Empirical Finance: Assignment 6

By Group 15

Lanlan Hou	2801069
Rick van der Ploeg	2782774
Sebastiaan van Dijen	2769505
Wa Chak Sou	2796840

School of Business and Economics

8th December 2025

QUESTION 1: Mean-variance optimization

a. Optimal portfolio weights based on a Mean-variance optimization

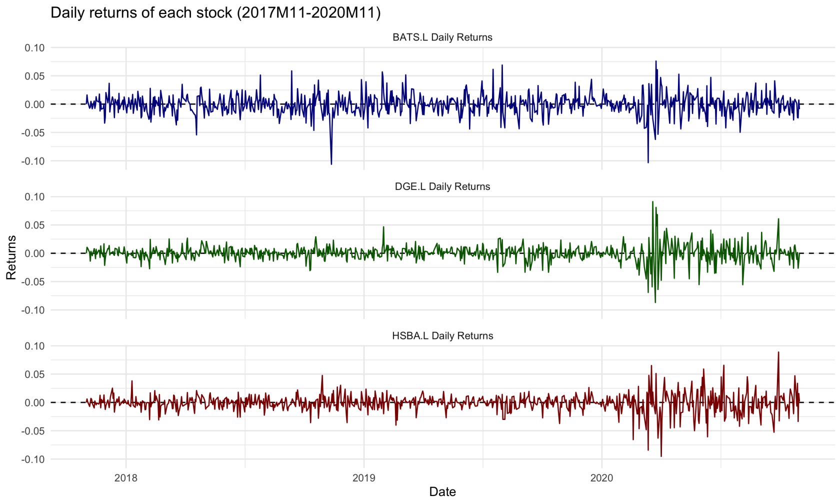
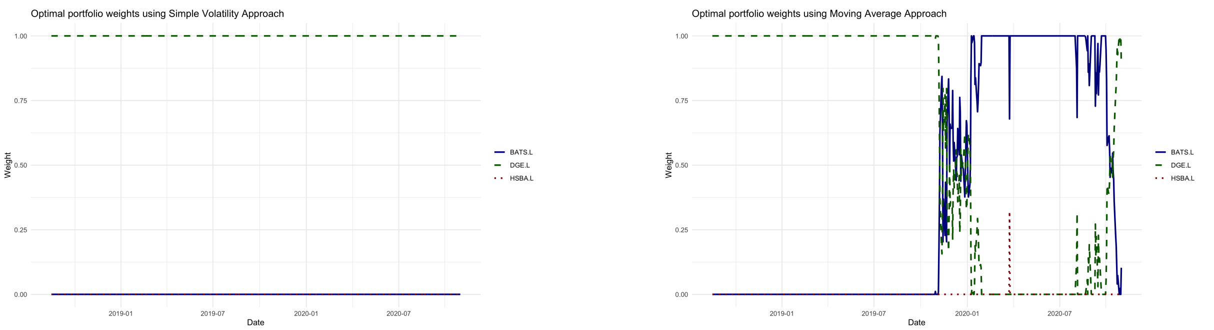


Figure 1.1.1 The daily returns of each stock within last three years.



(a) Simple Volatility

(b) Moving Average

Figure 1.1.2 The optimal portfolio weights based on a Mean-Variance Optimization using different volatility approaches

Figure 1.1.1 demonstrates the daily returns of the three selected stocks: **HSBA.L**, **DEO.L** and **BATS.L**, within the period November 2017 to November 2020. All three series fluctuate around zero and display volatility clustering, particularly toward the end of the sample. For the final three years, obtain optimal portfolio weights using mean–variance optimization under full-investment and no-short-selling constraints. The risk-aversion parameter is set to $\gamma = 2$, and the estimation window is $W_E = 200$ trading days, which balances responsiveness with statistical stability. Figure 1.1.2 reveals the optimal portfolio weights of each stock (darkblue solid line: BATS.L; darkgreen dashed line: DGE.L; darkred dotted line: HSBA.L) using two different volatility modelling methods: (a) *Simple Volatility* and (b) *Moving Average*. In the *Simple Volatility* approach, the covariance matrix at each date is estimated using all past data up to $t-1$. Since this approac varies very slowly over time, the optimization allocates almost the entire portfolio to **DGE.L**, which has the best long-run balance in return and risk over the other two stocks. Using the *Moving Average* volatility approach, the mean vector and covariance matrix are computed from the most recent $W_E = 200$ observations only. Consequently, the portfolio weights become time-varying and the allocation gradually moves away from **DGE.L**, shifts toward **BATS.L** after 2019. The occasional small weights assigned to **HSBA.L** when recent data make it relatively attractive.

b. Sharpe ratio of the optimal portfolio

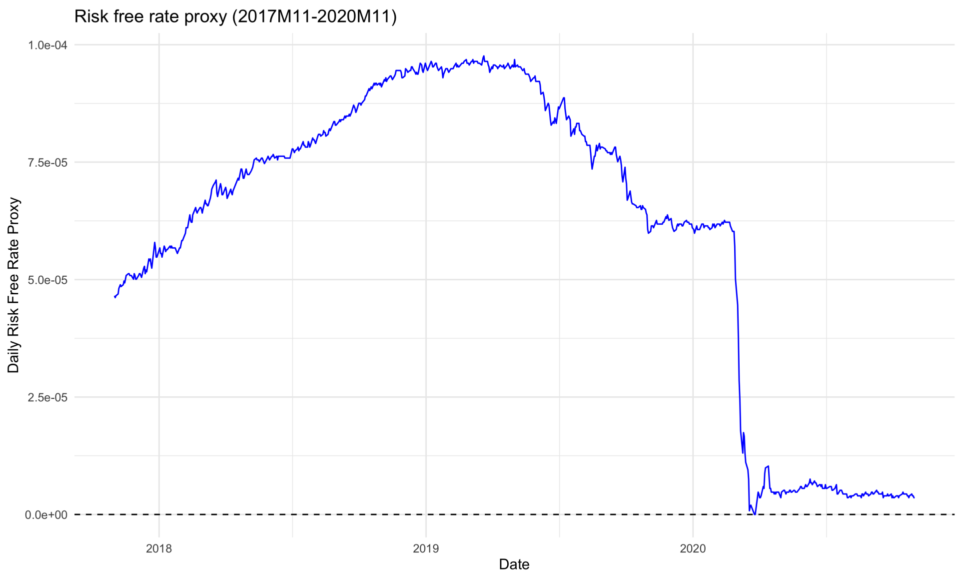
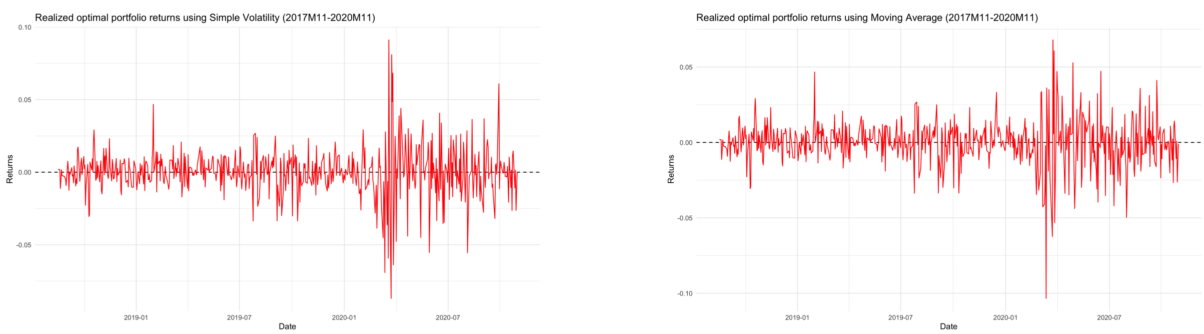


Figure 1.2.1 The appropriate risk-free rate within the last three years.



(a) Simple Volatility

(b) Moving Average

Figure 1.2.2 The realized portfolio returns using different volatility approaches

Volatility Estimation Method	**Sharp-Ratio**
Simple Volatility	-0.00996
Moving Average	-0.01158

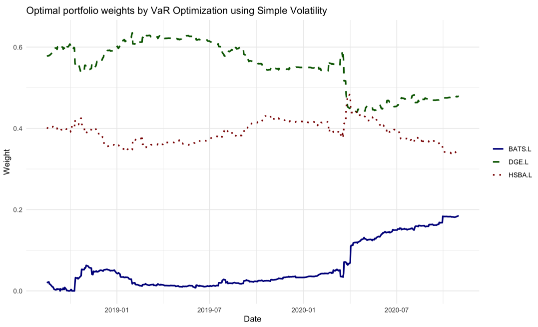
Table 1.2.1: Sharpe Ratio using different volatility estimation methods

Figure 1.2.1 plots the resulting daily risk-free rate. It shows a steady decline followed by a sharp drop in early 2020 due to the COVID-19 crisis. Figures 1.2.2 shows the realized returns of the optimal portfolios obtained from the *Simple Volatility* and *Moving Average* methods. The portfolio using *Moving Average* indicates more volatile returns because its weights adjust more rapidly to short-term changes in estimated volatility and returns.

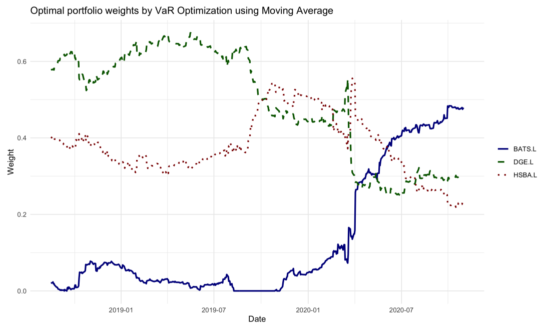
Table 1.2.1 reports the correponding Sharpe ratios for two volatility estimation methods. Both portfolios have negative Sharpe ratios, with *Simple Volatility* at –0.00996 and *Moving Average* at –0.01158. It indicates neither optimized portfolio delivered positive returns. The *Moving Average* approach performs slightly worse due to its higher return volatility and the turbulent market conditions during 2019–2020.

Question 2: Value-at-Risk optimization

a. Optimal portfolio weights by directly minimizing the Value-at-Risk(0.05)



(a) Simple Volatility



(b) Moving Average

Figure 2.1.1 The optimal portfolio weights based on a Value-at-Risk Optimization using different volatility approaches

Alongside with the Mean-Variance Optimization, VaR Optimization which minimizing the parametric VaR(0.05) of the portfolio is being applied to obtain the optimal portfolio weights. VaR is estimated with different volatility estimation methods, again based on the most recent $W_E = 200$ observations. The optimization is subject to full investment and no short-selling. VaR minimization is solved using constrained non-linear optimization rather than quadratic programming since it includes a square-root term.

Figure 2.1.1 shows the optimal portfolio weights based on VaR Optimization using two different volatility modelling methods: (a) *Simple Volatility* and (b) *Moving Average*. In *Simple Volatility* approaches, **DGE.L** receives the largest share, with **HSBA.L** and **BATS.L** holding smaller but gradually increasing positions. In *Moving Average* approaches, it provides much more time-varying allocations. The estimator reacts quickly to shifts in recent volatility during 2020 and the weight in **BATS.L** increases substantially while **DGE.L** and **HSBA.L** decline.

Compared with the optimal portfolios based on Mean–Variance Optimization in Question 1(a), the optimal portfolios based on VaR Optimization are less concentrated and maintain positive weights in multiple assets. This reflects that VaR Optimization focuses on downside risk, leading to more diversified allocations and stronger reactions to changes in recent volatility conditions.

b. Violation ratio and Sharpe ratio

Volatility Estimation Method	Violation Ratio	Sharpe Ratio
Simple Volatility	1.03757	-0.04779
Moving Average	0.89445	-0.05557

Table 2.2.1: Violation and Sharpe Ratio using different volatility estimation methods based on VaR Optimization

Table 2.2.1 reveals the Violation ratios and Sharpe ratios for the VaR Optimization portfolios under the two volatility estimation methods. For the Violation ratio, the *Simple Volatility* method results a violation ratio of 1.04, which is very close to the theoretical value. While the *Moving Average* method obtains 0.89, which means it produces slightly fewer violations than expected. Therefore, it is somehow more conservative.

For Sharpe ratio, both portfolios have negative Sharpe ratios, with *Simple Volatility* at –0.0478 and *Moving Average* at –0.0556. These indicate that neither strategy generated positive returns over the period. The *Moving Average* portfolio performs slightly worse, which is consistent with its more sensitive weight adjustments and higher realized volatility during the COVID-19 crisis.

Appendix

```r
# Load required packages
library(openxlsx)    # Excel file handling
library(lubridate)   # Date/time utilities
library(quantmod)    # Financial data & modeling
library(quadprog)    # Quadratic programming
library(tidyr)       # Data reshaping
library(dplyr)       # Data manipulation
library(tidyquant)   # Tidy interface for financial data
library(rugarch)     # GARCH/AR-GARCH models
library(nloptr)      # Nonlinear optimization
library(ggplot2)     # Data visualization

raw <- read.xlsx('student_groups_stocks_plus3.xlsx', sheet = 1)

groupNumber <- 15
startDate <- raw$Start.Date[groupNumber]
endDate <- raw$`End.Date.(+10y)`[groupNumber]
stocks_list <- c('HSBA.L','DGE.L','BATS.L')
# stocks_list <- c('HSBC','DEO','BTI') # USD

originalDate <- as.Date(startDate)
startDate_LastThreeYears <- originalDate %m+% years(7)

stockPrices <- tq_get(stocks_list,
                      from = startDate_LastThreeYears, to = endDate,
                      get = "stock.prices")

# Calculate daily returns
all_returns <- stockPrices %>%
  group_by(symbol) %>%
  tq_transmute(select = adjusted,
               mutate_fun = periodReturn,
               period = "daily",
               col_rename = "returns")

# Reshape dataset to wide
separate_returns <- all_returns %>%
  pivot_wider(
    names_from = symbol,  # The column that holds the new column names
    values_from = returns     # The column that holds the values to be spread
  )

custom_colors <- c('darkred','darkgreen','darkblue')  # List of colors
names(custom_colors) <- stocks_list        # Assign stock names to the colors

# Define the plot with facet_wrap to split by 'symbol' and remove legend
dailyReturnsPlot <-
  ggplot(all_returns,
         aes(x = date, y = returns, color = symbol, linetype = symbol)) +
  geom_hline(yintercept = 0, color = 'black',
             linewidth = 0.5, linetype = 'dashed') +
  geom_line() +  # Draw lines for each stock
```

```
  labs(title = 'Daily returns of each stock (2017M11-2020M11)',
       x = "Date", y = "Returns") +
  theme_minimal() +
  scale_color_manual(values = custom_colors) +
  scale_linetype_manual(values = c('solid', 'solid', 'solid')) +
  theme(legend.position = "none") +  # Remove legend
  facet_wrap(~ symbol, ncol = 1,
             labeller = labeller(symbol = function(x) paste(x, "Daily Returns")))  # Customize facet labels

# Export the plot
ggsave('figures/a_returns_plot.png', plot = dailyReturnsPlot,
       width = 10, height = 6, dpi = 300)
print(dailyReturnsPlot)

# Prepare the portfolio weight optimization
createResultDataFrame <- function(input_returns){
  # Create a new data frame with the specified dates and empty weight columns
  result_simple <- data.frame(
    date = input_returns$date[start_index:end_index])

  # Add empty columns for each stock name dynamically
  for (name in stocks_list) {
    result_simple[[name]] <- NA  # Initialize with NA for each stock
  }
  return(result_simple)
}

meanVarianceOptimization <- function(input_approach,
                                     input_stocksList,
                                     input_window,
                                     input_returns,
                                     input_gamma){
  # Create a new data frame with the specified dates and empty weight columns
  result_simple <- createResultDataFrame(input_returns = input_returns)

  # Obtain portfolio weights using selected approach
  for (i in start_index:end_index) {
    if (input_approach == 'simpleVolatility'){
      data_subset <- input_returns[1:(i-1), -1]
    } else if (input_approach == 'movingAverage') {
      data_subset <- input_returns[(i-input_window):(i-1), -1]
    }

    # Obtain mean and variance-covariance
    dvec = colMeans(data_subset, na.rm = TRUE)
    Dmat = input_gamma * cov(data_subset, use = "pairwise.complete.obs")

    # Compute and store weights
    outcome <- solve.QP(Dmat, dvec, Amat, bvec, meq)
    result_simple[(i-input_window),2:(K+1)] <- outcome$solution
  }

  # Plot the optimal weights
  result_expanded <- result_simple %>%
```

```
    pivot_longer(cols = input_stocksList,
                 names_to = "Stock",
                 values_to = "Weight")

  return(list(singleTable = result_expanded,
              expandedTable = result_simple))
}

portfolioPlot <- function(input_weightsTable,
                          input_title,
                          input_path){
  resultPlot <- ggplot(input_weightsTable,
         aes(x = date, y = Weight, color = Stock, linetype = Stock)) +
  geom_line(linewidth=1) +  # Draw lines for each stock
  labs(title = input_title,
       x = "Date", y = "Weight") +
  theme_minimal() +  # Use a minimal theme
  scale_color_manual(values = custom_colors) +
  scale_linetype_manual(values = c('solid', 'dashed', 'dotted')) +
  theme(legend.title = element_blank())  # Remove legend title
  #Export the plot
  ggsave(input_path, plot = resultPlot,
         width = 10, height = 6, dpi = 300)

  return(resultPlot)
}

# Estimation window length
W=200

# Extract the desired date range (from row W+1 to the end)
start_index = W+1
end_index <- nrow(separate_returns)

# Number of assets
K=length(separate_returns)-1

# Create Amat constraint matrix
first_row <- matrix(1, nrow = 1, ncol = K)
identity_matrix <- diag(K)
Amat <- t(rbind(first_row, identity_matrix))

# Create bvec vector holding the constraint values of object b to be
   optimized
bvec <- c(1, rep(0, K))

# meq: First how many constraints are equality constraints (sum of
   weights = 1)
meq <- 1

# Gamma
gamma = 2

SV_portfolio <- meanVarianceOptimization(
  input_approach = 'simpleVolatility',
  input_stocksList = stocks_list,
  input_window = W,
  input_returns = separate_returns,
```

```
  input_gamma = gamma)
MA_portfolio <- meanVarianceOptimization(
  input_approach = 'movingAverage',
  input_stocksList = stocks_list,
  input_window = W,
  input_returns = separate_returns,
  input_gamma = gamma)

simpleVolatility <- portfolioPlot(
  input_weightsTable = SV_portfolio$singleTable,
  input_title = 'Optimal portfolio weights using Simple Volatility
     Approach',
  input_path = 'figures/a_SV_optimizationPlot.png')
print(simpleVolatility)

movingAverage <- portfolioPlot(
  input_weightsTable = MA_portfolio$singleTable,
  input_title = 'Optimal portfolio weights using Moving Average
     Approach',
  input_path = 'figures/a_MA_optimizationPlot.png')
print(movingAverage)
```

Figure 1: Question 1(a)

```
# Import the rf rate from csv file
rfrate_data <- read.csv("risk_free_rate_proxy.csv")
rfrate_data <- na.omit(rfrate_data)
rfrate_data$date <- as.Date(rfrate_data$date, format = "%d/%m/%Y")

# Extract the rf with the last three years
rfrate_data <- subset(rfrate_data, date >= startDate_LastThreeYears &
    date <= endDate)

# Convert to daily returns (yield in percent per annuum) and add to df
    of realized returns
rfrate_data$rfrate_daily <- ((1+rfrate_data$rfrate/100)^(1/252)-1)

# Plot the rate
riskFreeRate_plot <- ggplot(rfrate_data, aes(x = date,y=rfrate_daily))
   +
  geom_hline(yintercept = 0, color = 'black',
             linewidth = 0.5, linetype = 'dashed') +
  geom_line(color = 'blue', linetype = "solid", linewidth=.5) +
  labs(title = 'Risk free rate proxy (2017M11-2020M11)',
       x = "Date", y = "Daily Risk Free Rate Proxy") +
  theme_minimal() +  # Use a minimal theme
  theme(legend.title = element_blank())  # Remove legend title
# Export the plot
ggsave('figures/b_riskFreeRate_plot.png', plot = riskFreeRate_plot,
       width = 10, height = 6, dpi = 300)
print(riskFreeRate_plot)

computeRealizedReturns <- function(input_stocksList,
                                   input_returns,
                                   input_portfolioWeighted){
  # Align by date (inner join)
  merged <- merge(input_portfolioWeighted, input_returns,
                  by = 'date', suffixes = c("_w", "_r"))

```

```
  # Extract weight and return matrices in same stock order
  W <- as.matrix(merged[ paste0(input_stocksList, "_w") ])
  R <- as.matrix(merged[ paste0(input_stocksList, "_r") ])

  # Portfolio return at each date: sum_j w_{t,j} * r_{t,j}
  portfolioReturns <- rowSums(W * R)

  result <- data.frame(date = merged$date,
                       realized_returns = portfolioReturns)

  return(result)
}

realizedReturnsPlot <- function(input_portfolioReturns,
                                input_title,
                                input_path){
  realized <-
    ggplot(input_portfolioReturns,
                   aes(x = date, y = realized_returns)) +
    geom_hline(yintercept = 0, color = 'black',
             linewidth = 0.5, linetype = 'dashed') +
    geom_line(color = 'red', linetype = "solid", linewidth=0.5) +
    labs(title = input_title,
       x = "Date", y = "Returns") +
    theme_minimal() +  # Use a minimal theme
    theme(legend.title = element_blank())  # Remove legend title
    # Export the plot
    ggsave(input_path, plot = realized,
           width = 10, height = 6, dpi = 300)
  return(realized)
}

computeSharpRatio <- function(input_portfolioReturns){
  # Merge datasets
  merged <- merge(input_portfolioReturns,
                  rfrate_data %>% select(date, rfrate_daily),
                  by="date", all.x=TRUE)

  # Compute the spread between portfolio and treasury returns
  merged$spread <- (merged$realized_returns - merged$rfrate_daily)

  # Compute Sharp Ratio
  sharpRatio = mean(merged$spread,na.rm=TRUE) / sd(merged$spread,na.rm=TRUE)

  return(sharpRatio)
}

# Create new dataframe with date and calculated portfolio returns
portfolio_returns_SV <- computeRealizedReturns(
  input_stocksList = stocks_list,
  input_returns = separate_returns,
  input_portfolioWeighted = SV_portfolio$expandedTable)

portfolio_returns_MA <- computeRealizedReturns(
  input_stocksList = stocks_list,
  input_returns = separate_returns,
  input_portfolioWeighted = MA_portfolio$expandedTable)
```

```
# Plot the realized
realized_SV <- realizedReturnsPlot(
  input_portfolioReturns = portfolio_returns_SV,
  input_title = 'Realized optimal portfolio returns using Simple
     Volatility (2017M11-2020M11)',
  input_path = 'figures/b_realized_SV_plot.png')
print(realized_SV)

# Plot the realized
realized_MA <- realizedReturnsPlot(
  input_portfolioReturns = portfolio_returns_MA,
  input_title = 'Realized optimal portfolio returns using Moving
     Average (2017M11-2020M11)',
  input_path = 'figures/b_realized_MA_plot.png')
print(realized_MA)

SR_SV <- computeSharpRatio(portfolio_returns_SV)
print(paste('The Sharp Ratio using Simple Volatility Approach is: ',
            round(SR_SV,5)))
SR_MA <- computeSharpRatio(portfolio_returns_MA)
print(paste('The Sharp Ratio using Moving Average Approach is: ',
            round(SR_MA,5)))
```

Figure 2: Question 1(b)

```
valueAtRisk_Parametric <- function(input_portfolioWeights,
                                   input_assetsMean,
                                   input_covarianceMatrix,
                                   input_p) {
  portfolioMean <- sum(input_portfolioWeights * input_assetsMean)
  portfolioSTDEV <- sqrt(as.numeric(t(input_portfolioWeights) %*% input
     _covarianceMatrix %*% input_portfolioWeights))
  VaR_returns <- -portfolioSTDEV*qnorm(input_p) - portfolioMean
  return(VaR_returns)
}

VaRMinimization <- function(input_dvec, input_Dmat) {
  K <- length(input_dvec)

  # Objective: minimize VaR(0.05)
  eval_f <- function(porfolioWeights) {
    valueAtRisk_Parametric(input_portfolioWeights = porfolioWeights,
                           input_assetsMean = input_dvec,
                           input_covarianceMatrix = input_Dmat,
                           input_p = 0.05)
  }

  # Equality constraint: sum(w) = 1
  eval_g_eq <- function(porfolioWeights) {
    sum(porfolioWeights) - 1
  }

  # No short-selling
  lowerBound <- rep(0, K)
  upperBound <- rep(1, K)

  result <- nloptr(
    x0 = rep(1/K, K),          # start from equal weights
```

```
    eval_f = eval_f, eval_g_eq = eval_g_eq,
    lb = lowerBound, ub = upperBound,
    opts = list(algorithm = "NLOPT_LN_COBYLA",
                maxeval = 1000, xtol_rel = 1e-6))

  return(as.numeric(result$solution))
}

VaROptimization <- function(input_approach,
                            input_stocksList,
                            input_window,
                            input_returns,
                            input_gamma) {
  # Create a new data frame with the specified dates and empty weight columns
  result_simple <- createResultDataFrame(input_returns = input_returns)
  VaR_list <- numeric(length(input_returns))

  for (i in start_index:end_index) {
    if (input_approach == 'simpleVolatility'){
      data_subset <- input_returns[1:(i-1), -1]
    } else if (input_approach == 'movingAverage') {
      data_subset <- input_returns[(i-input_window):(i-1), -1]
    }

    # Obtain mean and variance-covariance
    dvec <- colMeans(data_subset, na.rm = TRUE)
    Dmat <- input_gamma * cov(data_subset, use = "pairwise.complete.obs")

    # Compute and store weights
    outcome <- VaRMinimization(dvec, Dmat)
    VaR_list[i-input_window] <- valueAtRisk_Parametric(
      input_portfolioWeights = outcome,
      input_assetsMean = dvec,
      input_covarianceMatrix = Dmat,
      input_p = 0.05)
    result_simple[(i-input_window), 2:(length(input_stocksList)+1)] <- outcome
  }

  # Plot the optimal weights
  result_expanded <- result_simple %>%
    pivot_longer(cols = input_stocksList,
                 names_to = "Stock",
                 values_to = "Weight")

  return(list(singleTable = result_expanded,
              expandedTable = result_simple,
              VaR = VaR_list))
}

VaR_portfolio_SV <- VaROptimization(input_approach = 'simpleVolatility',
                                    input_stocksList = stocks_list,
                                    input_window = W,
                                    input_returns = separate_returns,
                                    input_gamma = gamma)
```

```

VaR_portfolio_MA <- VaROptimization(input_approach = 'movingAverage',
                                    input_stocksList = stocks_list,
                                    input_window = W,
                                    input_returns = separate_returns,
                                    input_gamma = gamma)

VaR_weights_plot_SV <- portfolioPlot(
  input_weightsTable = VaR_portfolio_SV$singleTable,
  input_title = 'Optimal portfolio weights by VaR Optimization using
     Simple Volatility',
  input_path  = 'figures/2a_VaR_optimization_SV_plot.png'
)

VaR_weights_plot_MA <- portfolioPlot(
  input_weightsTable = VaR_portfolio_MA$singleTable,
  input_title = 'Optimal portfolio weights by VaR Optimization using
     Moving Average',
  input_path  = 'figures/2a_VaR_optimization_MA_plot.png'
)

print(VaR_weights_plot_SV)
print(VaR_weights_plot_MA)
```

Figure 3: Question 2(a)

```
computeViolationRatio <- function(input_probability,
                                  input_testWindow,
                                  input_VaR, input_realized){
  expectedViolation <- input_probability * input_testWindow
  eta <- numeric(input_testWindow)
  for(i in 1:input_testWindow){
    if(input_realized[i] <= -input_VaR[i]){
      eta[i] <- 1
    } else {
      eta[i] <- 0
    }
  }
  violationRatio <- sum(eta == 1) / expectedViolation

  return(list(eta=eta, violationRatio=violationRatio))
}

portfolio_returns_VaR_SV <- computeRealizedReturns(
  input_stocksList       = stocks_list,
  input_returns          = separate_returns,
  input_portfolioWeighted = VaR_portfolio_SV$expandedTable
)

portfolio_returns_VaR_MA <- computeRealizedReturns(
  input_stocksList       = stocks_list,
  input_returns          = separate_returns,
  input_portfolioWeighted = VaR_portfolio_MA$expandedTable
)

vr_VaROptimization_SV <- computeViolationRatio(
  input_probability = 0.05,
  input_testWindow = length(portfolio_returns_VaR_SV$date),
  input_VaR = VaR_portfolio_SV$VaR,
```

```
  input_realized = portfolio_returns_VaR_SV$realized_returns)

vr_VaROptimization_MA <- computeViolationRatio(
  input_probability = 0.05,
  input_testWindow = length(portfolio_returns_VaR_MA$date),
  input_VaR = VaR_portfolio_MA$VaR,
  input_realized = portfolio_returns_VaR_MA$realized_returns)

vr_VaROptimization_SV$violationRatio
print(paste('The Violation Ratio using Simple Volatility Approach is: '
    ,
            round(vr_VaROptimization_SV$violationRatio,5)))
vr_VaROptimization_MA$violationRatio
print(paste('The Violation Ratio using Moving Averge Approach is: ',
            round(vr_VaROptimization_MA$violationRatio,5)))

SR_VaR_SV <- computeSharpRatio(portfolio_returns_VaR_SV)
print(paste('The Sharp Ratio using Simple Volatility Approach is: ',
            round(SR_VaR_SV,5)))
SR_VaR_MA <- computeSharpRatio(portfolio_returns_VaR_MA)
print(paste('The Sharp Ratio using Moving Average Approach is: ',
            round(SR_VaR_MA,5)))
```

Figure 4: Question 2(b)