Mean-Variance Optimal Risky Portfolio Using Simulation

Pranav B, Sanjeev H, Jared R, Ankita S

2022-10-30

INTRODUCTION

In this report, we present our findings based on the Mean-Variance Portfolio Theory, for locating an Optimal Risky Portfolio using the Monte Carlo Simulation (MCS) technique. The one that offers the highest return per unit of risk measured by the Sharpe Ratio is considered an optimal risk portfolio. Further, the Modern Portfolio Theory (MPT) states that investors are risk averse and given a level of risk, they will choose the portfolios that offer the most return. In order to do that, we need to optimize the portfolios.

This test can be done with either historical or "predicted" returns. We assign random weights to each of the strategies in the portfolio; with the sum of all the weights equal to 1 (or 100% of allocated capital). After assigning weights, we generate a risk-adjusted return value or Sharpe Ratio. We want this value to be as high as possible.

For ease of computation, we've made use of R packages such as quantmod, dplyr, and lubridate. The data is visualized using the ggplot2 package in order to find the optimal portfolio and the minimum variance portfolio.

Loading Relevant Packages

```
library(quantmod)  # For loading and managing financial data (default source: Yahoo Finance)
library(ggplot2)   # For data visualization
library(dplyr)     # For data manipulation
library(lubridate) # For easy handling of date-time data
library(yfR)       # For downloading and organizing financial data from Yahoo Finance
```

'yfR' is a new R package based on quantmod. It may be used to download and organize financial data from Yahoo Finance. As of Mar'22, there is not much documentation available on the web or in print regarding it's use. However, we tried installing the package and retrieving stock data. For ease of computation in this report, we'll be using quantmod package itself to retrieve data.

Next, we select the stocks we would like to include in our portfolio:

1. **General Electric Company** (GE)
2. **Exxon Mobil** (XOM)
3. **Greenbrier Companies Inc** (GBX)
4. **Starbucks Corporation** (SBUX)
5. **Pfizer Inc.** (PFE)
6. **Honda Motor Co Ltd** (HMC)
7. **AT&T Inc.** (AT&T (T))

Loading Data

We retrieve the stock data and create a list with their ticker symbols. To calculate the returns, we specify the time period in question.

```
begin_date = as_date("20140101")
end_date = as_date("20171231")
# Setting beginning date and end date

ticks <- c("GE", "XOM", "GBX", "SBUX","PFE","HMC","NVDA")
# Ticker symbols for required stocks

rfr <- .02 # Initializing the risk-free rate
```

Creating a Function (MeanVarPort) to Calculate Useful Portfolio Optimization Metrics for Simulated Portfolios

In the following code, we use **Sharpe Ratio** as a metric to provide information on how efficient a portfolio return is, concerning how risky is its composition. It is a risk-adjusted measure and we, therefore, want this to be as high as possible, as opposed to simply looking at return measures.

```
MeanVarPort <- function(ticks, begin_date, end_date, rfr)
# Initializing MeanVarPort function
{

#library(yfR)
#returnsyf <- yf_get(ticks,first_date = begin_date,
#last_date = end_date,type_return ='arit',freq_data = 'monthly',do_complete_data=TRUE)

#returnsyf <- na.omit(returnsyf)
#returnsyf
#retout<-returnsyf(ret_adjusted_prices)
#colnames(retout) <- ticks

#head(retout, 2)
#tail(retout, 2)

  retout <- NULL
  retout <- xts(retout)

  for(i in 1:length(ticks)){
  # Retrieving the monthly returns for the stocks
    prices = getSymbols(ticks[i], auto.assign = F)
    #using getSymbols to retrieve monthly adjusted prices
    returns <- periodReturn(prices, period = "monthly", type = "arithmetic")
    # using periodReturn to calculate returns
    retout <- merge.xts(retout, returns)
  # Retrieving monthly share data and merging the two data sets
  }

colnames(retout) <- ticks
# Initializing column names
retout <- na.omit(retout)
```

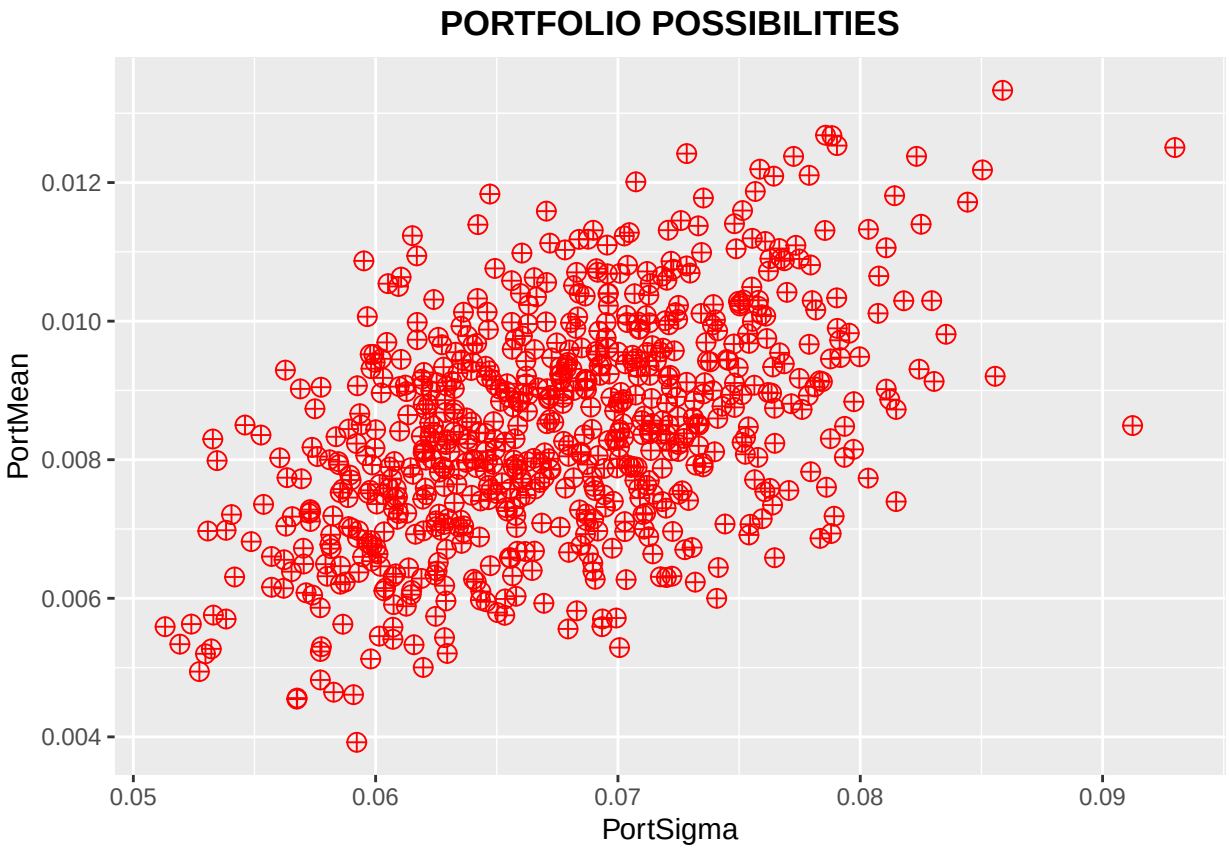
```r
# Omitting all NA values

meanret <- colMeans(retout, na.rm = T)
# Calculating mean returns and covariance
covar <- var(retout)
correlation <- cor(retout)
# Running correlation function to figure data patterns

set.seed(12)
# Setting the seed to 12 to get consistent results
niter <- length(ticks)*100
# N*100 (Simulating portfolio weights to construct 100N portfolios (7 securities))
randomnums <- data.frame(replicate(length(ticks), runif(niter,1,10)))
wt_sim <- randomnums/rowSums(randomnums)

weight <- matrix(data = NA, nrow = length(ticks), ncol = 1)
# Normalizing weights, so that we get proper weights that sum to 1

results <- matrix(data = NA, nrow = niter, ncol = length(ticks)+3)
# Setting weight and results matrices; 9 columns required in total

for( i in 1:niter){
# Giving weight to each stock
  for (k in 1:length(ticks)){
    results[i,k] = weight[k,1] = wt_sim[i,k]
  }

  results[i,length(ticks)+1] <- t(weight) %*% meanret
# Matrix multiplication of mean and the weights to calculate PortMean

  results[i,length(ticks)+2] <- sqrt(t(weight) %*% covar %*% weight)
# Matrix multiplication of transpose of weight with covariance and multiplication

  results[i,length(ticks)+3] <-
    (sqrt(12)*(results[i,length(ticks)+1]) - rfr)/results[i,length(ticks)+2]
# Resulting matrix to weight and taking square root to calculate PortSigma
#Getting an annualized Sharpe Ratio ((Portmean - risk free rate)/PortSigma)
}
  colnames(results) <- c(ticks, "PortMean", "PortSigma","SR")
 # Adding column names
results <- as.data.frame(results)
 # Converting to dataframe
head_results <- head(results,5)
 # Getting head from results

plot_1 <<- ggplot(data = results, aes(x = PortSigma, y = PortMean)) +
 # Plotting portfolio possibilities


minmret = min(results$PortMean)
 # Getting minimum and maximum of PortMean
maxmret = max(results$PortMean)
seqmret = seq(round(minmret,3)-.001, maxmret+.001,.001)
```

```
 # Getting rounded returns by sequences of one-thousandths

optim1 <- results %>% mutate(portnumber = index(results)) %>%
   # Getting optimum portfolios
  mutate(ints = cut(PortMean, breaks = seqmret),
         lower = as.numeric(sub("\\((.+),.*","\\1", ints))) %>%
  group_by(ints) %>%
  summarise(lowerval = min(lower),
            sig_optim = min(PortSigma),
            retn_optim = PortMean[which.min(PortSigma)],
            numb = length(PortSigma),
            portID = portnumber[which.min(PortSigma)])

plot_2 <- ggplot(data = optim1, aes(x = sig_optim, y = retn_optim)) +
  geom_point(pch = 10, colour = "red", size = 3)+
  annotate(geom= "segment",x=0.065,xend=0.062,y=0.012,
           yend=0.0112,colour='blue', size=3,alpha=0.6,arrow=arrow())+
  annotate("text", x = 0.066, y =0.0125, label = "optimal portfolio")+
  annotate(geom= "segment",x=0.055,xend=0.0515,y=0.006,yend=0.00559,
           zcolour='blue',size=3,alpha=0.6,arrow=arrow())+
  annotate("text", x = 0.059, y =0.0065, label = "minimum variance")+
  ggtitle("OPTIMAL PORTFOLIOS") + theme(plot.title = element_text(face = "bold",hjust = 0.5))

l <- list(meanret,covar,head_results,optim1,plot_1,plot_2)
 # Plotting list of optimum portfolios and annotating the optimum portfolio and

l
 # Setting up a list that will contain all outputs
}

MeanVarPort(ticks,begin_date,end_date,rfr) # Calling the function
```

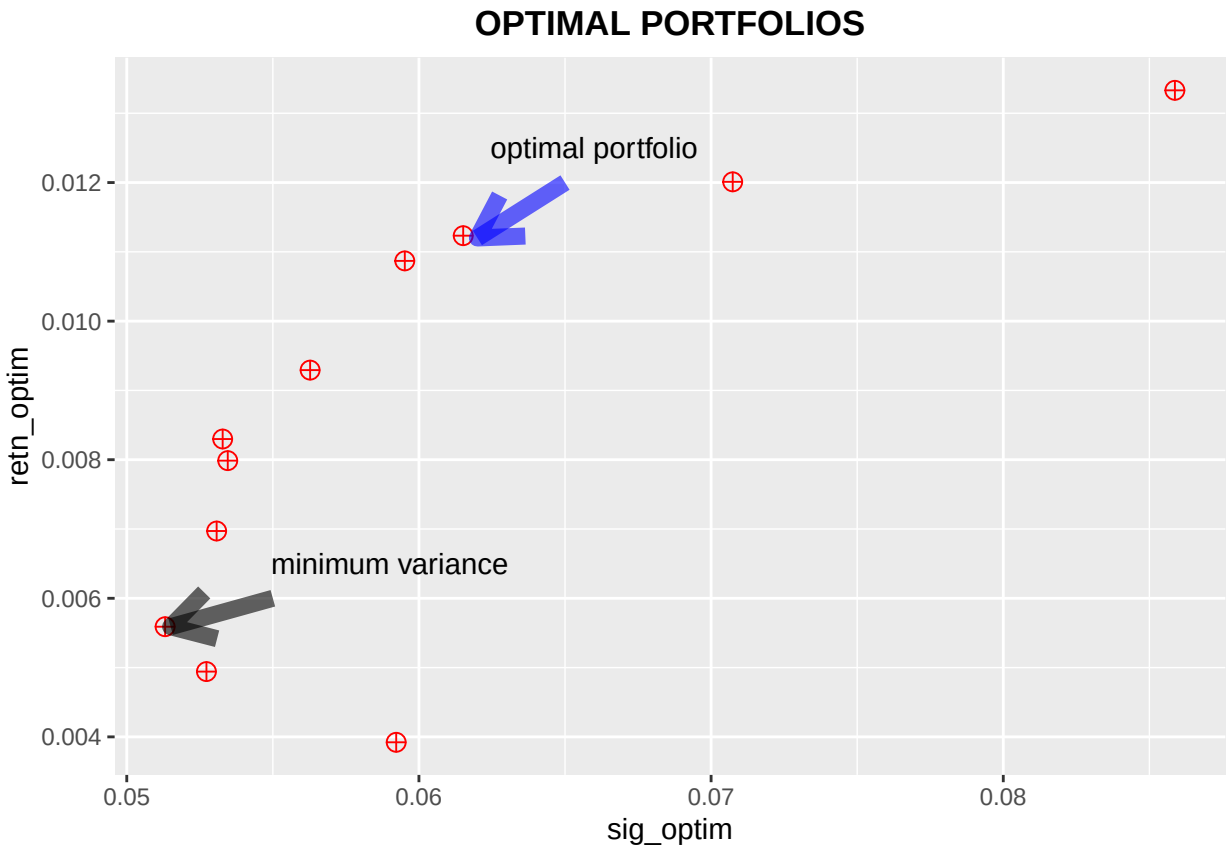
```
## Warning: Ignoring unknown parameters: zcolour
```

```
## [[1]]
##            GE           XOM           GBX          SBUX           PFE
## -0.0020535529  0.0042445814  0.0155332037  0.0113808069  0.0052379329
##           HMC          NVDA
## -0.0006990292  0.0255460588
##
## [[2]]
##                GE          XOM         GBX        SBUX          PFE          HMC
## GE   0.009534554 0.0030124962 0.008481668 0.003009762 0.0015931815 0.0026728923
## XOM  0.003012496 0.0046193387 0.003835148 0.000699276 0.0009632678 0.0014799708
## GBX  0.008481668 0.0038351479 0.033478656 0.005985388 0.0028268989 0.0056676045
## SBUX 0.003009762 0.0006992760 0.005985388 0.005970903 0.0016365403 0.0020825425
## PFE  0.001593181 0.0009632678 0.002826899 0.001636540 0.0036903571 0.0006484048
## HMC  0.002672892 0.0014799708 0.005667605 0.002082542 0.0006484048 0.0044565973
## NVDA 0.003984727 0.0020944666 0.008129490 0.003193484 0.0016668433 0.0033102152
##             NVDA
## GE   0.003984727
## XOM  0.002094467
## GBX  0.008129490
```

```
## SBUX 0.003193484
## PFE  0.001666843
## HMC  0.003310215
## NVDA 0.018311352
##
## [[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
##      PortMean  PortSigma         SR
## 1 0.008233120 0.05920785 0.14390597
## 2 0.007027397 0.06761299 0.06424233
## 3 0.009018927 0.08107091 0.13867465
## 4 0.009766939 0.06258960 0.22102185
## 5 0.010072156 0.07610063 0.19567476
##
## [[4]]
## # A tibble: 11 x 6
##    ints          lowerval sig_optim retn_optim  numb portID
##    <fct>            <dbl>     <dbl>      <dbl> <int>  <int>
##  1 (0.003,0.004]    0.003    0.0592    0.00392     1    350
##  2 (0.004,0.005]    0.004    0.0527    0.00494     6    187
##  3 (0.005,0.006]    0.005    0.0513    0.00559    36    331
##  4 (0.006,0.007]    0.006    0.0531    0.00697    90    584
##  5 (0.007,0.008]    0.007    0.0535    0.00799   146    490
##  6 (0.008,0.009]    0.008    0.0533    0.00830   159    401
##  7 (0.009,0.01]     0.009    0.0563    0.00929   145     20
##  8 (0.01,0.011]     0.01     0.0595    0.0109     74     48
##  9 (0.011,0.012]    0.011    0.0615    0.0112     30    438
## 10 (0.012,0.013]    0.012    0.0707    0.0120     12    693
## 11 (0.013,0.014]    0.013    0.0859    0.0133      1    408
##
## [[5]]
```


PORTFOLIO POSSIBILITIES
PortMean
0.012
0.010
0.008
0.006
0.004
0.05
0.06
0.07
0.08
0.09
PortSigma

```
##
## [[6]]
```



CODING ALGORITHM

1. Retrieve financial data (getSymbols() is used) and create a vector with respective ticker symbols. To calculate the monthly returns for the stocks, specify the time period in question (periodReturn() is used).
2. Retrieve monthly share data and merge the data set with returns.
3. Create an arbitrary function (we've used MeanVarPort) to calculate portfolio optimization metrics for simulated portfolios.
4. Calculate mean returns vector and covariance matrix using the functions 'colMeans' and 'var' respectively.
5. Set seed to 12 to get consistent results; provide input to simulate portfolio weights to construct 100N portfolios (7 securities in this case).
6. Normalize weights, so that we get proper weights that sum to 1; assign weight to each stock using a 'for' loop.
7. Use matrix multiplication of the transpose of the weights and the stock mean to calculate Portfolio Mean.
8. Take a matrix transpose of weights followed by matrix multiplication with covariance; the resultant matrix is multiplied by weights and then the square root is calculated for the final matrix in order to get Portfolio Sigma.
9. Calculate the Sharpe Ratio : ((Portmean - risk free rate)/PortSigma))

10. Plot the portfolio possibilities.
11. Retrieve min and max of PortMean; get rounded returns by sequences of one-thousandths.
12. Use dplyr chain commands to create optimal portfolio data frame.
13. Plot optimal portfolios and annotate both optimal portfolio and the minimum variance portfolio.

RESULTS

Refer Pg(6-7),

The output achieves the following:

- In [1], we get the mean return vector for each stock.
- In [2], we get the covariance of the daily stock returns.
- In [3], the information regarding Weights, Mean, Sigma, and SR for each simulated portfolio is displayed. We print only the first 5 portfolios to verify, although the code takes into account 100N portfolios.
- Further, the plots help visualize the optimal portfolio and the minimum variance portfolio.
- In the plot depicting 'Portfolio Possibilities', the desirable portfolios in the feasible set are upper left (i.e. more returns, less risk). This means that for a particular portfolio, we want the Mean (PortMean) to be high and the standard deviation (PortSigma) to be significantly low.

CONCLUDING REMARKS

Optimization problems such as the one we have above can be solved either analytically or computationally. For a portfolio to be optimal,it should have the highest return per risk. Here, we've tried to find the optimal portfolio through computational means. Based on the feasible set, we recognize two portfolios matching the required threshold of risk and return, and out of them, optimal portfolio in our judgement would be the one annotated 'optimal portfolio' in the plot, for the reason that it offers slightly better returns than the adjacent ones for the same value of risk. Through this simulation, we could find an optimal portfolio such that the selected portfolio dominates (and is not dominated by) other choices. Returns are high, and the risk is low.

Similarly, the minimum variance portfolio gives maximum returns (mean) for minimum risk (standard deviation). This portfolio might be suitable for risk-averse investors.

The problem inputs such as the choice of risk-free rate, number of Monte Carlo Simulations and even metrics to determine our optimal portfolio can be changed as per requirement.

Perspective on The Monte Carlo Simulation Method

The Monte Carlo Simulation (MCS) method is an excellent way of estimating uncertainty in a model, and it works really well when the models are nonlinear and chaotic. The notion is that if we are aware there are several components that make up a model, and each component has some element of randomness, we can find the whole range of possible outcomes by simulating the results of each element and understanding how it affects the overall model. In addition, the outcomes help us visualize the distribution of computed results.

In our opinion, these are the pros and cons of the MCS method:

Pros:

- Solving optimization problems using the traditional analytical approaches, such as linear algebra and matrix operations is complex (especially for large data sets). MCS helps avoid the complexity of mathematical computations as we get computerized simulations here.
- It provides the flexibility to factor in and simulate a large range of values for various inputs.
- The method is fast as it does not require repeated model-fitting. However, some argue that it is computationally inefficient at times when you have a large amount of variables bound to different constraints.
- The precision is higher due to smoothness of the sampling distribution.
- It is easy to visualize the output and reach conclusions regarding optimal portfolios.

Cons:

- It is important that the assumptions are fair in order to avoid an output that is misleading, because the output is only as good as the inputs.
- Another downfall is that MCS tends to underestimate the probability of extreme events such as a financial crisis.
- Further, a simulation like this is unable to factor in the behavioral aspects of finance and the irrationality exhibited by market participants.