Markowitz_Research_Me

24 July 2018

PART 1

In this part, we are going to study the Markowitz model using normal calculations or direct calculations

Markowitz Problem with 4 stocks

The following data shows historical adjusted closing prices for 5 stocks for the trading period of January 2017 to January 2018.

Loading the required packages

```
suppressMessages(library(quantmod))
suppressMessages(library(PerformanceAnalytics))
suppressMessages(suppressWarnings(library(timeSeries)))
suppressMessages(suppressWarnings(library(fPortfolio)))
suppressMessages(suppressWarnings(library(caTools)))
suppressMessages(library(dplyr))
suppressMessages(library(ggplot2))
suppressMessages(library(ggcorrplot))
suppressMessages(suppressWarnings(library(psych)))
library(dygraphs)
```

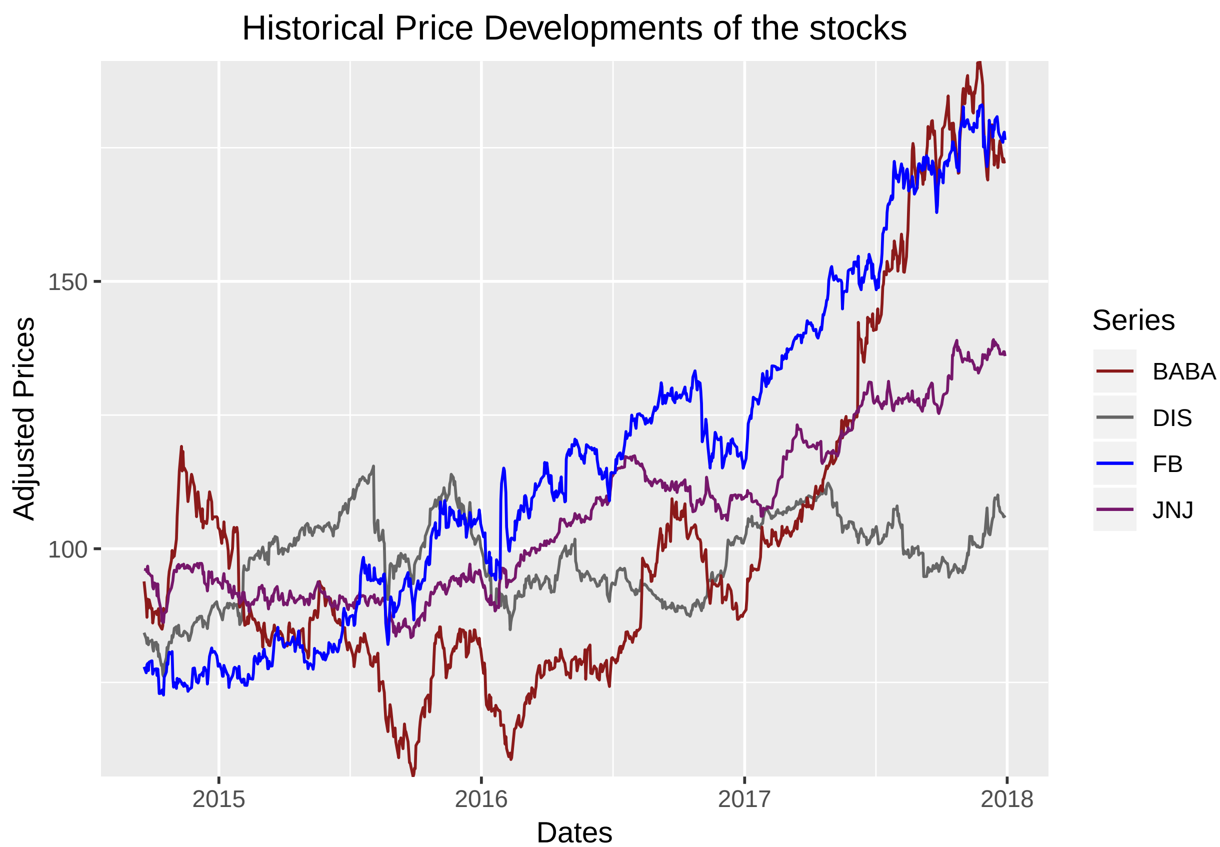
Obtaining the data and plot

```
begin = "2014-01-01"
end = "2018-01-01"
stocks = c("DIS","BABA","JNJ","FB")
suppressMessages(getSymbols(stocks,from=begin, to = end))
```

```
## [1] "DIS"  "BABA" "JNJ"  "FB"
```

```
prices = na.omit(merge(Ad(DIS),Ad(BABA),Ad(JNJ),Ad(FB)))
names(prices) <- c("DIS","BABA","JNJ","FB")

#write the prices to a csv file in your computer
write.zoo(prices,file = "Prices_of_4_stocks.csv", index.name = "Dates", sep = ",")
prices_matrix <- as.matrix(prices)
m = dim(prices_matrix)
#Plot of the price developments of the above data
ggplot(prices, aes(x = index(prices))) +
  geom_line(aes(y = prices$DIS,color = "DIS")) +
  ggtitle("Historical Price Developments of the stocks") +
  geom_line(aes(y = prices$BABA, color = "BABA")) +
  geom_line(aes(y = prices$JNJ, color = "JNJ")) +
  geom_line(aes(y = prices$FB, color = "FB")) +
```

```
  xlab("Dates") + ylab("Adjusted Prices") +
  theme(plot.title = element_text(hjust = 0.5), panel.border = element_blank()) +
  scale_y_continuous(expand = c(0,0)) +
  scale_colour_manual("Series",values=c("DIS"="gray40","BABA"="firebrick4",
                                        "FB"="blue","JNJ"="#76176b"))
```




```
head(prices,10)
```

```
##                 DIS  BABA      JNJ    FB
## 2014-09-19 84.29720 93.89 96.16410 77.91
## 2014-09-22 83.17931 89.89 96.06615 76.80
## 2014-09-23 82.26636 87.17 95.69215 78.29
## 2014-09-24 83.32836 90.57 96.74294 78.54
## 2014-09-25 82.04281 88.92 95.37157 77.22
## 2014-09-26 82.66695 90.46 95.37157 78.79
## 2014-09-29 82.75080 88.75 94.87290 79.00
## 2014-09-30 82.93709 88.85 94.91740 79.04
## 2014-10-01 81.50249 86.10 92.87820 76.55
## 2014-10-02 80.85040 87.06 92.47746 77.08
```

Dygraphs

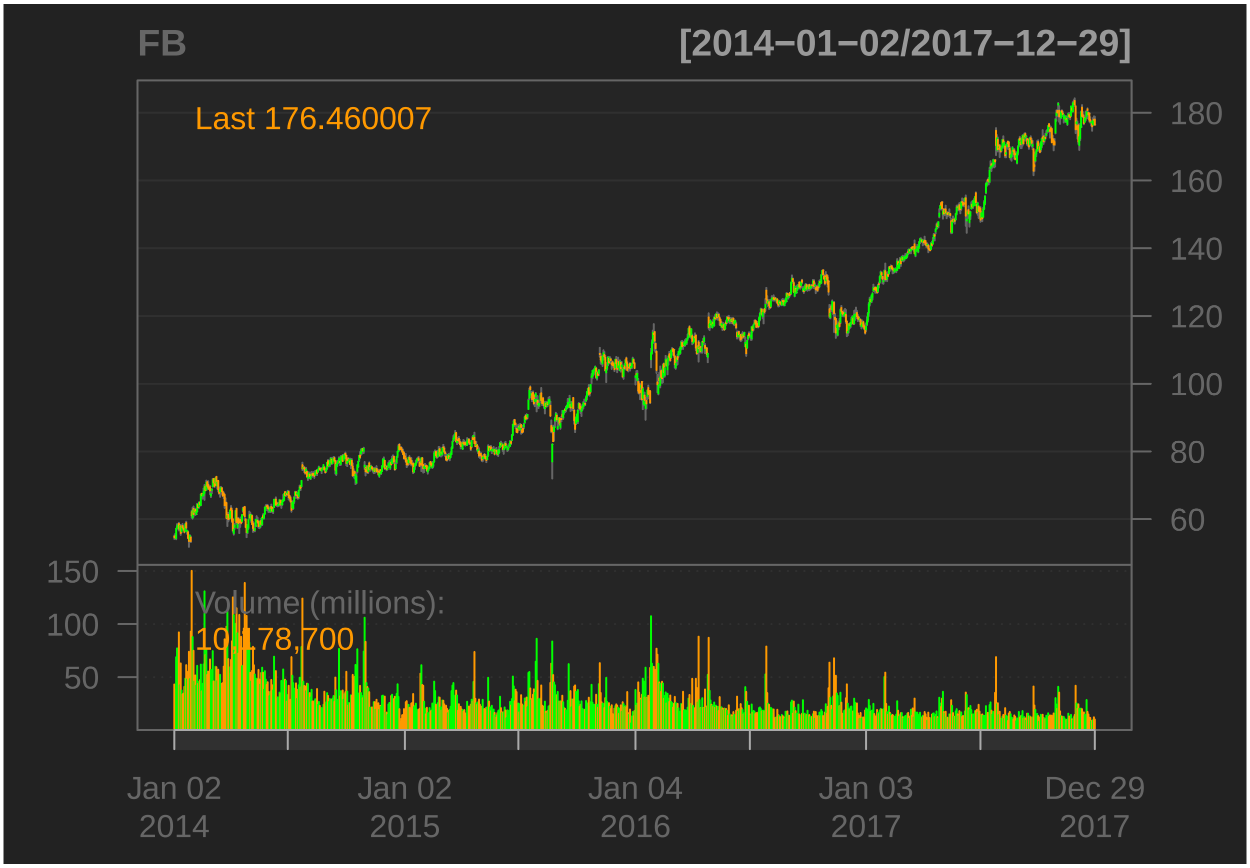
```
#plot dygraph from package dygraphs
#dygraph(prices,main = "Prices of stocks",xlab = "Dates",ylab = "Prices")
## Uncomment abive to see dygraphs
# basic time series plot with range slider underneath
```

```r
#dygraph(prices) %>% dyRangeSelector()
## uncomment above to see dygraphs
```

Chart series for the stocks

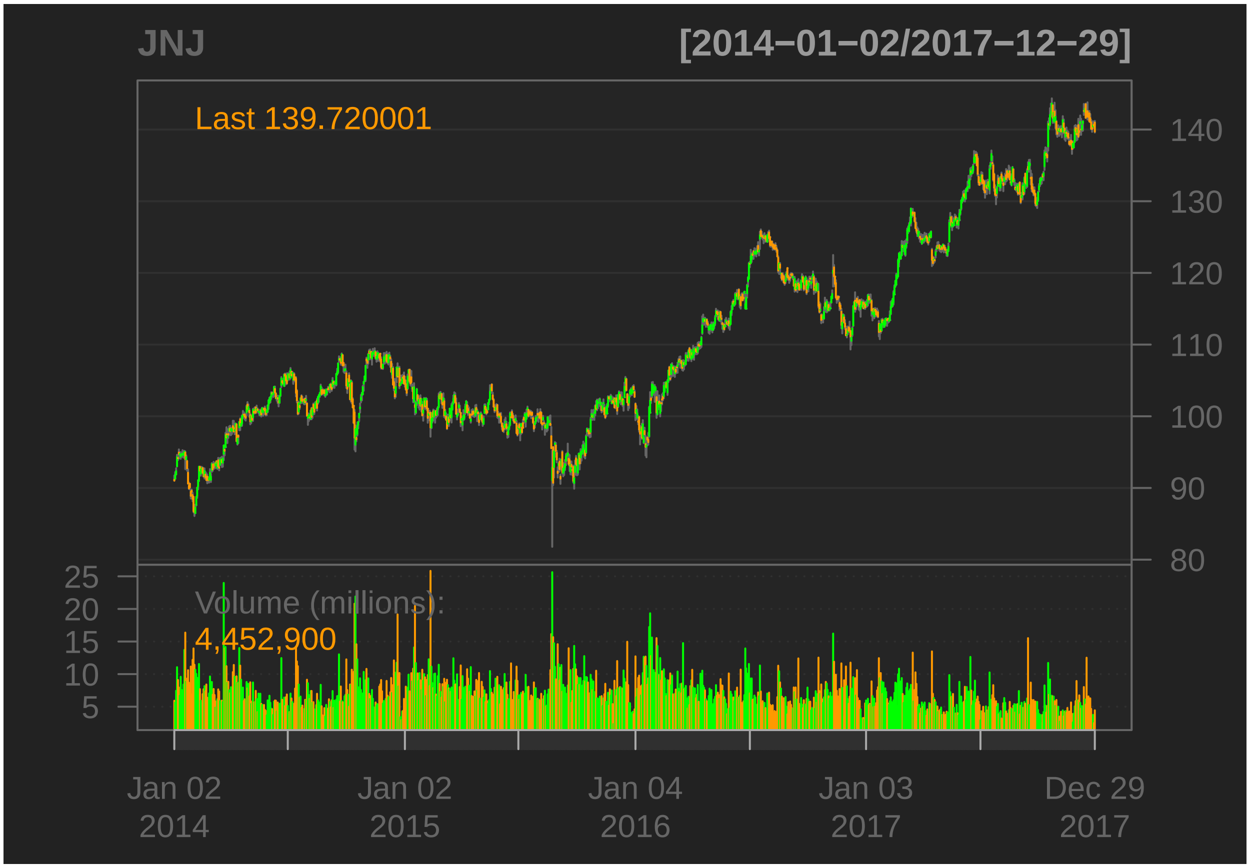
```r
chartSeries(BABA)
```


```r
chartSeries(FB)
```




```
chartSeries(DIS)
```


```
chartSeries(JNJ)
```



Corrplot for 2014-2016

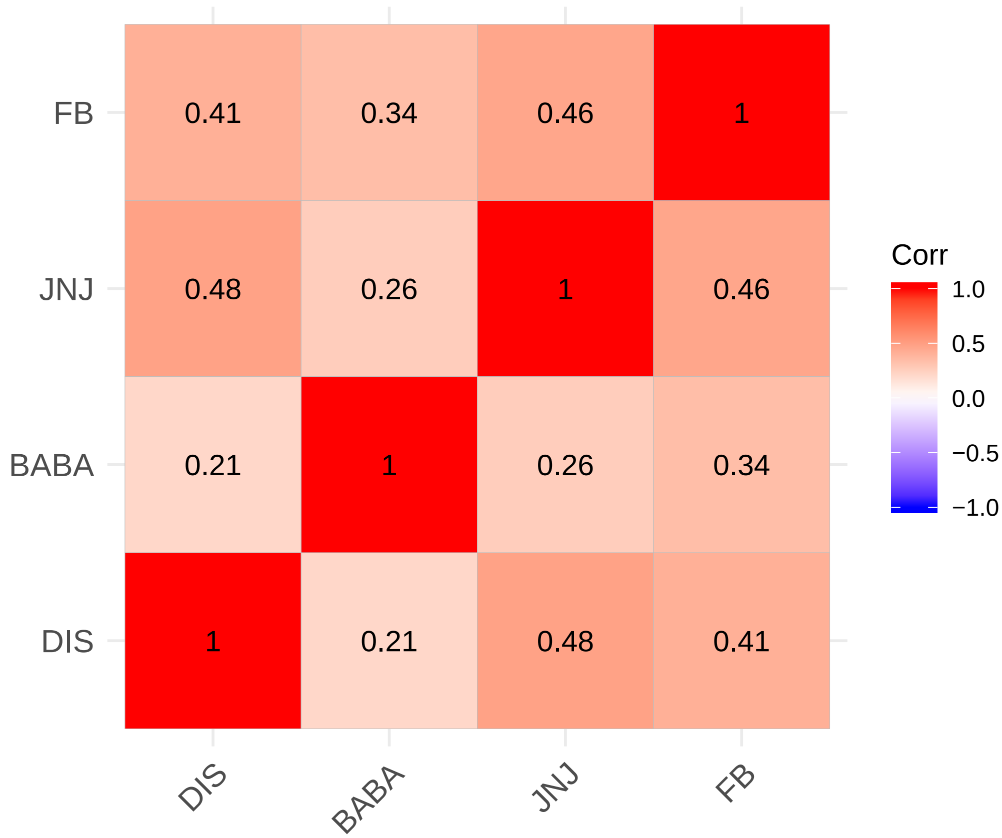
```
begin1 = "2014-01-01"
end1 = "2016-01-01"
stocks = c("DIS","BABA","JNJ","FB")
suppressMessages(getSymbols(stocks,from=begin1, to = end1))
```

```
## [1] "DIS"  "BABA" "JNJ"  "FB"
```

```
prices1 = na.omit(merge(Ad(DIS),Ad(BABA),Ad(JNJ),Ad(FB)))
names(prices1) <- c("DIS","BABA","JNJ","FB")
prices1<- as.matrix(prices1)
Xi1 = 365*diff(log(prices1))
corr1 <- round(cor(Xi1), 3)
head(corr1[, 1:4])
```

```
##        DIS  BABA   JNJ    FB
## DIS  1.000 0.211 0.484 0.410
## BABA 0.211 1.000 0.259 0.336
## JNJ  0.484 0.259 1.000 0.458
## FB   0.410 0.336 0.458 1.000
```

```
ggcorrplot(corr1,lab = TRUE)
```



Corrplot for 2016-2018

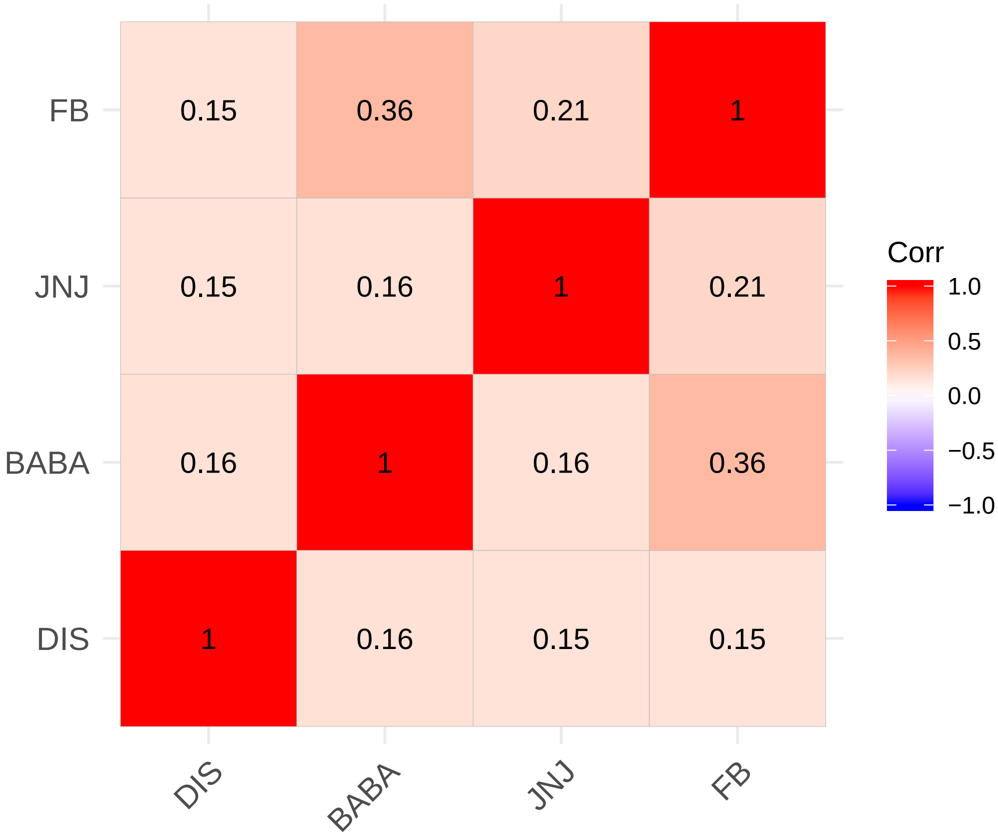
```
begin2 = "2016-01-01"
end2 = "2018-01-01"
stocks = c("DIS","BABA","JNJ","FB")
suppressMessages(getSymbols(stocks,from=begin2, to = end2))
```

```
## [1] "DIS"  "BABA" "JNJ"  "FB"
```

```
prices2 = na.omit(merge(Ad(DIS),Ad(BABA),Ad(JNJ),Ad(FB)))
names(prices2) <- c("DIS","BABA","JNJ","FB")
prices2<- as.matrix(prices2)
Xi2 = 365*diff(log(prices2))
corr2 <- round(cor(Xi2), 3)
head(corr2[, 1:4])
```

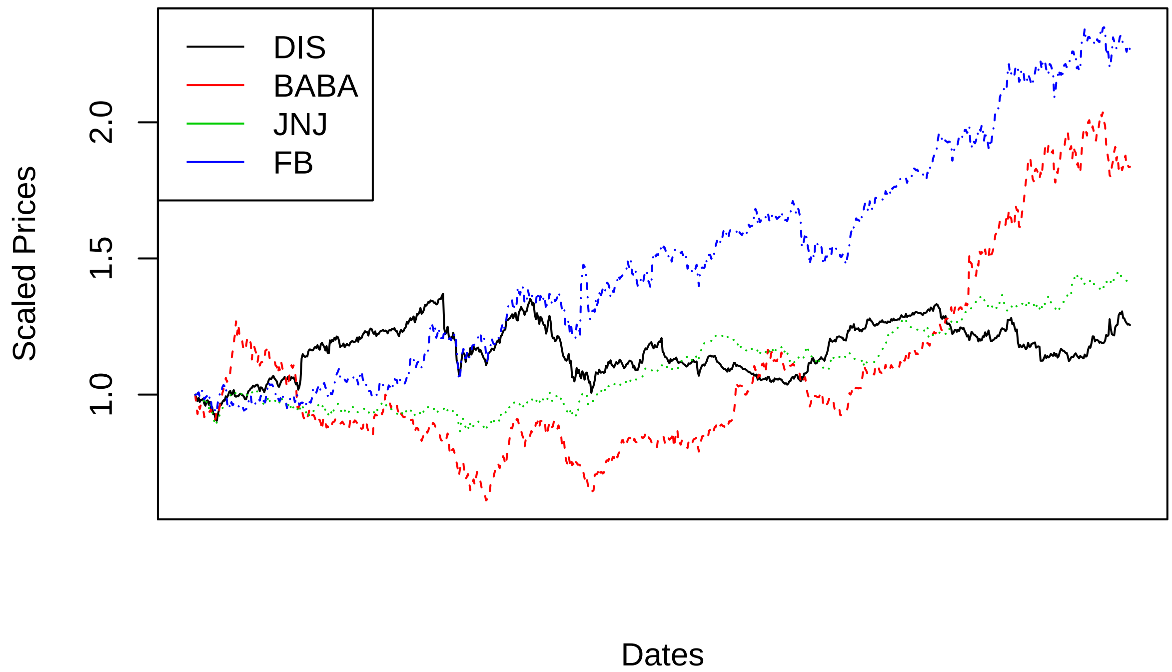
```
##        DIS  BABA   JNJ    FB
## DIS  1.000 0.162 0.149 0.152
## BABA 0.162 1.000 0.164 0.355
## JNJ  0.149 0.164 1.000 0.209
## FB   0.152 0.355 0.209 1.000
```

```r
ggcorrplot(corr2,lab = TRUE)
```



Matrix plot for scaled prices for the data

```r
#Scaled prices
prices.scaled = prices_matrix
m = dim(prices_matrix)
for (k in (1:m[2])){
  prices.scaled[,k] = prices.scaled[,k]/prices.scaled[1,k]
}

matplot(prices.scaled,type='l',xaxt="n",lwd =1,
        col=1:m[2],xlab="Dates",ylab="Scaled Prices")
legend("topleft", stocks, col=1:m[2],lwd=1)
```



Matrix of annualized returns,skewness and kurtosis

```
#Matrix of Annualized Returns
Xi = 365*diff(log(prices_matrix))
#round(Xi*100,2) #Annualized returns in percentages
# skewness and kurtosis from the package psych
skew(Xi)
```

```
## [1] -0.7823843  0.2267734  0.1575834  0.6631494
```

```
kurtosi(Xi)
```

```
##       DIS      BABA       JNJ        FB
##  9.333801  3.495925  2.692699 10.718660
```

Expected annualied returns and variance(biased and unbiased)

```
dim_xi = dim(Xi)
n = dim_xi[1]
j = dim_xi[2]
#first moment: expected annulized returns
returns = colMeans(Xi)
returns
```

```
##       DIS      BABA       JNJ        FB
## 0.1005307 0.2686097 0.1532568 0.3612616
```

```
apply(X=Xi, MARGIN=2,FUN = mean)*100
```

```
##      DIS     BABA      JNJ       FB
## 10.05307 26.86097 15.32568 36.12616
```

```r
#Second moment : variance
(apply(X=Xi, MARGIN = 2, FUN = var)) #biased
```

```
##      DIS     BABA      JNJ       FB
## 18.27308 50.49671 10.76208 31.17938
```

```r
(apply(X=Xi, MARGIN = 2, FUN = var) * (n-1)/n)*100 #unbiased
```

```
##      DIS     BABA      JNJ       FB
## 1825.096 5043.557 1074.905 3114.163
```

```r
#standard deviation
(apply(X=Xi, MARGIN = 2, FUN = sd))*100 #biased
```

```
##      DIS     BABA      JNJ       FB
## 427.4703 710.6103 328.0561 558.3850
```

```r
(apply(X=Xi, MARGIN = 2, FUN = sd) * (n-1)/n)*100 #unbiased
```

```
##      DIS     BABA      JNJ       FB
## 426.9527 709.7500 327.6589 557.7090
```

```r
# Alternative calculation for mean and variance
Return <- as.data.frame(Xi)
Mean <- sapply(Return,mean)
Variance <- sapply(Return, var)
SD <- sapply(Return,sd)
cbind(Mean,Variance,SD)
```

```
##           Mean Variance       SD
## DIS  0.1005307 18.27308 4.274703
## BABA 0.2686097 50.49671 7.106103
## JNJ  0.1532568 10.76208 3.280561
## FB   0.3612616 31.17938 5.583850
```

Covariance matrix

From below, we should note that the covariance matrix is symmetric and is also positive definite. Also, the diagonal of the covariance matrix are the variances of the stocks. And since it is symmetric, then also the covariance matrix is also invertible.

```r
cov(Xi)
```

```
##            DIS      BABA       JNJ        FB
## DIS  18.273082  5.788963  4.657648  6.717097
## BABA  5.788963 50.496707  5.157152 13.846795
## JNJ   4.657648  5.157152 10.762081  6.118848
## FB    6.717097 13.846795  6.118848 31.179377
```

```r
diag(cov(Xi))*100 #biased
```

```
##      DIS     BABA      JNJ       FB
## 1827.308 5049.671 1076.208 3117.938
```

```r
(cov(Xi)*(n-1)/n)
```

```
##            DIS      BABA       JNJ        FB
## DIS  18.250959  5.781954  4.652009  6.708965
## BABA  5.781954 50.435573  5.150908 13.830031
```

```
## JNJ   4.652009  5.150908 10.749052  6.111440
## FB    6.708965 13.830031  6.111440 31.141630
```

```r
diag(cov(Xi)*(n-1)/n) #unbiased
```

```
##      DIS     BABA      JNJ       FB
## 18.25096 50.43557 10.74905 31.14163
```

CAPM solution.

Covariance Matrix

```r
Covar = cov(Xi)*(n-1)/n
inv_C = solve(Covar)
Covar
```

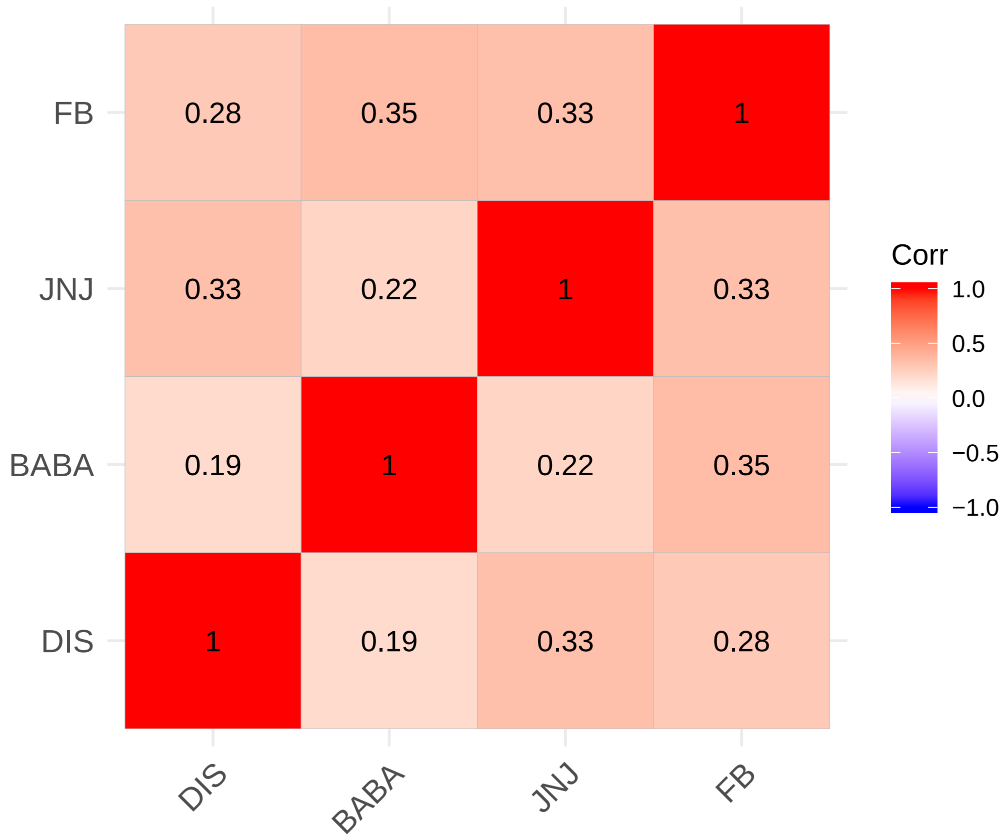
```
##            DIS      BABA       JNJ        FB
## DIS  18.250959  5.781954  4.652009  6.708965
## BABA  5.781954 50.435573  5.150908 13.830031
## JNJ   4.652009  5.150908 10.749052  6.111440
## FB    6.708965 13.830031  6.111440 31.141630
```

Correlation of the assets

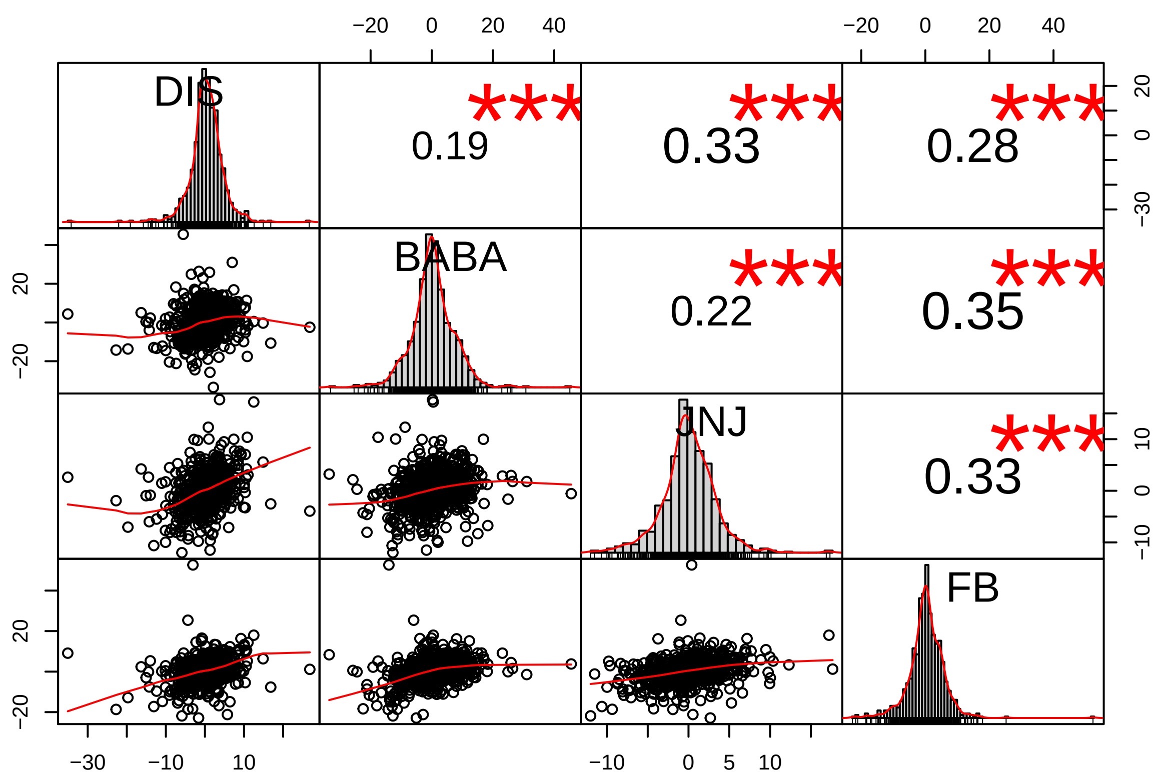
```r
corr <- round(cor(Xi), 3)
head(corr[, 1:4])
```

```
##        DIS  BABA   JNJ    FB
## DIS  1.000 0.191 0.332 0.281
## BABA 0.191 1.000 0.221 0.349
## JNJ  0.332 0.221 1.000 0.334
## FB   0.281 0.349 0.334 1.000
```

```r
ggcorrplot(corr,lab = TRUE)
```




```r
# correlation chart
chart.Correlation(Xi)
```



Inverse of the covariance matrix

Generally, we know that the inverse of a positive definite symmetric matrix is also symmetric. Clearly, the inverse of the covariance matrix below is alo symmetric.

```r
round(inv_C,2)
```

```
##        DIS  BABA   JNJ    FB
## DIS   0.06  0.00 -0.02 -0.01
## BABA  0.00  0.02  0.00 -0.01
## JNJ  -0.02  0.00  0.11 -0.02
## FB   -0.01 -0.01 -0.02  0.04
```

```r
round(inv_C*100,2)
```

```
##        DIS  BABA   JNJ    FB
## DIS   6.43 -0.29 -2.17 -0.83
## BABA -0.29  2.30 -0.49 -0.87
## JNJ  -2.17 -0.49 11.35 -1.54
## FB   -0.83 -0.87 -1.54  4.08
```

A,B,C and D values

```r
ones = rep(1,j)
A = as.numeric(returns%*% inv_C %*% returns)
B = as.numeric(returns%*% inv_C %*% ones )
C = as.numeric(ones %*% inv_C %*% ones )
D = A*C-B*B
first <- C/D * inv_C %*% returns
first2 <- B/D * inv_C %*% returns
first3 <- A/D * inv_C %*% ones
first4 <- B/D * inv_C %*% ones
first - first4
```

```
##            [,1]
## DIS  -2.7756963
## BABA  0.4695406
## JNJ  -1.5374357
## FB    3.8435914
```

```r
first3 - first2
```

```
##             [,1]
## DIS   0.71164775
## BABA -0.01911337
## JNJ   0.85326827
## FB   -0.54580265
```

Efficient Markowitz portfolio

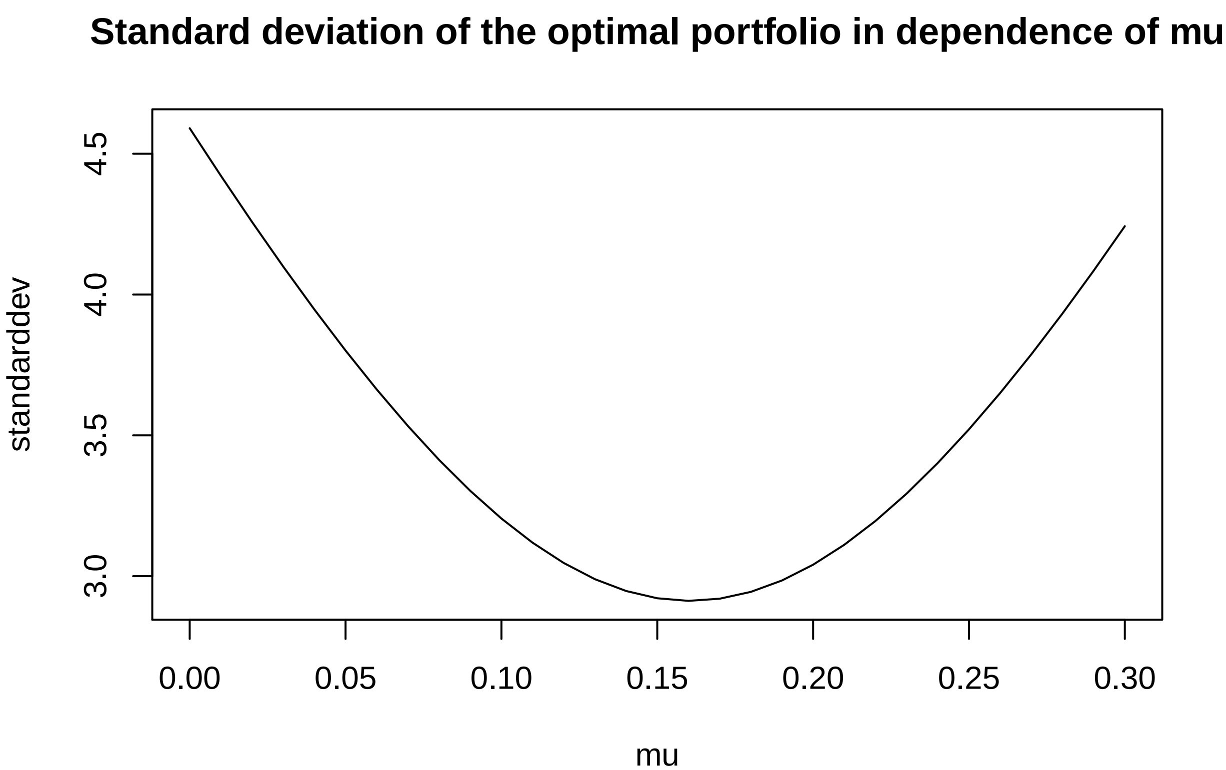
```r
mu = seq(from=0.00,to = 0.3,by=0.01)
ml = length(mu)
xm = matrix(nrow = j, ncol=ml)
for (k in 1:ml){
```

```
  xm[,k]= A*inv_C%*%ones- B*inv_C%*%returns+ mu[k]*(C*inv_C%*%returns -B*inv_C%*%ones)
}
xm = xm/D
colSums(xm)
```

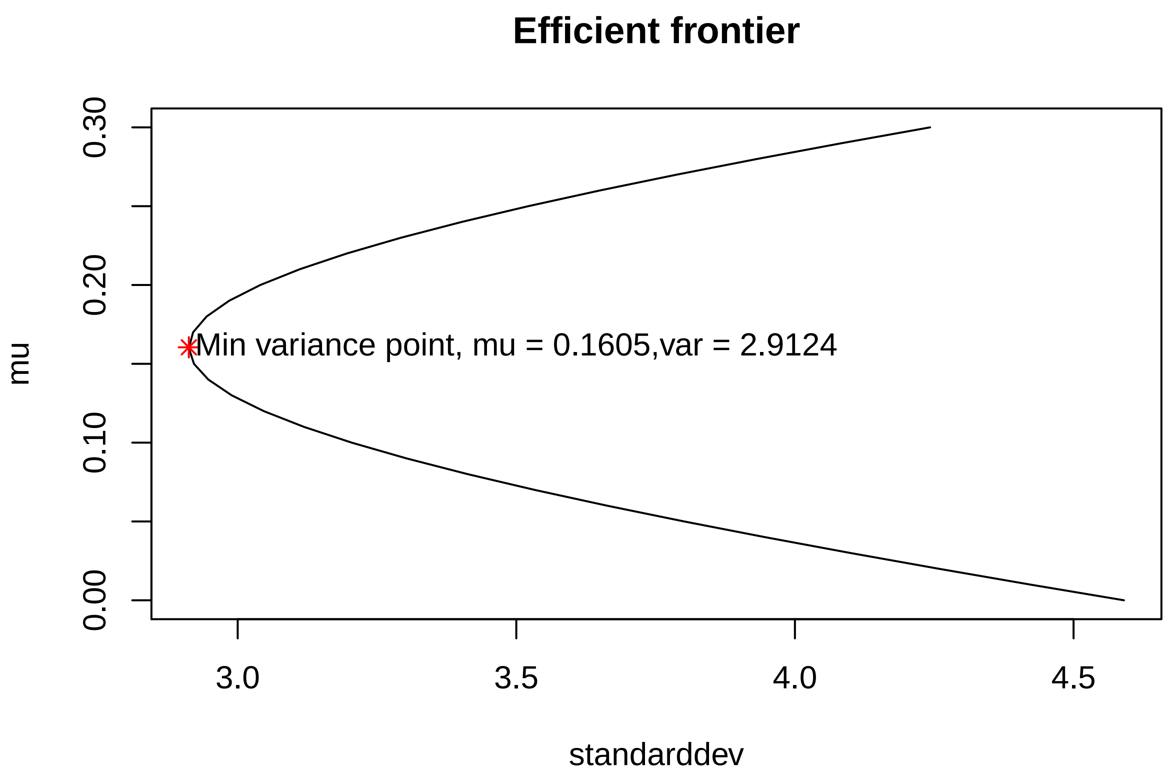
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

we know that standard deviation $= \sqrt{\frac{\mu^2 * c - 2 * \mu * b + a}{d}}$ where $d = a * c - b^2$ and so we can see the relationship between standard deviation and returns for our data in the following figure:

```
standarddev = sqrt((mu*mu*C-2*mu*B+A)/D)
var = mu*mu*C-2*mu*B+A/D
min_mu = B/C #minimum expected return
min_var = sqrt(1/C) #minimum standard dviation
sharpeRatio = min_mu/min_var # sharpe ratio
Expe_port = sharpeRatio*standarddev # portfolio returns for CML
plot(mu,standarddev,type="l",main="Standard deviation of the optimal portfolio in dependence of mu")
```



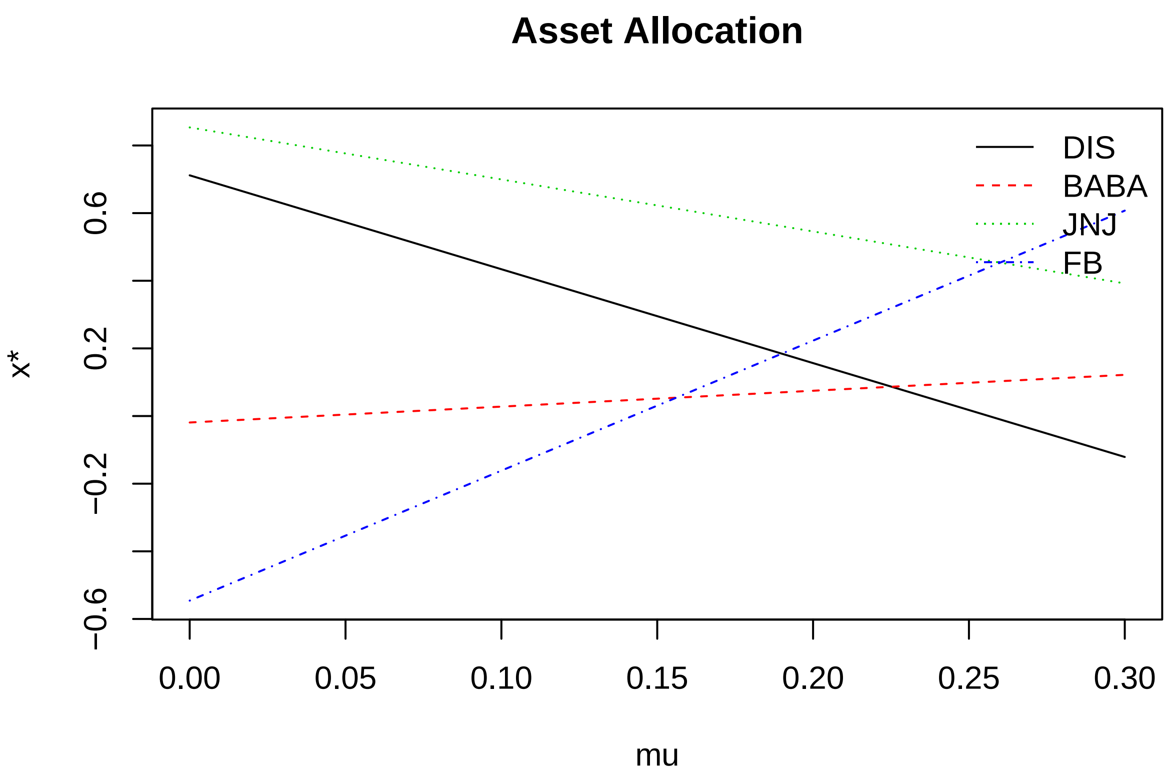

```
plot(standarddev,mu,type = "l", main = "Efficient frontier")
points(min_var,min_mu,pch = 8, col = "red",cex = 1)
text(3.5,min_mu,expression("Min variance point, mu = 0.1605,var = 2.9124"))
```



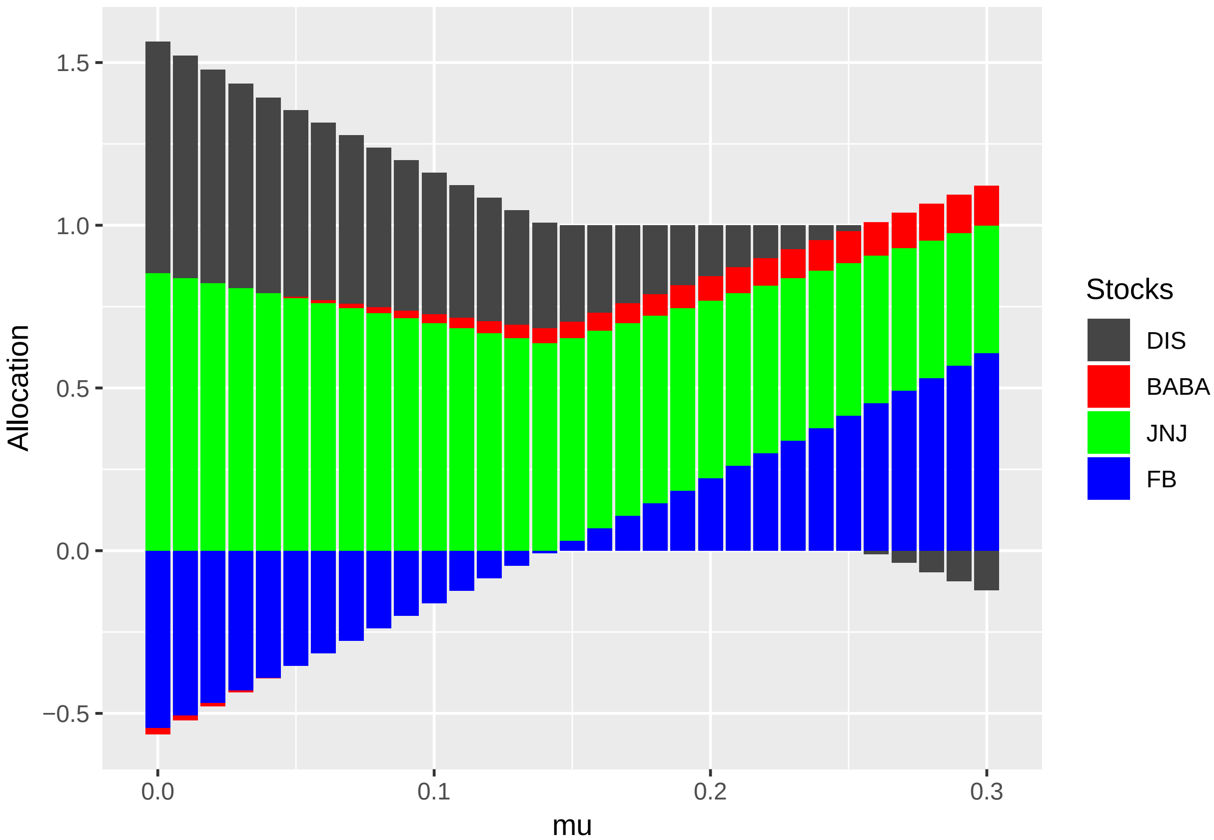
The portfolio with the smallest variance from our combinations is given by $\mu = \frac{b}{c} = 0.1604767 = 16.05\%$ which is clear from the diagram and the corresponding standard deviation is given by $\sigma = \sqrt{\frac{1}{C}} = 2.912357 = 291.24\%$.

Asset allocation figure.

```
matplot(mu,t(xm),type='l',col=1:j,lty=1:j,ylab="x*",main="Asset Allocation")
legend("topright", stocks,col=1:j, lty=1:j,bty="n",lwd=1)
```

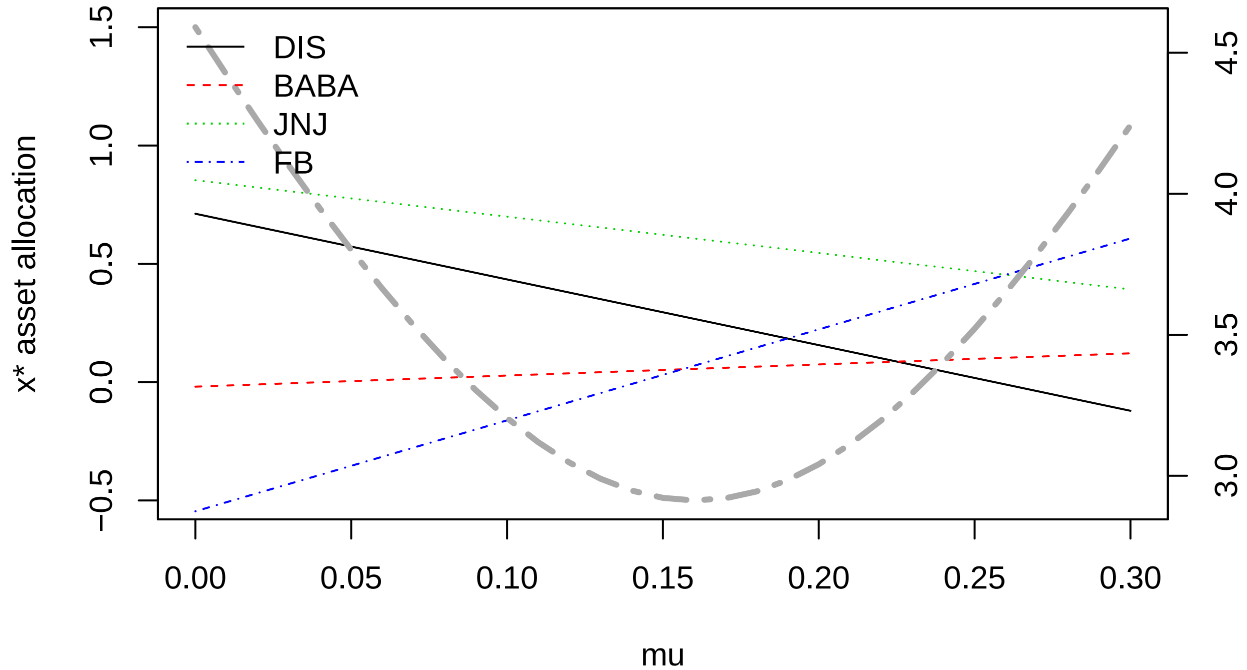



```r
# Or we can plot a stacked barplot which will appear a bit better than above
library(reshape2)
asset_alloc <- data.frame(mu,t(xm))
names(asset_alloc) <- c("mu","DIS","BABA","JNJ","FB")
asset_alloc <- melt(asset_alloc, id = "mu")
names(asset_alloc) <- c("mu","Stocks","Allocation")
ggplot() + geom_bar(aes(y = Allocation, x = mu, fill = Stocks), data = asset_alloc, stat="identity")+
  scale_fill_manual("Stocks",values =
                      c("FB"="#0000FF","JNJ"="#00FF00","BABA"="#FF0000","DIS"="#454545"))
```



Combined asset allocation and standard deviation

```
plot(mu,t(xm[1,]),ylim=c(-0.5,1.5),type='l',col=1,lty=1,ylab="x* asset allocation")
for (k in 2:j) lines(mu,t(xm[k,]),type='l',col=k,lty=k)
par(new=TRUE)
plot(mu, standarddev,type="l",col="darkgray",lwd=3,lty=6,xaxt="n",yaxt="n",xlab="",ylab="")
axis(4)
mtext("sd(x*(mu))",side=4,line=3,col="darkgray")
legend("topleft", c(stocks), col=c(1:j), lty=1:(j),bty="n",lwd=rep(1,j))
```




```
eff.frontier <- function (returns, short="no", max.allocation=NULL, risk.premium.up=.5, risk.increment=
        covariance <- cov(returns)
        print(covariance)
        n <- ncol(covariance)

# Create initial Amat and bvec assuming only equality constraint (short-selling is allowed, no allocati
Amat <- matrix (1, nrow=n)
bvec <- 1
meq <- 1

# Then modify the Amat and bvec if short-selling is prohibited
if(short=="no"){
Amat <- cbind(1, diag(n))
bvec <- c(bvec, rep(0, n))
}
# And modify Amat and bvec if a max allocation (concentration) is specified
if(!is.null(max.allocation)){
if(max.allocation > 1 | max.allocation <0){
stop("max.allocation must be greater than 0 and less than 1")
}
if(max.allocation * n < 1){
stop("Need to set max.allocation higher; not enough assets to add to 1")
}
Amat <- cbind(Amat, -diag(n))
bvec <- c(bvec, rep(-max.allocation, n))
}

# Calculate the number of loops based on how high to vary the risk premium and by what increment
loops <- risk.premium.up / risk.increment + 1
loop <- 1

# Initialize a matrix to contain allocation and statistics
```

```r
# This is not necessary, but speeds up processing and uses less memory
eff <- matrix(nrow=loops, ncol=n+3)
# Now I need to give the matrix column names
colnames(eff) <- c(colnames(returns), "Std.Dev", "Exp.Return", "sharpe")

# Loop through the quadratic program solver
for (i in seq(from=0, to=risk.premium.up, by=risk.increment)){
dvec <- colMeans(returns) * i # This moves the solution up along the efficient frontier
sol <- solve.QP(covariance, dvec=dvec, Amat=Amat, bvec=bvec, meq=meq)
eff[loop,"Std.Dev"] <- sqrt(sum(sol$solution *colSums((covariance * sol$solution))))
eff[loop,"Exp.Return"] <- as.numeric(sol$solution %*% colMeans(returns))
eff[loop,"sharpe"] <- eff[loop,"Exp.Return"] / eff[loop,"Std.Dev"]
eff[loop,1:n] <- sol$solution
loop <- loop+1
}

return(as.data.frame(eff))
}
```

Mean Variance Optimization

```r
library(quadprog)
eff <- eff.frontier(returns=Return, short="no", max.allocation=NULL, risk.premium.up=.5, risk.increment=
```

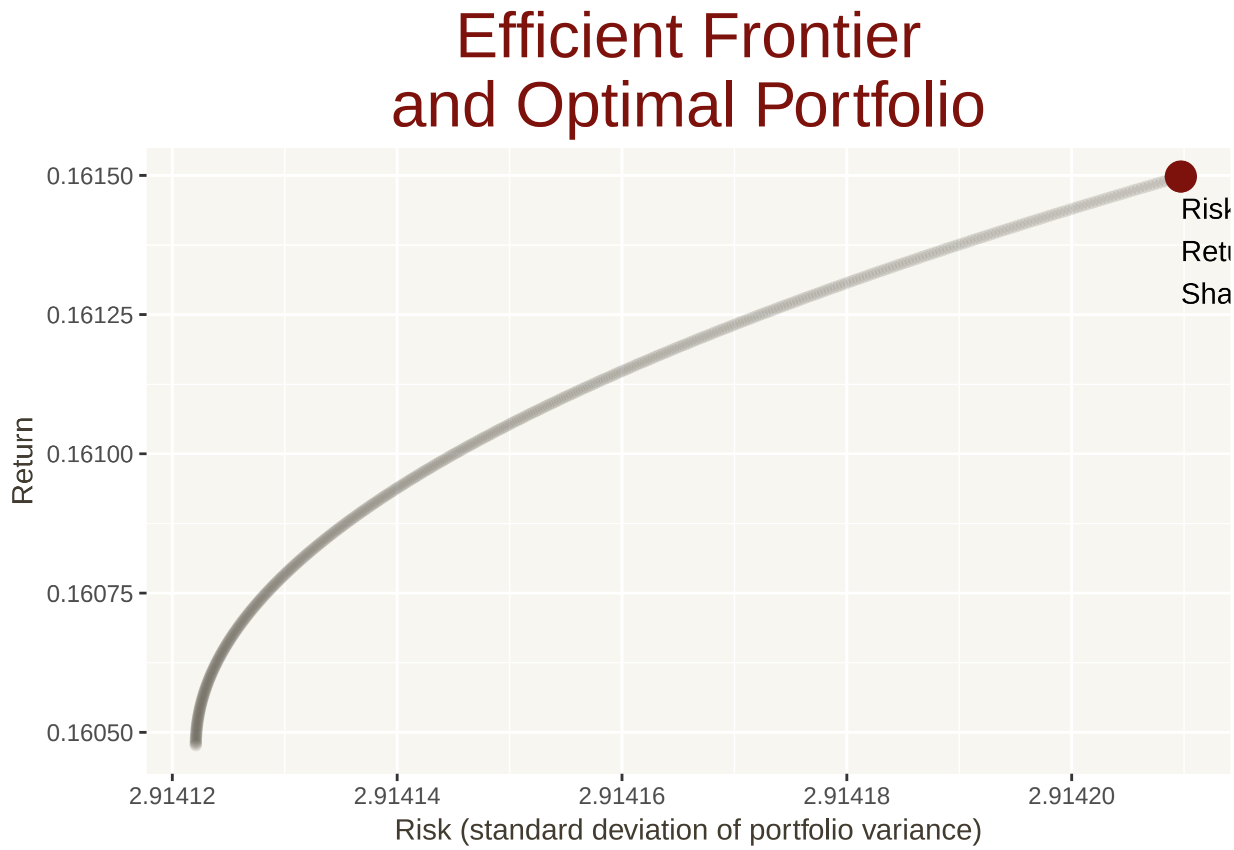
```
##            DIS      BABA       JNJ        FB
## DIS  18.273082  5.788963  4.657648  6.717097
## BABA  5.788963 50.496707  5.157152 13.846795
## JNJ   4.657648  5.157152 10.762081  6.118848
## FB    6.717097 13.846795  6.118848 31.179377
```

```r
eff.optimal.point <- eff[eff$sharpe==max(eff$sharpe),]*100
eff.optimal.point
```

```
##          DIS     BABA      JNJ       FB Std.Dev Exp.Return   sharpe
## 501 26.33781 5.671654 60.49754 7.492994 291.421   16.14981 5.541745
```

```r
eff.optimal.point <- eff[eff$sharpe==max(eff$sharpe),]
library(ggplot2)
# Color Scheme
 ealred  <- "#7D110C"
 ealtan  <- "#CDC4B6"
 eallighttan <- "#F7F6F0"
 ealdark  <- "#423C30"
ggplot(eff, aes(x=Std.Dev, y=Exp.Return)) + geom_point(alpha=.1, color=ealdark) +
 geom_point(data=eff.optimal.point, aes(x=Std.Dev, y=Exp.Return, label=sharpe), color=ealred, size=5) +
 annotate(geom="text", x=eff.optimal.point$Std.Dev, y=eff.optimal.point$Exp.Return,
 label=paste("Risk: ", round(eff.optimal.point$Std.Dev*100, digits=3),"\nReturn: ",
 round(eff.optimal.point$Exp.Return*100, digits=4),"%\nSharpe: ",
 round(eff.optimal.point$sharpe*100, digits=2), "%", sep=""), hjust=0, vjust=1.2) +
 ggtitle("Efficient Frontier\nand Optimal Portfolio") + labs(x="Risk (standard deviation of portfolio va
 theme(panel.background=element_rect(fill=eallighttan), text=element_text(color=ealdark),
 plot.title=element_text(size=24, color=ealred, hjust = 0.5))
```

```
## Warning: Ignoring unknown aesthetics: label
```



PART 2

We are going to use the package **fPortfolio** in this part to study the Markowitz model.

Makowitz model using fPortfolio package and others

I will be using the package "fPortfolio" to study the Markowitz model. This package is specifically geared towards portfolio optimization.

In order to construct a minimum variance portfolio, we will need 3 things:

- Historical Returns
- Historical volatility
- Covariance matrix and correlation matrix

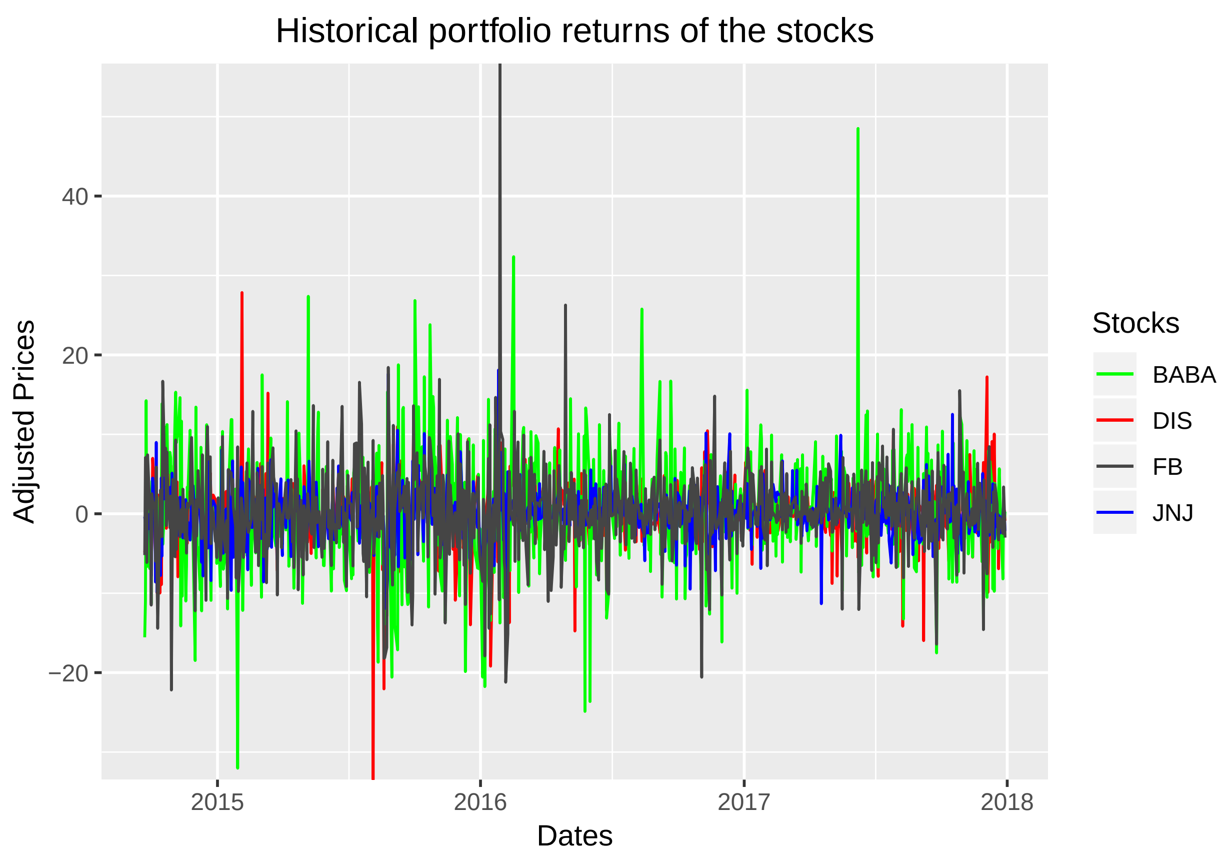
We are going to consider the following 4 stocks during the period of 2010-01-01 to 2018-01-01:

- DIS
- BABA
- JNJ
- FB

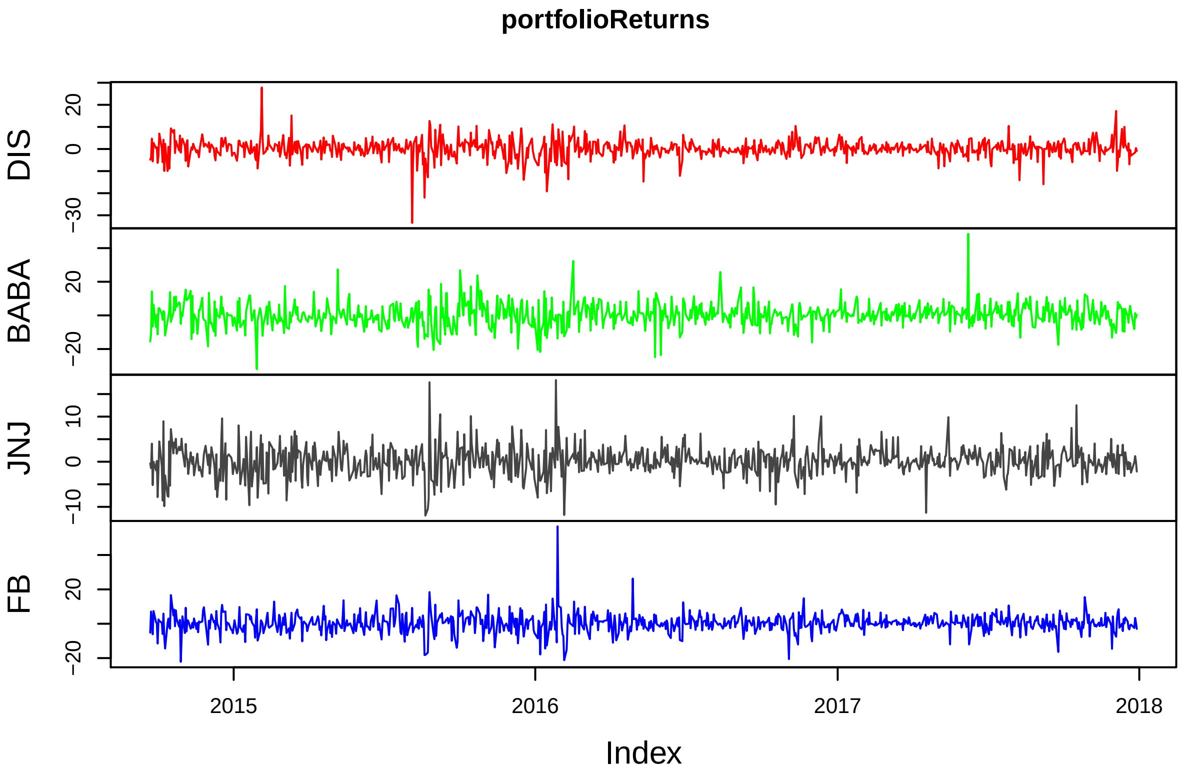
Obtaining data using the quantmod package

We will also be creating a time series object to allow us plot the Efficient frontier. We will use the "**getSymbols**" function from **quantmod** to be able to gather the prices for the stocks which we will use to calculate returns and convert the data into a time series object.

```
tickers <- c("DIS", "BABA", "JNJ", "FB")
#Calculate Returns: Daily
portfolioPrices <- NULL
for (Ticker in tickers)
  portfolioPrices <- cbind(portfolioPrices,
                           suppressMessages(getSymbols(Ticker,from="2014-01-01",
                                      to="2018-01-01",auto.assign=FALSE)[,4]))
portfolioPrices <- portfolioPrices[apply(portfolioPrices,1,function(x) all(!is.na(x))),]
colnames(portfolioPrices) <- tickers
#Calculate Returns: Daily RoC
portfolioReturns <- na.omit(ROC(portfolioPrices, type="discrete"))*365
colnames(portfolioReturns) <- tickers
#Plot of the returns developments of the above data
ggplot(portfolioReturns, aes(x = index(portfolioReturns))) +
  geom_line(aes(y = portfolioReturns$DIS,color = "DIS")) +
  ggtitle("Historical portfolio returns of the stocks") +
  geom_line(aes(y = portfolioReturns$BABA, color = "BABA")) +
  geom_line(aes(y = portfolioReturns$JNJ, color = "JNJ")) +
  geom_line(aes(y = portfolioReturns$FB, color = "FB")) +
  xlab("Dates") + ylab("Adjusted Prices") +
  theme(plot.title = element_text(hjust = 0.5), panel.border = element_blank()) +
  scale_y_continuous(expand = c(0,0)) +
  scale_colour_manual("Stocks",values=c("DIS"="#FF0000","BABA"="#00FF00",
                                        "JNJ"="#0000FF","FB"="#454545"))
```



```r
plot.zoo(portfolioReturns,col = c("DIS"="#FF0000","BABA"="#00FF00",
                                  "FB"="#0000FF","JNJ"="#454545"))
```




```r
portfolioReturns <- as.timeSeries(portfolioReturns)
#Checking on the dimension of the portfolio prices
dim(portfolioPrices)
```

```
## [1] 827   4
```

```r
dim(portfolioReturns)
```

```
## [1] 826   4
```

```r
head(portfolioPrices)
```

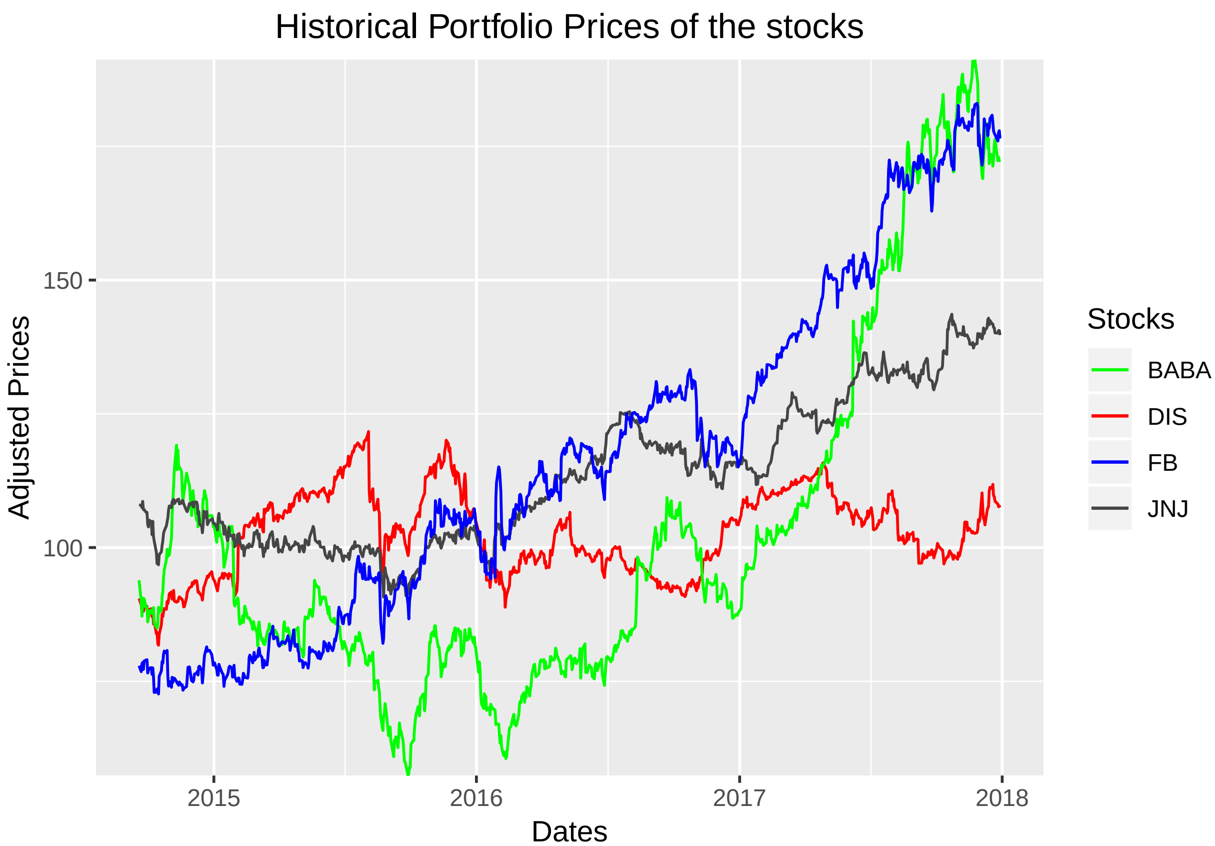
```
##              DIS  BABA    JNJ    FB
## 2014-09-19 90.49 93.89 107.99 77.91
## 2014-09-22 89.29 89.89 107.88 76.80
## 2014-09-23 88.31 87.17 107.46 78.29
## 2014-09-24 89.45 90.57 108.64 78.54
## 2014-09-25 88.07 88.92 107.10 77.22
## 2014-09-26 88.74 90.46 107.10 78.79
```

```r
head(portfolioReturns)
```

```
## GMT
##                  DIS       BABA        JNJ         FB
## 2014-09-22 -4.840302 -15.550112 -0.3717971 -5.2002355
## 2014-09-23 -4.006060 -11.044614 -1.4210166  7.0813704
## 2014-09-24  4.711807  14.236558  4.0080030  1.1655384
## 2014-09-25 -5.631067  -6.649561 -5.1739725 -6.1344537
## 2014-09-26  2.776760   6.321417  0.0000000  7.4210048
```

```
## 2014-09-29  0.370199  -6.899731 -1.9084865  0.9728345
```

```r
#Plot of the price developments of the above data
ggplot(portfolioPrices, aes(x = index(portfolioPrices))) +
  geom_line(aes(y = portfolioPrices$DIS,color = "DIS")) +
  ggtitle("Historical Portfolio Prices of the stocks") +
  geom_line(aes(y = portfolioPrices$BABA, color = "BABA")) +
  geom_line(aes(y = portfolioPrices$JNJ, color = "JNJ")) +
  geom_line(aes(y = portfolioPrices$FB, color = "FB")) +
  xlab("Dates") + ylab("Adjusted Prices") +
  theme(plot.title = element_text(hjust = 0.5), panel.border = element_blank()) +
  scale_y_continuous(expand = c(0,0)) +
  scale_colour_manual("Stocks",values=c("DIS"="#FF0000","BABA"="#00FF00",
                                        "JNJ"="#454545","FB"="#0000FF"))
```



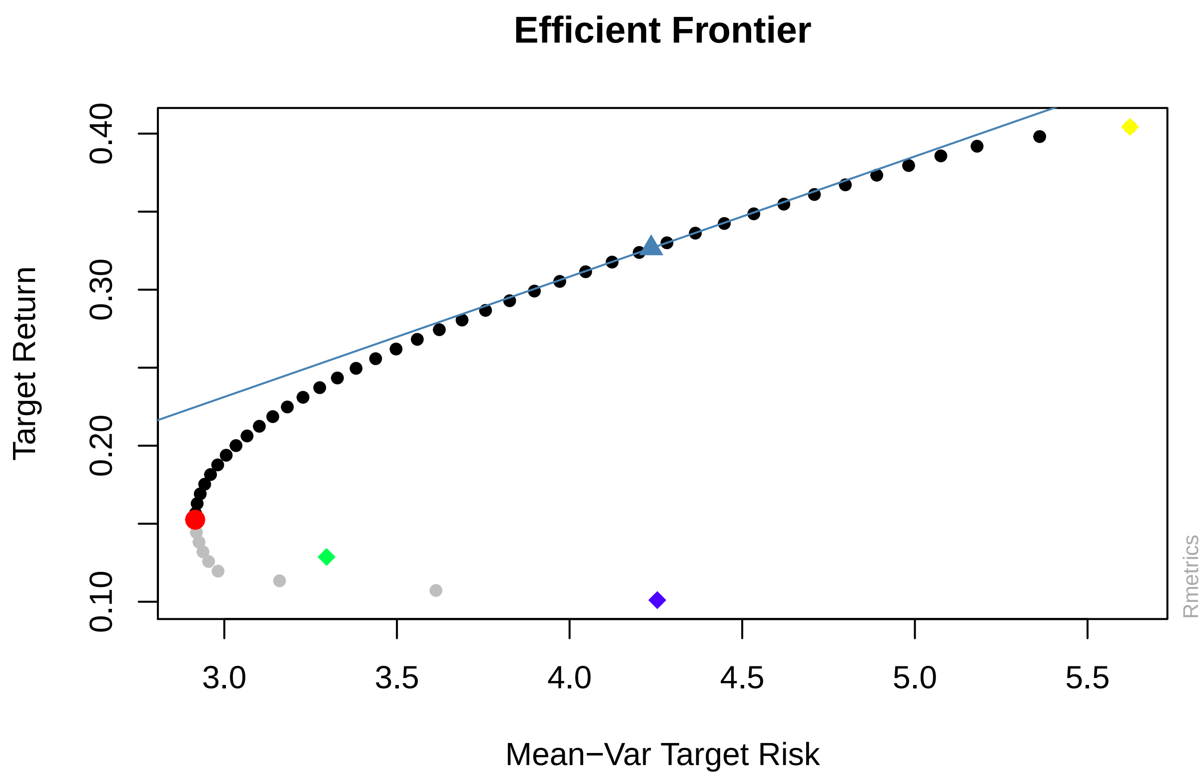
Portfolio frontier calculation

I will now calculate and plot the efficient frontier by using the function " **portfolioFrontier**". I will also output the covariance matrix and correlation matrix for our portfolio of assets. We can also extract certain portfolios sucsh as the Minimun Variance Portfolio or Maximum Return Portfolio.

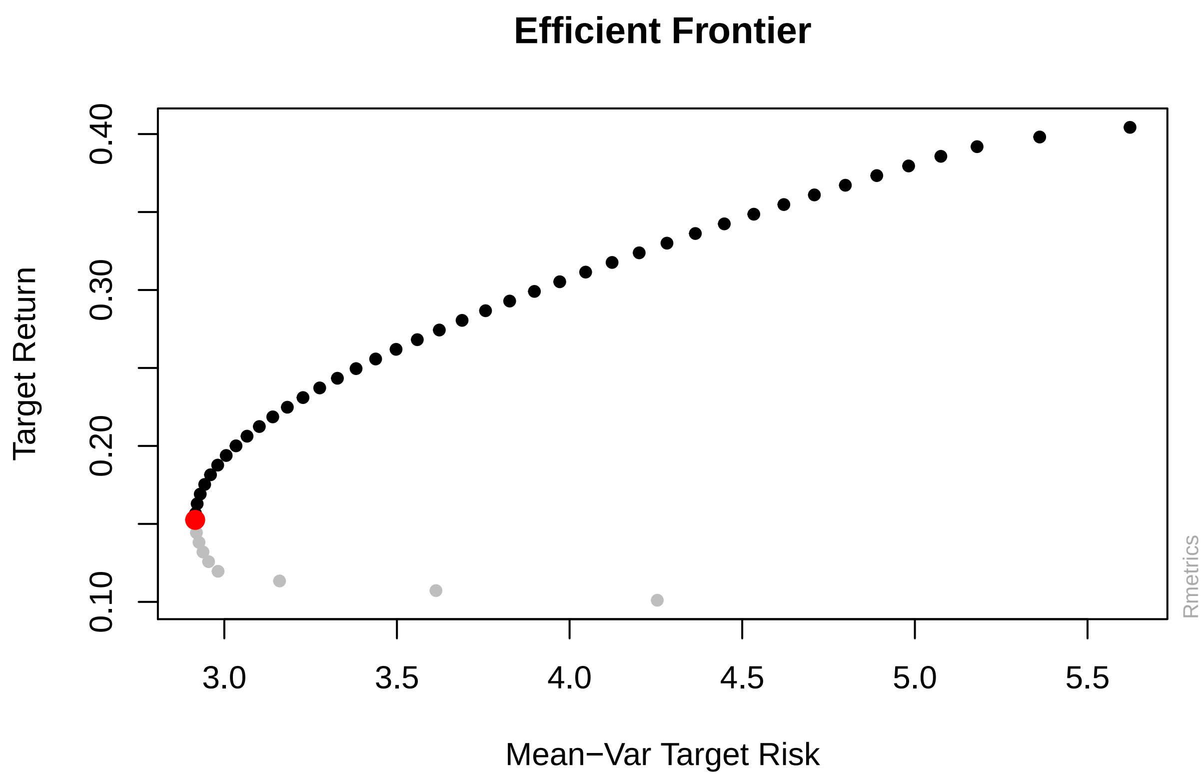
We will also examine the weights of each point on the frontier graphically. We will also annualize the data and plot the risk returns on a scatter plot also. Also, we will plot the Sharpe Ratio of each point on the frontier on a scatter graph.

We will also see the Value-at-risk and conditional value-at-risk of the portfolio returns at different levels

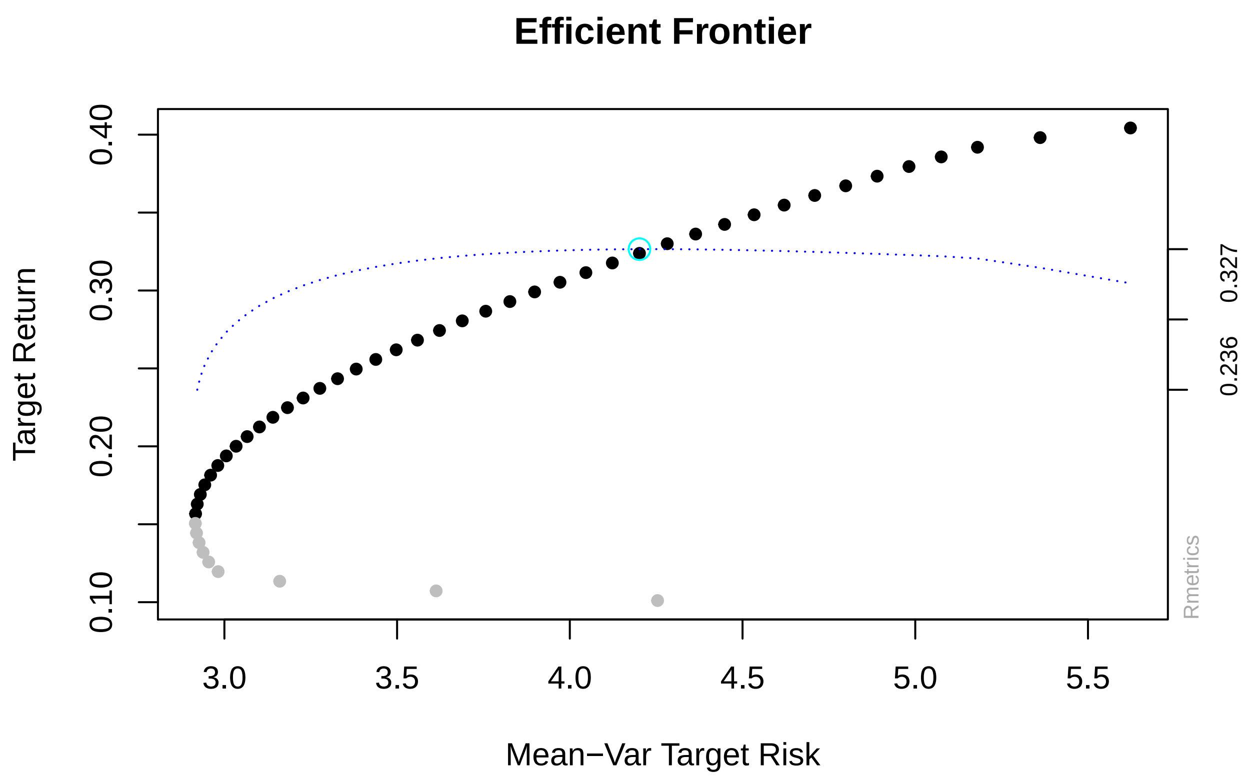
```r
# calculate the efficient frontier
effFrontier <- portfolioFrontier(portfolioReturns,
                                 constraints = "LongOnly")
plot(effFrontier,c(1,2,3,4))
```




```r
plot(effFrontier,c(1,2))
```




```
plot(effFrontier,c(1,8))
```

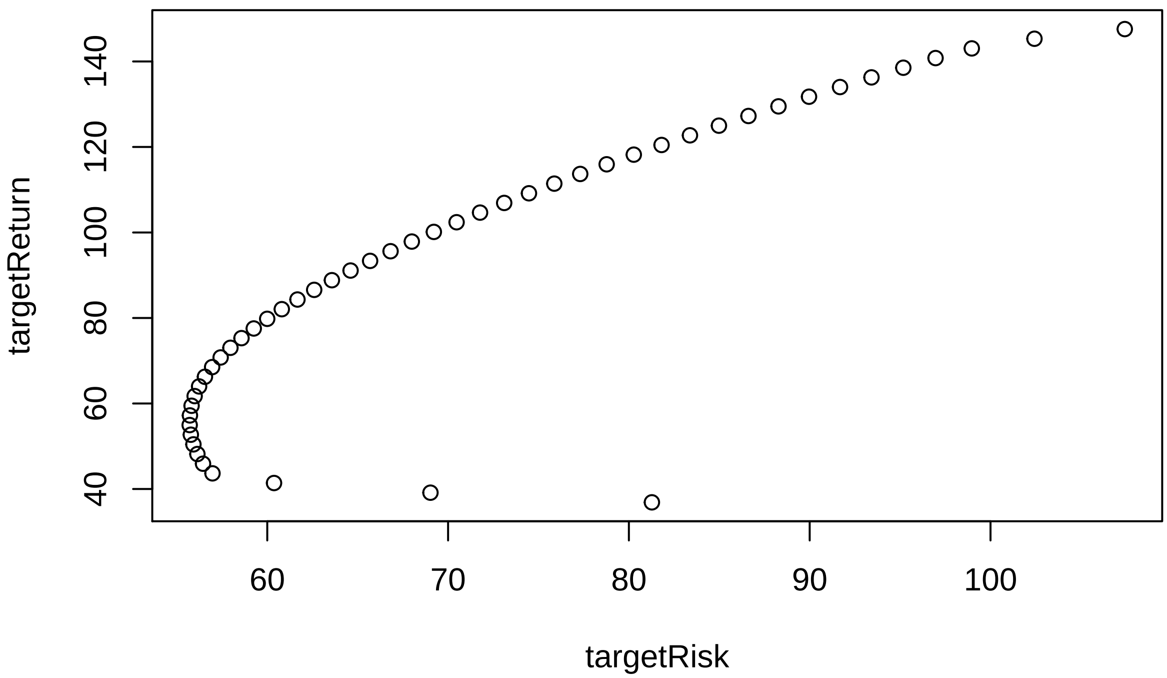



```
#Plot Frontier Weights (Can Adjust Number of Points)
#get allocations for each instrument for each point on the efficient frontier
frontierWeights <- getWeights(effFrontier)
```

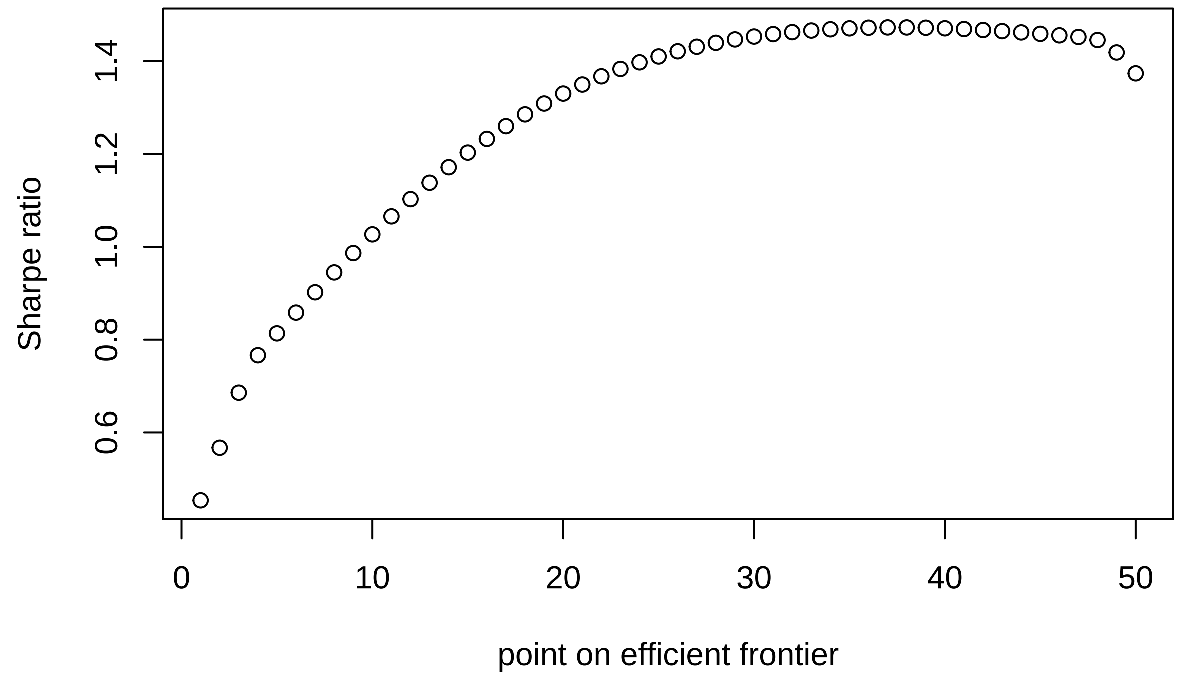
```r
colnames(frontierWeights) <- tickers
risk_return <- frontierPoints(effFrontier)
write.csv(risk_return, "risk_return.csv")

#Output Correlation
cor_matrix <- cor(portfolioReturns)
cov_matrix <- cov(portfolioReturns)
write.csv(cov_matrix, "covmatrix.csv")
cov_matrix
```

```
##            DIS      BABA       JNJ        FB
## DIS  18.096436  5.647140  4.629781  6.640169
## BABA  5.647140 50.863675  5.113207 13.648713
## JNJ   4.629781  5.113207 10.864806  6.147258
## FB    6.640169 13.648713  6.147258 31.617699
```

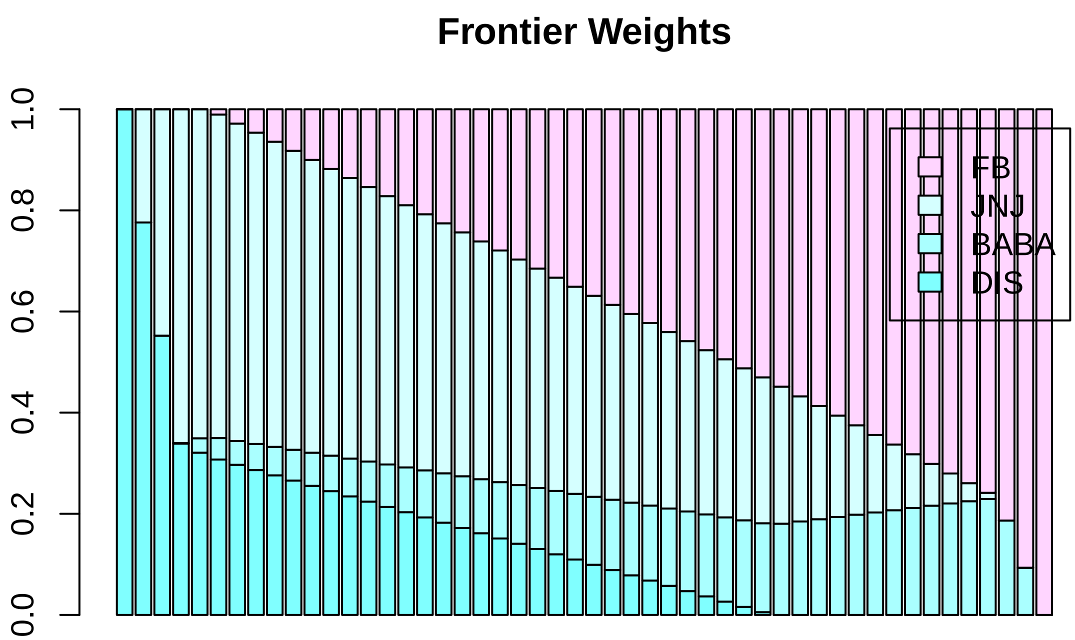
```r
#Annualize Data
#get risk and return values for points on the efficient frontier
riskReturnPoints <- frontierPoints(effFrontier)
annualizedPoints <- data.frame(targetRisk=riskReturnPoints[, "targetRisk"]*sqrt(365),
                               targetReturn=riskReturnPoints[,"targetReturn"]*365)
plot(annualizedPoints)
```




```r
# plot Sharpe ratios for each point on the efficient frontier
riskFreeRate <- 0
plot((annualizedPoints[,"targetReturn"]-riskFreeRate) / annualizedPoints[,"targetRisk"],
     xlab="point on efficient frontier", ylab="Sharpe ratio")
```




```
#Plot Frontier Weights (Need to transpose matrix first)
barplot(t(frontierWeights), main="Frontier Weights",
        col=cm.colors(ncol(frontierWeights)+2),
        legend=colnames(frontierWeights))
```




```
#Get Minimum Variance Port, Tangency Port, etc.
#mvp = minimum variance portfolio
mvp <- minvariancePortfolio(portfolioReturns,
                            spec=portfolioSpec(), constraints="LongOnly")
mvp
```

```
##
## Title:
```

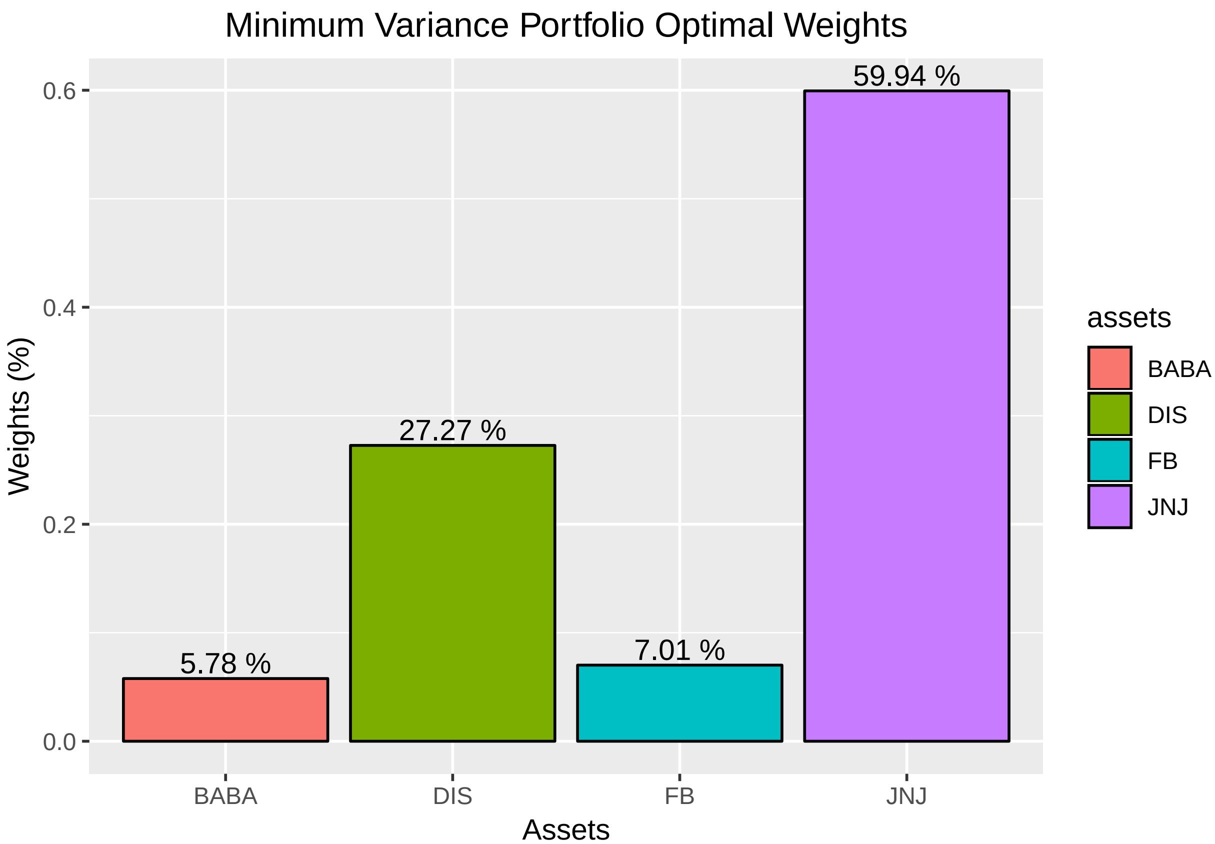
```
##  MV Minimum Variance Portfolio
##  Estimator:         covEstimator
##  Solver:            solveRquadprog
##  Optimize:          minRisk
##  Constraints:       LongOnly
##
## Portfolio Weights:
##    DIS   BABA    JNJ     FB
## 0.2727 0.0578 0.5994 0.0701
##
## Covariance Risk Budgets:
##    DIS   BABA    JNJ     FB
## 0.2727 0.0578 0.5994 0.0701
##
## Target Returns and Risks:
##   mean    Cov   CVaR    VaR
## 0.1526 2.9157 6.7967 4.8266
##
## Description:
##  Thu Jan 24 09:38:05 2019 by user: kirui
```

```
tangencyPort <- tangencyPortfolio(portfolioReturns,
                                  spec=portfolioSpec(),constraints="LongOnly")
tangencyPort
```

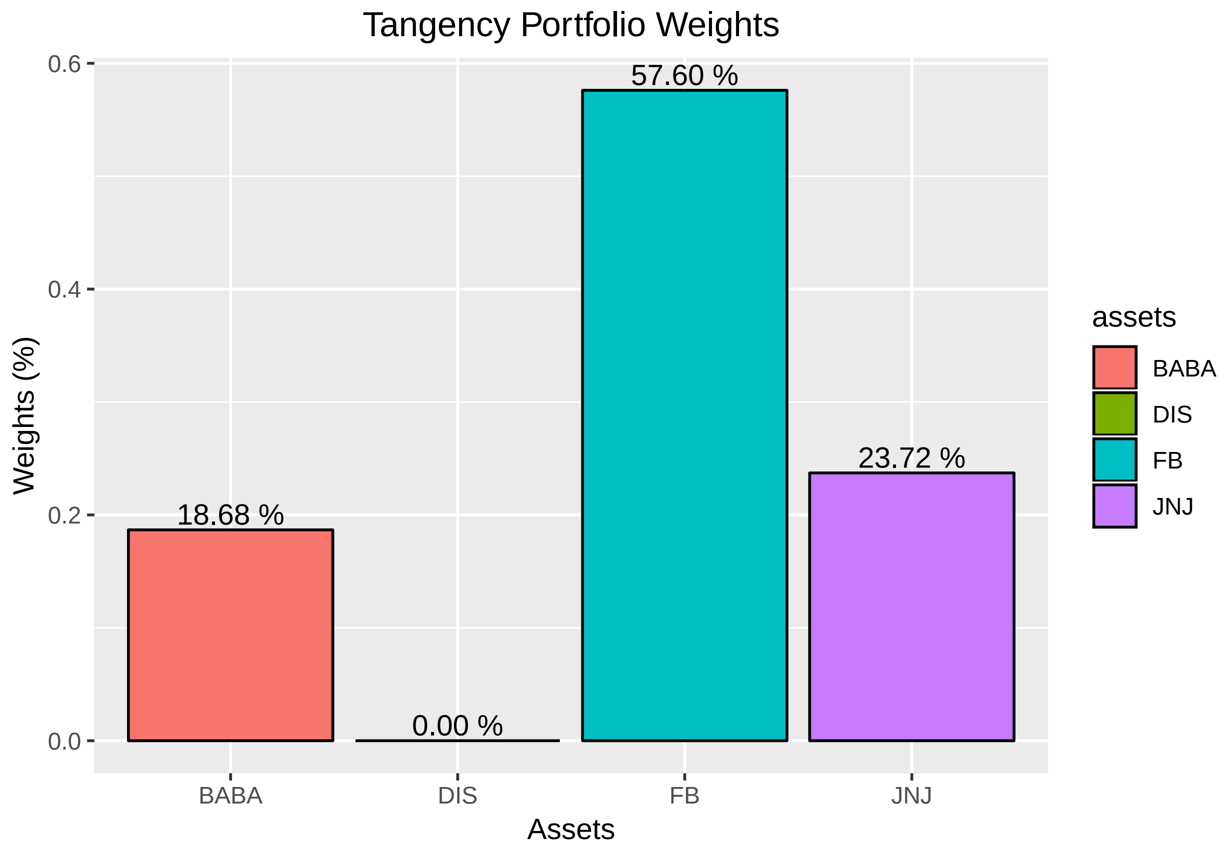
```
##
## Title:
##  MV Tangency Portfolio
##  Estimator:         covEstimator
##  Solver:            solveRquadprog
##  Optimize:          minRisk
##  Constraints:       LongOnly
##
## Portfolio Weights:
##    DIS   BABA    JNJ     FB
## 0.0000 0.1868 0.2372 0.5760
##
## Covariance Risk Budgets:
##    DIS   BABA    JNJ     FB
## 0.0000 0.1933 0.0935 0.7132
##
## Target Returns and Risks:
##   mean    Cov   CVaR    VaR
## 0.3265 4.2364 9.8855 6.3517
##
## Description:
##  Thu Jan 24 09:38:05 2019 by user: kirui
```

```
mvpweights <- getWeights(mvp) #mininum variance portfolio weights
tangencyweights <- getWeights(tangencyPort) #tangency portfolio weights
#ggplot of MVP Weights
df <- data.frame(mvpweights)
assets <- colnames(frontierWeights)
ggplot(data=df, aes(x=assets, y=mvpweights, fill=assets)) +
  geom_bar(stat="identity", position=position_dodge(),colour="black") +
```

```
  geom_text(aes(label=sprintf("%.02f %%",mvpweights*100)),
            position=position_dodge(width=0.9), vjust=-0.25, check_overlap = TRUE) +
              ggtitle("Minimum Variance Portfolio Optimal Weights")+
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(x= "Assets", y = "Weights (%)")
```




```
#ggplot of tangency portfolio weights
dft <- data.frame(tangencyweights)
assets <- colnames(frontierWeights)
ggplot(data=dft, aes(x=assets, y=tangencyweights, fill=assets)) +
  geom_bar(stat="identity", position=position_dodge(),colour="black") +
  geom_text(aes(label=sprintf("%.02f %%",tangencyweights*100)),
            position=position_dodge(width=0.9), vjust=-0.25, check_overlap = TRUE) +
  ggtitle("Tangency Portfolio Weights")+ theme(plot.title = element_text(hjust = 0.5)) +
  labs(x= "Assets", y = "Weights (%)")
```



Note:

- The function **portfolioFrontier** calculates the whole efficient frontier. The portfolio information consists of five arguments: data, specifications, constraints, title and description. Tha range of the frontier is determined from the range of asset returns, and the number of equidistant points in the returns, is calculated from the number of frontier points hold in the specification structure.
- An efficient portfolio ia a portfolio which lies on the efficient frontier. The **efficientPortfolio** function returns the properties of the efficient portfolio.
- The function **tangencyPortfolio** returns the portfolio with the highest return/risk ratio on the efficient frontier. For the Markowitz, this is the same as the Share Ratio. To find this point on the frontier thr return/risk ratio calculated from the target return and target risk returned by the function **efficientPortfolio**.
- The function **minvariancePortfolio** returns the portfolio with the minimal risk on the efficient frontier. To find the minimal risk point, the target risk returned by the function **efficientPortfolio** is minimized.
- The function **maxreturnPortfolio** returns the portfolio with the maximal return for a fixed target risk.

You will realize that we have multiplied the daily returns by 252 and multiplied the deviation by square root of 252. The reason is that the Sharpe Ratio is typically defined in terms of annual return and annual deviation. As everyone has said, you go from daily returns to annual returns by assuming daily returns are independent and identically distributed.

With that assumption, you get annual return by multiplying by daily return by 252 (compounding makes little difference when daily return is 1). You get annual deviation by multiplying daily deviation by square root of 252

Examining the portfolio with constraints

Portfolio with Short selling allowed

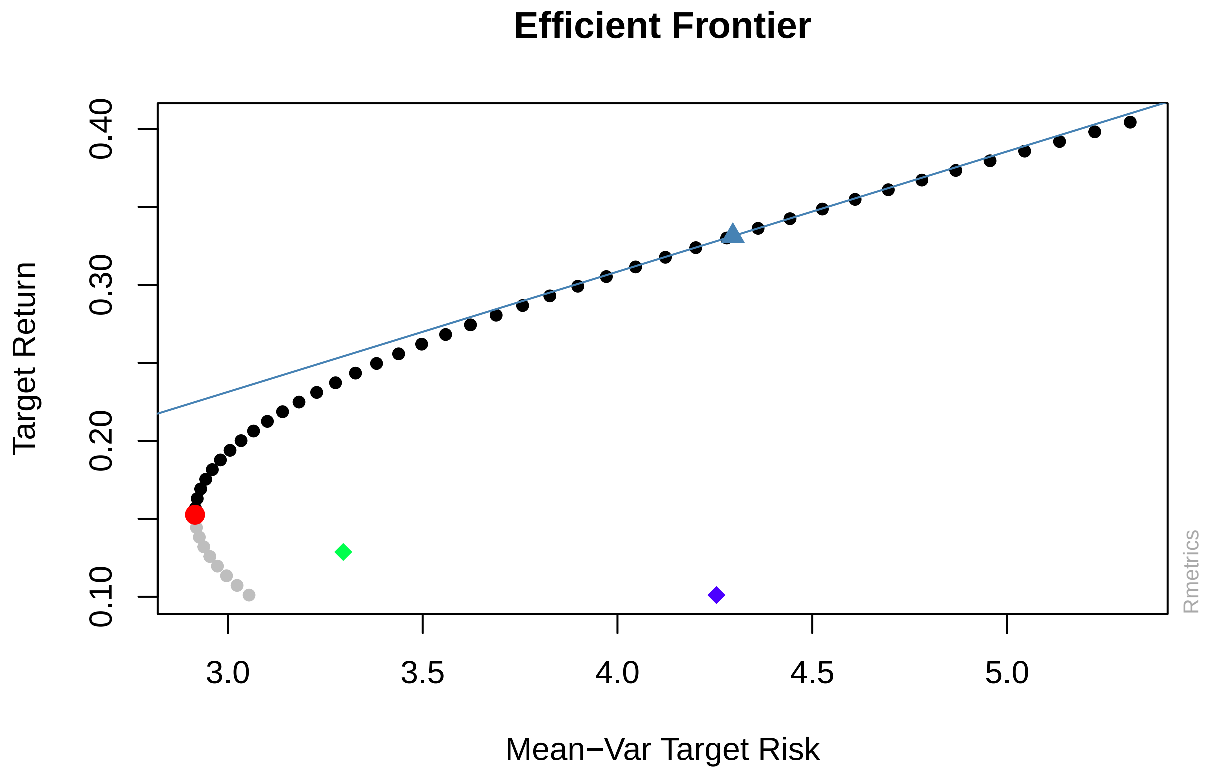
We will now examine the portfolio having some constraints and specifications. We will alow shortselling in this portfolio and set the minimum and maximimum weight in one given asset throughout the entire list of tickers.

```
#Set Specs
Spec = portfolioSpec()
setSolver(Spec) = "solveRshortExact" #set the solver to use
setTargetRisk(Spec) = .12 #set the target risk level.
constraints <- c("minW[1:length(tickers)]=-1","maxW[1:length(tickers)]=.60", "Short")

effFrontierShort <- portfolioFrontier(portfolioReturns, Spec, constraints = constraints)
```

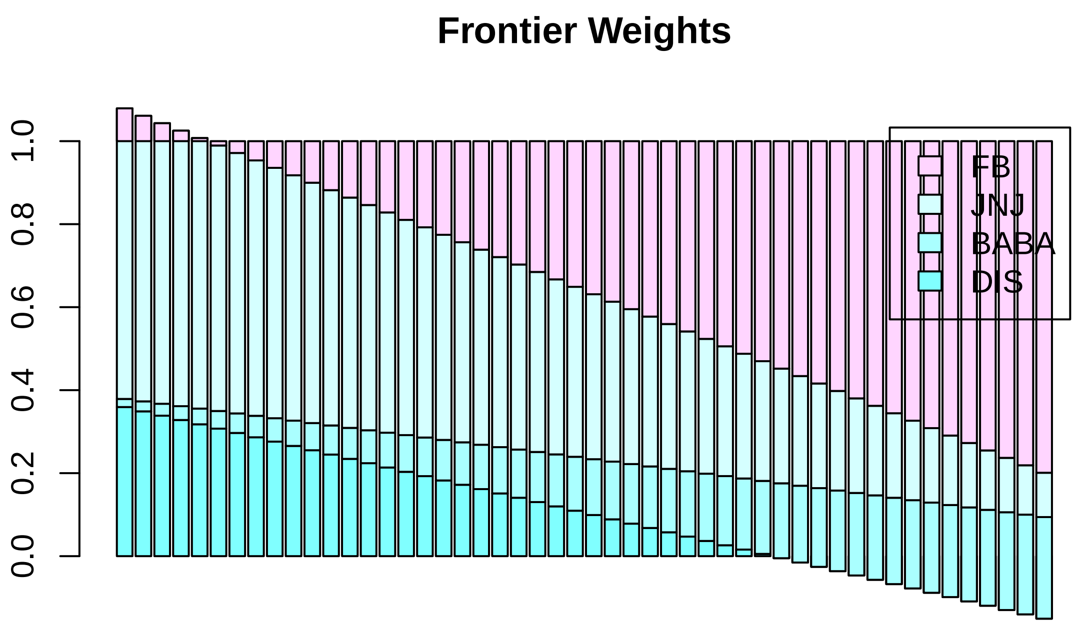
```
## Warning in as.vector(invSigma %*% one)/(one %*% invSigma %*% one): Recycling array of length 1 in ve
##   Use c() or as.vector() instead.
```

```
weights <- getWeights(effFrontierShort)
write.csv(weights, "weightsShort.csv")
colnames(weights) <- tickers
#Plot the efficient frontier with minimun variance portfolio and tangency portfolio.
plot(effFrontierShort, c(1, 2, 3,4))
```

```
## Warning in as.vector(invSigma %*% one)/(one %*% invSigma %*% one): Recycling array of length 1 in ve
##   Use c() or as.vector() instead.
```



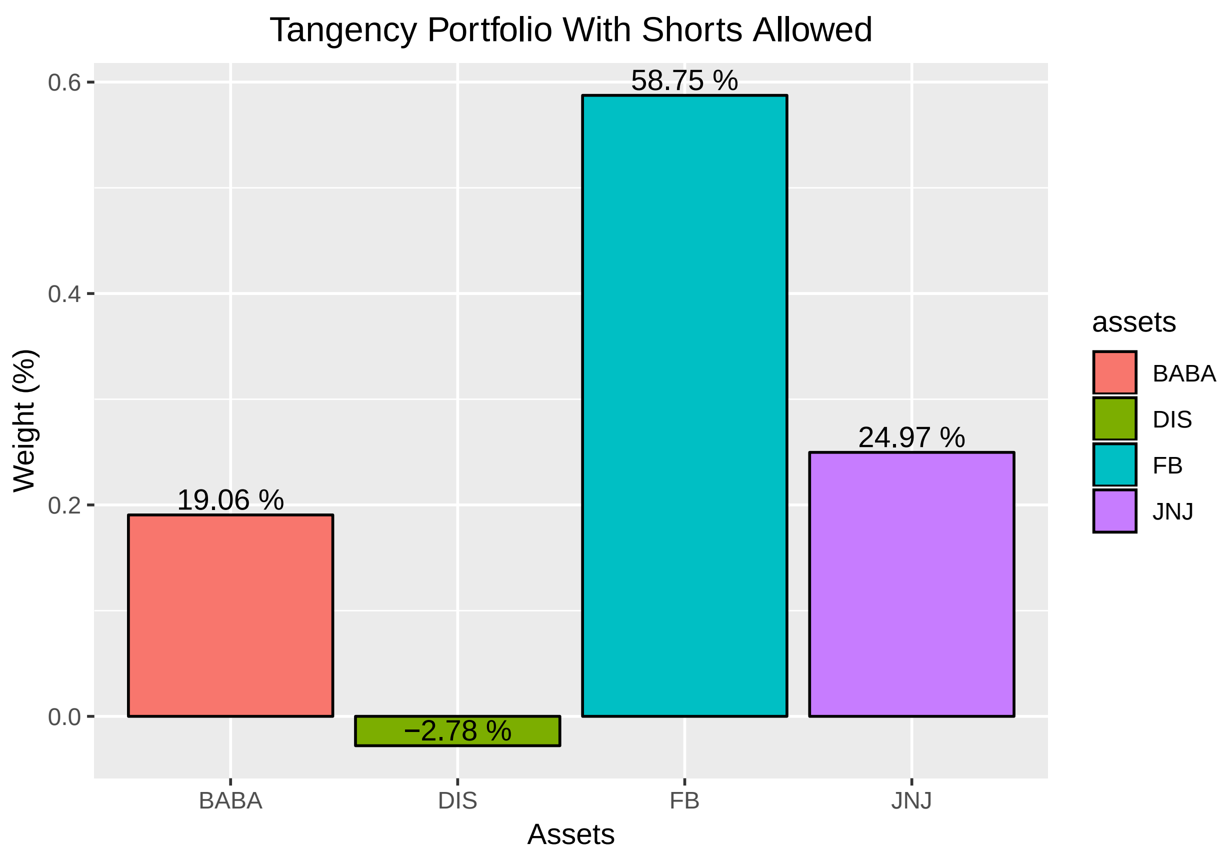

```
#Plot Frontier Weights (Need to transpose matrix first)
barplot(t(weights), main="Frontier Weights",
        col=cm.colors(ncol(weights)+2), legend=colnames(weights))
```




```
effPortShort <- minvariancePortfolio(portfolioReturns, Spec, constraints=constraints)
```

```
## Warning in as.vector(invSigma %*% one)/(one %*% invSigma %*% one): Recycling array of length 1 in ve
##   Use c() or as.vector() instead.
```

```
optWeights <- getWeights(effPortShort)
tanPortShort <- tangencyPortfolio(portfolioReturns, Spec, constraints=constraints)
tanWeights <- getWeights(tanPortShort)
#maxR <- maxreturnPortfolio(portfolioReturns , Spec, constraints=constraints)
#maxWeights <- getWeights(maxR)

#ggplot MVP Weights
df <- data.frame(tanWeights)
assets <- colnames(frontierWeights)
ggplot(data=df, aes(x=assets, y=tanWeights, fill=assets)) +
  geom_bar(stat="identity", position=position_dodge(),colour="black") +
  geom_text(aes(label=sprintf("%.02f %%",tanWeights*100)),
            position=position_dodge(width=0.9), vjust=-0.25, check_overlap = TRUE) +
  ggtitle("Tangency Portfolio With Shorts Allowed")+
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(x= "Assets", y = "Weight (%)")
```



Markowitz model with a risk free asset

function to create asset allocation

```
graph.asset.allocation = function(portfolio,sd,mu, png.name="optimal-asset-allocation-mu.png",title="Ma
  nr = dim(portfolio)[1]
  titles = row.names(portfolio)
  #print(portfolio)
  #print(titles)

  par(mar=c(5,4,4,5)+.1)
  plot(mu,t(portfolio[1,]),ylim=c(min(portfolio),max(portfolio)),type='l',col=1,lty=1,ylab="x* asset al
  for (j in 2:nr) lines(mu,t(portfolio[j,]),type='l',col=j,lty=j)
  par(new=TRUE)
  plot(mu, sd,type="l",col="darkgray",lwd=3,lty=6,xaxt="n",yaxt="n",xlab="",ylab="")
  axis(4)
  mtext("sd(x*(mu))",side=4,line=3,col="darkgray")
  legend("topleft", c(titles), col=c(1:nr), lty=1:nr,bty="n",lwd=rep(1,nr))
}
```

```
markowitz.portfolio.cash = function(mu.ret, Cov.Matrix=diag(length(mu.ret)), mu.portfolio.min = 0, r0=0
  if (missing(mu.ret)) stop("need vector of expected asset returns: mu.ret")
  titles= names(mu.ret)

  nr = length(mu.ret)
  if (sum(dim(Cov.Matrix)==nr)<2) stop("wrong dimensions")
```

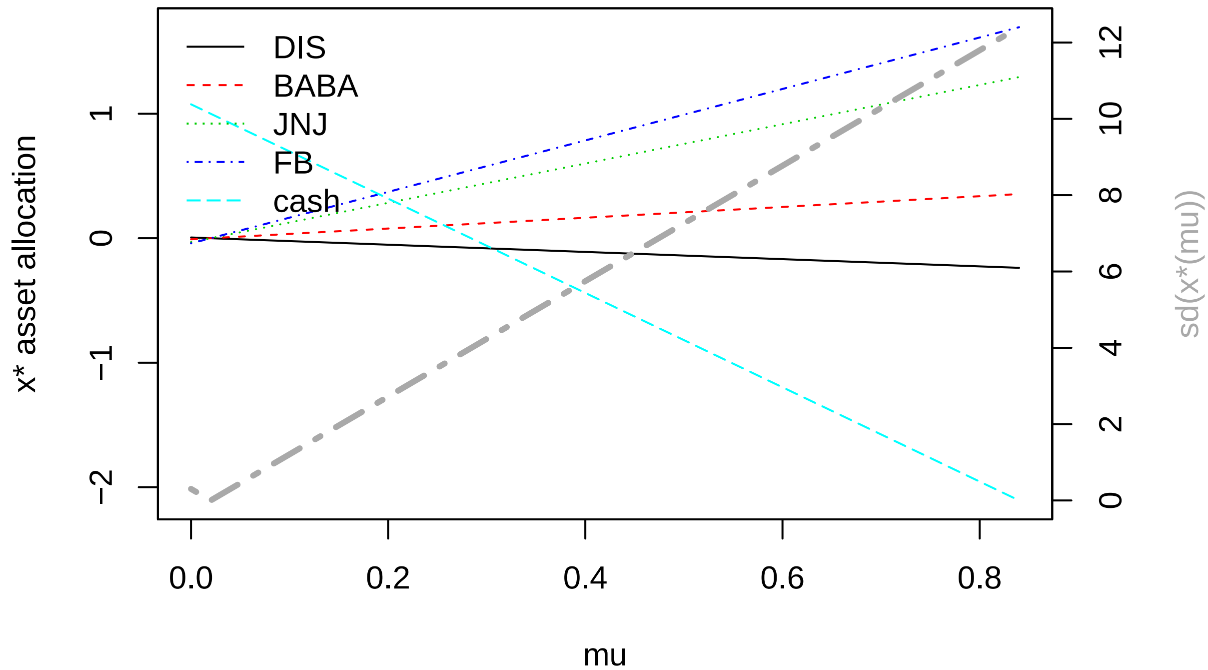
```
  ones = rep(1,nr)
  Cov.inv = solve(Cov.Matrix)

  m = length(mu.portfolio.min)
  xm.r0 = matrix(nrow = nr+1, ncol=m)
  #first nr compontens usual assets x^star, nr+1 = cash(r0) x0^star

  a = as.numeric(mu.ret%*% Cov.inv %*% mu.ret)
  b = as.numeric(mu.ret%*% Cov.inv %*% ones )
  c = as.numeric(ones %*% Cov.inv %*% ones )
  d = a-2*b*r0+c*r0*r0

  for (k in 1:m){
    xm.r0[,k] = c((mu.portfolio.min[k]-r0)*(Cov.inv%*%mu.ret - r0 * Cov.inv%*%ones),d-(b-r0*c)*(mu.portf
  }

  if (length(titles)==0) titles=paste(rep("asset",nr),1:nr)
  print(titles)
  row.names(xm.r0) = c(titles,"cash")
  standarddev = sqrt(((mu.portfolio.min-r0)^2/d))
  return.list = list(xm.r0/d,standarddev)
  names(return.list) = c("efficient_portfolio","standarddev")
  return(return.list)
}

r0 = 0.02

mu.new = seq(from=0.0, by=0.01, to=0.84)

portfolio.riskfree = markowitz.portfolio.cash(returns,Covar,mu.new,r0)
```

```
## [1] "DIS"  "BABA" "JNJ"  "FB"
```

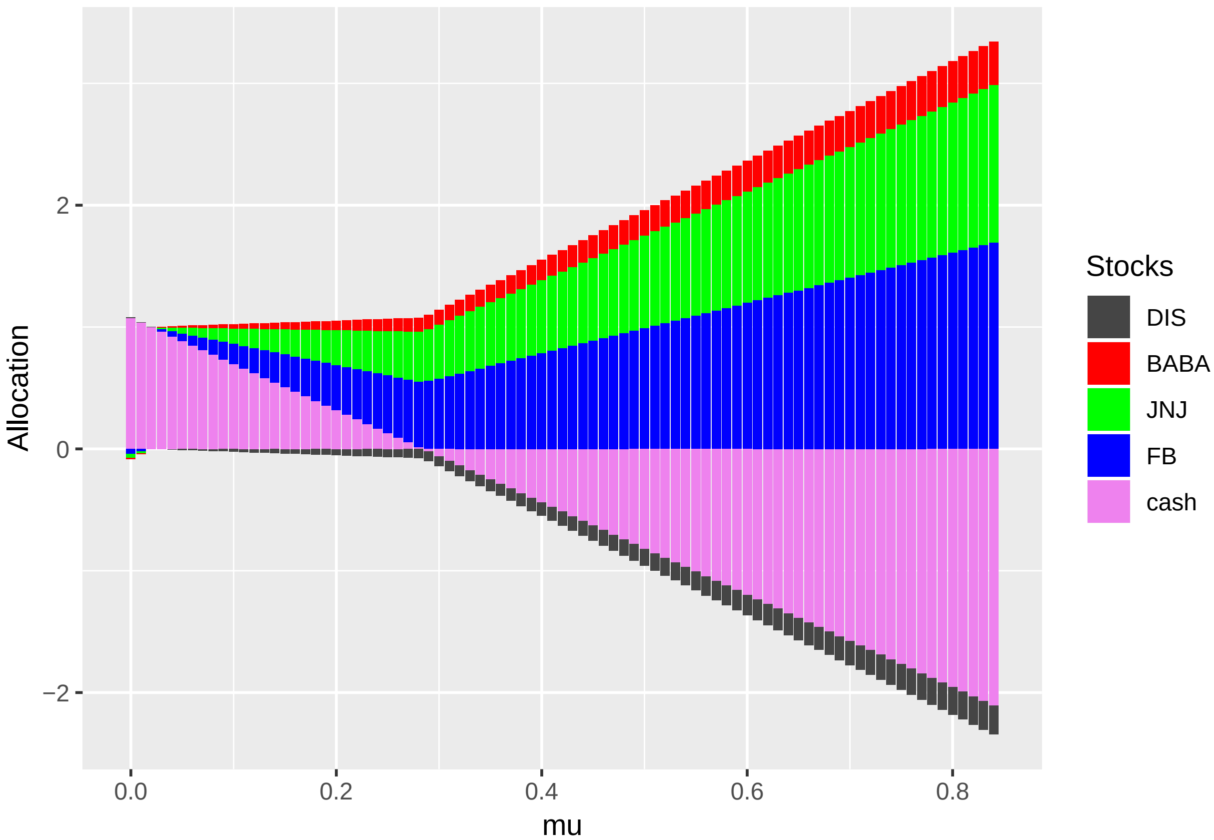
```
colSums(portfolio.riskfree$efficient_portfolio)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [71] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
graph.asset.allocation(portfolio=portfolio.riskfree$efficient_portfolio,sd=portfolio.riskfree$standardde
                       mu=mu.new)
```




```r
# Using stacked barplot
asset_free <- data.frame(mu.new,t(portfolio.riskfree$efficient_portfolio))
library(reshape2)
asset_free <- melt(asset_free, id = "mu.new")
names(asset_free) <- c("mu","Stocks","Allocation")
ggplot() + geom_bar(aes(y = Allocation, x = mu, fill = Stocks), data = asset_free, stat="identity")+
  scale_fill_manual("Stocks",values =
                      c("FB"="#0000FF","JNJ"="#00FF00","BABA"="#FF0000","DIS"="#454545","cash"="violet")
```



numerical solution for our problem with risk free asset

```
(mu.ret <- returns)
```

```
##       DIS      BABA       JNJ        FB
## 0.1005307 0.2686097 0.1532568 0.3612616
```

```
r0 = 0.02
(a = as.numeric(mu.ret%*% inv_C %*% mu.ret))
```

```
## [1] 0.005081476
```

```
(b = as.numeric(mu.ret%*% inv_C %*% ones ))
```

```
## [1] 0.01892006
```

```
(c = as.numeric(ones %*% inv_C %*% ones ))
```

```
## [1] 0.1178991
```

```
(d = a-2*b*r0+c*r0*r0)
```

```
## [1] 0.004371833
```

```
(mu_0 = (a - b*r0)/(b-c*r0))
```

```
## [1] 0.2839664
```

```
denominator <- r0 - mu_0
sol_risky <- (1/(b-r0*c))*((inv_C %*% returns) - (r0 * inv_C %*% ones))
sol_risky
```

```
##              [,1]
## DIS  -0.07655683
## BABA  0.11422041
## JNJ   0.41668814
## FB    0.54564828
```