University of Strathclyde
Department of Computer Science

M.Sc. Financial Technology
(2017/2018)

CS983 – Evolutionary Computation for Finance

Assessment

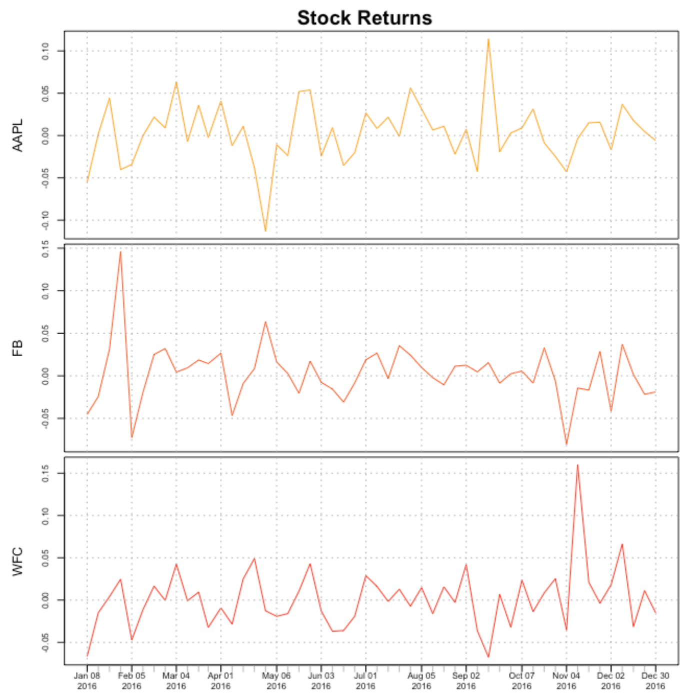
"Report of investigating the development of a Genetic Algorithm to build optimal investments portfolio"

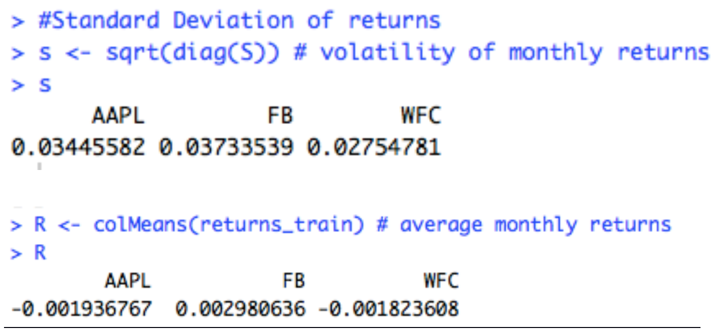
Joshua Eick - 201779687

March 15, 2018

Report of investigated the development of a GA-based solution to optimize the weightings of a given portfolio of assets

I. Choice of assets




```
> #Standard Deviation of returns
> s <- sqrt(diag(S)) # volatility of monthly returns
> s
      AAPL         FB        WFC
0.03445582 0.03733539 0.02754781
```

```
> R <- colMeans(returns_train) # average monthly returns
> R
        AAPL           FB          WFC
-0.001936767  0.002980636 -0.001823608
```

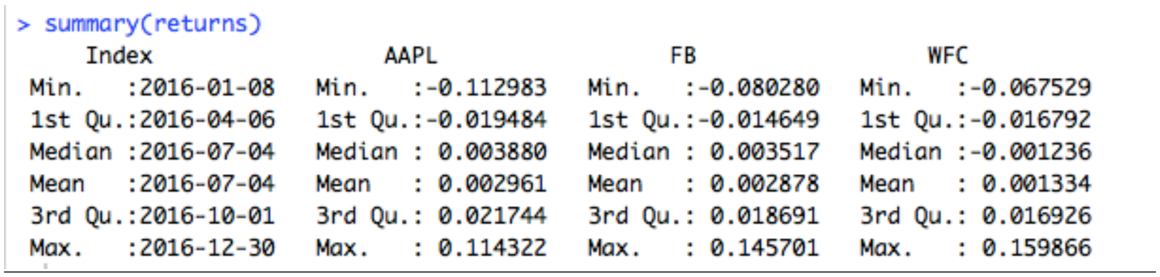
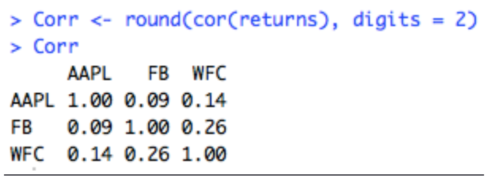

```
> summary(returns)
     Index                 AAPL                FB                 WFC
 Min.   :2016-01-08   Min.   :-0.112983   Min.   :-0.080280   Min.   :-0.067529
 1st Qu.:2016-04-06   1st Qu.:-0.019484   1st Qu.:-0.014649   1st Qu.:-0.016792
 Median :2016-07-04   Median : 0.003880   Median : 0.003517   Median :-0.001236
 Mean   :2016-07-04   Mean   : 0.002961   Mean   : 0.002878   Mean   : 0.001334
 3rd Qu.:2016-10-01   3rd Qu.: 0.021744   3rd Qu.: 0.018691   3rd Qu.: 0.016926
 Max.   :2016-12-30   Max.   : 0.114322   Max.   : 0.145701   Max.   : 0.159866
```

Illustration 1

The development of the application of Genetic Algorithm is good optimize investment portfolio? In order to understand the process of GA to optimize investment portfolio it's decide to take as perspective of an investor. It's chosen three big stocks from different sectors base on the market capitalization in United States, which are Apple, Facebook, and Wells Fargo. Hence, the portfolio can mitigate unsystematic risk. Also, the position it's place is on risk averse investor/conservative. Therefore, the first criteria was market capitalization to reduce volatility because the bigger the company, the less volatility. It's shows in the illustration above that the standard deviation is around 2% to 3%, which is very small and that way you reduce risk. In addition, in the table above provide the mean of the weekly return of these stocks and is approximately .20. This indicates that are very conservative stocks.



```
> Corr <- round(cor(returns), digits = 2)
> Corr
     AAPL   FB  WFC
AAPL 1.00 0.09 0.14
FB   0.09 1.00 0.26
WFC  0.14 0.26 1.00
```

Illustration 2

The other main criteria which is take into account to select the stocks is diversification between sectors and products. For instance, Apple is from the sector of technology, Facebook is from the social media sector and Well Fargo is from the banking sector. In additions, this companies are so big that have diversity of products, which means if one product go do not have profitability the companies still performing good because have other products on the market and as well distributed products around the word. Thus, these companies are diversified among products and geographically. The correlation matrix above shows that the correlation among companies is small positive correlation. Its, suggest it that is not that much advantage of diversification among this portfolio.

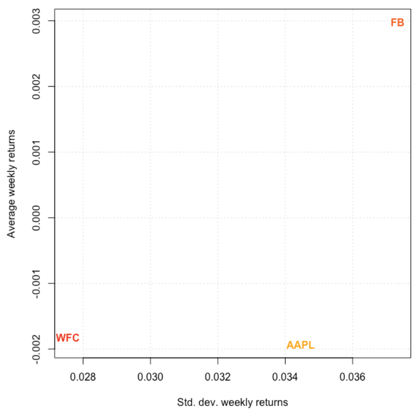
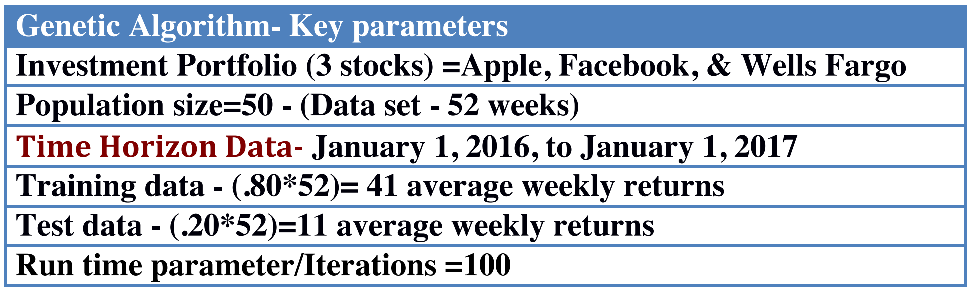


Illustration 3

Finally, its show in the illustration above the trade of return and risk form different companies. Indeed, initial this plot shows that GA will place more weight on the Facebook stock because have higher return and small weight on Well Fargo stock because have less return.

Overall, my data set are these 3 stocks mention above from the time horizon of January 1, 2016, to January 1, 2107. Having said that, above all, the main criteria I take three securities/stocks is to mitigate estimation risk problem and in that way can be investigate deeply the process of GA. Estimation risk problem in this case is when the expected returns are time varying, so the estimation are changing over time. Thus, fewer stocks in the portfolio equals less data so require less estimation. As result reduce estimation error.

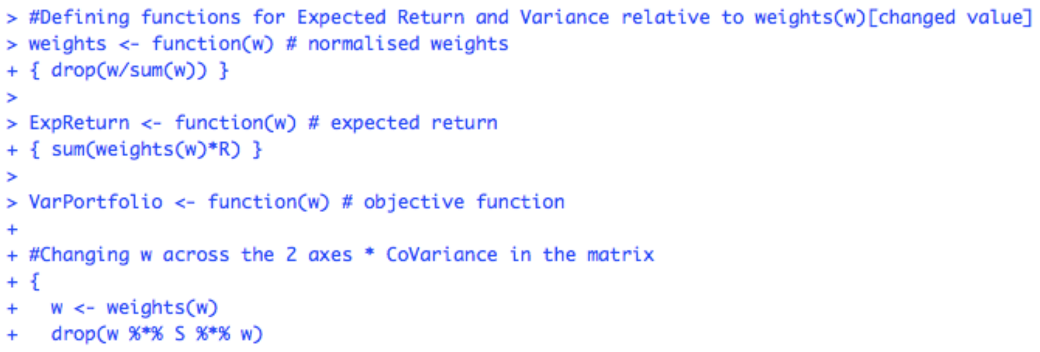
II.Details of the GA



Genetic Algorithm- Key parameters
Investment Portfolio (3 stocks) =Apple, Facebook, & Wells Fargo
Population size=50 - (Data set - 52 weeks)
Time Horizon Data- January 1, 2016, to January 1, 2017
Training data - (.80*52)= 41 average weekly returns
Test data - (.20*52)=11 average weekly returns
Run time parameter/Iterations =100

Illustration 4

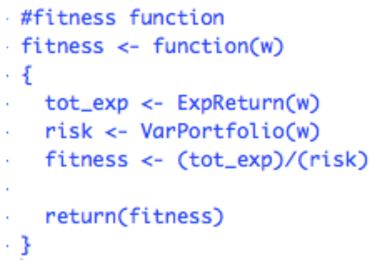
First, it's divide the data between 80% percent of training set and 20% test set because is widely agree that is good proportion to divide the data.



```
> #Defining functions for Expected Return and Variance relative to weights(w)[changed value]
> weights <- function(w) # normalised weights
+ { drop(w/sum(w)) }
>
> ExpReturn <- function(w) # expected return
+ { sum(weights(w)*R) }
>
> VarPortfolio <- function(w) # objective function
+
+ #Changing w across the 2 axes * CoVariance in the matrix
+ {
+   w <- weights(w)
+   drop(w %*% S %*% w)
```

Illustration 5

Secondly, in order to build the fitness function to optimize the investment portfolio it's defined the following variables: weights, expected return, standard deviation.

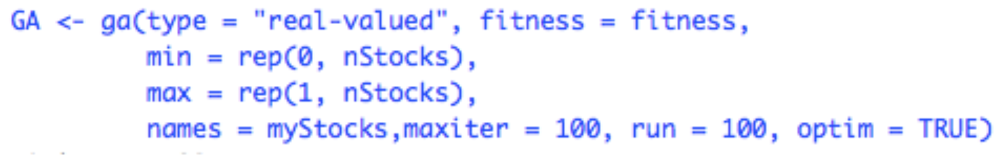


```
#fitness function
fitness <- function(w)
{
  tot_exp <- ExpReturn(w)
  risk <- VarPortfolio(w)
  fitness <- (tot_exp)/(risk)

  return(fitness)
}
```

Illustration 6

Third, after define this variable it's built fitness function. In fact, our fitness function is going to combine risk and return on one function because GA Package can't permit a multi-objective function. Multi – objective function means risk and return variable separate in a different function. The fitness function illustrates above represent the maximum sharpe ratio. In other words, my fitness function represents the maximum sharpe ratio between all possible combinations of weights between 3 stocks (securities) that GA generated. The reason why is the maximum Sharpe ratio is because investment portfolio optimization is a key measure to select the optimal portfolio. According to Nickolas Lioudisv (2017) is a measure of return that is used to compare the performance by adjusting for risk. The sharpe ratio means that for each unit of risk (standard deviation/volatility) you receive x% of return. In other words, the fitness function equals the formula of sharpe ratio which is the total return divide by risk (variance). In addiction, higher sharpe ratio equals when risk free rate line is tangent with the efficient frontier/Pareto Front.

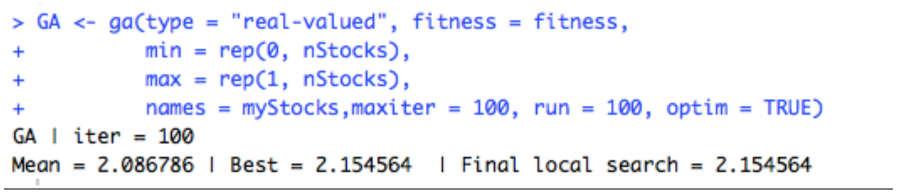


```
GA <- ga(type = "real-valued", fitness = fitness,
         min = rep(0, nStocks),
         max = rep(1, nStocks),
         names = myStocks,maxiter = 100, run = 100, optim = TRUE)
```

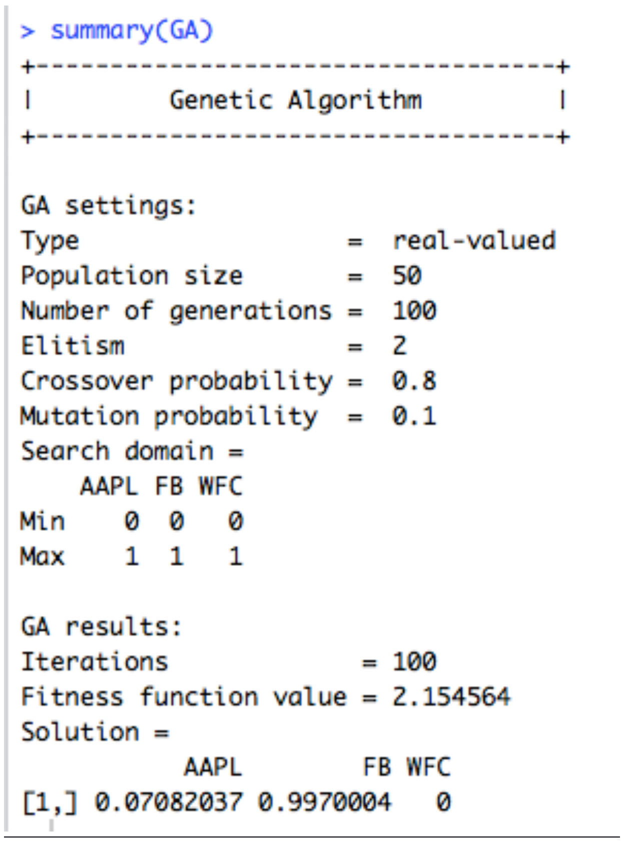
Illustration 7

Having said that, it's build the command to run the Genetic Algorithm. The process of GA works, first it's select the number of chromosomes in this case three chromosomes which represent each weight of each stock that with GA we going to find the optimal weights. These chromosomes are the type of real-valued. This means that the weight are real data and not binary Second, we input the fitness function, which already explain above. Afterward, is input the "min" and "max" input, which demonstrate the maximum weight can have a stocks in the portfolio (100% or the minimum weight of 0%). Thus, the assets take values between 0 to 1. In the above illustration "names" means the data uploaded from yahoo. The "run" command indicates how many times the is going to run the Genetic Algorithms to find the higher sharpe ratio

III. Results on the evolved solution



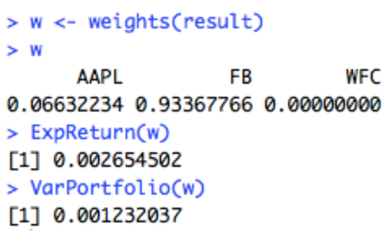
```
> GA <- ga(type = "real-valued", fitness = fitness,
+          min = rep(0, nStocks),
+          max = rep(1, nStocks),
+          names = myStocks,maxiter = 100, run = 100, optim = TRUE)
GA | iter = 100
Mean = 2.086786 | Best = 2.154564  | Final local search = 2.154564
```



```
> summary(GA)
+-----------------------------------+
|         Genetic Algorithm         |
+-----------------------------------+

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  100
Elitism               =  2
Crossover probability =  0.8
Mutation probability  =  0.1
Search domain =
    AAPL FB WFC
Min    0  0   0
Max    1  1   1

GA results:
Iterations             = 100
Fitness function value = 2.154564
Solution =
           AAPL        FB WFC
[1,] 0.07082037 0.9970004   0
```



```
> w <- weights(result)
> w
      AAPL         FB        WFC
0.06632234 0.93367766 0.00000000
> ExpReturn(w)
[1] 0.002654502
> VarPortfolio(w)
[1] 0.001232037
```

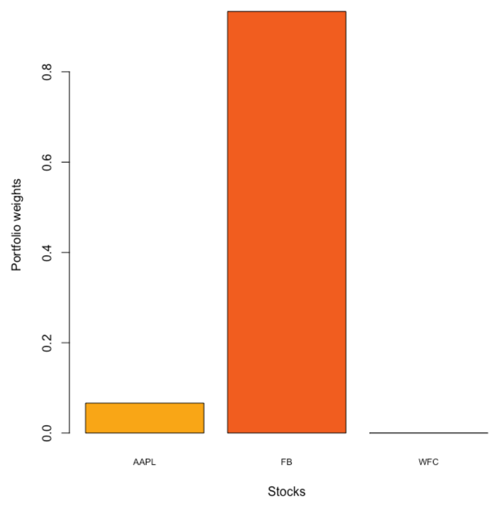


Illustration 8

Evaluating the performance of the application of GA, the illustration above shows the result of the fitness function over training data and it's identify the optimal investment portfolio. The best solution to optimize the portfolio of three stocks is to invest the money in the company of Apple, Facebook and Wells Fargo based on these weight respectively, 6.63%, 93.36% and 0% (shows in the bar graph). Thus, investing more in Facebook will give more risk-adjusted returns. My opinion and I presume it's widely agree that this portfolio with this respectively weight is the optimal portfolio because is the higher sharper ratio in the efficient frontier/Pareto Front.

The most significant value is the fitness value, which is the maximum sharpe ratio and equals 2.15. This means that with one unit of risk this portfolio with the optimal weights will achieve 2.15% of return. In addiction, the mean of the sharpe ratio of the 100 generations is 2.09%. The expected return calculate by GA with the optimal weights was .26%. Indeed, the GA is good tool because maintain a constant fitness value and higher expected return.

Other important variable in order to explain the performance of GA is the population size, which is 50. Population size means that the application of Genetic Algorithm takes 50 possible vectors combinations of weights to calculate the fitness value. In other words, each generations of 50 population size take the higher sharpe ratio. Then, each generation will take the higher fitness value from the 50 population size/vectors, in this case 50. The iterations which is the same as generations is 100 and as well means how many times is going to run the application of GA. The crossover probability of 80% means that the population/vectors of 50 is going to take 80% from each vector to make a new chromosome/weight/son Then, the mutation occur with 10% probability.

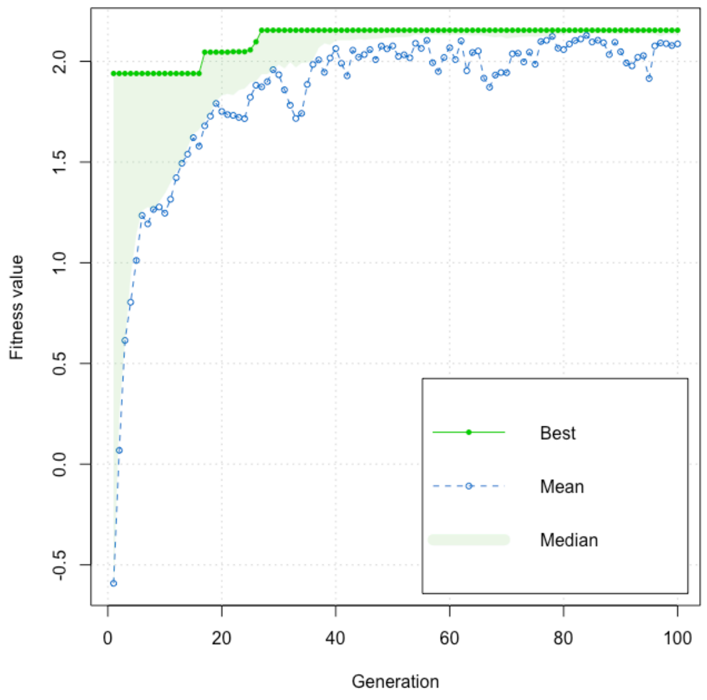
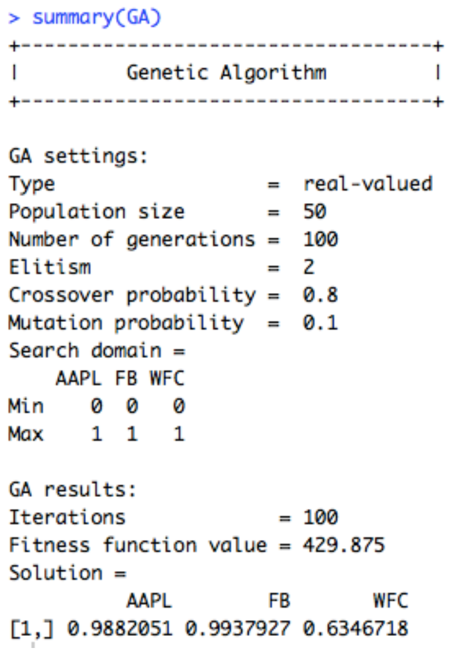


Illustration 9

Evaluating the performance of GA with the fitness value throughout the illustration above shows that after 30 generation the curve flatten (green curve). The application of GA identify a mean of fitness value/sharpe ratio is 2.09. Therefore, you do not have to increase the command of generations more than 100 iterations to achieve the best fitness value. In other words, you can have a higher fitness value/higher sharpe ratio between the range of 30 and 60 generations because in this range the fitness value remain constant at the higher mean and median.



```
> summary(GA)
+-----------------------------------+
|         Genetic Algorithm         |
+-----------------------------------+

GA settings:
Type                  =  real-valued
Population size       =  50
Number of generations =  100
Elitism               =  2
Crossover probability =  0.8
Mutation probability  =  0.1
Search domain =
    AAPL  FB  WFC
Min    0   0    0
Max    1   1    1

GA results:
Iterations             = 100
Fitness function value = 429.875
Solution =
          AAPL        FB       WFC
[1,] 0.9882051 0.9937927 0.6346718
```

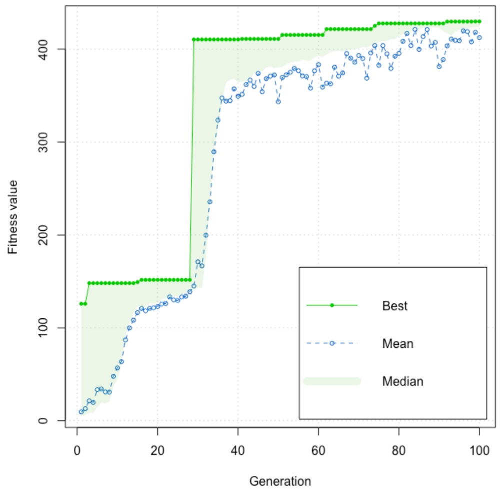
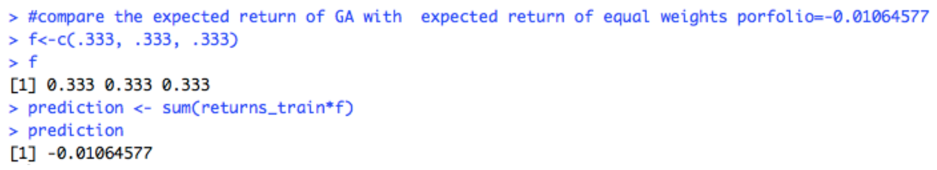


Illustration 10

On the other hand, initially the balance used for return and the risk criteria was the same percent for both, which is 50% of return and 50% of risk. The impact altering this balance to a 30% return and 70% risk in terms of fitness value was that the fitness value increase significantly to 429.0. In fact, choosing this balance does not make sense in finance principles. However, in terms of analyze the process of GA it's suggest that fitness function and generations in GA have a relationship. The relationship shows that depend of the data/formula of the fitness function how much generations will take to produce the best fitness value. Therefore, in this case took around 40 to 60 generations to arrive to the highest fitness value. Hence, this study/test indicates that changing the balance of return and risk results in a possible explanation that 100 iterations parameter is good estimate to generate the best fitness value in general.

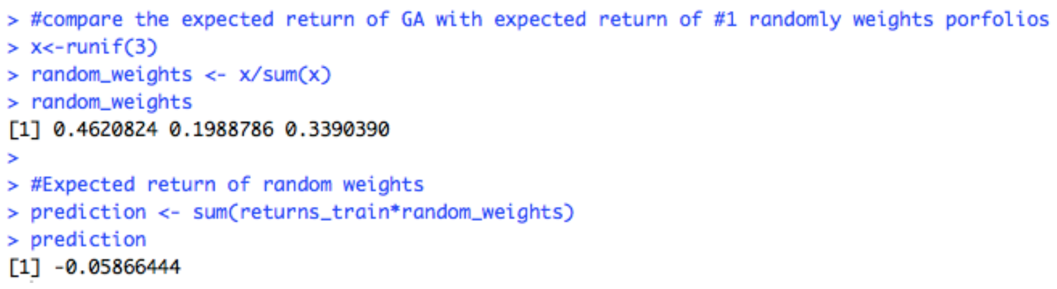
IV. Analysis of the evolved portfolio(s)



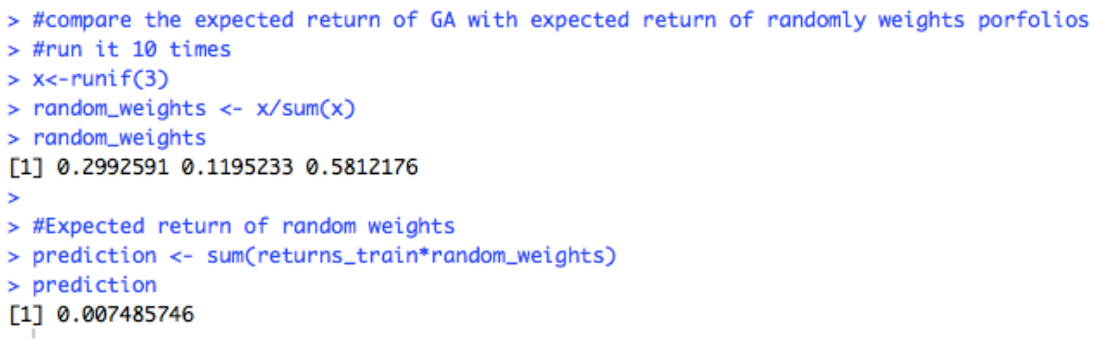
```
> #compare the expected return of GA with  expected return of equal weights porfolio=-0.01064577
> f<-c(.333, .333, .333)
> f
[1] 0.333 0.333 0.333
> prediction <- sum(returns_train*f)
> prediction
[1] -0.01064577
```

Illustration 11

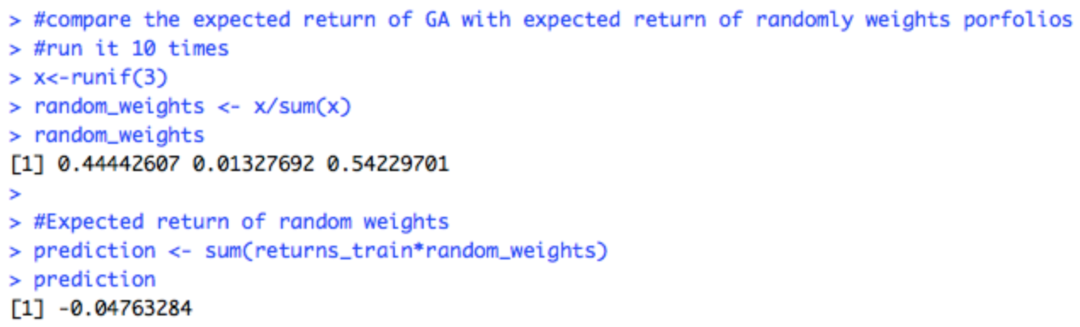
Furthermore, in order to compare how superior is the application of GA to build optimal portfolio to make investment decision it's create an equal weighted portfolio and randomly weighted portfolios. In order to compare it with the measure of expected return of the optimal portfolio generate by GA. First, I create an equal weighed portfolio and the weekly expected return was -1.06% and the weekly expected return of optimal portfolio build by GA was .26%. Thus, GA generates a better portfolio of investment compared to the equal weighted portfolio based on expected weekly returns.



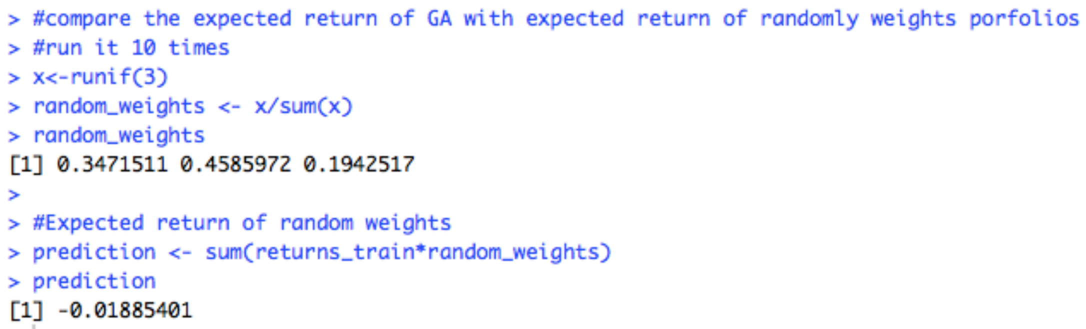
```
> #compare the expected return of GA with expected return of #1 randomly weights porfolios
> x<-runif(3)
> random_weights <- x/sum(x)
> random_weights
[1] 0.4620824 0.1988786 0.3390390
>
> #Expected return of random weights
> prediction <- sum(returns_train*random_weights)
> prediction
[1] -0.05866444
```



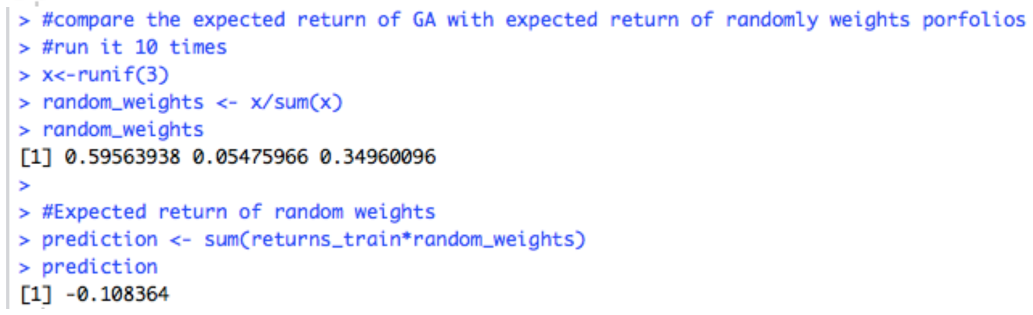
```
> #compare the expected return of GA with expected return of randomly weights porfolios
> #run it 10 times
> x<-runif(3)
> random_weights <- x/sum(x)
> random_weights
[1] 0.2992591 0.1195233 0.5812176
>
> #Expected return of random weights
> prediction <- sum(returns_train*random_weights)
> prediction
[1] 0.007485746
```



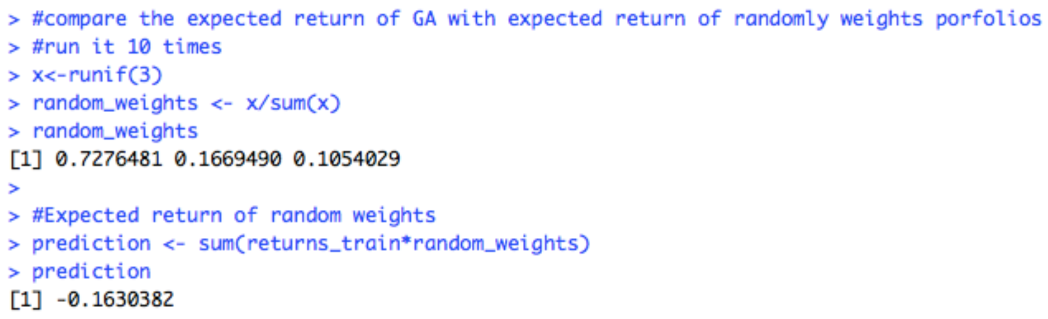
```
> #compare the expected return of GA with expected return of randomly weights porfolios
> #run it 10 times
> x<-runif(3)
> random_weights <- x/sum(x)
> random_weights
[1] 0.44442607 0.01327692 0.54229701
> 
> #Expected return of random weights
> prediction <- sum(returns_train*random_weights)
> prediction
[1] -0.04763284
```



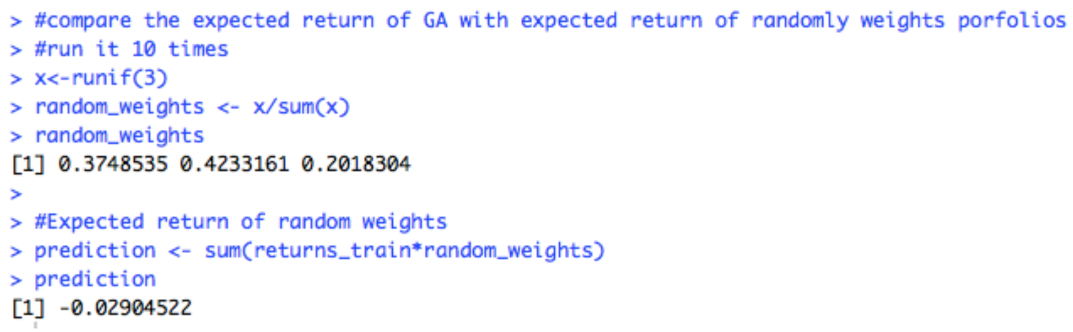
```
> #compare the expected return of GA with expected return of randomly weights porfolios
> #run it 10 times
> x<-runif(3)
> random_weights <- x/sum(x)
> random_weights
[1] 0.3471511 0.4585972 0.1942517
> 
> #Expected return of random weights
> prediction <- sum(returns_train*random_weights)
> prediction
[1] -0.01885401
```



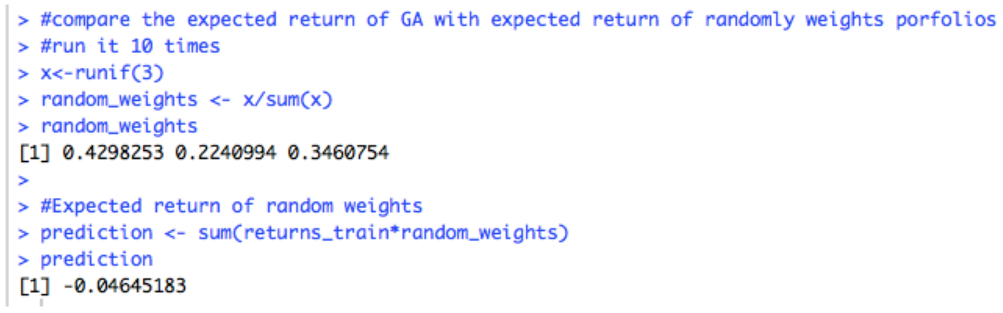
```
> #compare the expected return of GA with expected return of randomly weights porfolios
> #run it 10 times
> x<-runif(3)
> random_weights <- x/sum(x)
> random_weights
[1] 0.59563938 0.05475966 0.34960096
> 
> #Expected return of random weights
> prediction <- sum(returns_train*random_weights)
> prediction
[1] -0.108364
```



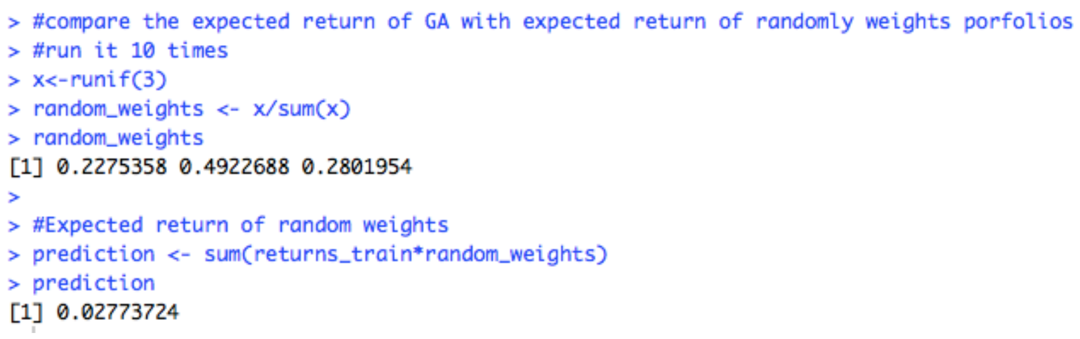
```
> #compare the expected return of GA with expected return of randomly weights porfolios
> #run it 10 times
> x<-runif(3)
> random_weights <- x/sum(x)
> random_weights
[1] 0.7276481 0.1669490 0.1054029
> 
> #Expected return of random weights
> prediction <- sum(returns_train*random_weights)
> prediction
[1] -0.1630382
```



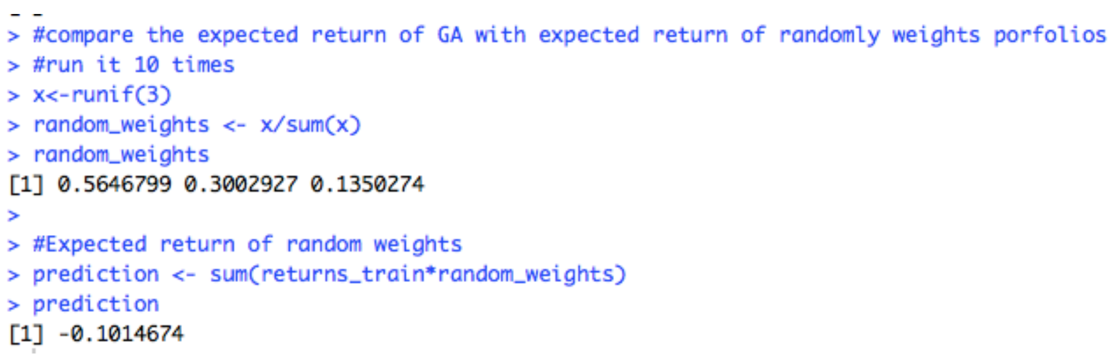
```
> #compare the expected return of GA with expected return of randomly weights porfolios
> #run it 10 times
> x<-runif(3)
> random_weights <- x/sum(x)
> random_weights
[1] 0.3748535 0.4233161 0.2018304
>
> #Expected return of random weights
> prediction <- sum(returns_train*random_weights)
> prediction
[1] -0.02904522
```



```
> #compare the expected return of GA with expected return of randomly weights porfolios
> #run it 10 times
> x<-runif(3)
> random_weights <- x/sum(x)
> random_weights
[1] 0.4298253 0.2240994 0.3460754
>
> #Expected return of random weights
> prediction <- sum(returns_train*random_weights)
> prediction
[1] -0.04645183
```



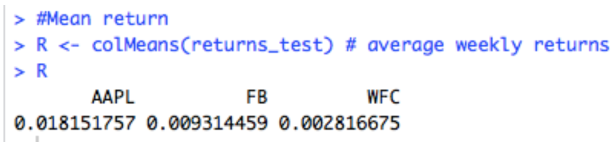
```
> #compare the expected return of GA with expected return of randomly weights porfolios
> #run it 10 times
> x<-runif(3)
> random_weights <- x/sum(x)
> random_weights
[1] 0.2275358 0.4922688 0.2801954
>
> #Expected return of random weights
> prediction <- sum(returns_train*random_weights)
> prediction
[1] 0.02773724
```



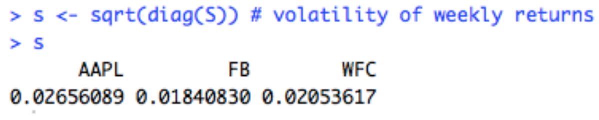
```
> #compare the expected return of GA with expected return of randomly weights porfolios
> #run it 10 times
> x<-runif(3)
> random_weights <- x/sum(x)
> random_weights
[1] 0.5646799 0.3002927 0.1350274
>
> #Expected return of random weights
> prediction <- sum(returns_train*random_weights)
> prediction
[1] -0.1014674
```

Illustration 12

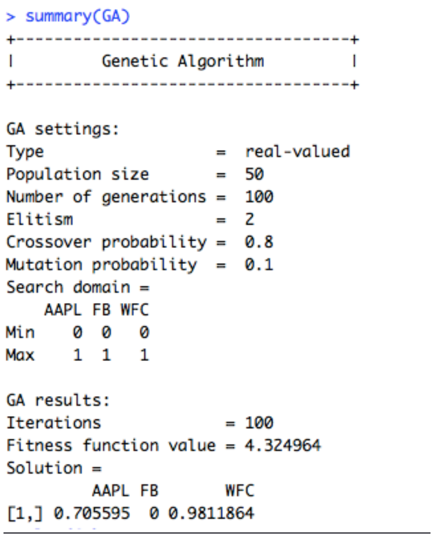
Secondly, to better verify the performance of the GA building an optimal investment portfolio it's generated ten random weighted portfolios that is shows in the illustration above to compare with the measure of expected return. As result, the optimal portfolio generate by GA outperformed the randomly portfolio by nine times. Represents that 90% of time the expected return of the optimal portfolio outperformed the randomly created portfolios. Hence, it's suggested according to the result that GA is good application tool to optimized portfolio.



```
> #Mean return
> R <- colMeans(returns_test) # average weekly returns
> R
       AAPL          FB         WFC 
0.018151757 0.009314459 0.002816675 
```



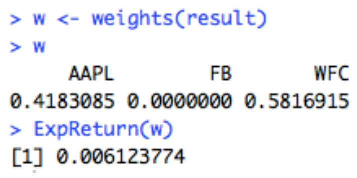
```
> s <- sqrt(diag(S)) # volatility of weekly returns
> s
      AAPL         FB        WFC 
0.02656089 0.01840830 0.02053617 
```



```
> summary(GA)
+-----------------------------------+
|         Genetic Algorithm         |
+-----------------------------------+

GA settings: 
Type                  =  real-valued 
Population size       =  50 
Number of generations =  100 
Elitism               =  2 
Crossover probability =  0.8 
Mutation probability  =  0.1 
Search domain = 
    AAPL FB WFC
Min    0  0   0
Max    1  1   1

GA results: 
Iterations             = 100 
Fitness function value = 4.324964 
Solution = 
         AAPL FB       WFC
[1,] 0.705595  0 0.9811864
```



```
> w <- weights(result)
> w
     AAPL        FB       WFC 
0.4183085 0.0000000 0.5816915 
> ExpReturn(w)
[1] 0.006123774
```

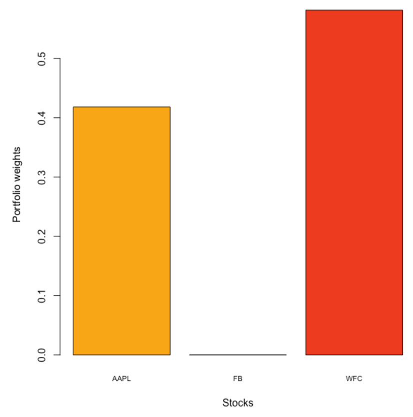


Illustration 13

Finally, in order to see how the portfolio performs on the future it's taken the test set data of the year (2016-2017) that is the last 11 weeks of the year and calculate the expected weekly return. Afterward, it's compare with the average weekly-expected return of the optimal portfolio of the training set data of the year (2016-2017) that is the first 41 weeks. Shows in the illustration above that the fitness value/sharpe ratio with the test set was 4.32 and the expected return was .61%. In contrast, the sharpe ratio and expected return calculate by GA initially (training set) was respectively, 2.15 and .26%. Indeed, it's believe that GA is good tool because maintain a constant fitness value and constant expected return.

In conclusion, it's suggests that the evolve process of GA for the optimization of investment portfolio create value on the investment decision. Hence, can generate the higher fitness value/sharpe ratio, which means higher risk adjusted returns for investor.

References:

Nickolas Lioudis (2017) Understanding The Sharpe Ratio Available at: https://www.investopedia.com/articles/07/sharpe_ratio.asp (Accessed: March 6, 2018)

Appendices

```
#Joshua Eick

library(quantmod)
library(GA)
library(timeSeries)


#part I
#Uploading my stocks and my returns are weekly
myStocks <- c("AAPL", "FB", "WFC")
getSymbols(myStocks, src = "yahoo", from="2016-01-01", to="2017-01-01",
   freq="week")
returns <- lapply(myStocks, function(s)
  weeklyReturn(eval(parse(text = s)),
             subset = "2016::2017"))

returns <- do.call(cbind,returns)
colnames(returns) <- myStocks
#52 weeks
nrow(returns)
returns



#summary returns
summary(returns)


#SPLIT TRAIN AND TEST
smp_size <- floor(0.80 * nrow(returns))


train_ind <- sample(seq_len(nrow(returns)), size = smp_size)

returns_train <- returns[train_ind, ]
returns_train
returns_test <- returns[-train_ind, ]
returns_test


#Plot weeklyReturns

plot(as.timeSeries(returns), at = "chic", minor.ticks="month",
    mar.multi = c(0.2, 5.1, 0.2, 1.1), oma.multi = c(4, 0, 4, 0),
    col = .colorwheelPalette(10), cex.lab = 0.8, cex.axis = 0.8)
title("Stock Returns")


#Number of Stocks
nStocks <- ncol(returns_train) # number of portfolio assets
nStocks
#Mean return
R <- colMeans(returns_train) # average weekly returns
R

#Correlation Matrix of Returns
Corr <- round(cor(returns), digits = 2)
Corr
pairs(returns, cex.labels = NULL)

#Covariance of Returns
S <- cov(returns_train) # covariance matrix of weekly returns
S


#Standard Deviation of returns
s <- sqrt(diag(S))
s

#Part II

#Plotting Mean Return to Std.Dev
plot(s, R, type = "n", panel.first = grid(),
    xlab = "Std. dev. weekly returns", ylab = "Average weekly returns")
text(s, R, names(R), col = .colorwheelPalette(10), font = 2)


#Defining functions
weights <- function(w) # normalised weights
{ drop(w/sum(w)) }

ExpReturn <- function(w) # expected return
{ sum(weights(w)*R) }

VarPortfolio <- function(w) # objective function

  #Changing w across the 2 axes * CoVariance in the matrix
{
  w <- weights(w)
  drop(w %*% S %*% w)
```

```
}



#fitness function
fitness <- function(w)
{
  tot_exp <- ExpReturn(w)
  risk <- VarPortfolio(w)
  fitness <- (tot_exp)/(risk)

  return(fitness)
}



GA <- ga(type = "real-valued", fitness = fitness,
         min = rep(0, nStocks),
         max = rep(1, nStocks),
         names = myStocks,maxiter = 100, run = 100, optim = TRUE)



#part IIIa
summary(GA)
plot(GA)
GA@solution


result=GA@solution/sum(GA@solution)
result


#Optimal weights
w <- weights(result)
w
#Expected return using the optimal weights
ExpReturn(w)
#Expected variance using the otimal weights
VarPortfolio(w)

barplot(w, xlab = "Stocks", ylab = "Portfolio weights",
        cex.names = 0.7, col = .colorwheelPalette(10))
```

```
#IIIb Impact of changing balance of risk and return

#Uploading my stocks and my returns are weekly
myStocks <- c("AAPL", "FB", "WFC")
getSymbols(myStocks, src = "yahoo", from="2016-01-01", to="2017-01-01",
   freq="week")
returns <- lapply(myStocks, function(s)
  weeklyReturn(eval(parse(text = s)),
              subset = "2016::2017"))

returns <- do.call(cbind,returns)
colnames(returns) <- myStocks
#52 weeks
nrow(returns)
returns



#summary returns
summary(returns)


#SPLIT TRAIN AND TEST
smp_size <- floor(0.80 * nrow(returns))


train_ind <- sample(seq_len(nrow(returns)), size = smp_size)

returns_train <- returns[train_ind, ]
returns_train
returns_test <- returns[-train_ind, ]
returns_test


#Plot weeklyReturns for Portfolio Constituents
```

```
plot(as.timeSeries(returns), at = "chic", minor.ticks="month",
    mar.multi = c(0.2, 5.1, 0.2, 1.1), oma.multi = c(4, 0, 4, 0),
    col = .colorwheelPalette(10), cex.lab = 0.8, cex.axis = 0.8)
title("Stock Returns")


#Number of Stocks
nStocks <- ncol(returns_train) # number of portfolio assets
nStocks
#Mean return
R <- colMeans(returns_train) # average weekly returns
R

#Correlation Matrix of Returns
Corr <- round(cor(returns), digits = 2)
Corr
pairs(returns, cex.labels = NULL)

#Covariance of Returns
S <- cov(returns_train) # covariance matrix of weekly returns
S


#Standard Deviation of returns
s <- sqrt(diag(S)) # volatility of weekly returns
s

#Part II

#Plotting Mean Return to Std.Dev
plot(s, R, type = "n", panel.first = grid(),
    xlab = "Std. dev. weekly returns", ylab = "Average weekly returns")
text(s, R, names(R), col = .colorwheelPalette(10), font = 2)


#Defining functions
weights <- function(w) # normalised weights
{ drop(w/sum(w)) }

ExpReturn <- function(w) # expected return
{ sum(weights(w)*R) }

VarPortfolio <- function(w) # objective function

  #Changing w across the 2 axes * CoVariance in the matrix
{
  w <- weights(w)
  drop(w %*% S %*% w)
}


#fitness function
w=0.3
fitness <- function(w)
{
  tot_exp <- ExpReturn(w)
  risk <- VarPortfolio(w)
  fitness <- ((tot_exp)*w)/((risk)*(1-w))

  return(fitness)
}




GA <- ga(type = "real-valued", fitness = fitness,
         min = rep(0, nStocks),
         max = rep(1, nStocks),
         names = myStocks,maxiter = 100, run = 100, optim = TRUE)


summary(GA)
plot(GA)
GA@solution


result=GA@solution/sum(GA@solution)
result

#Conlusion what is change is the process more longer to indentify the fitness and
   have been constant

#Optimal weights
w <- weights(result)
w
#Expected return
ExpReturn(w)
#Expecteed return
VarPortfolio(w)

barplot(w, xlab = "Stocks", ylab = "Portfolio weights",
       cex.names = 0.7, col = .colorwheelPalette(10))
```

```
#part IVa

#Uploading my stocks and my returns are weekly
myStocks <- c("AAPL", "FB", "WFC")
getSymbols(myStocks, src = "yahoo", from="2016-01-01", to="2017-01-01",
   freq="week")
returns <- lapply(myStocks, function(s)
  weeklyReturn(eval(parse(text = s)),
             subset = "2016::2017"))

returns <- do.call(cbind,returns)
colnames(returns) <- myStocks
#52 weeks
nrow(returns)
returns



#summary returns
summary(returns)


#SPLIT TRAIN AND TEST
smp_size <- floor(0.80 * nrow(returns))


train_ind <- sample(seq_len(nrow(returns)), size = smp_size)
```

```
returns_train <- returns[train_ind, ]
returns_train
returns_test <- returns[-train_ind, ]
returns_test


#Plot weeklyReturns for Portfolio Constituents

plot(as.timeSeries(returns), at = "chic", minor.ticks="month",
    mar.multi = c(0.2, 5.1, 0.2, 1.1), oma.multi = c(4, 0, 4, 0),
    col = .colorwheelPalette(10), cex.lab = 0.8, cex.axis = 0.8)
title("Stock Returns")


#Number of Stocks
nStocks <- ncol(returns_train) # number of portfolio assets
nStocks
#Mean return
R <- colMeans(returns_train) # average weekly returns
R

#Correlation Matrix of Returns
Corr <- round(cor(returns), digits = 2)
Corr
pairs(returns, cex.labels = NULL)

#Covariance of Returns
S <- cov(returns_train) # covariance matrix of weekly returns
S


#Standard Deviation of returns
s <- sqrt(diag(S)) # volatility of weekly returns
s

#Part II

#Plotting Mean Return to Std.Dev
plot(s, R, type = "n", panel.first = grid(),
    xlab = "Std. dev. weekly returns", ylab = "Average weekly returns")
text(s, R, names(R), col = .colorwheelPalette(10), font = 2)


#Defining functions for Expected Return and Variance relative to weights(w)[changed
   value]
```

```
weights <- function(w) # normalised weights
{ drop(w/sum(w)) }

ExpReturn <- function(w) # expected return
{ sum(weights(w)*R) }

VarPortfolio <- function(w) # objective function

  #Changing w across the 2 axes * CoVariance in the matrix
{
  w <- weights(w)
  drop(w %*% S %*% w)
}



#fitness function
fitness <- function(w)
{
  tot_exp <- ExpReturn(w)
  risk <- VarPortfolio(w)
  fitness <- (tot_exp)/(risk)

  return(fitness)
}



GA <- ga(type = "real-valued", fitness = fitness,
         min = rep(0, nStocks),
         max = rep(1, nStocks),
         names = myStocks,maxiter = 100, run = 100, optim = TRUE)


summary(GA)
plot(GA)
GA@solution


#normalized the GA solution
result=GA@solution/sum(GA@solution)
result

result=GA@solution/sum(GA@solution)
result
```

```
#Optimal weights GA
w <- weights(result)
w
#Expected return GA
ExpReturn(w)
#Expecteed return GA
VarPortfolio(w)

barplot(w, xlab = "Stocks", ylab = "Portfolio weights",
        cex.names = 0.7, col = .colorwheelPalette(10))


returns



#compare the expected return of GA with  expected return of equal weights porfolio
f<-c(.333, .333, .333)
f
prediction <- sum(returns_train*f)
prediction

#compare the expected return of GA with expected return of randomly weights
   porfolios
#run it 10 times
x<-runif(3)
random_weights <- x/sum(x)
random_weights

#Expected return of random weights
prediction <- sum(returns_train*random_weights)
prediction
```

```
#VIb

#Uploading my stocks and my returns are weekly
myStocks <- c("AAPL", "FB", "WFC")
getSymbols(myStocks, src = "yahoo", from="2016-01-01", to="2017-01-01",
   freq="week")
returns <- lapply(myStocks, function(s)
  weeklyReturn(eval(parse(text = s)),
            subset = "2016::2017"))

returns <- do.call(cbind,returns)
colnames(returns) <- myStocks
#52 weeks
nrow(returns)
returns




#summary returns
summary(returns)


#SPLIT TRAIN AND TEST
smp_size <- floor(0.80 * nrow(returns))


train_ind <- sample(seq_len(nrow(returns)), size = smp_size)

returns_train <- returns[train_ind, ]
returns_train
returns_test <- returns[-train_ind, ]
returns_test


#Plot weeklyReturns for Portfolio Constituents

plot(as.timeSeries(returns), at = "chic", minor.ticks="month",
    mar.multi = c(0.2, 5.1, 0.2, 1.1), oma.multi = c(4, 0, 4, 0),
    col = .colorwheelPalette(10), cex.lab = 0.8, cex.axis = 0.8)
title("Stock Returns")


#Number of Stocks
```

```
nStocks <- ncol(returns_test) # number of portfolio assets
nStocks
#Mean return
R <- colMeans(returns_test) # average weekly returns
R

#Correlation Matrix of Returns
Corr <- round(cor(returns), digits = 2)
Corr
pairs(returns, cex.labels = NULL)

#Covariance of Returns
S <- cov(returns_test) # covariance matrix of monthly returns
S


#Standard Deviation of returns
s <- sqrt(diag(S)) # volatility of weekly returns
s

#Part II

#Plotting Mean Return to Std.Dev
plot(s, R, type = "n", panel.first = grid(),
    xlab = "Std. dev. weekly returns", ylab = "Average weekly returns")
text(s, R, names(R), col = .colorwheelPalette(10), font = 2)


#Defining functions
weights <- function(w) # normalised weights
{ drop(w/sum(w)) }

ExpReturn <- function(w) # expected return
{ sum(weights(w)*R) }

VarPortfolio <- function(w) # objective function

  #Changing w across the 2 axes * CoVariance in the matrix
{
  w <- weights(w)
  drop(w %*% S %*% w)
}



#fitness function
```

```
fitness <- function(w)
{
  tot_exp <- ExpReturn(w)
  risk <- VarPortfolio(w)
  fitness <- (tot_exp)/(risk)

  return(fitness)
}



GA <- ga(type = "real-valued", fitness = fitness,
         min = rep(0, nStocks),
         max = rep(1, nStocks),
         names = myStocks,maxiter = 100, run = 100, optim = TRUE)




summary(GA)
plot(GA)
GA@solution


result=GA@solution/sum(GA@solution)
result


#Optimal weights
w <- weights(result)
w
#Expected return using the optimal weights
ExpReturn(w)
#Expected variance using the otimal weights
VarPortfolio(w)

barplot(w, xlab = "Stocks", ylab = "Portfolio weights",
        cex.names = 0.7, col = .colorwheelPalette(10))
```