Portfolio Optimization Using Simulation

Khizar Tahir

A. Introduction

Portfolio Managers across the globe have a key task of Portfolio Optimization which is the process of finding the optimal weights for each of the selected securities from a set of possible or feasible portfolios in such a way that the portfolio is mean-variance efficient. In portfolio management, an activity conducted by many firms, a critical issue after identifying what assets to include in portfolio becomes “in what proportions”. They have a tough decision to make. Do they want to have the same proportions for each security or is there an optimal set of weights that would produce the best portfolio results. The process to identify the best is called optimal portfolio construction.

There are multiple approaches to identify the optimal portfolio. One is algorithmic/mathematical that solves for optimal portfolio computationally and determines the set of optimal weights. The other is Simulation in which we simulate lots of “what if” scenarios (portfolios), develop some metrics and then by scanning, we determine which among them is the best - meaning the one that dominates all others.

B. Project overview

This project focuses on the Simulation method of Portfolio Optimization. Using a random sampling of portfolio weights, portfolios are to be constructed and compared in mean-variance space using the Metric Sharpe Ratio:

$$S_p = \frac{E(r_p) - r_f}{\sigma_p}$$

$where:$
$S_p = SharpeRatio,$
$E(r_p) = ExpectedReturn,$
$r_f = RiskFreeRate,$
$\sigma_p = Risk,$

The project has 2 main components leading to optimal portfolio selection:

> myMeanVarPort function
>
> This function analyzes monthly stock-return data from quantmod package to output mean-variance metrics for a collection of stocks as well as information about simulated portfolios. It takes in 4 arguments: vector of TICKERs, a begin date, an end date, and risk-free rate. It then outputs a **List** containing the **vector** of stock means; the covariance **matrix**; relevant information (weights, mean, sigma) for each simulated portfolio.

> ggplot2 graph
>
> A plot needs to be created using information obtained in part 1 that displays the simulated portfolios on a graph and has an annotation signifying the optimal portfolio.

C. Code and Explanation

Part 1: R function

The function constructed has 8 sub-parts:

1.0 Importing required packages and libraries.

1.1 Taking Input

The function takes in the four inputs defined in section B. It creates a variable that stores the start and end dates in format that can be used to subset later on.

1.2 Obtaining monthly stock-returns data

We use the quantmod package to access data. We create a time series variable. Based on vector of Tickers from input, we fetch prices for those tickers from quantmod. Then use those prices data to fetch period returns for month-wise specified period. We merge this data with our original time series to have a time series data of relevant tickers and their monthly stock-returns. We name the data columns, subset data for our specified duration from part 1.1, and drop NA values. Our returns dataframe is now ready.

1.3 Calculating means of stocks data and storing in appropriate formats for calculations and output.

We use columnmean function to achieve this and store a dataframe, vector, and matrix of the result.

1.4 Calculating Covariance matrix for returns data.

Used var function on the returns dataframe.

1.5 Simulating weights.

This is where we create a set of possible weights for simulation. We set.seed(12) to have static random values, set number of portfolios to 100N, where N is the number of securities. With 7 securities as argument, the simulation ran for 100*7 = 700 portfolios.

Then, we fill the dataframe of weights with random numbers between 1 and 10. These cannot be used just yet because the sum of weights can at max be 1 (fully invested). At this time, the total is more. So we calculate simulated weights where we divide each security in a portfolio with its rowsum. After this, all rows add up to one and weights are usable.

1.6 Initializing weights and results matrices.

We initialize these for use in our for loop for calculations and storing results. We cannot use the simulated weights matrix from 1.5 because it has weights for all portfolios and would mess up our matrix multiplication. For each portfolio, we need a weight matrix that is n x 1 where n is the number of securities.

1.7 For loop to calculate metrics and store results: mean (Expected return), standard deviation (Risk), and Sharpe Ratio for each portfolio.

We use 2 for loops for this. The first (outer) runs for each row (portfolio) and the second (inner) runs for each column to populate weights matrix and the first N columns of Results matrix with simulated weights. For each outer loop iteration, we calculate mean and sigma using matrix multiplication formulas below:

$Mean$:

$$\bar{R}_p = w' \bar{R}$$

$St.dev$:

$$\sigma_p^2 = w' \sum w$$

1.8 Final output

We name columns of results dataframe appropriately, and store **Vector** of StockMeans, covariance matrix, and simulation Results in Output as a **List**.

```
# 1.0: Importing required packages and libraries.
library(quantmod)
library(ggplot2)
library(dplyr)
```

```
# 1.1 Taking Input
myMeanVarPort <- function(ticks, startDate, endDate, rfRate){
  duration <- paste(startDate, endDate, sep = "/")
  # 1.2 Obtaining monthly stock-returns data
  retout <- NULL
  retout <- xts(retout)

  for(i in 1:length(ticks)){
    prices = getSymbols(ticks[i], auto.assign = F)
    returns <- periodReturn(prices, period = "monthly",
                            type = "arithmetic")
    retout <- merge.xts(retout, returns)
  }
  colnames(retout) <- ticks
  retout <- retout[duration]
  retout <- na.omit(retout)

  # 1.3 Calculating means of stocks data and storing in appropriate formats for calculations and output.
  # DF
  stockMeans <- colMeans(retout,na.rm = T)
  stockMeans = round(stockMeans, 5)
  # as Vector
  vectorStockMeans <- as.vector(stockMeans)
  # as matrix for multiplication
  matrixStockMeans <- as.matrix(stockMeans)

  # 1.4 Calculating Covariance matrix for returns data.
  covar <- var(retout)
  covar = round(covar, 8)

  # 1.5 Simulating weights.
  set.seed(12)
  niter <- 100*length(ticks)   # Set the number of iterations here. 100 multiple by Num of securities -
 7.
  randomnums <- data.frame(replicate(length(ticks), runif(niter, 1, 10)))
  sim_weights <- randomnums/rowSums(randomnums)

  # 1.6 Initializing weights and results matrices.
  weights <- matrix(data = NA, nrow = length(ticks), ncol = 1)
  Results <- matrix(data = NA, nrow = niter, ncol = length(ticks)+3)

  # 1.7 For loop to calculate metrics and store results: mean (Expected return), standard deviation (Ris
k), and Sharpe Ratio for each portfolio
  for (i in 1:niter){
    # inner loop places weights into Results
    for (k in 1:length(ticks)) {
             Results[i,k] = weights[k,1] = sim_weights[i,k]
    }

    Results[i,8] <- t(weights) %*% matrixStockMeans # portfolio mean
    Results[i,9] <- sqrt(t(weights) %*% covar %*% weights) # portfolio sigma
    Results[i,10] <- (Results[i,8] - rfRate)/Results[i,9] # Sharpe ratio
  }

  # 1.8 Final output
  colnames(Results) <- c(ticks, "PortMean", "PortSigma", "SharpeRatio")
  Results <- as.data.frame(Results)

  output <- list(vectorStockMeans, covar, Results)

  return(output)
}
```

Running the function.

It will output:

1. Type of returned object. It will be a list.
2. Stock Means as a Vector.
3. Covariance Matrix.
4. Head of the Portfolio results.

```
tickList <- c("GE","XOM","GBX","SBUX","PFE","HMC","NVDA")
begin <- "20140101"
end <- "20171231"
riskFreeRate <- 0

funcResult = myMeanVarPort(tickList, begin, end, riskFreeRate)
print(typeof(funcResult))
```

```
## [1] "list"
```

```
print(funcResult[[1]])
```

```
## [1] -0.00837 -0.00314  0.01536  0.00902  0.00442 -0.00246  0.05817
```

```
print(funcResult[[2]])
```

```
##              GE        XOM        GBX       SBUX        PFE        HMC
## GE   0.00294255 0.00103470 0.00121121 0.00080728 0.00064653 0.00062168
## XOM  0.00103470 0.00168517 0.00162533 0.00023608 0.00053446 0.00020432
## GBX  0.00121121 0.00162533 0.01036163 0.00006399 0.00157675 0.00212741
## SBUX 0.00080728 0.00023608 0.00006399 0.00211533 0.00052364 0.00068028
## PFE  0.00064653 0.00053446 0.00157675 0.00052364 0.00188741 0.00083543
## HMC  0.00062168 0.00020432 0.00212741 0.00068028 0.00083543 0.00316065
## NVDA 0.00135432 0.00109161 0.00318265 0.00036711 0.00036748 0.00116384
##            NVDA
## GE   0.00135432
## XOM  0.00109161
## GBX  0.00318265
## SBUX 0.00036711
## PFE  0.00036748
## HMC  0.00116384
## NVDA 0.01112564
```

```
print(head(funcResult[[3]]))
```

```
##           GE        XOM        GBX       SBUX       PFE        HMC      NVDA
## 1 0.04591352 0.12500521 0.11204062 0.14103162 0.2409166 0.20463972 0.1304527
## 2 0.17032874 0.09825230 0.16522497 0.17753802 0.1241027 0.19185299 0.0727003
## 3 0.22918285 0.03894855 0.24130303 0.05836888 0.1482585 0.11881530 0.1651229
## 4 0.09390609 0.21051359 0.07800949 0.03641164 0.2498903 0.08883252 0.2424364
## 5 0.07590857 0.19030177 0.25140423 0.10023674 0.1593137 0.08202033 0.1408146
## 6 0.03183638 0.12317640 0.22950334 0.17884515 0.1092784 0.11006486 0.2172955
##      PortMean  PortSigma SharpeRatio
## 1 0.010366106 0.03713121   0.2791750
## 2 0.006710637 0.03799180   0.1766338
## 3 0.012160559 0.04662683   0.2608061
## 4 0.015068165 0.04154418   0.3627022
## 5 0.012226386 0.04463796   0.2739011
## 6 0.017337440 0.04636429   0.3739395
```

Part 2: Plotting Graph.

This is achieved in sub-parts:

2.1: Storing results dataframe.

We create a new variable “portResults” to store only the portfolio results from Part 1.

2.2: Obtaining Optimal portfolio mathematically and printing it.

We store the location of the portfolio with the highest Sharpe Ration (already calculated in 1.7). Then we subset that location from our results to obtain the optimal portfolio.

2.3 Preparation for Graph of simulated portfolios and Capital Allocation Line.

We understand that in the graph of simulated portfolios plotted with portfolio sigma on x-axis and portfolio mean on y-axis, the Capital Allocation Line (CAL) has the y-intercept at (0, risk-free rate) and is tangent to the efficient frontier. The portfolio at the point of contact/tangent is the optimal portfolio.

We can conclude that a line passing through (0, risk-free rate) and (sigma of optimal portfolio, mean of optimal portfolio) having gradient = Sharpe Ratio of Optimal portfolio will be the CAL. We can plot it to have ease in identifying optimal portfolio from 700 portfolio plots. We do all the calculations making this plot possible in this part.

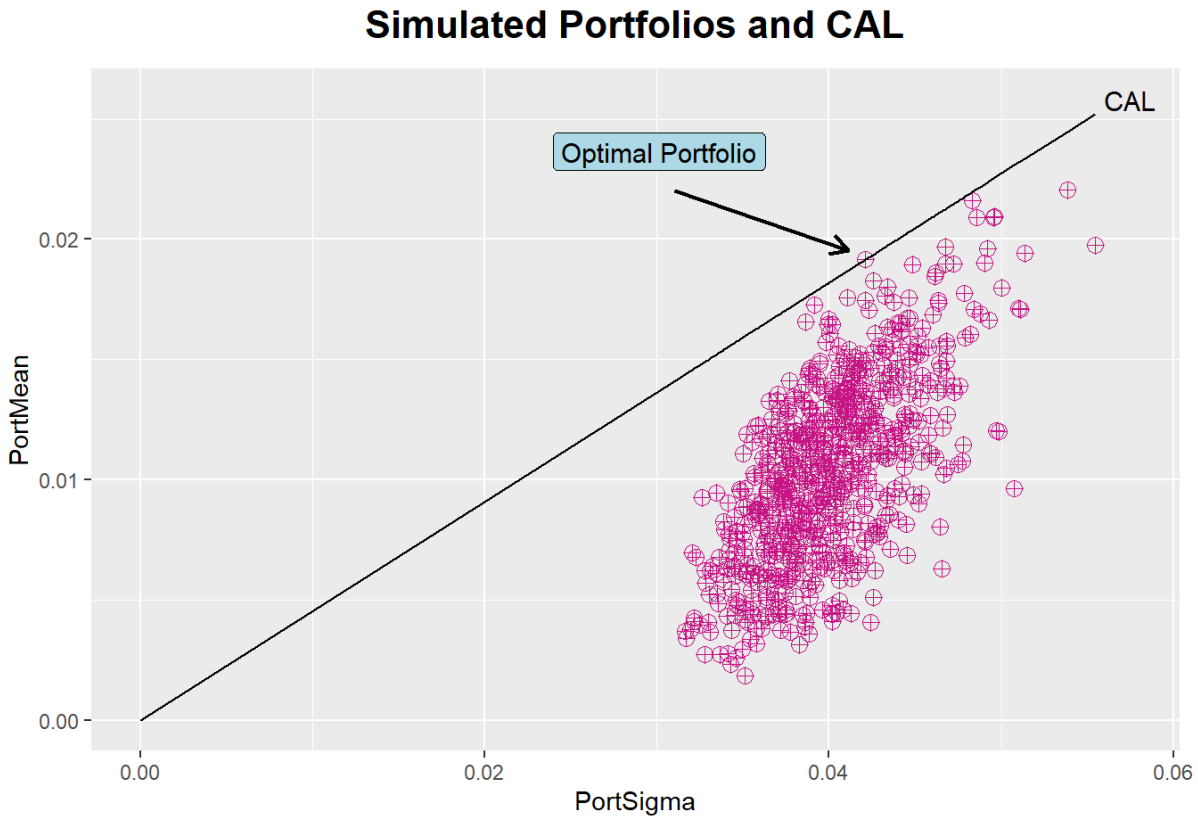
2.4: Final plotting and annotation.

So we plot the two: simulated portfolios, and CAL. At the point of tangency of the two, we annotate for the Optimal Portfolio.

```
# 2.1: Storing results dataframe.
portResults = funcResult[[3]]

# 2.2: Obtaining Optimal portfolio mathematically.
maxSharpe_loc <- which.max(portResults$SharpeRatio)
optPort <- portResults[maxSharpe_loc,]
print(optPort)
```

```
##             GE       XOM        GBX      SBUX       PFE       HMC     NVDA
## 356 0.05440971 0.1053346 0.05791203 0.1985004 0.2221315 0.0786257 0.283086
##       PortMean  PortSigma SharpeRatio
## 356 0.01914936 0.04212125   0.4546247
```

```
# 2.3 Preparation for Graph of simulated portfolios and Capital Allocation Line.
# Gradient of CAL = Sharpe ratio of optimal portfolio.
gradCAL <- (optPort$PortMean - riskFreeRate) / (optPort$PortSigma - 0)
# some additional values for CAL
xvals <- c(0, portResults$PortSigma)
yvals <- gradCAL*xvals + riskFreeRate
dat <- data.frame(x1 = xvals, y1 = yvals)

# 2.4: Final plotting and annotation.
myplot2 <- ggplot(data = portResults) +
  ggtitle('Simulated Portfolios and CAL') +
  theme(plot.title = element_text(size=16, face="bold", margin = margin(10, 0, 10, 0), hjust = 0.5)) +
  geom_point(aes(x = PortSigma, y = PortMean), pch = 10, colour = "mediumvioletred", size = 3) +
  geom_line(data = dat, aes(x = x1, y = y1), color = 'black') +
  annotate("text", x = 0.0575, y = 0.0258, label = "CAL", size=4, colour='black') +
  annotate("label", x = optPort$PortSigma - 0.012, y = optPort$PortMean + 0.0045, label = "Optimal Portfo
lio", size=4, colour='black', fill='lightblue') +
  geom_segment(aes(x = 0.031, y = 0.022, xend = optPort$PortSigma - 0.001, yend = optPort$PortMean + 0.00
04), colour='black', size=0.8 ,arrow = arrow(length = unit(0.3, "cm")))

print(myplot2)
```



D. Conclusion.

From this project, we learned how the simulation method can be effectively used to simulate hundreds (in this case) but thousands of portfolios in no time at all. We observed how just knowing column means and covariance of returns are not enough and how the mathematics combines with the computing and visualizing power of R to generate insights that are so valuable. Moreover, computationally finding Optimal portfolio and then being able to visualize using ggplot2 is a real plus. This can help firms make informed decisions on where to invest, and most importantly, how much to invest in each security.