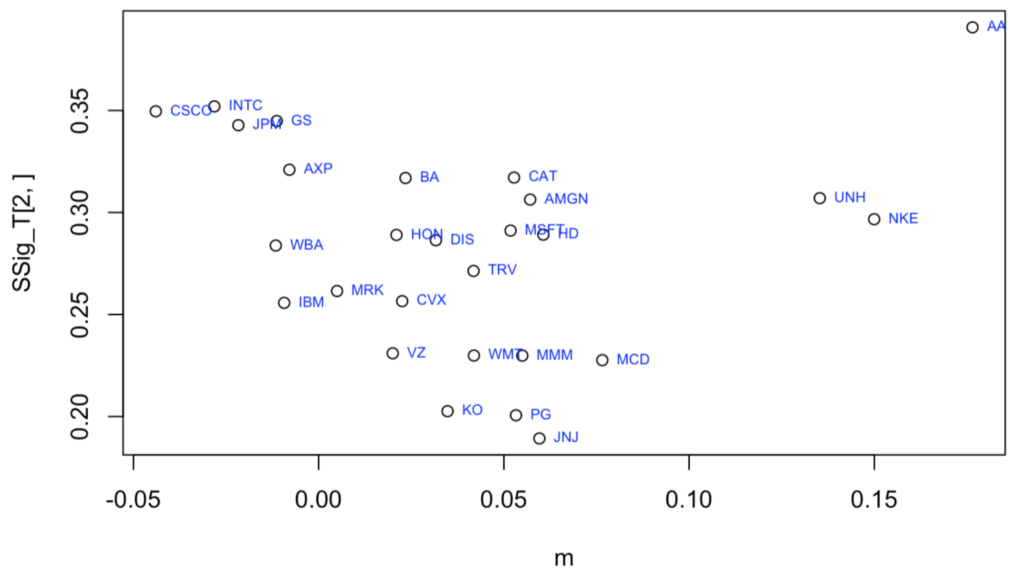
1.Performance Summary

Q1 Summary consists of a 3 × 3 table.

	Mu_T <dbl>	Sig_T <dbl>	SR_T <dbl>
Min	-1.24007798	0.09057667	-1.7884057
Mean	0.08582597	0.28297726	0.4968323
Max	1.19018268	1.01529441	3.8749416

Q2: Plot the asset mean returns against their volatilities



Q3(a): Report the relative performance measures in a 5 × 3 summary table

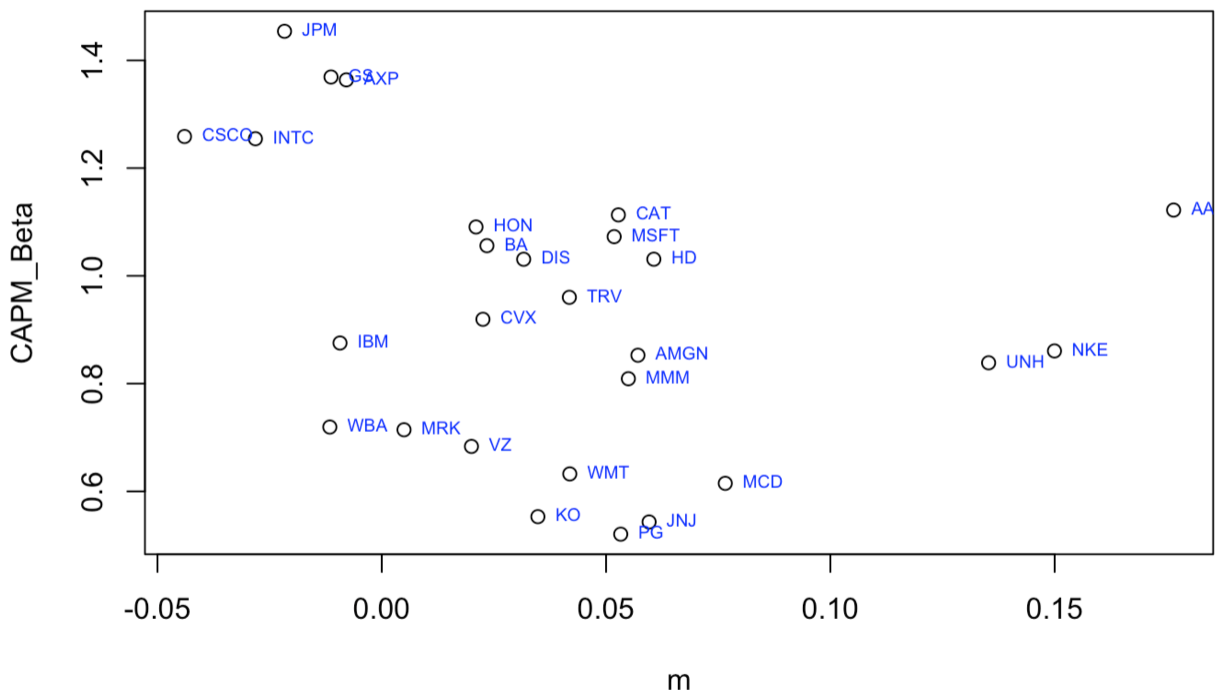
	Alpha	Beta	Tracking Error	Information Ratio	Treynor Ratio
Minimum	-0.0002000000	0.5205000	0.1825000	-0.28830000	-0.03490000
Meu	0.0001074074	0.9375519	0.2465667	-0.01945185	0.04856667
Maximum	0.0008000000	1.4540000	0.3669000	0.40860000	0.17430000

Q3(b): Best & Worst Stocks

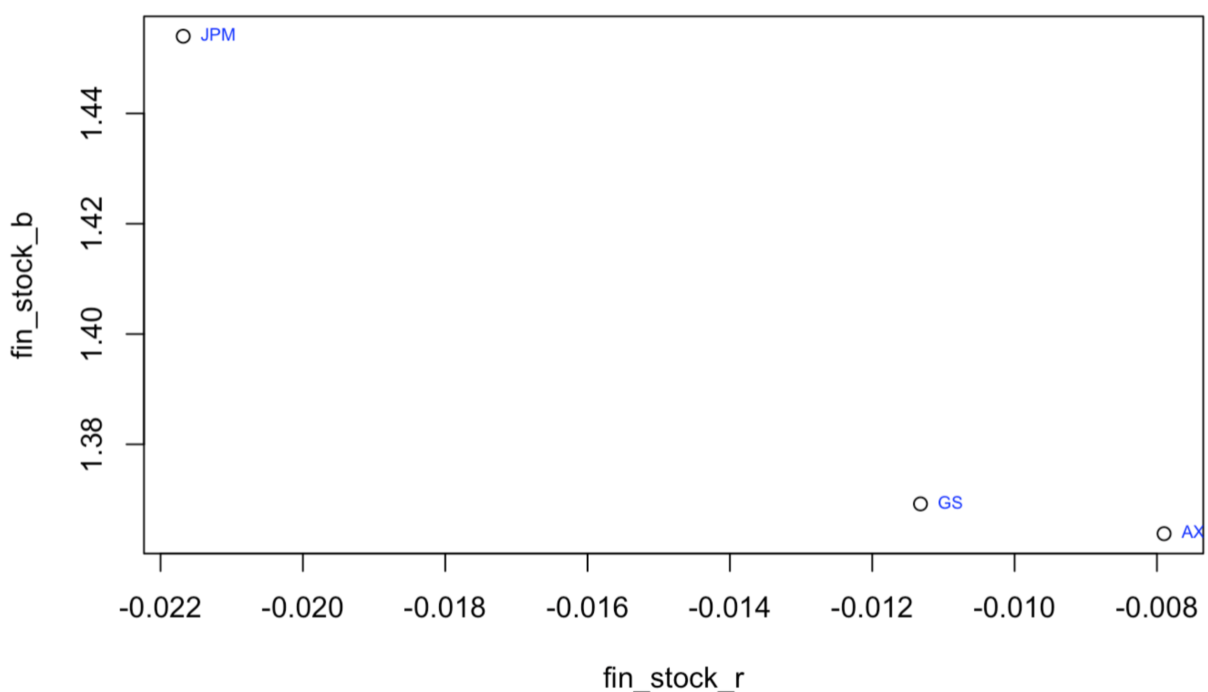
Best: NKE relative high Treynor Ratio with low Tracking Error

Worst: CSCO, the stock has the lowest negative A negative ratio Treynor Ratio, indicates that the investment has performed worse than a risk free instrument.

Q4(a): Plot the mean return of each asset against its beta.



Q4(b):Plot JPM, GS, and AXP



2 Back-Testing

Q1

[1] "SPY"

the range of in-sample is 501

the range of out-of-sample is 440

Q2

the 2-year cumulative return of portfolio 1 is 0.1175789

the 2-year cumulative return of portfolio 2 is 0.2184694

the 2-year cumulative return of portfolio 2 is 0.1173892

weight1_percentage <fctr>	weight2_percentage <fctr>	weight3_percentage <fctr>
2.45054041472494%	7.46419991674283%	3.7037037037037%
3.2369676844427%	5.42822130699952%	3.7037037037037%
3.71471609723677%	4.65673520289233%	3.7037037037037%
2.14613891430803%	18.7642997770272%	3.7037037037037%
1.86925594309663%	8.42023379619387%	3.7037037037037%
2.92617772514037%	7.80690503826973%	3.7037037037037%
3.52302976475353%	-1.62702566543796%	3.7037037037037%
4.13440264818881%	0.621470468173537%	3.7037037037037%
2.63489967206207%	-8.67533607200302%	3.7037037037037%
4.05769518692003%	4.66787082169104%	3.7037037037037%

1-10 of 27 rows Previous 1 2 3 Next

Q3.1

Q3.2

portfolio <fctr>	beta <fctr>	alpha <fctr>	SR <fctr>
portfolio1	0.979822688501692	-0.0924436090781786	3.98967722299554
portfolio2	1.12420177904119	-0.0583139690860298	6.29792561726823
portfolio3	1.01756812460256	-0.0409413400844204	3.85197803484115

3 rows

the 2-year cumulative return of portfolio 1 is 0.1175789

the 2-year cumulative return of portfolio 2 is 0.2184694

the 2-year cumulative return of portfolio 2 is 0.1173892

Q3.3

Portofolio 2 should be pick, since it has the largest cumulative return while the sharpe rate is big enough
Since each of three portfolio fails to “beat the market”, the EMH has been buttressed by this example
2
Another possible explanation is that since the Dow&Jones is constituted by large company, according to the ‘the small company e ect’ In the behaviour finance, the return of Dow&Jones fails to beat the overall stock market which includes some small company

3 Random Numbers and Monte Carlo Simulation

Q1. Game 2 Phase 2:

The maximum amount I am willing to pay is 0.059, with 10^5 simulation times.

```
```{r}
N = 10^5
seq1 <- numeric()
for (n in 1:N) {
  c1 <- sample(1:6,6,TRUE)
  c2 <- (max(c1)-min(c1)) < 3
  seq1 <- c(seq1,c2)
}

round(mean(seq1),4)
```
```

```
[1] 0.059
```

Q2. Breaking Even:

The K value makes this game break-even is 0.2499388(is near 0.25), with 10^5 simulation times.

```
```{r}
seq3 <- numeric()
for (i in 1:10^5){
  seq2 <- numeric()
  while(sum(seq2)<2){
   seq2 <- c(seq2,sample(0:1,1))
  }
  time <- length(seq2)
  seq3 <- c(seq3,time)
}

100000/sum(seq3)
```
```

```
[1] 0.2499388
```

Q3. Pi

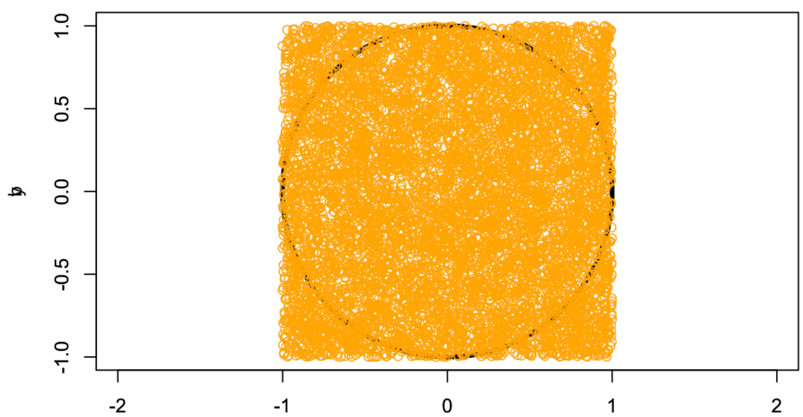
The idea to calculate pi value is based on the formula for calculating the area of circle:
when the radius of A is 1, the area of circle A is pi; therefore, the probability that one point (a,b), a,b are range from -1 to 1, located in the circle A, is equal to pi value.

```{r}
f <- function(n){
  seq4 <- numeric()
  a <- runif(n,-1,1)
  b <- runif(n,-1,1)
  distance <- sqrt(a^2+b^2)
  I <- distance<1
  p <- 4*mean(I)

  library(ggplot2)
  f=seq(0,2*pi,0.001)
  x=sin(f)
  y=cos(f)
  plot(x,y,type='l',xlim=c(-1,1),ylim=c(-1,1),asp=1,col="black",lwd = 4)
  par(new=TRUE)
  plot(a,b,xlim=c(-1,1),ylim=c(-1,1),asp=1,col="orange",lwd=1)
  return (p)
}

f(8000)
```

The pi value is 3.146 with 8000 simulation times.



This chat is circle A, whose radius is 1, and 8000 simulation plots.

Q4

(a)

```
$$
E[Y_k] =E[X^k]= \int_0^1 x^k dx =  \frac{x^{k+1}}{k+1} |_0^1 = \frac{1}{k+1}\,\,\,\,\, if \,\,\,\,\,k>-1
$$
```

$$E[Y_k] = E[X^k] = \int_0^1 x^k dx = \frac{x^{k+1}}{k+1}|_0^1 = \frac{1}{k+1} \quad if \quad k>-1$$

```
According to expression of the expected value of $Y_k$:
$$
V[Y_k] = E[Y_k^2]-E^2[Y_k] = E[X^{2k}] - E^2[X^k] = \frac{1}{2k+1} - \frac{1}{(k+1)^2}\,\,\,\,\,if\,\,\,\,k>-1
$$
```

$$V[Y_k] = E[Y_k^2] - E^2[Y_k] = E[X^{2k}] - E^2[X^k] = \frac{1}{2k+1} - \frac{1}{(k+1)^2} \quad if \quad k>-1$$

(b)

Because $E[Y_k]$ and $V[Y_k]$ are finite, $k \neq 1/2$ and $k> -1$, and because $V[Y_k]>0$, $k \neq 0$.

(c)

```{r}
Q <- function (k){
  X <- runif(10^5,0,1)
  Y <- X^k
  result <- c(mean(Y),var(Y))
  return (result)
}

C <- function(k){
  M <- (1-0^(k+1))/(k+1)
  V <- (1-0^(2*k+1))/(2*k+1)-((1-0^(k+1))/(k+1))^2
  return (c(M,V))
}

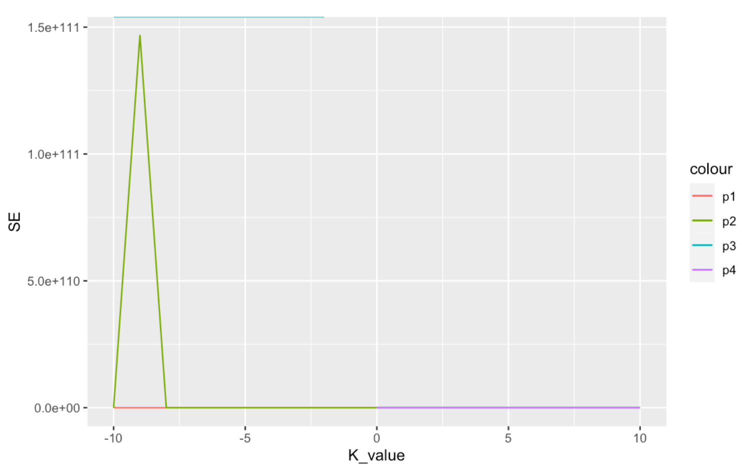
table <- data.frame(t(rbind(sapply(c(-10:10),Q),sapply(c(-10:10),C))),row.names = c(-10:10))
colnames(table) <- c("SE","SV","CE","CV")
table$K_value = c(-10:10)
table

ggplot(table, aes(x=K_value)) +
  geom_line(aes(y=SE, color="p1")) +
  geom_line(aes(y=SV, color="p2")) +
  geom_line(aes(y=CE, color="p3"))+
  geom_line(aes(y=CV, color="p4"))
```

SE","SV","CE","CV" stand for simulate mean, simulate variance, calculated mean and calculated variance respectively. Get the following table and chart:

	SE <dbl>	SV <dbl>	CE <dbl>	CV <dbl>	K_value <int>
-10	2.908000e+44	8.456424e+93	Inf	NaN	-10
-9	1.211319e+53	1.467294e+111	Inf	NaN	-9
-8	3.032499e+33	3.566222e+71	Inf	NaN	-8
-7	1.198433e+28	9.267226e+60	Inf	NaN	-7
-6	6.638596e+22	4.388001e+50	Inf	NaN	-6
-5	1.014769e+23	1.029644e+51	Inf	NaN	-5
-4	9.810555e+12	3.617821e+30	Inf	NaN	-4
-3	6.691123e+10	4.014237e+26	Inf	NaN	-3
-2	1.119744e+05	1.884359e+14	Inf	NaN	-2
-1	1.151884e+01	1.810231e+05	NaN	NaN	-1

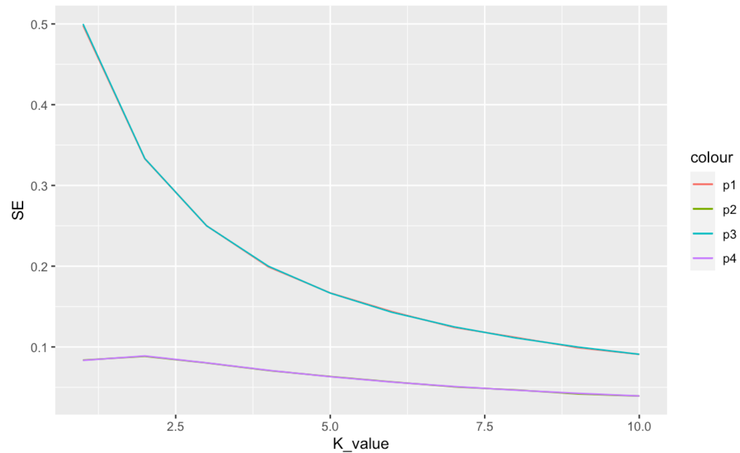
1-10 of 21 rows Previous 1 2 3 Next



Since when K is range from -10 to -1, the calculated mean and calculated variance is infinite or NaN value, the lines in the chart are irregular.

(d)

Based on the restrictions of K value in (b), K is larger than -1 and not equal to 0, I draw a new chart:



From the above chat we can see that p1(simulate mean value) and p3(calculated mean value) overlap, p3(simulate variance value) and p4(calculated variance value) overlap, which means simulated values are nearly equal to calculated values.

Bonus:

First Let's look at simple case, if the required head number is 1 instead of 2:
Assume that we need to use expected n times to finish the simple game: then according to the first roll result, it satisfies the equation:

$$nk = \frac{1}{2} * k + \frac{1}{2}(n+1)k$$

then, we have n = 2, which is the expected game times in simple the case.
Let's see the complex case when required head number is 2:
Again, assume that we need to use expected m times to finish the complex game: then according to the first roll result, if we success, we will enter the simple game, if we fails, we will return to the begin of complex game, which satisfies the equation:

$$mk = \frac{1}{2} * (1+2)k + \frac{1}{2} * (1+m)k$$

then, the expected value of m is 4: then because mk=1, k=1/4
Notice here, k is not equal to E[1/X], k = 1/E[X], for example, if we play 100000 times game, the expected roll time is 400000, to make this total game fair, the expected value of k should be 1/4. Making every game fair and calculating average is a wrong method.

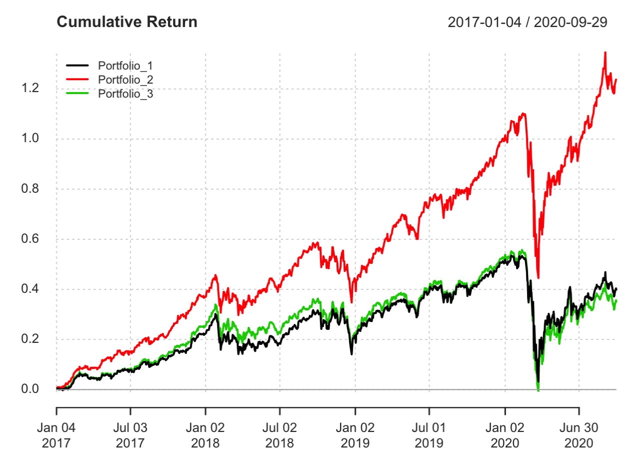
4 Value at Risk and Stress Testing

Task 1

Portfolio 1 -- GMV portfolio; Portfolio 2 -- Sharpe-ratio portfolio; Portfolio 3 -- naive portfolio

We refer to PerformanceAnalytics package by (Peterson and Carl 2018) to visualize the performance of the three portfolios.

```
chart.CumReturns(R_p, main = "Cumulative Return", legend.loc = "topleft")
```

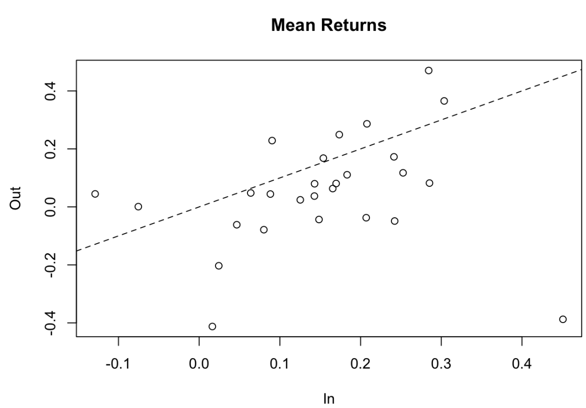


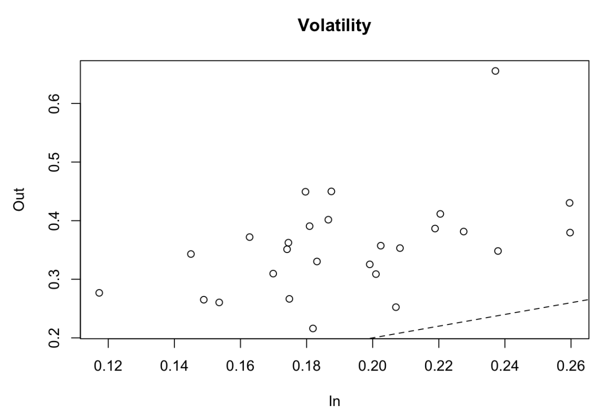
Descriptive summary

	Portfolio_1	Portfolio_2	Portfolio_3
mean	0.1082536	0.2362662	0.1023947
std	0.1937506	0.2028444	0.2109120
SR	0.5587267	1.1647660	0.4854856

Question 1

To get started with the backtesting, we split the portfolio return into two periods in-sample (IN) and out-of-sample (OUT). Then estimate μ_i and σ_i using the IN to construct Portfolio 1, 2 and 3. After that, estimate the corresponding parameters from the OUT to demonstrate the sensitivity of each over time.



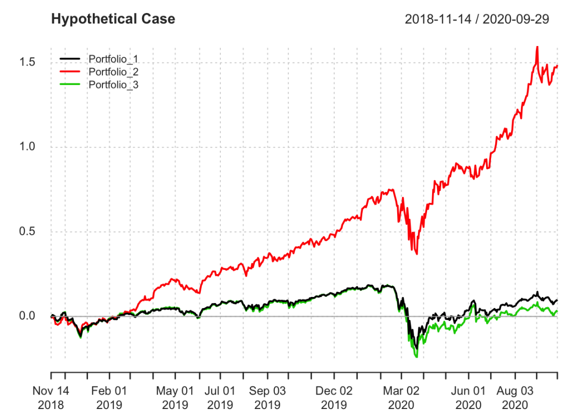


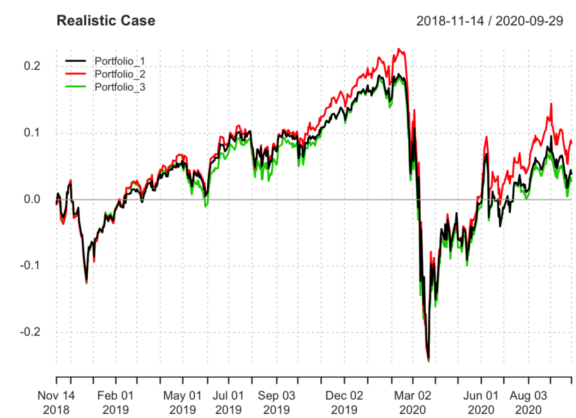
It appears that σ s exhibit a lower sensitivity than μ s, making the mean returns are more susceptible to model risk.

```
print(mean((M_in - M_out) ^ 2))
## [1] 0.05103022
```

```
print(mean((S_in - S_out) ^ 2))
## [1] 0.032454
```

Take a look on how the portfolio strategy performs



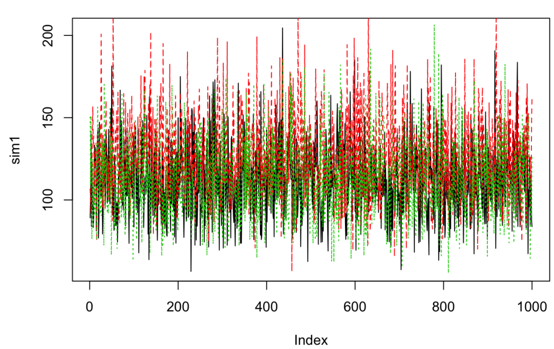


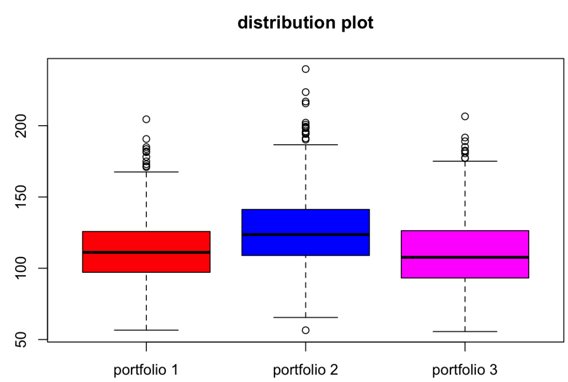
Mean and Standard deviation for p=1,2,3 using the daily returns in the OUT period:

	Portfolio_1	Portfolio_2	Portfolio_3
miu	0.0003058959	0.002083584	0.0002077179
sigma	0.0155675296	0.017153771	0.0173447471

Question 2

Simulated path of Geometric Brownian Motion





Portfolio 1 is a Global Minimum Variance portfolio which focuses on controlling the risk to the lowest, therefore which do have the lowest volatility but lack considerable rewards similar to portfolio 3 that simply equally distribute the funds. While Sharpe-ratio portfolio (p2) has a consideration on the risk-adjusted return of stocks allowing investors to have a bias on reward(highest mean) within undertaking a higher risk(highest variance).

Question 3

The expected value one year from now on is

```
                      p1       p2       p3
expected value 83.77982 163.1078 127.7484
```

Question 4

With 95% level of confidence, the Value-at-Risk is

```
          p1       p2      p3
VaR 34.57601 37.77015 34.0068
```

Task 2

Referring to the SPY ETF as the markets, directly, we use the table.CAPM from the PerformanceAnalytics package to attain a number of statistics.

```
SPY <- get(getSymbols("SPY", from = "2017-01-01", to = "2020-09-30"))[, 6]
R_m <- na.omit(log(SPY / lag(SPY)))
names(R_m) <- "SPY"

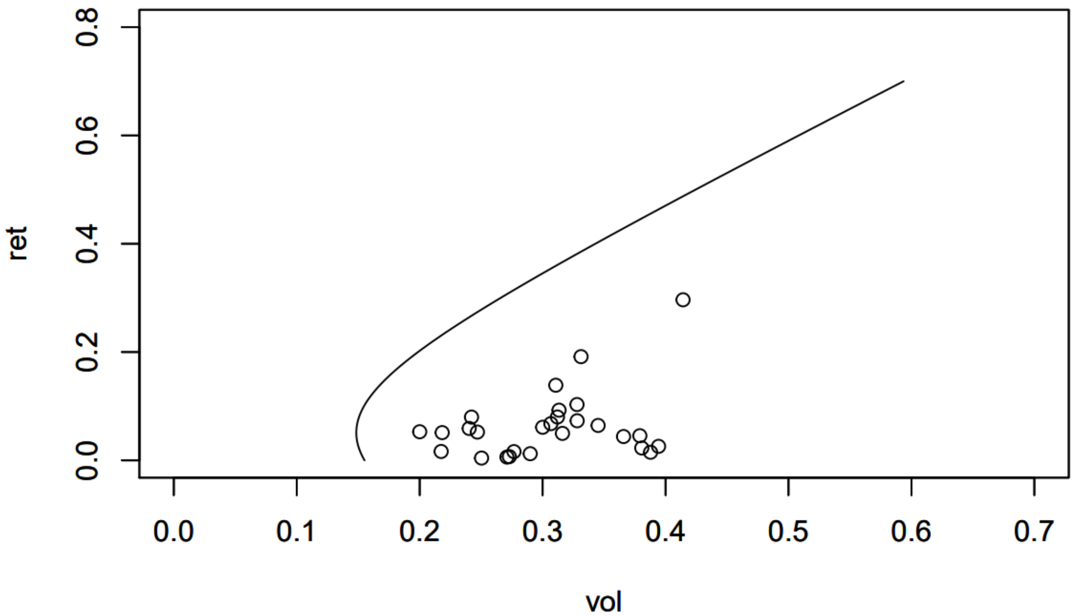
table.CAPM(R_p_out, R_m)
```

VaR(0.05) for each portfolio is

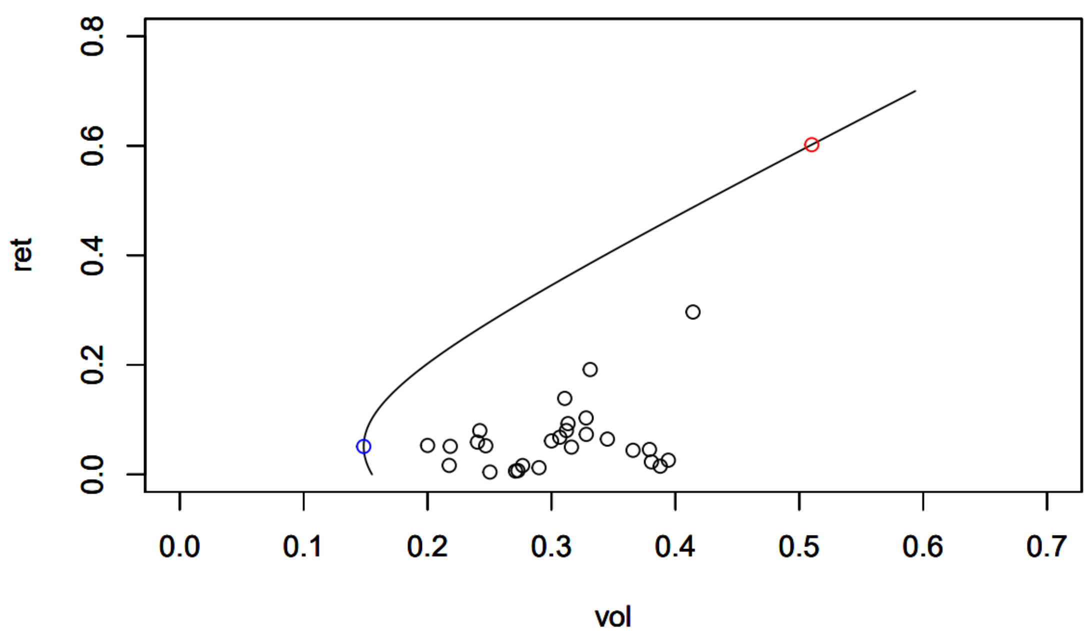
```
          p1       p2       p3
VaR 3.014278 3.906139 3.149279
```

5 Mean-Variance Efficient Frontier

Q1. Plot the MVEF

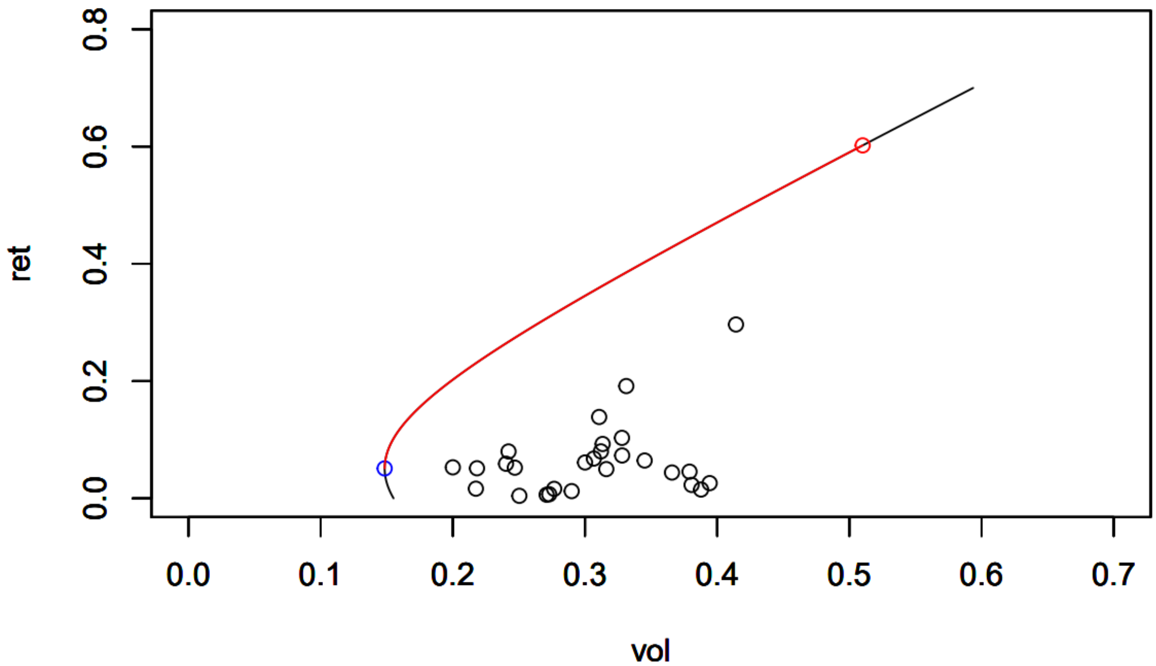


Q2. Highlight SR & GMV points



Q3. Highlight EF & the economic rationale of theta

If theta>1, it means that we will short the SR portfolio and long the low risk portfolio. In other words, I will sell the SR portfolio despite not having the SR portfolio to get the money, and use the money and the principle to buy the low risk portfolio.



Appendix

```
library(quantmod)

library(lubridate)

install.packages('PerformanceAnaly
tics' )

install.packages('matrixStats' )

library(matrixStats)

library(PerformanceAnalytics)

##Part 1

v <-
c('AAPL','CSCO','HON',"KO","NK
E",'WBA',"AMGN","CVX",

"IBM","MCD",'PG','WMT',"AXP",
"DIS","INTC","MMM",

"TRV","BA","GS","JNJ","MRK","
UNH","CAT",

    'HD','JPM',"MSFT","VZ")

P <-
get(getSymbols(v,from="1999-05-0
1", to="2020-09-30"))

P1 <- AAPL$'AAPL.Adjusted'

P2 <- CSCO$'CSCO.Adjusted'

P3 <- HON$'HON.Adjusted'

P4 <- KO$'KO.Adjusted'

P5 <- NKE$'NKE.Adjusted'

P6 <- WBA$'WBA.Adjusted'

P7 <- AMGN$'AMGN.Adjusted'

P8 <- CVX$'CVX.Adjusted'

P9 <- IBM$'IBM.Adjusted'

P10 <- MCD$'MCD.Adjusted'

P11 <- PG$'PG.Adjusted'

P12 <- WMT$'WMT.Adjusted'

P13 <- AXP$'AXP.Adjusted'

P14 <- DIS$'DIS.Adjusted'

P15 <- INTC$'INTC.Adjusted'

P16 <- MMM$'MMM.Adjusted'

P17 <- TRV$'TRV.Adjusted'

P18 <- BA$'BA.Adjusted'

P19 <- GS$'GS.Adjusted'

P20 <- JNJ$'JNJ.Adjusted'

P21 <- MRK$'MRK.Adjusted'

P22 <- UNH$'UNH.Adjusted'

P23 <- CAT$'CAT.Adjusted'

P24 <- HD$'HD.Adjusted'

P25 <- JPM$'JPM.Adjusted'

P26 <- MSFT$'MSFT.Adjusted'

P27 <- VZ$'VZ.Adjusted'

Price <-
cbind(P1,P2,P3,P4,P5,P6,P7,P8,

P9,P10,P11,P12,P13,P14,P15,

P16,P17,P18,P19,P20,P21,P22,

        P23,P24,P25,P26,P27)

` ` `

## Compute the daily return

` ` `{r}

R <- na.omit(log(Price/lag(Price)))

m <- Return.annualized(R)

std <- colSds(R)

dt <- 1/252

sig <- (1/sqrt(dt))*std

` ` `

## Q1(a): Daily return in annual
terms & summary for each measure

` ` `{r}

choose_years <- 1999:2020

R_sub <- R[year(R) %in%
choose_years,]

R_sub <- R["1999-01-01/",]

Mu_T <-
252*apply.yearly(R_sub,mean)

Sig_T <-
cbind(sqrt(252)*apply.yearly(R_sub
[,1],sd),
sqrt(252)*apply.yearly(R_sub[,2],sd
),
sqrt(252)*apply.yearly(R_sub[,3],sd
),
sqrt(252)*apply.yearly(R_sub[,4],sd
),
sqrt(252)*apply.yearly(R_sub[,5],sd
),
sqrt(252)*apply.yearly(R_sub[,6],sd
),
sqrt(252)*apply.yearly(R_sub[,7],sd
),
sqrt(252)*apply.yearly(R_sub[,8],sd
),
sqrt(252)*apply.yearly(R_sub[,9],sd
),
sqrt(252)*apply.yearly(R_sub[,10],s
d) ,
sqrt(252)*apply.yearly(R_sub[,11],s
```

d),
sqrt(252)*apply.yearly(R_sub[,12],sd),
sqrt(252)*apply.yearly(R_sub[,13],sd),
sqrt(252)*apply.yearly(R_sub[,14],sd),
sqrt(252)*apply.yearly(R_sub[,15],sd),
sqrt(252)*apply.yearly(R_sub[,16],sd),
sqrt(252)*apply.yearly(R_sub[,17],sd),
sqrt(252)*apply.yearly(R_sub[,18],sd),
sqrt(252)*apply.yearly(R_sub[,19],sd),
sqrt(252)*apply.yearly(R_sub[,20],sd),
sqrt(252)*apply.yearly(R_sub[,21],sd),
sqrt(252)*apply.yearly(R_sub[,22],sd),
sqrt(252)*apply.yearly(R_sub[,23],sd),
sqrt(252)*apply.yearly(R_sub[,24],sd),
sqrt(252)*apply.yearly(R_sub[,25],sd),
sqrt(252)*apply.yearly(R_sub[,26],sd),
sqrt(252)*apply.yearly(R_sub[,27],sd))

SR_T <- Mu_T/Sig_T

SMu_T<-rbind(colMins(Mu_T),colMeans(Mu_T),colMaxs(Mu_T))

SSig_T<-rbind(colMins(Sig_T),colMeans(Sig_T),colMaxs(Sig_T))

SSR_T<-rbind(colMins(SR_T),colMeans(SR_T),colMaxs(SR_T))

row.names(SMu_T)<-c('Mins','Means','Maxs')

row.names(SSig_T)<-c('Mins','Means','Maxs')

row.names(SSR_T)<-c('Mins','Means','Maxs')

print('Summary Mu of total')

SMu_T

print('Summary Sig of total ')

SSig_T

print('Summary SR of total')

SSR_T

```

## Q1(b): Summary consists of a 3 × 3 table.

```{r}

Performance1 <- data.frame(c(min(Mu_T),mean(Mu_T),max(Mu_T)),

c(min(Sig_T),mean(Sig_T),max(Sig_T)),

c(min(SR_T),mean(SR_T),max(SR_T)))

colnames(Performance1)<-(c('Mu_T','Sig_T','SR_T'))

rownames(Performance1)<-(c('Min','Mean','Max'))

Performance1

```

Q2: Plot the asset mean returns against their volatilities

```{r}

SSig_T[2,]

plot(m,SSig_T[2,])

#olMeans(Sig_T)

colnames(SSig_T)<-c('AAPL','CSCO','HON',"KO","NKE",'WBA',"AMGN","CVX",

"IBM","MCD",'PG','WMT',"AXP","DIS","INTC","MMM",

"TRV","BA","GS","JNJ","MRK","UNH","CAT",

'HD','JPM',"MSFT","VZ")

text(m, SSig_T[2,], colnames(SSig_T), cex=0.6, pos=4, col="blue")

```

Merge the SPY daily returns

```{r}

v <- 'SPY'

P <- get(getSymbols(v,from="1999-05-01", to="2020-09-30"))

Ps <- P$'SPY.Adjusted'

Ps <- na.omit(Ps)

Rs <- na.omit(log(Ps/lag(Ps)))

R_merge <- na.omit(merge(Rs,R,all = FALSE))

names(R_merge) <- c('SPY','AAPL','CSCO','HON',"KO","NKE",'WBA',"AMGN","CVX",

"IBM","MCD",'PG','WMT',"AXP","DIS","INTC","MMM",

"TRV","BA","GS","JNJ","MRK","UNH","CAT",

'HD','JPM',"MSFT","VZ")

```
```

Q3: compute the following measures for each stock:

(a) Jensen’s α

(b) Market β

(c) Treynor Ratio (TR)

(d) Tracking error (ω)

(e) Information Ratio (IR)

```{r}

CAPM_TOTAL <- table.CAPM(R_merge[,2:28],R_merge[,1])

CAPM_Alpha <- CAPM_TOTAL['Alpha',]

CAPM_Beta <- CAPM_TOTAL['Beta',]

CAPM_TE <- CAPM_TOTAL['Tracking Error',]

CAPM_IR <- CAPM_TOTAL['Information Ratio',]

CAPM_TR <- CAPM_TOTAL['Treynor Ratio',]

CAPM <- rbind(CAPM_Alpha, CAPM_Beta, CAPM_TE, CAPM_IR, CAPM_TR)

```

Q3(1): Report the relative performance measures in a 5 × 3 summary table

```{r}

Minimum <- c(min(CAPM[1,]),min(CAPM[2,]),min(CAPM[3,]),min(CAPM[4,]),min(CAPM[5,]))

Maximum <- c(max(CAPM[1,]),max(CAPM[2,]),max(CAPM[3,]),max(CAPM[4,]),max(CAPM[5,]))

Meu<-rowMeans(CAPM)

Performance2 <- rbind(Minimum,Meu,Maximum)

Performance2

```

Q4(a): Plot the mean return of each asset against its beta.

```{r}

plot(m,CAPM_Beta)

colnames(CAPM_Beta)<-c('AAPL','CSCO','HON',"KO","NKE",'WBA',"AMGN","CVX",


                       "IBM","MCD",'PG','WMT',"AXP","DIS","INTC","MMM",


                       "TRV","BA","GS","JNJ","MRK","UNH","CAT",

       'HD','JPM',"MSFT","VZ")

text(x=m, y=CAPM_Beta, colnames(CAPM_Beta), cex=0.6, pos=4, col="blue")

```

Q4(b): Plot JPM, GS, and AXP

```{r}

fin_stock_r <- c(m[,25],m[,19],m[,13])

fin_stock_b <- CAPM_Beta[c('JPM','GS','AXP')]

plot(fin_stock_r,fin_stock_b)

text(x=fin_stock_r, y=fin_stock_b, colnames(fin_stock_b), cex=0.6, pos=4, col="blue")

```

```{r}

CAPM_Beta['JPM to SPY']

m[,25]

CAPM_Beta['GS to SPY']

m[,19]

CAPM_Beta['AXP to SPY']

m[,13]

ax.text(-0.02168128, 1.454, 'JPM', fontsize=12, color = "r", style = "italic", weight = "light", verticalalignment='center', horizontalalignment='right', rotation=90)

ax.text(-0.01132219, 1.3692, 'GS', fontsize=12, color = "r", style = "italic", weight = "light", verticalalignment='center', horizontalalignment='right', rotation=90)

ax.text(-0.00789982, .3638, 'JPM', fontsize=12, color = "r", style = "italic", weight = "light", verticalalignment='center', horizontalalignment='right', rotation=90)

```

2 Back-Testing ## Q1

```{r include=FALSE} #preparation library(quantmod) library(tictoc, quietly = T) library(tidyverse, quietly = T) ```

```{r include=FALSE}

#import the data of stocks

tic()
getSymbols("AAPL",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("AMGN",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("AXP",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("BA",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("CAT",src="yahoo",from="2017-01-01",to="2020-09-30")

getSymbols("CSCO",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("CVX",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("DIS",src="yahoo",from="2017-01-01",to="2020-09-30")

getSymbols("GS",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("HD",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("HON",src="yahoo"

,from="2017-01-01",to="2020-09-30")
getSymbols("IBM",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("INTC",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("JNJ",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("JPM",src="yahoo",from="2017-01-01",to="2020-09-30")

getSymbols("KO",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("MCD",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("MMM",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("MRK",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("MSFT",src="yahoo",from="2017-01-01",to="2020-09-30")

getSymbols("NKE",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("PG",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("TRV",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("UNH",src="yahoo",from="2017-01-01",to="2020-09-30")
getSymbols("VZ",src="yahoo",from="2017-01-01",to="2020-09-30")

getSymbols("WBA",src="yahoo",from="2017-01-01",to="2020-

09-30")
getSymbols("WMT",src="yahoo",from="2017-01-01",to="2020-09-30")

toc() ```

```{R include=FALSE}
AAPL_Return = dailyReturn(AAPL[,4],type='log')
AMGN_Return = dailyReturn(AMGN[,4],type='log ')

AXP_Return = dailyReturn(AXP[,4],type = 'log')

MRK_Return = dailyReturn(MRK[,4],type = 'log')

dailyReturn(SPY[,4],type = 'log')

MarketReturn = as.data.frame(MarketReturn[-1,]) ```

```{r echo=FALSE}

df_return = as.data.frame(Return_of_stocks)

#delete first col

df = df_return[-1,]

#in sample start_date = 1 end_date = 501

#out of sample start_date1 = 502 end_date1 = 941

df_IN = df[start_date : end_date,]
df_OUT = df[start_date1 : end_date1,]

#-------------------------- -------------------------- ---

cat('the range of in-sample is', nrow(df_IN),'\n')

cat('the range of out-of-sample is', nrow(df_OUT),'\n')
#-------------------------- -------------------------- ----

```

Q2

```{r echo=FALSE}

Market <- MarketReturn[start_date1 : end_date1,]

sigma <- vector(length = 27)

for(i in 1:27){

sigma[i] = sd(df_IN[,i])

}

#portfolio 1

weight1 <- vector(length = 27) denominator1 <- 0

BA_Return dailyReturn(BA[,4],type='log') CAT_Return dailyReturn(CAT[,4],type='log')

= =

CSCO_Return dailyReturn(CSCO[,4],type

'log')

CVX_Return dailyReturn(CVX[,4],type='log') DIS_Return = dailyReturn(DIS[,4],type='log') GS_Return = dailyReturn(GS[,4],type = 'log') HD_Return = dailyReturn(HD[,4],type='log')

HON_Return = dailyReturn(HON[,4],type='log') IBM_Return = dailyReturn(IBM[,4],type = 'log') INTC_Return = dailyReturn(INTC[,4],type='log') JNJ_Return = dailyReturn(JNJ[,4],type = 'log') JPM_Return = dailyReturn(JPM[,4],type='log')

KO_Return = dailyReturn(KO[,4],type='log') MCD_Return = dailyReturn(MCD[,4],type = 'log') MMM_Return = dailyReturn(MMM[,4],type='log')

= =

=

MSFT_Return dailyReturn(MSFT[,4],type 'log')

= =

NKE_Return dailyReturn(NKE[,4],type='log') PG_Return dailyReturn(PG[,4],type = 'log') TRV_Return = dailyReturn(TRV[,4],type='log') UNH_Return dailyReturn(UNH[,4],type = 'log') VZ_Return = dailyReturn(VZ[,4],type = 'log')

WBA_Return = dailyReturn(WBA[,4],type = 'log') WMT_Return = dailyReturn(WMT[,4],type = 'log')
```

```{r include=FALSE}
Return_of_stocks =

merge(AAPL_Return,AMGN_Return,AXP_Return,BA_Return,CAT_Return,

CSCO_Return,CVX_Return,DIS_Return,GS_Return,HD_Return,

HON_Return,IBM_Return,INTC_Return,JNJ_Return,JPM_Return,

KO_Return,MCD_Return,MMM_Return,MRK_Return,MSFT_Return,

NKE_Return,PG_Return,TRV_Return,UNH_Return,VZ_Return,

WBA_Return,WMT_Return)

```

```{r echo=FALSE}

#market portfolio
getSymbols("SPY",src="yahoo",from="2017-01-01",to="2020-09-30")

MarketReturn =

= =

=

for(j in 1:27){

denominator1 = denominator1

+ (1/(sigma[j])^2) }

for(k in 1:27){

weight1[k] =

(1/(sigma[k])^2)/denominator1 }

#porfolio 2 denominator2=0

weight2 <- vector(length = 27) for(j in 1:27){

denominator2 = denominator2 + (sum(df_IN[,j])/2*(sigma[j]))

}

for (z in 1:27) {

weight2[z] =

(sum(df_IN[,z])/2*(sigma[z]))/denominator2

}

#portfolio 3

weight3 = vector(length = 27)

for (g in 1:27) { weight3[g] = 1/27

}

```

```{r echo=FALSE}

#calculate return

```
#p1

portfolio1_Return = vector(length =
nrow(df_OUT))

for (x1 in (1:nrow(df_OUT))) {
daily_return =

sum(weight1*df_OUT[x1,])
portfolio1_Return[x1] =

daily_return }

#p2

portfolio2_Return =

vector(length = nrow(df_OUT))

for (x2 in (1:nrow(df_OUT))) {
daily_return =

sum(weight2*df_OUT[x2,])
portfolio2_Return[x2] =

daily_return }

#p3

portfolio3_Return = vector(length =
nrow(df_OUT))

for (x3 in (1:nrow(df_OUT))) {
daily_return =

sum(weight3*df_OUT[x3,])
portfolio3_Return[x3] =

daily_return }

#Return of each porfolio cat("the
2-year cumulative return of
portfolio 1 is",
sum(portfolio1_Return), "\n")
cat("the 2-year cumulative return of
portfolio 2 is",
sum(portfolio2_Return), "\n")
cat("the 2-year cumulative return of
portfolio 2 is",
sum(portfolio3_Return), "\n")

weight1_percentage <-
paste(weight1*100, "%", sep='')
weight2_percentage <-
paste(weight2*100, "%", sep='')
weight3_percentage <-
paste(weight3*100, "%", sep='')

#table of weights (weights_table <-
as.data.frame(cbind(weight1_pe
rcentage,weight2_percentage,w
eight3_percentage)))

```

## Q3.1

```{r echo=FALSE} #plot

Market_cumulative <- vector(length
= 440)

for (a1 in 1:440) {
Market_cumulative[a1] =

sum(Market[1:a1]) }

#cumulative return of p1
portfolio1_cumulative <-
vector(length = 440)

for (a2 in 1:440) {
portfolio1_cumulative[a2] =

sum(portfolio1_Return[1:a2]) }

#cumulative return of p2
portfolio2_cumulative <-
vector(length = 440)

for (a3 in 1:440) {
portfolio2_cumulative[a3] =

sum(portfolio2_Return[1:a3]) }

#cumulative return of p3
portfolio3_cumulative <-
vector(length = 440)

for (a4 in 1:440) {
portfolio3_cumulative[a4] =

sum(portfolio3_Return[1:a4]) }

```

```{r echo=FALSE}

date_number = seq(1,
length(Market_cumulative), 1)
date_number =
as.vector(date_number)

cumulative_return <-
cbind(date_number,Market_cu
mulative,portfolio1_cumulative,
portfolio2_cumulative,portfolio3
_cumulative)

cumulative_return =
as.data.frame(cumulative_return)

ggplot(cumulative_return,
aes(x=date_number))
+geom_line(aes(y=Market_cum
ulative, color="market")) +

geom_line(aes(y=portfolio1_cu

mulative, color="p1")) +

geom_line(aes(y=portfolio2_cu
mulative, color="p2")) +

geom_line(aes(y=portfolio3_cu
mulative, color="p3"))

```

```{r echo=FALSE}

beta1 =
(cov(portfolio1_Return,Market))/
(sd(Market)^2)

beta2 =
(cov(portfolio2_Return,Market))/
(sd(Market)^2)

beta3 =
(cov(portfolio3_Return,Market))/
(sd(Market)^2)

```

```{r echo=FALSE}

alpha1 =
sum(portfolio1_Return)/1.75-
beta1*sum(Market)/1.75

# "1.75" repersents 1.75 years
```

alpha2 = sum(portfolio2_Return)/1.75-beta2*sum(Market)/1.75

alpha3 = sum(portfolio2_Return)/1.75-beta3*sum(Market)/1.75

```

```{r echo=FALSE}

SR1 = (sum(portfolio1_Return)/1.75)/(sd(portfolio1_Return))

SR2 = (sum(portfolio2_Return)/1.75)/(sd(portfolio2_Return))

SR3 = (sum(portfolio3_Return)/1.75)/(sd(portfolio3_Return))

portfolio <- c("portfolio1", "portfolio2", "portfolio3")

beta <- c(beta1, beta2, beta3) alpha <- c(alpha1, alpha2, alpha3)

SR <- c(SR1, SR2, SR3) ```

## Q3.2

```{r echo=FALSE} (summary_table <- as.data.frame(cbind(portfolio, beta, alpha, SR)))

#performance

cat("the 2-year cumulative return of portfolio 1 is", sum(portfolio1_Return), "\n" ) cat("the 2-year cumulative return of portfolio 2 is", sum(portfolio2_Return), "\n" ) cat("the 2-year cumulative return of portfolio 2 is", sum(portfolio3_Return), "\n" )

```

Q3.3

Portofolio 2 should be pick, since it has the largest cumulative return while the sharpe rate is big enough

Since each of three portfolio fails to "beat the market", the EMH has been buttressed by this example

Another possible explaination is that since the Dow&Jones is constituted by large company, accroding to the 'the small company effect'

In the behaviour finance, the return of Dow&Jones fails to beat the overall stock market which includes some small company

3 Random Numbers and Monte Carlo Simulation

Q1 Game2 Pharse2

```{r echo=FALSE}

N=10^5

seq1 <- numeric() for(nin1:N){

I use 10^5 as the simulate times in this question.

## Q2 Breaking even ```{r echo=FALSE} seq3 <- numeric() for (i in 1:10^5){

seq2 <- numeric() while(sum(seq2)<2){

seq2 <- c(seq2,sample(0:1,1)) }

time <- length(seq2)

seq3 <- c(seq3,time) }

100000/sum(seq3)

```

100000 is used as the simulate times in this question.

##Q3Pi

```{r echo=FALSE} f <- function(n){

seq4 <- numeric()

a <- runif(n,-1,1)

b <- runif(n,-1,1)

distance <- sqrt(a^2+b^2) I <- distance<1

p <- 4*mean(I)

library(ggplot2) f=seq(0,2*pi,0.001) x=sin(f)

y=cos(f) plot(x,y,type='l',xlim=c(-

1,1),ylim=c(-1,1),asp=1,col="black",lwd =2)

par(new=TRUE)

plot(a,b,xlim=c(-1,1),ylim=c(-1,1),asp=1,col="orange",lwd=1)

return (p) }

f(1000)

```

In the chart, I plot a circle,whose radius is 1, to stand for pi value.

##Q4

(a)

$$

E[Y_k] =E[X^k]= \int_0^1 x^k dx = \frac{x^{k+1}}{k+1} |_0^1 =

c1 <- sample(1:6,6,TRUE) c2 <- (max(c1)-min(c1)) < 3 seq1 <- c(seq1,c2)

}

round(mean(seq1),4) ```

\frac{1}{k+1}\,\,\,\,\, if \,\,\,\,\,k>- 1

$$

According to expression of the expected value of Y_k:

$$

V[Y_k] = E[Y_k^2]-E^2[Y_k] = E[X^{2k}] - E^2[X^k] = \frac{1}{2k+1} - \frac{1}{(k+1)^2}\,\,\,\,\,if\,\,\,\,k >-1

$$

(b)

Because $E[Y_k]$ and $V[Y_k]$ are finite, $k \neq - \frac{1}{2}$ and $k > -1$, and because $V[Y_k]>0$, $k \neq 0$.

(c)

```{r echo=FALSE} Q <- function (k){

X <- runif(10^5,0,1)

Y <- X^k

result <- c(mean(Y),var(Y)) return (result)

}

C <- function(k){

M <- (1-0^(k+1))/(k+1)

V <- (1-0^(2*k+1))/(2*k+1)-

((1-0^(k+1))/(k+1))^2 return (c(M,V))

}

table <- data.frame(t(rbind(sapply(c(- 10:10),Q),sapply(c(- 10:10),C))),row.names = c(- 10:10))

colnames(table) <- c("SE","SV","CE","CV") table$K_value = c(-10:10)

table

ggplot(table, aes(x=K_value)) + geom_line(aes(y=SE,

color="p1")) + geom_line(aes(y=SV,

color="p2")) + geom_line(aes(y=CE,

color="p3"))+ geom_line(aes(y=CV,

color="p4")) geom_line(aes(y=CV,

```

SE","SV","CE","CV" stand for simulate mean, simulate variance, calculated mean and calculated variance respectively.

```{r echo=FALSE}

#remove 10 rows contain missing data:

table1 <- data.frame(t(rbind(sapply(c(0:10 ),Q),sapply(c(0:10),C))),row.nam es = c(0:10))

colnames(table1) <- c("SE","SV","CE","CV") table1$K_value = c(0:10)

table1

ggplot(table1, aes(x=K_value)) + geom_line(aes(y=SE,

color="p1")) + geom_line(aes(y=SV,

color="p2")) + geom_line(aes(y=CE,

color="p3"))+ geom_line(aes(y=CV,

color="p4"))

```

When k <=-1, the calculated mean value and variance are infinate, I draw a new picture, which doesn't contain missing data.

###(d)

```{r echo=FALSE}

table2 <- data.frame(t(rbind(sapply(c(1:10 ),Q),sapply(c(1:10),C))),row.nam es = c(1:10))

colnames(table2) <- c("SE","SV","CE","CV") table2$K_value = c(1:10)

table2

ggplot(table2, aes(x=K_value)) + geom_line(aes(y=SE,

color="p1")) + geom_line(aes(y=SV,

color="p2")) + geom_line(aes(y=CE,

color="p3"))+

color="p4"))

```

Under the condicition k > -1 & K not eaual to 0 from (b), from the above chat we can see that p1(simulate mean value) and p3(calculated mean value) overlap, p3(simulate variance value) and p4(calculated variance value) overlap, which means simulated values are nearly equal to calculated values.

Bonus

First Let's look at simple case, if the required head number is 1 instead of 2:

Assume that we need to use expected n times to finish the simple game: then according to the first roll result, it satisfies the equation:

$$

nk = \frac{1}{2}*k + \frac{1}{2}(n+1)k

$$

then, we have $n = 2$, which is the expected game times in simple case.

Let's see the complex case when required head number is 2:

Again, assume that we need to use expected m times to finsh the complex game: then according to the first roll result, if we success, we will enter the simple game, if we fails, we will return to the begin of complex game, which satisfies the equation:

$$

mk = \frac{1}{2}*(1+2)k + \frac{1}{2}*(1+m)k

$$

then, the expected value of m is 4: then because $mk=1$ $\to$ $k=\frac{1}{4}$.

Notice here, $k \neq E[1/X]$, $k = 1/E[X]$, for example, if we play 100000 times game, the expected roll time is 400000, to make this total game fair, the expected value of k should be $\frac{1}{4}$. Making every game fair and calculating average is a wrong method.

```{r include=FALSE}
library(lubridate)
library(PerformanceAnalytics)

stocks = c( "AAPL", "CSCO", "HON", "KO", "NKE", "WBA", "AMGN", "CVX", "IBM", "MCD", "PG", "WMT", "AXP", "DIS", "INTC", "MMM", "TRV", "BA", "GS", "JNJ", "MRK", "UNH", "CAT", "HD", "JPM", "MSFT", "VZ"

)

stklist = lapply(stocks, function(x) {

try(get(getSymbols(x, from = "2017-01-01", to = "2020-09- 30")), silent = TRUE)

})

min.dates = sapply(stklist,

function(x) { as.character(min(date(x)))

})

keep.tics = date(min.dates) == names(table(min.dates))

stklist = stklist[keep.tics]

#compute return

P_adj = lapply(stklist, function(x){x[, 6]})

P = Reduce(merge, P_adj)

R = na.omit(log(P / lag(P)))

M_r = apply(R, 2, mean) * 252

#volatility

S = apply(R, 2, sd) * sqrt(252)

df = data.frame(M_r, S)
#df$top=(df$M>quantile(df$M_r,0.75))*1

#downside risk

df$VaR = -apply(R, 2, function(x) {

quantile(x, 0.05) })

#sharp-ratio

df$SR = with(df, M_r / S) ```

# 4 Value at Risk and Stress Testing

## Task 1

### 1. Calibrate the price path for each portfolio

Before implementing, we take a brief review on the three portfolio.

The weight allocated to asset i in Portfolio 1 is given by

$$ \omega_i^\sigma=\frac{\frac{1}{\sigma_j^2}}{\sum_{j=1}^d\frac{1}{\sigma_j^2}}

$$

where $\sigma_i$ is the volatility of asset i $\forall$i = 1,...,27 and $\sum_{i=1}^{27}\omega_i^\sigma=1$. On the other hand, the weight allocated to asset i in

Portfolio 2 is $$\omega_i^{SR}=\frac{SR_i}{\sum_{j=1}^dSR_j}$$

where $SR_i$ denotes the SR of stock i $\forall$i=1,...,27 and $\sum_{i=1}^{27}\omega_i^{SR}=1$.

Portfolio 3 allocates equal weights to each asset such that $$\omega_i^N=\frac{1}{27}$$
```{r echo=FALSE}

#portfolio return
W1=(1/(df$S^2))/sum(1/ (df$S ^ 2))

W2 = df$SR / sum(df$SR)
W3=W2/W2*1/27

names(W1) <- names(W2) <- names(W3) <- names(R)

R_p1 <- as.matrix(R) %*% W1
R_p2 <- as.matrix(R) %*% W2
R_p3 <- as.matrix(R) %*% W3

#reformat back to xts and merge together

R_p1 <- as.xts(data.frame(Portfolio_1 = R_p1), dateFormat = "Date") R_p2 <- as.xts(data.frame(Portfolio_2 = R_p2), dateFormat = "Date") R_p3 <- as.xts(data.frame(Portfolio_3 = R_p3), dateFormat = "Date")

R_p <- merge(R_p1, R_p2, R_p3)
```

We refer to PerformanceAnalytics package by (Peterson and Carl 2018) to visualize the performance of the three portfolio.

```{r echo=FALSE}
chart.CumReturns(R_p, main = "Cumulative Return", legend.loc = "topleft")

```

```{r echo=FALSE}
chart.Drawdown(R_p, main = "Drawdown")

```

Descriptive summary

```{r echo=FALSE} my.sum = function(x){

Mean = mean(x)

Std = sd(x)

SR = sqrt(252) * Mean / Std M <- rbind(Mean * 252, Std

* sqrt(252), SR) return(M)

}

# run over each column

sum.i <- apply(R_p, 2, my.sum)
rownames(sum.i) <- c("mean", "std", "SR")

sum.i

```

To get started with the backtesting, we split the sample into two periods in-sample (IN) and out-of-sample (OUT).

```{r echo=FALSE} # backtesting

#separate

in_index <- 1:floor(0.5 * nrow(R))
R_in <- R[in_index, ]

R_out <- R[(max(in_index) + 1):nrow(R), ]

tail(R_in,2)

head(R_out,2)

```

Then estimate μ_i and σ_i using the IN to construct Portfolio 1, 2 and 3. After that, estimate the corresponding the parameters from the OUT to demonstrate the sensitivity of each over time. ```{r echo=FALSE}

#compute M and S using the in-sample

M_in <- apply(R_in, 2, mean) * 250

S_in <- apply(R_in, 2, sd) * sqrt(250)

#compute M and S using the out-sample

M_out <- apply(R_out, 2, mean) * 250

S_out <- apply(R_out, 2, sd) * sqrt(250)

plot(M_out ~ M_in,

ylab = "Out",

xlab = "In",

main = "Mean Returns")

abline(a=0,b=1,lty= "dashed")

plot(S_out ~ S_in, ylab = "Out",

xlab = "In",

main = "Volatility")
abline(a=0,b=1,lty= "dashed")

```

It appears that $\sigma$s exhibit a lower sensitivity than $\mu$s, making the mean returns are more susceptible to model risk. ```{r echo=FALSE} print(mean((M_in - M_out) ^ 2)) print(mean((S_in - S_out) ^ 2)) ```

Take a look on how the portfolio strategy performs on OUT

```{r echo=FALSE}

port_ret_f <- function(W1, W2, W3){

R_p1 <- as.matrix(R_out) %*% W1

R_p2 <- as.matrix(R_out) %*% W2

R_p3 <- as.matrix(R_out) %*% W3

(1/(S_in^2))/sum(1/(S_in^2)) W2 <- SR_in/sum(SR_in)
W3<-W2/W2*1/27 names(W1) <- names(W2) <- names(W3) <- names(R)

R_p_in <- port_ret_f(W1, W2, W3)

chart.CumReturns(R_p_in, main = "Realistic Case", legend.loc = "topleft")

```

For hypothesis(OUT)

```{r echo=FALSE}

#hypothesis

SR_out <- M_out / S_out W1<-(1/(S_out^2))/sum(1 / (S_out ^ 2))

W2 <- SR_out / sum(SR_out) W3<-W2/W2*1/27 names(W1) <- names(W2) <- names(W3) <- names(R) R_p_out <- port_ret_f(W1, W2, W3)

chart.CumReturns(R_p_out, main = "Hypothetical Case", legend.loc = "topleft")

```

Mean and Standard deviation for p=1,2,3 using the daily returns in the OUT period:

```{r echo=FALSE}

rbind(miu = apply(R_p_out, 2, mean),sigma = apply(R_p_out, 2, sd))

```


2. Insights

```{r echo=FALSE}

#GBM

m = as.numeric(sum.i[1,])

s = as.numeric(sum.i[2,]) N=1000

S=100

T_end = 252 / 252

gbm_path <- function(N, mu, sig, T_end, S) {

R_t <- rnorm(N, T_end * (mu - 0.5 * sig ^ 2), sig * sqrt(T_end))

R_p1 as.xts(data.frame(Portfolio_1 R_p1), dateFormat = "Date")

<- =

R_p2 as.xts(data.frame(Portfolio_2 = R_p2), dateFormat = "Date")

R_p3 <-

as.xts(data.frame(Portfolio_3 R_p3), dateFormat = "Date")

=

R_p <- merge(R_p1, R_p2, R_p3,all = F)

return(R_p) }

```

For reality(IN)

```{r echo=FALSE}

#reality

SR_in <- M_in/S_in

W1 <-

<-

S_t <- S * exp(R_t)

return(S_t) }

sim1 <- gbm_path(N, m[1], s[1], 1, S)

sim2 <- gbm_path(N, m[2], s[2], 1, S)

sim3 <- gbm_path(N, m[3], s[3], 1, S)

```


Simulated path of Geometric Brownian Motion

```{r echo=FALSE}

plot(sim1, type = "l")

lines(sim2, col = 2, lty = 2) lines(sim3, col = 3, lty = 3) ```

&lt;br&gt;

Distrbution plot&lt;br&gt; ```{r echo=FALSE} boxplot(

sim1,

sim2,

sim3,

main = "distribution plot", at = c(1, 2, 3),

names = c("portfolio 1", "portfolio 2", "portfolio 3"),

col = c(2, 4, 6) )

```


Portfolio 1 is a Global

Minimum Variance portfolio which focus on controling the risk to the lowest, therefore which do have the lowest volatility but lack considerable rewards similar to portfolio 3 that simply equaly distribute the funds. While Sharp-ratio portfolio (p2) has a consideration on the risk- adjusted return of stocks allowing investers to have a bias on reward(highest mean) within undertaking a higher risk(highest variance).

3. Expected value of each portfolio one year from now The

expected value one year from now on is

```{r echo=FALSE}

#expected value

exp = cbind(p1 = sim1[length(sim1)],

p2 = sim2[length(sim2)],

p3 = sim3[length(sim3)]) exp = data.frame(exp, row.names = "expected value") exp

```


4. Value at risk

With 95% level of confidence, the Value-at-Risk is

```{r echo=FALSE}

#VaR

VaR1 <- mean(sim1) - quantile(sim1, 0.05)

VaR2 <- mean(sim2) - quantile(sim2, 0.05)

VaR3 <- mean(sim3) - quantile(sim3, 0.05) names(VaR1) = names(VaR2) = names(VaR3) = "VaR"

VaR = cbind(p1 = VaR1, p2 = VaR2, p3 = VaR3)

VaR

#rbind(exp, VaR)

```


Task 2

Refering to the SPY ETF as the markets, directly, we use the table.CAPM from the PerformanceAnalytics package to attain a number of statistics. ```{r echo=FALSE}

SPY <-

get(getSymbols("SPY", from = "2017-01-01", to = "2020-09-30"))[, 6]

R_m <- na.omit(log(SPY / lag(SPY)))

names(R_m) <- "SPY"

table.CAPM(R_p_out, R_m) table.Stats(R_p_out)

```

&lt;br&gt;

VaR(0.05) for each portfolio is ```{r echo=FALSE}

sig_m = apply(R_p, 2, sd)

sig = apply(R_p_out, 2, sd) + data.frame(table.CAPM(R_p_out, R_m))["Beta", ]*sig_m * 0.1 sim11 = gbm_path(N, m[1], as.numeric(sig[1]), 1, S)

sim22 = gbm_path(N, m[2], as.numeric(sig[2]), 1, S)

sim33 = gbm_path(N, m[3], as.numeric(sig[3]), 1, S)

VaR11 <- mean(sim11) - quantile(sim11, 0.05)

VaR22 <- mean(sim22) - quantile(sim22, 0.05)

VaR33 <- mean(sim33) - quantile(sim33, 0.05) names(VaR11)=names(VaR22)= names(VaR33)="VaR"

VaRr = cbind(p1 = VaR11, p2 = VaR22, p3 = VaR33)

VaRr

```

5 Mean-Variance Efficient Frontier

Get data

```{r include=FALSE} library(PerformanceAnalytics) library(quantmod)

stocklist <- c(

"AAPL","CSCO","HON","KO","NK E","WBA",

"AMGN","CVX","IBM","MCD","PG","WMT",

"AXP","DIS","INTC","MMM","TRV ",

"BA","GS","JNJ","MRK","UNH",

"CAT","HD","JPM","MSFT","VZ" )

i=1 repeat{

data = getSymbols(stocklist[i], from = "1999-5-1",

to = "2020-9-30") i=i+1

if (i == 28){ break

} }

m = merge(dailyReturn(AAPL), dailyReturn(AMGN),

dailyReturn(AXP), dailyReturn(BA), dailyReturn(CAT), dailyReturn(CSCO), dailyReturn(CVX), dailyReturn(DIS), dailyReturn(GS), dailyReturn(HD), dailyReturn(HON),
```

dailyReturn(IBM), dailyReturn(INTC), dailyReturn(JNJ), dailyReturn(JPM), dailyReturn(KO), dailyReturn(MCD), dailyReturn(MMM), dailyReturn(MRK), dailyReturn(MSFT), dailyReturn(NKE), dailyReturn(PG), dailyReturn(TRV), dailyReturn(UNH), dailyReturn(VZ), dailyReturn(WBA), dailyReturn(WMT))

m <- setNames(m,

c("AAPL","AMGN","AXP","BA","C AT","CSCO",

"CVX","DIS","GS","HD","HON","I BM","INTC",

"JNJ","JPM","KO","MCD","MMM" ,"MRK","MSFT",

"NKE","PG","TRV","UNH","VZ"," WBA","WMT")

)

m <- na.omit(m)

```

## 1. Plot the MVEF

} ```{r echo=FALSE,

message=FALSE, warning=FALSE}

I <- matrix(data = 1,nrow = 27, ncol = 1)

I_trans <- matrix(data = 1 ,nrow = 1, ncol = 27)

I1 <- matrix(data = 1,nrow = 27, ncol = 27)

ret <- Return.annualized(m) vol <- StdDev.annualized(m) cov <- cov(m)*252 #sigma cov_reverse <- solve(cov)

optimal_solution <- function(Sigma, r, rho){

I <- matrix(data = 1,nrow = 27, ncol = 1)

a = r %*% solve(Sigma) %*% t(r) b = t(I) %*% solve(Sigma) %*% t(r) c = r %*% solve(Sigma) %*% I

d = t(I) %*% solve(Sigma) %*% I

ma = matrix(data=c(c,a,d,b), nrow=2,byrow=T)

lamb = solve(ma) %*% c(rho,1)

lambda1 = lamb[1] lambda2 = lamb[2]

weights = lambda1*solve(Sigma) %*% I + lambda2*solve(Sigma) %*%t(r)

min_sigma2 = t(weights) %*% Sigma %*% weights

return(list(weights, min_sigma2))

}

## plot MVEF

m_range = seq(0,0.7,0.001) edge = rep(0,length(m_range)) i=1

for (mu in m_range){

edge[i] = sqrt(optimal_solution(cov,ret,mu )[[2]])

i = i + 1

plot(vol,ret, xlim=c(0,0.7), ylim =c(0,0.8))

par(new=T)

plot(edge, m_range, type='l',xlim=c(0,0.7), ylim =c(0,0.8), xlab='vol',ylab='ret') ```

## 2. Highlight SR & GMV points ```{r echo=FALSE}

plot(vol,ret, xlim=c(0,0.7), ylim =c(0,0.8))

par(new=T)

plot(edge, m_range, type='l',xlim=c(0,0.7), ylim =c(0,0.8), xlab='vol',ylab='ret')

t = which.max(m_range/edge) par(new=T) plot(edge[t],m_range[t],xlim=c(0 ,0.7), ylim =c(0,0.8), col='red', xlab='vol',ylab='ret')

y = which.min(edge) par(new=T) plot(edge[y],m_range[y],xlim=c( 0,0.7), ylim =c(0,0.8), col='blue', xlab='vol',ylab='ret', main ='efficient frontier')

```

3.

(a) Highlight the frontier ```{r echo=FALSE}

plot(vol,ret, xlim=c(0,0.7), ylim =c(0,0.8))

par(new=T)

plot(edge, m_range, type='l',xlim=c(0,0.7), ylim =c(0,0.8), xlab='vol',ylab='ret')

t = which.max(m_range/edge) par(new=T) plot(edge[t],m_range[t],xlim=c(0 ,0.7), ylim =c(0,0.8), col='red', xlab='vol',ylab='ret')

y = which.min(edge) par(new=T)

plot(edge[y],m_range[y],xlim=c(0,0.7), ylim =c(0,0.8), col='blue',

```
xlab='vol',ylab='ret', main
='efficient frontier')

w0 =
optimal_solution(cov,ret,m_rang
e[y])[[1]]

w_SR =
optimal_solution(cov,ret,m_rang
e[t])[[1]]

vol1 = vector()

ret1= vector()

for (theta in seq(0,1,0.01)){

w = w0 * theta + w_SR*(1-theta)

vol1 = c(vol1,sqrt(t(w) %*% cov
%*% w))

ret1 = c(ret1,sum(w*t(ret))) }

par(new=T)

plot(vol1,ret1,
type='l',xlim=c(0,0.7), ylim
=c(0,0.8),
xlab='vol',ylab='ret',col='red')
```

(b) Economic rationale

If theta>1, it means that we will short the SR portfolio and long the low risk portfolio. In other words, I will sell the SR portfolio despite not having the SR portfolio to get the money, and use the money and the principle to buy the low risk portfolio.