PortfolioConstruction

Reng Chiz Der

2024-03-05

Portfolio Construction Test

Please note that the portfolio used are the same across the two tasks (bootstrap and portfolio) ### Environment Setup

```r
library(quadprog)
library(ROI)
library(ROI.plugin.glpk)
library(ROI.plugin.quadprog)
library(ROI.plugin.symphony)
library(quantmod)
library(zoo)
library(xts)
library(ggplot2)
library(PerformanceAnalytics)
library(PortfolioAnalytics)
```

Getting data

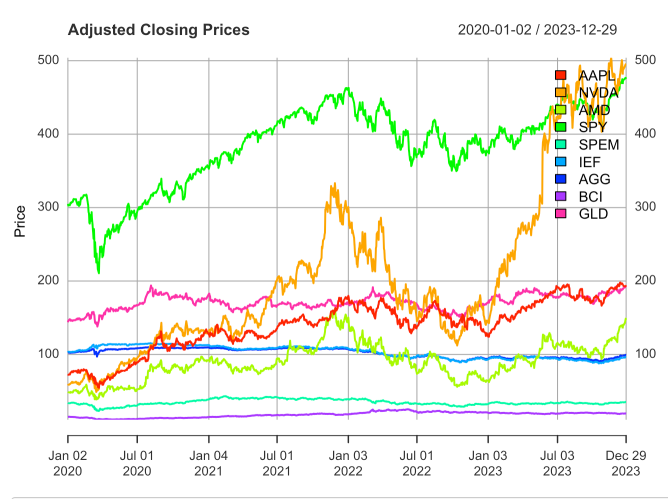
```r
fetch_and_preprocess <- function(tickers) {
  stocks <- lapply(tickers, function(ticker) {
    stock_data <- getSymbols(ticker, src = 'yahoo', from = "2020-01-01", to = "2024-01-01", auto.assign = FALSE)
    colnames(stock_data) <- gsub(".*\\.", "", colnames(stock_data)) # Remove the prefix
    adjusted_data <- stock_data[, "Adjusted", drop = FALSE] # Select only the adjusted closing value
    if ("Adjusted" %in% colnames(stock_data)) {
      colnames(adjusted_data) <- ticker
      return(adjusted_data)
    } else {
      stop("Adjusted closing value not found for", ticker)
    }
  })

  # Preprocess the data
  for (i in seq_along(stocks)) {
    if (!is.null(stocks[[i]])) {
      # Calculate the number of missing values
      missing_values <- colSums(is.na(stocks[[i]]))

      # Replace missing values with the previous day's values
      stocks[[i]] <- na.locf(stocks[[i]])
    }
  }

  # Combine the data into a single xts object
  combined_data <- do.call(merge, stocks)
  return(combined_data)
}

# Example usage:
stocks <- c('AAPL', 'NVDA', 'AMD')
stock_data <- fetch_and_preprocess(stocks)

# S&P500, Emerging Market ETF
etfs <- c('SPY', 'SPEM')
etf_data <- fetch_and_preprocess(etfs)

# fixed income (bonds): IEF (treasury bond ETF), AGG (US Aggregate Bond ETF)
fis <- c('IEF', 'AGG')
fi_data <- fetch_and_preprocess(fis)

comms <- c('BCI')
comm_data <- fetch_and_preprocess(comms)

# SPDR  Gold Share ETF
gold <- c('GLD')
gold_data <- fetch_and_preprocess(gold)

# Long volatility
long_vol <- c('VIXY')
long_vol_data <- fetch_and_preprocess(long_vol)
```

Portfolio

```r
# Create a sample portfolio of different assets
portfolio <- cbind(stock_data, etf_data, fi_data, comm_data, gold_data)
portfolio_na_counts <- colSums(is.na(portfolio))
portfolio_na_counts
```

```
## AAPL NVDA  AMD  SPY SPEM  IEF  AGG  BCI  GLD
##    0    0    0    0    0    0    0    0    0
```

```r
plot(portfolio, main = "Adjusted Closing Prices", xlab = "Date", ylab = "Price", col = rainbow(ncol(portfolio)),
legend.loc = "topright")
```




```r
# Convert the data from adjusted closing price to daily return
portfolio_dr <- Return.calculate(portfolio, method = "log")
portfolio_dr <- portfolio_dr * 100
# Remove first day w/o return
portfolio_dr <- portfolio_dr[-1, ]
# portfolio_dr_dis <- Return.calculate(portfolio, method = "discrete")
# Remove first day w/o return
# portfolio_dr_dis <- portfolio_dr_dis[-1, ]
assets <- colnames(portfolio_dr)
annual_rf = 0.04
```

Common Constraints

1. Full investment (weights sum to 1)
2. Long-only = no short positions

Minimum Variance Portfolio

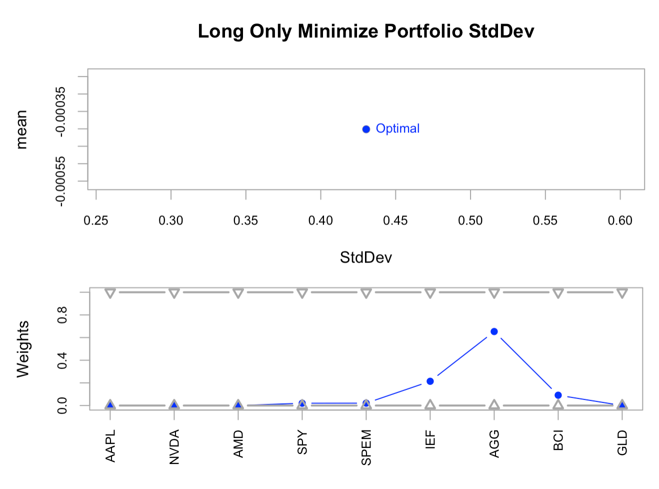
```r
# https://github.com/braverock/PortfolioAnalytics/blob/master/demo/demo_min_StdDev.R
port_spec_mv <- portfolio.spec(assets=assets)
port_spec_mv <- add.constraint(portfolio=port_spec_mv, type="full_investment")
port_spec_mv <- add.constraint(portfolio=port_spec_mv, type="long_only")
port_spec_mv <- add.objective(portfolio=port_spec_mv, type="risk", name="StdDev")
print(port_spec_mv)
```

```
## **************************************************
## PortfolioAnalytics Portfolio Specification
## **************************************************
##
## Call:
## portfolio.spec(assets = assets)
##
## Number of assets: 9
## Asset Names
## [1] "AAPL" "NVDA" "AMD"  "SPY"  "SPEM" "IEF"  "AGG"  "BCI"  "GLD"
##
## Constraints
## Enabled constraint types
##      - full_investment
##      - long_only
##
## Objectives:
## Enabled objective names
##      - StdDev
```

```r
min_var_portfolio <- optimize.portfolio(R=portfolio_dr, portfolio=port_spec_mv,
                                        optimize_method="ROI",
                                        trace=TRUE)
print(min_var_portfolio)
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = portfolio_dr, portfolio = port_spec_mv,
##     optimize_method = "ROI", trace = TRUE)
##
## Optimal Weights:
##   AAPL   NVDA    AMD    SPY   SPEM    IEF    AGG    BCI    GLD
## 0.0000 0.0000 0.0000 0.0205 0.0211 0.2141 0.6532 0.0912 0.0000
##
## Objective Measure:
## StdDev
## 0.4303
```

```r
plot(min_var_portfolio, risk.col="StdDev", main="Long Only Minimize Portfolio StdDev")
```

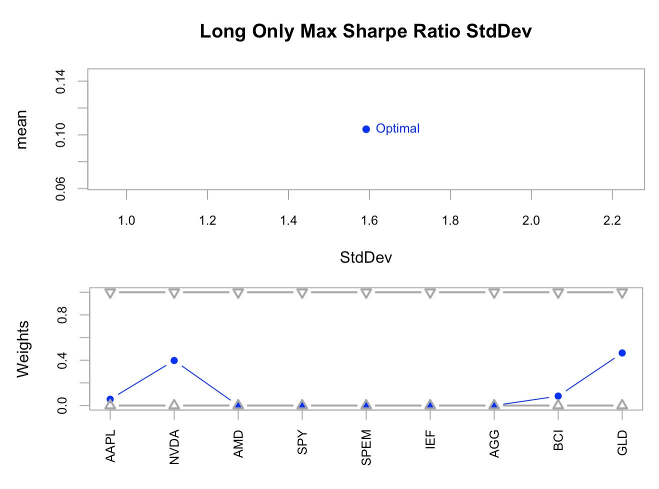


Markowitz Portfolio

```r
# https://github.com/braverock/PortfolioAnalytics/blob/master/demo/demo_max_Sharpe.R
port_spec_sr <- portfolio.spec(assets=assets)
port_spec_sr <- add.constraint(portfolio=port_spec_sr, type="full_investment")
port_spec_sr <- add.constraint(portfolio=port_spec_sr, type="long_only")
port_spec_sr <- add.objective(portfolio=port_spec_sr, type="return", name="mean")
port_spec_sr <- add.objective(portfolio=port_spec_sr, type="risk", name="StdDev")
max_sr_portfolio <- optimize.portfolio(R=portfolio_dr, portfolio=port_spec_sr,
                                   optimize_method="ROI",
                                   maxSR=TRUE, trace=TRUE, rf=annual_rf)
print(max_sr_portfolio)
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = portfolio_dr, portfolio = port_spec_sr,
##     optimize_method = "ROI", trace = TRUE, maxSR = TRUE, rf = annual_rf)
##
## Optimal Weights:
##   AAPL   NVDA    AMD    SPY   SPEM    IEF    AGG    BCI    GLD
## 0.0553 0.3975 0.0000 0.0000 0.0000 0.0000 0.0000 0.0832 0.4640
##
## Objective Measure:
## StdDev
##  1.592
##
##
##   mean
## 0.1042
```

```r
plot(max_sr_portfolio, risk.col="StdDev", main="Long Only Max Sharpe Ratio StdDev")
```



Comparison of Portfolios

```r
extractObjectiveMeasures(min_var_portfolio)
```

```
## $StdDev
##    StdDev
## 0.4303332
```

```r
extractObjectiveMeasures(max_sr_portfolio)
```

```
## $StdDev
##   StdDev
## 1.591931
##
## $mean
##      mean
## 0.1041612
```

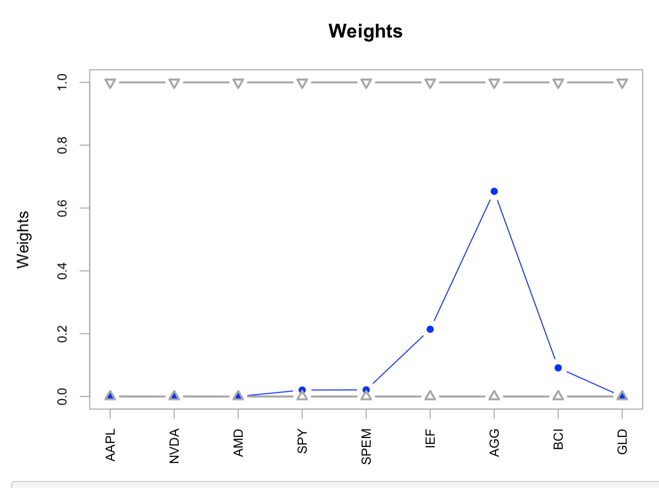
```r
extractWeights(min_var_portfolio)
```

```
##          AAPL          NVDA           AMD           SPY          SPEM
##  0.000000e+00 -3.705634e-18 -2.294035e-18  2.051965e-02  2.105622e-02
##           IEF           AGG           BCI           GLD
##  2.140757e-01  6.531764e-01  9.117207e-02 -3.453993e-18
```

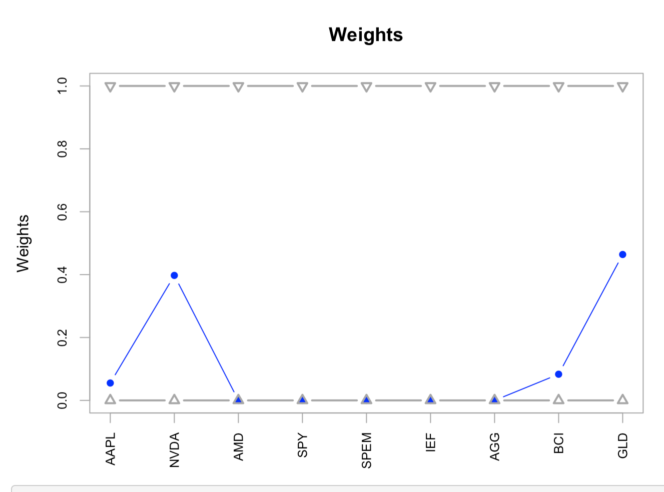
```r
extractWeights(max_sr_portfolio)
```

```
##          AAPL          NVDA           AMD           SPY          SPEM
##  5.528497e-02  3.974808e-01  1.166956e-24  2.092352e-16 -3.546199e-17
##           IEF           AGG           BCI           GLD
##  1.494729e-15 -1.528393e-16  8.324074e-02  4.639935e-01
```

```r
chart.Weights(min_var_portfolio)
```




```r
chart.Weights(max_sr_portfolio)
```




```r
extractStats(min_var_portfolio)
```

```
##        StdDev           out        w.AAPL        w.NVDA         w.AMD
##  4.303332e-01  1.851867e-01  0.000000e+00 -3.705634e-18 -2.294035e-18
##         w.SPY        w.SPEM         w.IEF         w.AGG         w.BCI
##  2.051965e-02  2.105622e-02  2.140757e-01  6.531764e-01  9.117207e-02
##         w.GLD
## -3.453993e-18
```

```r
extractStats(max_sr_portfolio)
```

```
##        StdDev          mean           out        w.AAPL        w.NVDA
##  1.591931e+00  1.041612e-01  2.534245e+00  5.528497e-02  3.974808e-01
##         w.AMD         w.SPY        w.SPEM         w.IEF         w.AGG
##  1.166956e-24  2.092352e-16 -3.546199e-17  1.494729e-15 -1.528393e-16
##         w.BCI         w.GLD
##  8.324074e-02  4.639935e-01
```

```r
evaluate_portfolio <- function(portfolio_dr, weights) {
  portfolio_returns <- Return.portfolio(portfolio_dr, weights=weights)
  mean_return <- mean(portfolio_returns)
  std_dev <- sd(portfolio_returns)

  ann_mean_return <- mean_return * 252  # Assuming 252 trading days per year
  ann_std_dev <- std_dev * sqrt(252)    # Assuming 252 trading days per year

  # Calculate risk-adjusted measures
  sharpe_ratio <- ann_mean_return / ann_std_dev
  sortino_ratio <- SortinoRatio(portfolio_returns)

  evaluation_metrics <- data.frame(
    Mean_Return = mean_return,
    Std_Deviation = std_dev,
    Annualized_Mean_Return = ann_mean_return,
    Annualized_Std_Deviation = ann_std_dev,
    Sharpe_Ratio = sharpe_ratio,
    Sortino_Ratio = sortino_ratio
  )

  return(evaluation_metrics)
}

evaluate_portfolio(portfolio_dr, min_var_portfolio$weights)
```

```
##                         Mean_Return Std_Deviation Annualized_Mean_Return
## Sortino Ratio (MAR = 0%)   0.1503491       3.42554               37.88796
##                         Annualized_Std_Deviation Sharpe_Ratio
## Sortino Ratio (MAR = 0%)                 54.37876    0.6967419
##                         portfolio.returns
## Sortino Ratio (MAR = 0%)         0.0605626
```

```r
evaluate_portfolio(portfolio_dr, max_sr_portfolio$weights)
```

```
##                         Mean_Return Std_Deviation Annualized_Mean_Return
## Sortino Ratio (MAR = 0%)    0.226284       3.43122               57.02356
##                         Annualized_Std_Deviation Sharpe_Ratio
## Sortino Ratio (MAR = 0%)                 54.46893     1.046901
##                         portfolio.returns
## Sortino Ratio (MAR = 0%)        0.09950134
```

Consideration for Retirement Planning

Additional Constraints

1. Diversification: Diversification is important for retirement planning
2. Conditional Value-at-Risk: Help control risk of large loss
3. Specify target return of 6% which is slightly higher than current risk-free rate of 4%

Risk Metric

1. Average Length (Retiree cannot rely on portfolio with long drawdown duration)

Return Metric

1. Omega Sharpe Ratio (Focus on downside risk)
2. Burke Ratio (Maximum Drowdown is important for retiree as they are dependent on withdrawals)
3. Sortino Ratio (Picked to maximize capital preservation)

```r
# Diversity - rebalancing over time?
rp <- portfolio.spec(assets = assets)
rp <- add.constraint(rp, type="full_investment")
rp$constraints[[1]]$min_sum=0.99
rp$constraints[[1]]$max_sum=1.01
rp <- add.constraint(rp, type = "long_only")
rp <- add.constraint(rp, type = "CVaR", level = 0.95, enabled = TRUE)
rp <- add.constraint(rp, type = "return", return_target = 0.06, enabled = TRUE)
rp <- add.constraint(rp, type = "diversification", div_target = 0.75)

CumOmegaSharpeRatio <- function(R, weights, MAR=0) {
  portfolio_returns <- Return.portfolio(R, weights = weights)
  omega_sharpe_ratio <- OmegaSharpeRatio(portfolio_returns, MAR = MAR)
  return(omega_sharpe_ratio)
}

# Both Maximum Drawdown and Drawdown duration are important for retiree as they are dependent on withdrawals and
further
# can't rely on portfolio with long drawdown duration
# Burke Ratio (return)
CumBurkeRatio <- function(R, weights) {
  portfolio_returns <- Return.portfolio(R, weights = weights)
  burke_ratio <- BurkeRatio(portfolio_returns)
  return(burke_ratio)
}
# Average Length
CumAverageLength <- function(R, weights) {
  portfolio_returns <- Return.portfolio(R, weights = weights)
  average_length <- AverageLength(portfolio_returns)
  return(average_length)
}

rp <- add.objective(rp, type = "risk", name = 'CumAverageLength')
rp <- add.objective(rp, type = "return", name = 'CumOmegaSharpeRatio')
# rp <- add.objective(rp, type = 'return', name = 'CumBurkeRatio')

# Sortino Ratio is picked to maximize capital preservation (return)
rp <- add.objective(rp, type = "return", name = "SortinoRatio")
rp_portfolio <- optimize.portfolio(R=portfolio_dr, portfolio=rp, rf=annual_rf, search_size=1000)
```

```
## Iteration: 1 bestvalit: 1562.251383 bestmemit:    0.004000    0.104000    0.096000    0.054000    0.306000
0.000000    0.000000    0.260000    0.178000
## Iteration: 2 bestvalit: 1540.579689 bestmemit:    0.004000    0.052000    0.184000    0.216000    0.022000
0.008000    0.210000    0.020000    0.292000
## Iteration: 3 bestvalit: 1540.579689 bestmemit:    0.004000    0.052000    0.184000    0.216000    0.022000
0.008000    0.210000    0.020000    0.292000
## Iteration: 4 bestvalit: 1538.572341 bestmemit:    0.038000    0.202000    0.038000    0.006000    0.346000
0.176000    0.020000    0.040000    0.130000
## Iteration: 5 bestvalit: 1531.382308 bestmemit:    0.018000    0.018000    0.054000    0.036409    0.320470
0.166035    0.238900    0.142657    0.007218
## Iteration: 6 bestvalit: 1531.382308 bestmemit:    0.018000    0.018000    0.054000    0.036409    0.320470
0.166035    0.238900    0.142657    0.007218
## Iteration: 7 bestvalit: 1531.382308 bestmemit:    0.018000    0.018000    0.054000    0.036409    0.320470
0.166035    0.238900    0.142657    0.007218
## Iteration: 8 bestvalit: 1531.382308 bestmemit:    0.018000    0.018000    0.054000    0.036409    0.320470
0.166035    0.238900    0.142657    0.007218
## Iteration: 9 bestvalit: 1530.491653 bestmemit:    0.004000    0.068000    0.228000    0.236000    0.302000
0.000000    0.068000    0.028000    0.060000
## Iteration: 10 bestvalit: 1530.491653 bestmemit:    0.004000    0.068000    0.228000    0.236000    0.302000
0.000000    0.068000    0.028000    0.060000
## Iteration: 11 bestvalit: 1530.491653 bestmemit:    0.004000    0.068000    0.228000    0.236000    0.302000
0.000000    0.068000    0.028000    0.060000
## Iteration: 12 bestvalit: 1530.491653 bestmemit:    0.004000    0.068000    0.228000    0.236000    0.302000
0.000000    0.068000    0.028000    0.060000
## Iteration: 13 bestvalit: 1530.491653 bestmemit:    0.004000    0.068000    0.228000    0.236000    0.302000
0.000000    0.068000    0.028000    0.060000
## Iteration: 14 bestvalit: 1530.491653 bestmemit:    0.004000    0.068000    0.228000    0.236000    0.302000
0.000000    0.068000    0.028000    0.060000
## Iteration: 15 bestvalit: 1530.491653 bestmemit:    0.004000    0.068000    0.228000    0.236000    0.302000
0.000000    0.068000    0.028000    0.060000
## Iteration: 16 bestvalit: 1530.491653 bestmemit:    0.004000    0.068000    0.228000    0.236000    0.302000
0.000000    0.068000    0.028000    0.060000
## Iteration: 17 bestvalit: 1530.491653 bestmemit:    0.004000    0.068000    0.228000    0.236000    0.302000
0.000000    0.068000    0.028000    0.060000
## Iteration: 18 bestvalit: 1530.491653 bestmemit:    0.004000    0.068000    0.228000    0.236000    0.302000
0.000000    0.068000    0.028000    0.060000
## Iteration: 19 bestvalit: 1530.491653 bestmemit:    0.004000    0.068000    0.228000    0.236000    0.302000
0.000000    0.068000    0.028000    0.060000
## Iteration: 20 bestvalit: 1530.491653 bestmemit:    0.004000    0.068000    0.228000    0.236000    0.302000
0.000000    0.068000    0.028000    0.060000
## Iteration: 21 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 22 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 23 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 24 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 25 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 26 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 27 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 28 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 29 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 30 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 31 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 32 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 33 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 34 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 35 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## Iteration: 36 bestvalit: 1529.958175 bestmemit:    0.014000    0.210000    0.086000    0.010000    0.216979
0.130000    0.008000    0.024000    0.310000
## [1] 0.014000 0.210000 0.086000 0.010000 0.216979 0.130000 0.008000 0.024000
## [9] 0.310000
```

```r
rp_portfolio
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = portfolio_dr, portfolio = rp, search_size = 1000,
##     rf = annual_rf)
##
## Optimal Weights:
##  AAPL  NVDA   AMD   SPY  SPEM   IEF   AGG   BCI   GLD
## 0.014 0.210 0.086 0.010 0.217 0.130 0.008 0.024 0.310
##
## Objective Measures:
## CumAverageLength
##            5.281
##
##
## CumOmegaSharpeRatio
##              0.3046
##
##
## SortinoRatio
##       0.1374
```

Comparisons

```r
evaluate_portfolio(portfolio_dr, rp_portfolio$weights)
```

```
##                         Mean_Return Std_Deviation Annualized_Mean_Return
## Sortino Ratio (MAR = 0%)   0.3658941      7.169461               92.2053
##                         Annualized_Std_Deviation Sharpe_Ratio
## Sortino Ratio (MAR = 0%)                 113.8117    0.8101569
##                         portfolio.returns
## Sortino Ratio (MAR = 0%)         0.1373825
```

1. We can see that Minimum Variance Portfolio indeed produce lowest standard deviation but also lower mean return compared to Markowitz and our Retirement Portfolio
2. The retirement portfolio provides higher mean return at the expense of higher risk in standard deviation.
3. The choice of portfolios however depend on the risk aversion and preferences of the manager

References

1. [https://bookdown.org/compfinezbook/introcompfinr/]
2. [https://rpubs.com/Sergio_Garcia/intermediate_portfolio_analysis_r]