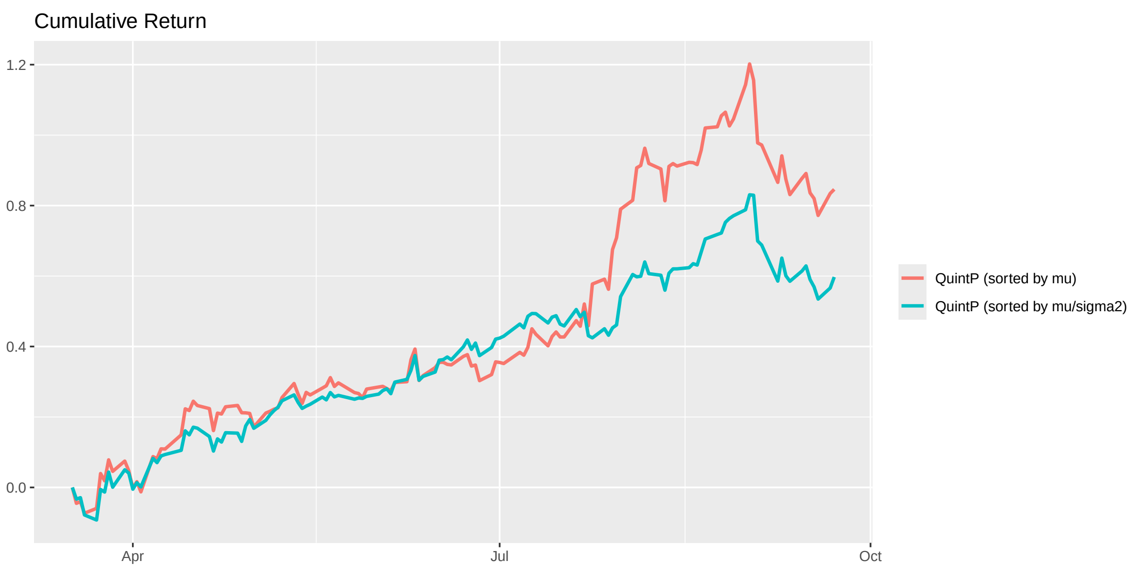
```r
T <- nrow(stock_prices)
T_trn <- round(0.70*T)

QuintP_mu <- function(dataset, ...) {
  N <- ncol(dataset$prices)
  X <- diff(log(dataset$prices))[-1]
  mu <- colMeans(X)
  idx <- order(mu, decreasing = TRUE)
  w <- rep(0, N)
  w[idx[1:round(N/5)]] <- 1/round(N/5)
  return(w)
}

QuintP_mu_over_sigma2 <- function(dataset, ...) {
  N <- ncol(dataset$prices)
  X <- diff(log(dataset$prices))[-1]
  mu <- colMeans(X)
  Sigma <- cov(X)
  idx <- order(mu/diag(Sigma), decreasing = TRUE)
  w <- rep(0, N)
  w[idx[1:round(N/5)]] <- 1/round(N/5)
  return(w)
}
```

At this point, we can conveniently use the R package portfolioBactest to perform the backtest (either a simple static one or a rolling-window one or even multiple backtests), as well as to compute performance measures, and plot the results as follows:

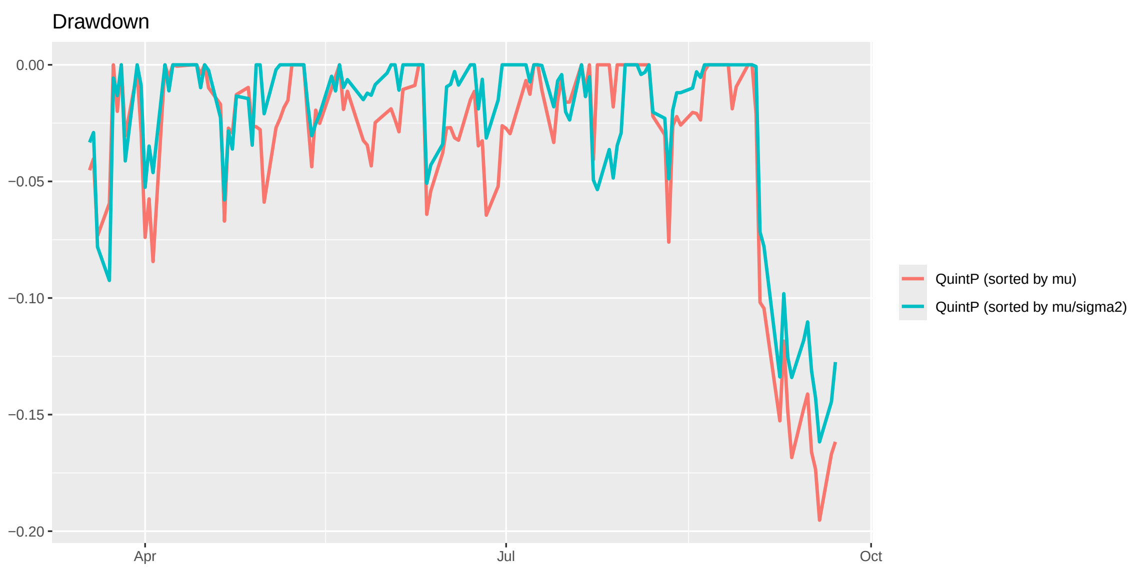
```r
# Backtest via the portfolioBacktest package
library(portfolioBacktest)

bt <- portfolioBacktest(
  portfolio_funs = list("QuintP (sorted by mu)"        = QuintP_mu,
                        "QuintP (sorted by mu/sigma2)" = QuintP_mu_over_sigma2),
  dataset_list = list("dataset1" = list("prices" = stock_prices)),
  lookback = T_trn, optimize_every = 10000, rebalance_every = 1
  )

# Print some performance measures
bt_summary <- backtestSummary(bt)
summaryTable(bt_summary, measures = c("Sharpe ratio", "max drawdown"))
```

```
                             Sharpe ratio max drawdown
QuintP (sorted by mu)                2.73         0.20
QuintP (sorted by mu/sigma2)         2.52         0.16
```

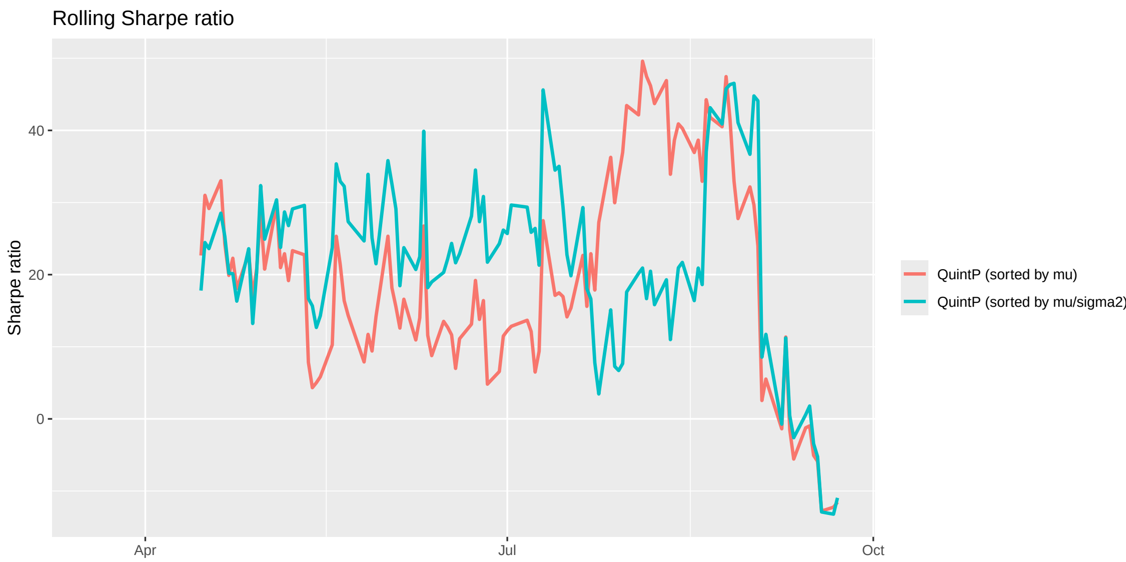
```r
# Plots
backtestChartCumReturn(bt)
```




```r
backtestChartDrawdown(bt)
```




```r
backtestChartSharpeRatio(bt, lookback = 20)
```

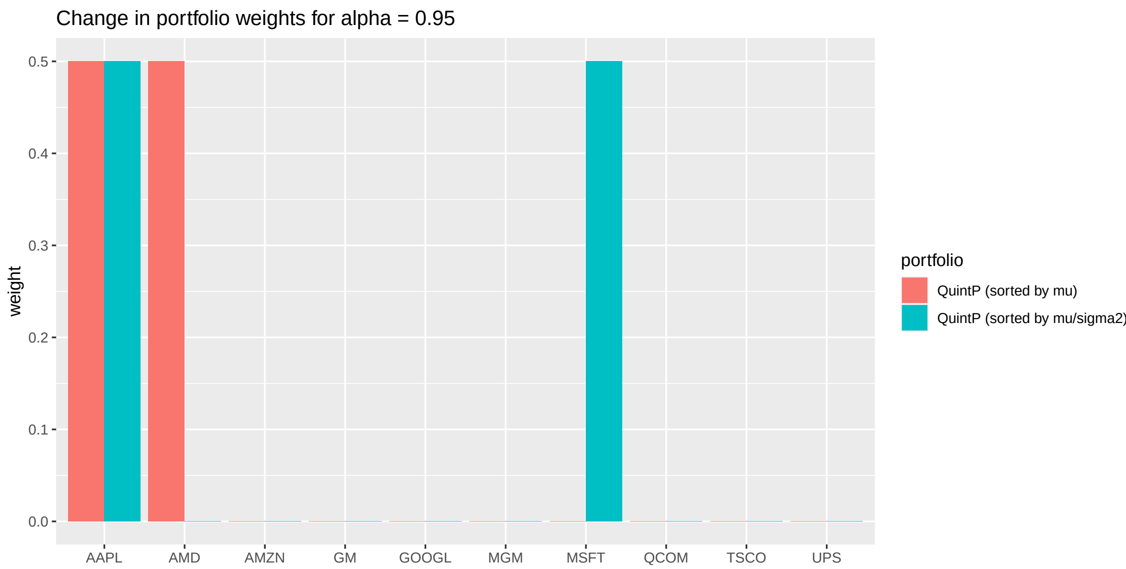


For educational purposes, however, we will proceed in the following with the manual calculation and plotting for the case of the simple static backtest.

```
w_QuintP_mu              <- QuintP_mu(list("prices" = stock_prices[1:T_trn]))
w_QuintP_mu_over_sigma2 <- QuintP_mu_over_sigma2(list("prices" = stock_prices[1:T_trn]))
w_all <- cbind("QuintP (sorted by mu)"        = w_QuintP_mu,
               "QuintP (sorted by mu/sigma2)" = w_QuintP_mu_over_sigma2)
```

c. Plot and compare the different portfolio allocations:

```
library(ggplot2)
library(reshape2)

data.frame("stocks" = names(stock_prices), w_all, check.names = FALSE) |>
  melt(id.vars = "stocks") |>
  ggplot(aes(x = stocks, y = value, fill = variable)) +
  geom_bar(stat="identity", position = "dodge") +
  labs("fill" = "portfolio") +
  labs(title = "Change in portfolio weights for alpha = 0.95", x = NULL, y = "weight")
```

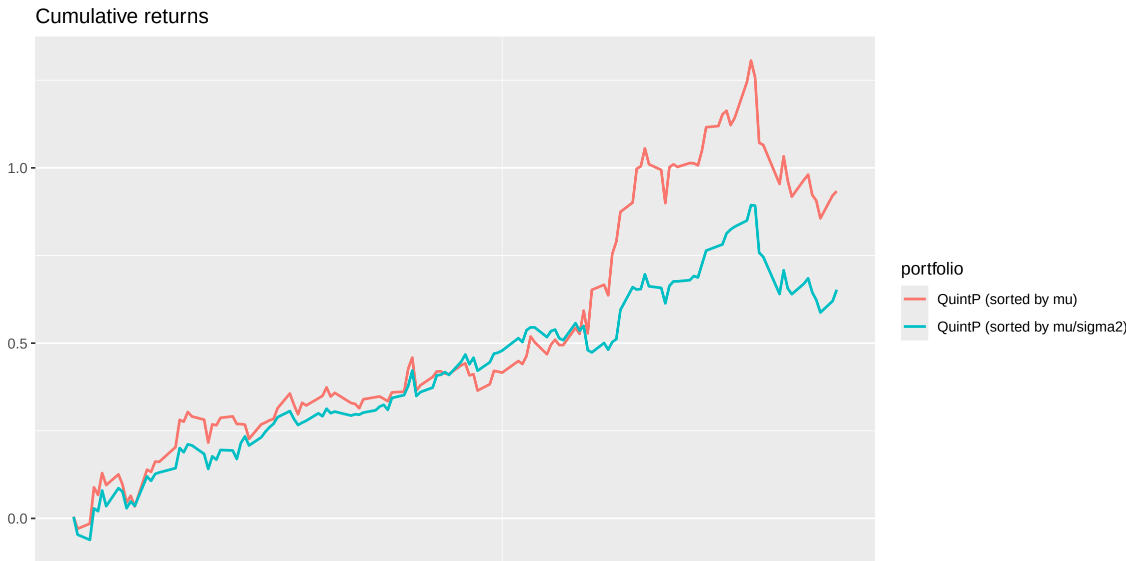


d. Using the remaining 30% of the data, assess the portfolios in terms of cumulative returns, volatility-adjusted cumulative returns, Sharpe ratio, and drawdown.

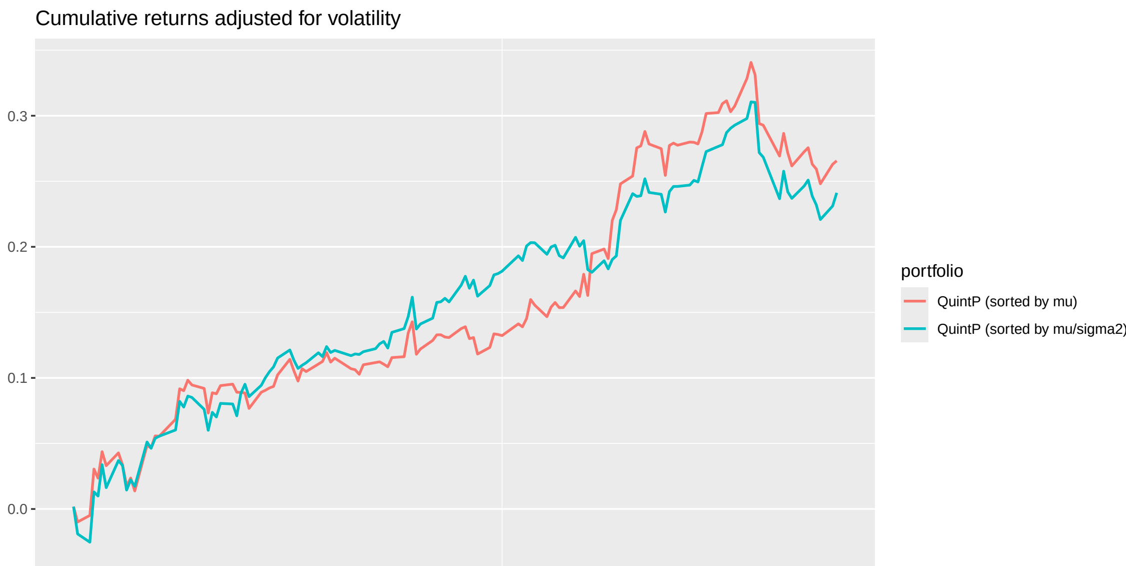
```
# Test data
stock_prices_tst <- stock_prices[-c(1:T_trn)]
X_tst <- stock_prices_tst/lag(stock_prices_tst) - 1

# Naive backtest (assuming daily rebalancing and no transaction cost)
portf_ret <- na.omit(xts(X_tst %*% w_all, index(X_tst)))
head(portf_ret)
```

```
           QuintP (sorted by mu) QuintP (sorted by mu/sigma2)
2020-03-19           0.005118845                  0.004398766
2020-03-20          -0.034383419                 -0.050526158
2020-03-23           0.015002953                 -0.015609147
2020-03-24           0.105165326                  0.095618189
2020-03-25          -0.019959001                 -0.007545154
2020-03-26           0.058469050                  0.057591404
```

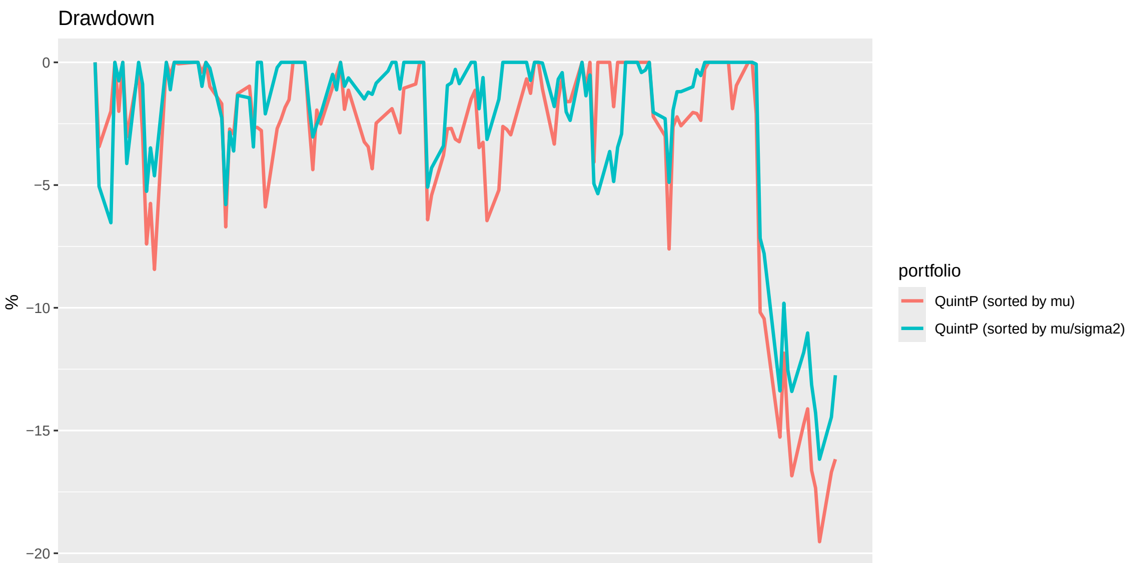
```
# Cumulative returns
cumreturns <- cumprod(1 + portf_ret) - 1

cumreturns |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 0.8) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Cumulative returns", x = NULL, y = NULL, color = "portfolio")
```




```
# Adjust returns by target volatility
target_vol <- 0.01  # target volatility
portf_vols <- apply(portf_ret, 2, sd)
portf_ret_vol_adj <- target_vol * sweep(portf_ret, 2, portf_vols, "/")
cumreturns_vol_adj <- cumprod(1 + portf_ret_vol_adj) - 1

cumreturns_vol_adj |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 0.8) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Cumulative returns adjusted for volatility",
       x = NULL, y = NULL, color = "portfolio")
```




```
# Drawdown
library(PerformanceAnalytics)

(100 * Drawdowns(portf_ret)) |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 1) +
  scale_x_date(date_breaks = "2 year", date_labels = "%Y") +
  labs(title = "Drawdown", x = NULL, y = "%", color = "portfolio")
```




```
# Compute Sharpe ratio
t(SharpeRatio.annualized(portf_ret, scale = 252))
```

```
                                Annualized Sharpe Ratio (Rf=0%)
QuintP (sorted by mu)                                  5.512187
QuintP (sorted by mu/sigma2)                           4.270759
```

Exercise 6.5: Risk-based portfolios

Repeat Exercise 6.4 with the following risk-based portfolios:

a. GMVP
b. IVolP
c. MDivP
d. MDecP

Solution

Market data corresponding to N stocks:

```r
library(xts)
library(pob)       # Market data used in the book

# Use data from package pob
data(SP500_2015to2020)
stock_prices <- SP500_2015to2020$stocks[
  "2019::",
  c("AAPL", "AMZN", "AMD", "GM", "GOOGL", "MGM", "MSFT", "QCOM", "TSCO", "UPS")
  ]
```

Using 70% of the data, compute the portfolios: GMVP, IVolP, MDivP, and MDecP:

```r
library(CVXR)

T <- nrow(stock_prices)
T_trn <- round(0.70*T)

# Define the portfolio functions
design_GMVP <- function(Sigma) {
  w <- Variable(nrow(Sigma))
  prob <- Problem(Minimize(quad_form(w, Sigma)),
                  constraints = list(w >= 0, sum(w) == 1))
  result <- solve(prob)
  w <- as.vector(result$getValue(w))
  return(w)
}

GMVP <- function(dataset, ...) {
  N <- ncol(dataset$prices)
  X <- diff(log(dataset$prices))[-1]
  Sigma <- cov(X)
  w <- design_GMVP(Sigma)
  return(w)
}
```

```r
design_IVolP <- function(Sigma) {
  sigma <- sqrt(diag(Sigma))
  w <- 1/sigma
  w <- w/sum(w)
  return(w)
}

IVolP <- function(dataset, ...) {
  N <- ncol(dataset$prices)
  X <- diff(log(dataset$prices))[-1]
  Sigma <- cov(X)
  w <- design_IVolP(Sigma)
  return(w)
}
```

```r
design_MSRP <- function(mu, Sigma) {
  w_ <- Variable(nrow(Sigma))
  prob <- Problem(Minimize(quad_form(w_, Sigma)),
                  constraints = list(w_ >= 0, t(mu) %*% w_ == 1))
  result <- solve(prob)
  w <- as.vector(result$getValue(w_)/sum(result$getValue(w_)))
  names(w) <- colnames(Sigma)
  return(w)
}

MDivP <- function(dataset, ...) {
  N <- ncol(dataset$prices)
  X <- diff(log(dataset$prices))[-1]
  Sigma <- cov(X)
  w <- design_MSRP(mu = sqrt(diag(Sigma)), Sigma)
  return(w)
}
```

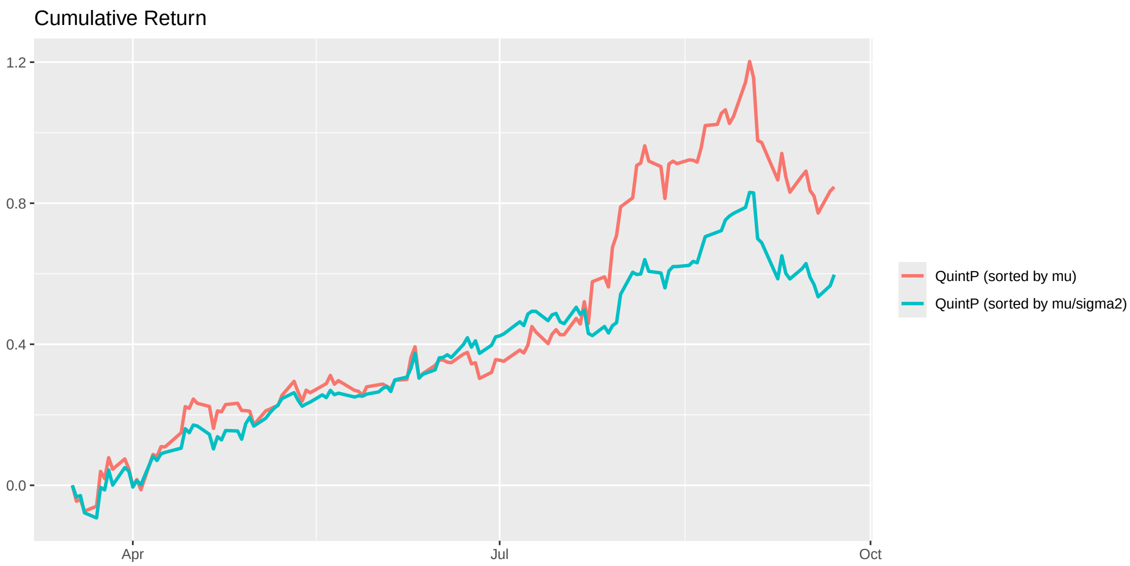
```r
design_MDecP <- function(Sigma) {
  C <- diag(1/sqrt(diag(Sigma))) %*% Sigma %*% diag(1/sqrt(diag(Sigma)))
  colnames(C) <- colnames(Sigma)
  return(design_GMVP(Sigma = C))
}

MDecP <- function(dataset, ...) {
  N <- ncol(dataset$prices)
  X <- diff(log(dataset$prices))[-1]
  Sigma <- cov(X)
  w <- design_MDecP(Sigma)
  return(w)
}
```

At this point, we can conveniently use the R package portfolioBactest to perform the backtest (either a simple static one or a rolling-window one or even multiple backtests), as well as to compute performance measures, and plot the results as follows:

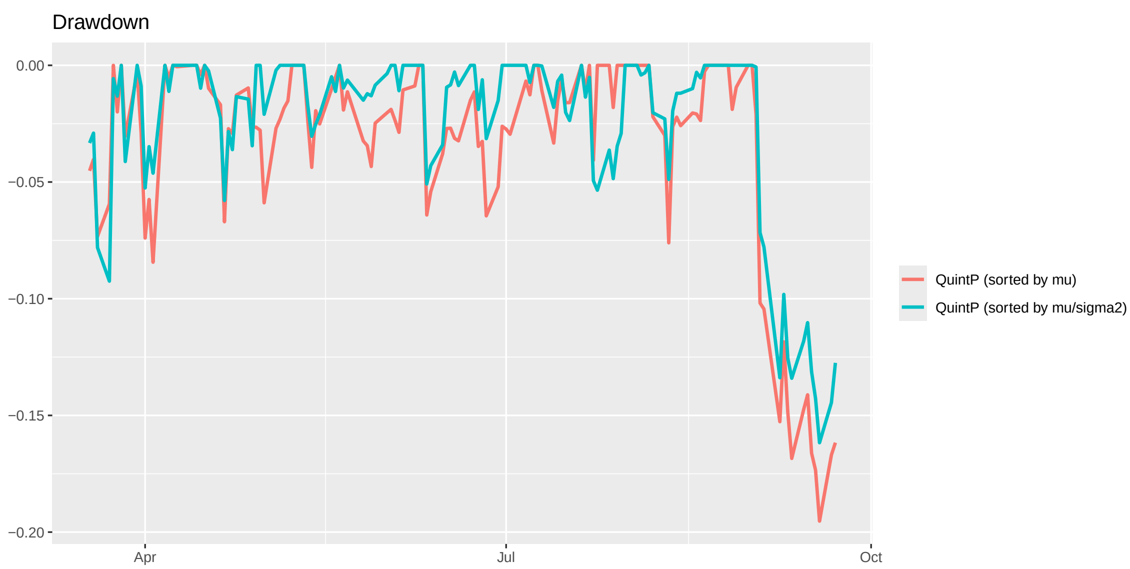
```r
# Backtest via the portfolioBacktest package
library(portfolioBacktest)

bt <- portfolioBacktest(
  portfolio_funs = list("QuintP (sorted by mu)"        = QuintP_mu,
                        "QuintP (sorted by mu/sigma2)" = QuintP_mu_over_sigma2),
  dataset_list = list("dataset1" = list("prices" = stock_prices)),
  lookback = T_trn, optimize_every = 10000, rebalance_every = 1
  )

# Print some performance measures
bt_summary <- backtestSummary(bt)
summaryTable(bt_summary, measures = c("Sharpe ratio", "max drawdown"))
```

```
                             Sharpe ratio max drawdown
QuintP (sorted by mu)                2.73         0.20
QuintP (sorted by mu/sigma2)         2.52         0.16
```

```r
# Plots
backtestChartCumReturn(bt)
```




```r
backtestChartDrawdown(bt)
```

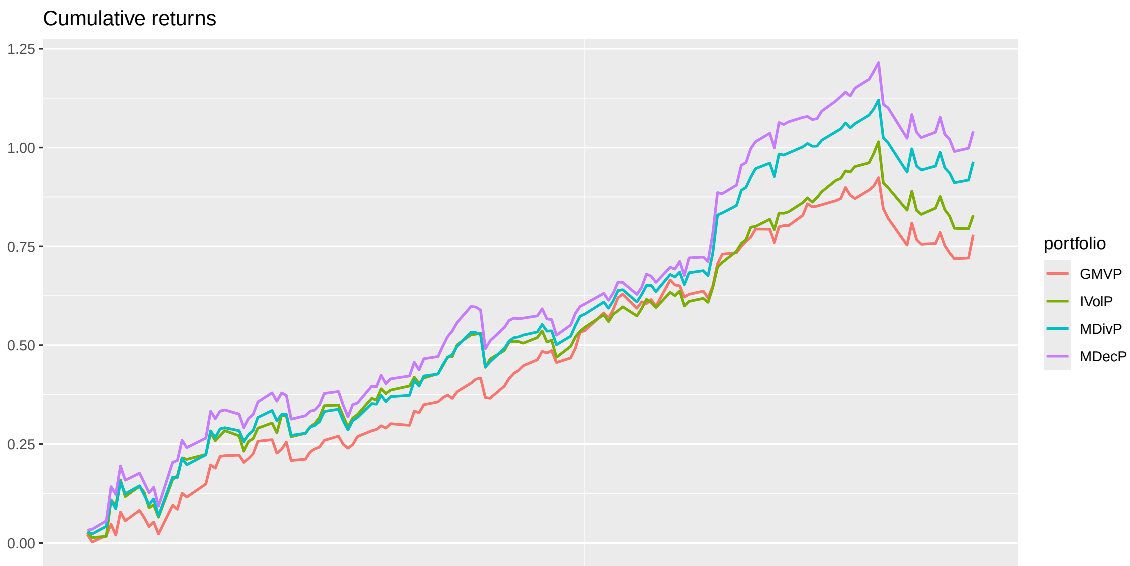


For educational purposes, we now perform each calculation manually for the case of the simple static backtest:

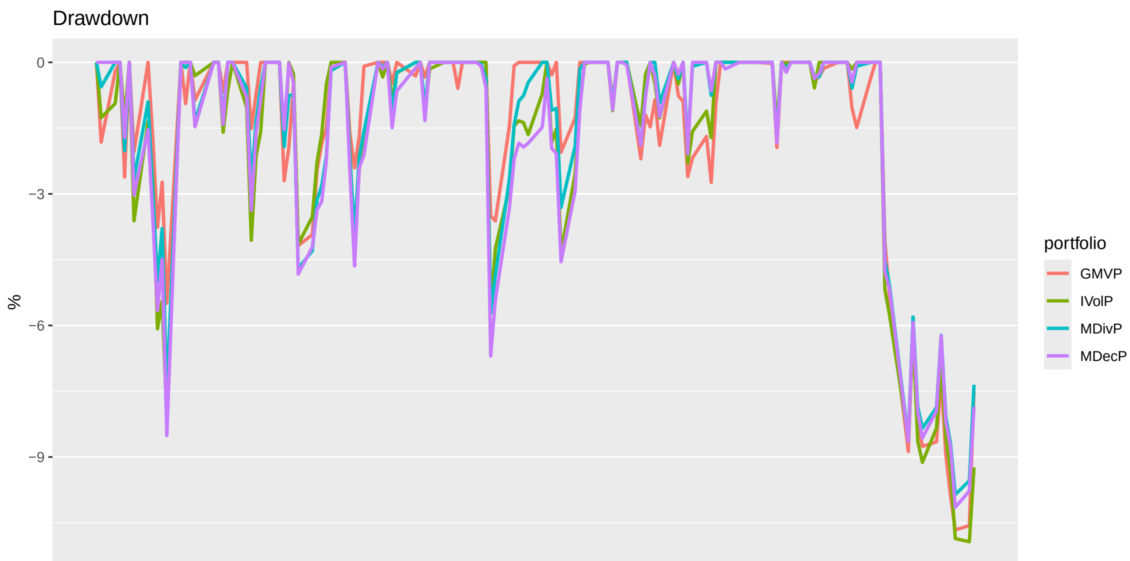
```r
# Calculate the portfolios
w_GMVP  <- GMVP(list("prices" = stock_prices[1:T_trn]))
w_IVolP <- IVolP(list("prices" = stock_prices[1:T_trn]))
w_MDivP <- MDivP(list("prices" = stock_prices[1:T_trn]))
w_MDecP <- MDecP(list("prices" = stock_prices[1:T_trn]))
w_all <- cbind("GMVP"   = w_GMVP,
               "IVolP"  = w_IVolP,
               "MDivP"  = w_MDivP,
               "MDecP"  = w_MDecP)

# Test data
stock_prices_tst <- stock_prices[-c(1:T_trn)]
X_tst <- stock_prices_tst/lag(stock_prices_tst) - 1

# Naive backtest (assuming daily rebalancing and no transaction cost)
portf_ret <- na.omit(xts(X_tst %*% w_all, index(X_tst)))
head(portf_ret)
```

```
                  GMVP        IVolP        MDivP        MDecP
2020-03-19  0.02094994  0.026315748  0.028528061  0.032118352
2020-03-20 -0.01823261 -0.012608456 -0.005595404  0.002282527
2020-03-23  0.01647562  0.003282223  0.018623227  0.020424825
2020-03-24  0.02764933  0.090446659  0.063819024  0.082148370
2020-03-25 -0.02617099 -0.012841624 -0.020134447 -0.017177485
2020-03-26  0.05707912  0.059104034  0.063526198  0.064064395
```

```r
# Cumulative returns
cumreturns <- cumprod(1 + portf_ret) - 1

cumreturns |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 0.8) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Cumulative returns", x = NULL, y = NULL, color = "portfolio")
```




```
# Drawdown
library(PerformanceAnalytics)

(100 * Drawdowns(portf_ret)) |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 1) +
  scale_x_date(date_breaks = "2 year", date_labels = "%Y") +
  labs(title = "Drawdown", x = NULL, y = "%", color = "portfolio")
```




```
# Compute Sharpe ratio
t(SharpeRatio.annualized(portf_ret, scale = 252))
```

```
      Annualized Sharpe Ratio (Rf=0%)
GMVP                         7.214367
IVolP                        6.729882
MDivP                        8.290347
MDecP                        8.389598
```