Project 2

Minxuan Wang, Ziwen Gu, Yining Qian

November 2017

1 Question 1

(a)

Firstly, I downloaded the data of stock prices from "YahooFinance" and used the "monthlyreturn" function to calculate the monthly return of stocks.
After numerious trials in R, we believe the ten companies below is the best portfolio for us.

```
> library("quantmod")
> startdate<-as.Date("2015-11-07")
> enddate<-as.Date("2017-11-07")
> getSymbols("GOOG",src="yahoo",from=startdate,to=enddate)
[1] "GOOG"
> google<-monthlyReturn(GOOG,subset=NULL,type='arithmetic',
leading=TRUE)
> google<-as.numeric(google)
> getSymbols("NFLX",src="yahoo",from=startdate,to=enddate)
[1] "NFLX"
> Netflix<-monthlyReturn(NFLX,subset=NULL,type='arithmetic',
leading=TRUE)
> Netflix<-as.numeric(Netflix)
> getSymbols("UNH",src="yahoo",from=startdate,to=enddate)
[1] "UNH"
> unitedhealth<-monthlyReturn(UNH,subset=NULL,type='arithmetic',
leading=TRUE)
> unitedhealth<-as.numeric(unitedhealth)
> getSymbols("NVDA",src="yahoo",from=startdate,to=enddate)
[1] "NVDA"
> Nvidia<-monthlyReturn(NVDA,subset=NULL,type='arithmetic',
leading=TRUE)
> Nvidia<-as.numeric(Nvidia)
> getSymbols("GE",src="yahoo",from=startdate,to=enddate)
[1] "GE"
> GeneralEle<-monthlyReturn(GE,subset=NULL,type='arithmetic',
```

```
leading=TRUE)
> GeneralEle<-as.numeric(GeneralEle)
> getSymbols("SBUX",src="yahoo",from=startdate,to=enddate)
[1] "SBUX"
> starbucks<-monthlyReturn(SBUX,subset=NULL,type='arithmetic',
leading=TRUE)
> starbucks<-as.numeric(starbucks)
> getSymbols("ELLI",src="yahoo",from=startdate,to=enddate)
[1] "ELLI"
> elli<-monthlyReturn(ELLI,subset=NULL,type='arithmetic',
leading=TRUE)
> elli<-as.numeric(elli)
> getSymbols("AWK",src="yahoo",from=startdate,to=enddate)
[1] "AWK"
> americawaters<-monthlyReturn(AWK,subset=NULL,type='arithmetic',
leading=TRUE)
> americawaters<-as.numeric(americawaters)
> getSymbols("EW",src="yahoo",from=startdate,to=enddate)
[1] "EW"
> edwlifescience<-monthlyReturn(EW,subset=NULL,type='arithmetic',
leading=TRUE)
> edwlifescience<-as.numeric(edwlifescience)
> getSymbols("COR",src="yahoo",from=startdate,to=enddate)
[1] "COR"
> coresite<-monthlyReturn(COR,subset=NULL,type='arithmetic',
leading=TRUE)
> coresite<-as.numeric(coresite)
> combinearraies<-data.frame(google,Netflix,unitedhealth,Nvidia,
GeneralEle,starbucks,elli,americawaters,edwlifescience,coresite)

> cov(combinearraies)
                      google       Netflix  unitedhealth        Nvidia
google          2.221796e-03  0.0023258261 -5.535521e-05  0.0020255293
Netflix         2.325826e-03  0.0108420237 -6.369698e-04  0.0039961768
unitedhealth   -5.535521e-05 -0.0006369698  1.505439e-03  0.0010157444
Nvidia          2.025529e-03  0.0039961768  1.015744e-03  0.0142826581
GeneralEle     -4.264063e-04 -0.0008491352  4.726770e-04  0.0003040551
starbucks       5.698522e-04 -0.0007817911  5.286558e-04  0.0012897869
elli            1.405401e-04 -0.0015542018 -8.470545e-04 -0.0025408743
americawaters  -2.575395e-04 -0.0014922522  6.654290e-04 -0.0018716772
edwlifescience  1.175887e-04 -0.0032298385 -3.685838e-04 -0.0014610311
coresite       -3.657737e-04 -0.0012391034  2.023150e-04 -0.0020920896
                  GeneralEle     starbucks          elli americawaters
google         -4.264063e-04  0.0005698522  0.0001405401 -2.575395e-04
Netflix        -8.491352e-04 -0.0007817911 -0.0015542018 -1.492252e-03
unitedhealth    4.726770e-04  0.0005286558 -0.0008470545  6.654290e-04
```

```
Nvidia          3.040551e-04  0.0012897869 -0.0025408743 -1.871677e-03
GeneralEle      2.521199e-03  0.0003505678 -0.0005110718 -2.215327e-04
starbucks       3.505678e-04  0.0018042945  0.0003483361  2.909454e-04
elli           -5.110718e-04  0.0003483361  0.0098177694  9.831749e-04
americawaters  -2.215327e-04  0.0002909454  0.0009831749  2.226591e-03
edwlifescience -1.087949e-05 -0.0008415861  0.0015349455  3.898216e-05
coresite        1.561984e-04  0.0005122152  0.0011691938  1.811415e-03
               edwlifescience      coresite
google           1.175887e-04 -0.0003657737
Netflix         -3.229838e-03 -0.0012391034
unitedhealth    -3.685838e-04  0.0002023150
Nvidia          -1.461031e-03 -0.0020920896
GeneralEle      -1.087949e-05  0.0001561984
starbucks       -8.415861e-04  0.0005122152
elli             1.534945e-03  0.0011691938
americawaters    3.898216e-05  0.0018114152
edwlifescience   8.239354e-03  0.0014870285
coresite         1.487028e-03  0.0043613537

> cov2cor(cov(combinearraies))
                    google    Netflix unitedhealth      Nvidia   GeneralEle
google          1.00000000  0.4738817  -0.03026737  0.35956873 -0.180164004
Netflix         0.47388172  1.0000000  -0.15766382  0.32113305 -0.162412008
unitedhealth   -0.03026737 -0.1576638   1.00000000  0.21905254  0.242621533
Nvidia          0.35956873  0.3211330   0.21905254  1.00000000  0.050669217
GeneralEle     -0.18016400 -0.1624120   0.24262153  0.05066922  1.000000000
starbucks       0.28461411 -0.1767591   0.32076553  0.25407371  0.164367058
elli            0.03009136 -0.1506419  -0.22032983 -0.21457160 -0.102724016
americawaters  -0.11579011 -0.3037154   0.36345437 -0.33189922 -0.093500546
edwlifescience  0.02748318 -0.3417268  -0.10465454 -0.13468163 -0.002387033
coresite       -0.11750322 -0.1801946   0.07895613 -0.26507276  0.047104489
                 starbucks        elli americawaters edwlifescience
google          0.28461411  0.03009136   -0.115790111    0.027483176
Netflix        -0.17675913 -0.15064191   -0.303715437   -0.341726805
unitedhealth    0.32076553 -0.22032983    0.363454371   -0.104654542
Nvidia          0.25407371 -0.21457160   -0.331899223   -0.134681626
GeneralEle      0.16436706 -0.10272402   -0.093500546   -0.002387033
starbucks       1.00000000  0.08276341    0.145156856   -0.218272116
elli            0.08276341  1.00000000    0.210283002    0.170663227
americawaters   0.14515686  0.21028300    1.000000000    0.009101209
edwlifescience -0.21827212  0.17066323    0.009101209    1.000000000
coresite        0.18259466  0.17867732    0.581282365    0.248063031
                  coresite
google         -0.11750322
Netflix        -0.18019458
unitedhealth    0.07895613
```

```
Nvidia         -0.26507276
GeneralEle      0.04710449
starbucks       0.18259466
elli            0.17867732
americawaters   0.58128236
edwlifescience  0.24806303
coresite        1.00000000
```

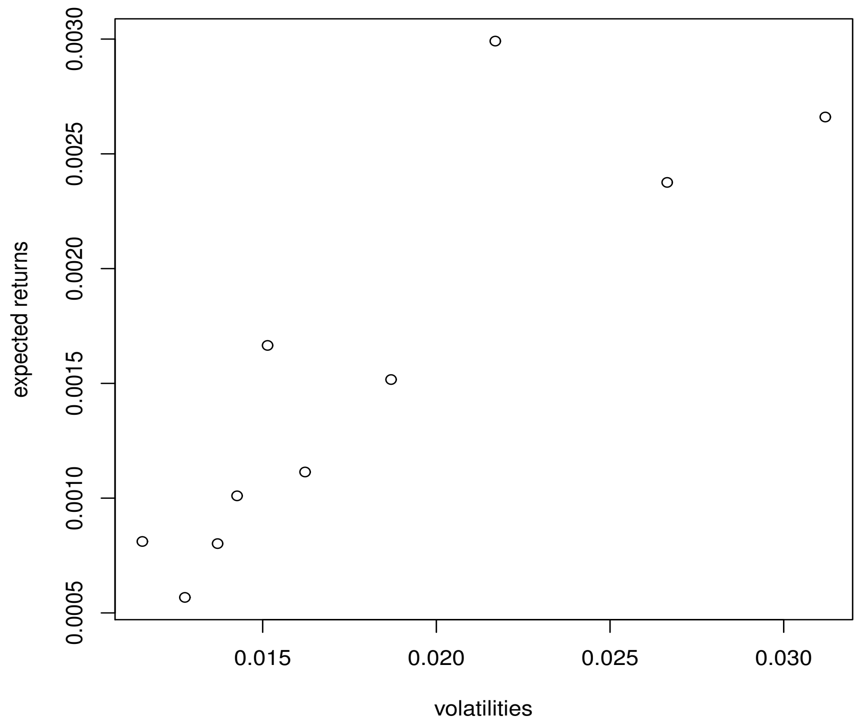
As we can see from the above, it is the Variance-Covariance Matrix for 10 companies.Also, we use the function of "Cov2Cor" to convert a Covariance Matrix into a Corrleation Matrix.

By definition,correlation means the degree to which two things behave in the same way. So to make a great portfolio, we prefer the stocks with small correlations between each other so that the risk could be reduced between different stocks across differnt industries.According to the matrix of variance and corrleation above, we can see that among ten companies, the correlation efficient is really small and it also contains some negative numbers inside, which means two stocks behave in two different directions.

(b)

```
> getSymbols(c("GOOG", "AWK", "NFLX", "ELLI", "NVDA",
"UNH","SBUX","GE","COR","EW"),from="2015-1-5", to="2016-10-22")
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
 [1] "GOOG" "AWK"  "NFLX" "ELLI" "NVDA" "UNH"
 "SBUX" "GE"   "COR"  "EW"
> prices.data<- merge.zoo(GOOG[,6],AWK[,6],NFLX[,6],ELLI[,6],
NVDA[,6],UNH[,6],SBUX[,6],GE[,6],COR[,6],EW[,6])
> returns.data <- CalculateReturns(prices.data)
> returns.data <- na.omit(returns.data)
> head(returns.data)
           GOOG.Adjusted AWK.Adjusted NFLX.Adjusted ELLI.Adjusted
2015-01-06  -0.023177070  0.002636193  -0.017120620  -0.007190701
2015-01-07  -0.001713269  0.012769925   0.005191927   0.019480494
2015-01-08   0.003153060  0.003893867   0.022188202   0.043361098
2015-01-09  -0.012950575  0.002955308  -0.015457744   0.015496595
2015-01-12  -0.007295889 -0.003130699  -0.031765320  -0.028208116
2015-01-13   0.007369820  0.002216925   0.015556893   0.006186105
           NVDA.Adjusted UNH.Adjusted SBUX.Adjusted   GE.Adjusted
2015-01-06  -0.030318396 -0.002017819  -0.008137146 -0.0215447391
2015-01-07  -0.002605654  0.010210330   0.024611813  0.0004155181
```

```
2015-01-08   0.037617650  0.047733293   0.016136870  0.0120432905
2015-01-09   0.004028150 -0.009360043  -0.032731032 -0.0139515406
2015-01-12  -0.012537516 -0.011280390   0.005514501 -0.0020808203
2015-01-13  -0.001523755  0.005070613   0.007976894 -0.0050041536
           COR.Adjusted   EW.Adjusted
2015-01-06  0.004961447 -5.947738e-03
2015-01-07  0.024191667  2.346090e-02
2015-01-08  0.018799965  2.423072e-02
2015-01-09 -0.017979876  7.517837e-05
2015-01-12  0.006504428  1.599572e-02
2015-01-13  0.011010108 -1.234384e-02
> colnames(returns.data) <- c("GOOG", "AWK", "NFLX", "ELLI",
"NVDA", "UNH","SBUX","GE","COR","EW")
> head(returns.data)
                   GOOG          AWK         NFLX         ELLI         NVDA
2015-01-06 -0.023177070  0.002636193 -0.017120620 -0.007190701 -0.030318396
2015-01-07 -0.001713269  0.012769925  0.005191927  0.019480494 -0.002605654
2015-01-08  0.003153060  0.003893867  0.022188202  0.043361098  0.037617650
2015-01-09 -0.012950575  0.002955308 -0.015457744  0.015496595  0.004028150
2015-01-12 -0.007295889 -0.003130699 -0.031765320 -0.028208116 -0.012537516
2015-01-13  0.007369820  0.002216925  0.015556893  0.006186105 -0.001523755
                    UNH         SBUX            GE          COR
2015-01-06 -0.002017819 -0.008137146 -0.0215447391  0.004961447
2015-01-07  0.010210330  0.024611813  0.0004155181  0.024191667
2015-01-08  0.047733293  0.016136870  0.0120432905  0.018799965
2015-01-09 -0.009360043 -0.032731032 -0.0139515406 -0.017979876
2015-01-12 -0.011280390  0.005514501 -0.0020808203  0.006504428
2015-01-13  0.005070613  0.007976894 -0.0050041536  0.011010108
                      EW
2015-01-06 -5.947738e-03
2015-01-07  2.346090e-02
2015-01-08  2.423072e-02
2015-01-09  7.517837e-05
2015-01-12  1.599572e-02
2015-01-13 -1.234384e-02
> meanReturns <- colMeans(returns.data)
> sd=apply(returns.data,2,"sd")
> plot(sd,meanReturns,ylab="expected returns",xlab="volatilities")
```



(c)

```
> stockModel<-function(stockReturns,drop=NULL,Rf=0,shortSelling=c("y",
+ "n"),model=c("none","SIM","CCM","MGM"),industry=NULL,
+ index=NULL,get=c("overlapOnly","all"),freq=c("month",
+ "week","day"),start="1970-01-01",end=NULL,recentLast=FALSE,
+ rawStockPrices=FALSE)
+ {
+ if(!is.vector(stockReturns)&!is.factor(stockReturns)&
+ !is.matrix(stockReturns)&!(class(stockReturns)%in%
+ c("stockReturns","stockModel"))){
+ stop("The\"stockReturns\"variableisnotrecognized.")
+ }
+ tM<-list()
+ class(tM)<-"stockModel"
+ tM$model<-model[1]
+ if(is.numeric(tM$model)){
+ tM$model<-c("none","SIM","CCM","MGM","MIM")[tM$model]
```

```
+ }
+ tM$ticker<-NA
+ tM$index<-ifelse(is.null(index),NA,index)
+ tM$theIndex<-NA
+ tM$industry<-NA
+ if(!is.null(industry)[1]){
+ tM$industry<-as.character(industry)
+ }
+ tM$returns<-NA
+ tM$marketReturns<-NA
+ tM$n<-NA
+ tM$start<-NA
+ tM$end<-NA
+ tM$period<-NA
+ tM$R<-NA
+ tM$COV<-NA
+ tM$sigma<-NA
+ temp<-c("y","yes","Y","Yes","YES",TRUE)
+ tM$shorts<-ifelse(shortSelling[1]%in%temp,TRUE,FALSE)
+ tM$Rf<-Rf
+ tM$alpha<-NA
+ tM$vAlpha<-NA
+ tM$beta<-NA
+ tM$vBeta<-NA
+ tM$betaAdj<-FALSE
+ tM$MSE<-NA
+ tM$RM<-NA
+ tM$VM<-NA
+ tM$rho<-NA
+ if(model[1]=="SIM"){
+ if(is.null(index)[1]){
+ stop("Variable\"index\"isrequiredforthesingleindexmodel.")
+ }
+ index<-index[1]
+ }
+ elseif(tM$model=="MGM"){
+ if(is.null(tM$industry)[1]){
+ stop("Variable\"industry\"isrequiredforthemultigroupmodel.")
+ }
+ }
+ elseif(tM$model=="none"&!tM$shorts){
+ warning("Shortsalesarealwayspermittedwhennomodelisspecified.")
+ tM$shorts<-TRUE
+ }
+ if(is.vector(stockReturns)|is.factor(stockReturns)){
+ if(!is.character(stockReturns)&!is.factor(stockReturns)){
```

```
+ stop("Variable\"stockReturns\"notrecognized.")
+ }
+ stockReturns<-getReturns(stockReturns,freq,get,start,
+ end)
+ temp<-stockModel(stockReturns,drop=drop,Rf=Rf,
+ shortSelling=shortSelling,model=model,index=index,
+ industry=industry)
+ return(temp)
+ }
+ elseif(is.matrix(stockReturns)){
+ n<-dim(stockReturns)[1]
+ if(recentLast){
+ stockReturns<-stockReturns[n:1,]
+ }
+ if(rawStockPrices){
+ rn<-rownames(stockReturns)[1:(n-1)]
+ temp<-stockReturns[-n,]-stockReturns[-1,]
+ stockReturns<-temp/stockReturns[-1,]
+ rownames(stockReturns)<-rn
+ }
+ rn<-rownames(stockReturns)
+ cn<-colnames(stockReturns)
+ start<-ifelse(is.null(rn[1]),start,rn[1])
+ end<-ifelse(is.null(end),rev(rn)[1],end)
+ period<-freq[1]
+ if(is.null(cn)[1]){
+ Ticker<-NA
+ }
+ else{
+ Ticker<-cn
+ }
+ stockReturns<-list(R=stockReturns,ticker=Ticker,
+ period=period,start=start,end=end)
+ class(stockReturns)<-"stockReturns"
+ }
+ elseif(class(stockReturns)=="stockModel"){
+ stockR<-list()
+ stockR$R<-stockReturns$returns
+ stockR$ticker<-stockReturns$ticker
+ stockR$period<-stockReturns$period
+ stockR$start<-stockReturns$start
+ stockR$end<-stockReturns$end
+ stockReturns<-stockR
+ class(stockReturns)<-"stockReturns"
+ }
+ sR<-stockReturns
```

```
+ tM$ticker<-sR$ticker
+ tM$returns<-sR$R
+ tM$start<-sR$start
+ tM$end<-sR$end
+ tM$period<-sR$period
+ if(!is.null(drop)[1]){
+ if(length(tM$ticker)==length(industry)){
+ tM$industry<-tM$industry[-drop]
+ }
+ if(length(tM$ticker)==dim(tM$returns)[2]){
+ tM$returns<-tM$returns[,-drop]
+ }
+ tM$ticker<-tM$ticker[-drop]
+ }
+ tM$R<-apply(tM$returns,2,mean)
+ tM$COV<-cov(tM$returns)
+ tM$n<-dim(tM$returns)[1]
+ tM$sigma<-sqrt(diag(tM$COV))
+ if(!is.na(tM$index)){
+ tM$marketReturns<-as.matrix(tM$returns[,index],ncol=1)
+ colnames(tM$marketReturns)<-tM$ticker[index]
+ }
+ if(tM$model=="SIM"){
+ tM$theIndex<-tM$ticker[index]
+ tM$ticker<-tM$ticker[-index]
+ tM$industry<-tM$industry[-index]
+ tM$sigma<-tM$sigma[-index]
+ getRegCoef<-function(R,COV,index,n){
+ RM<-R[index]
+ VM<-diag(COV)[index]
+ beta<-COV[index,-index]/VM
+ alpha<-R[-index]-beta*RM
+ MSE<-(n-1)*(diag(COV)[-index]-beta^2*VM)/(n-
+ 2)
+ VBeta<-MSE/(n*VM)
+ VAlpha<-MSE*rep((RM^2+ VM)/(n*VM),length(VBeta))
+ R<-R[-index]
+ COV<-matrix(VM,length(R),length(R))
+ COV<-t(COV*beta)*beta
+ diag(COV)<-diag(COV)+ MSE
+ return(list(R=R,COV=COV,RM=RM,VM=VM,alpha=alpha,
+ vAlpha=VAlpha,beta=beta,vBeta=VBeta,
+ MSE=MSE))
+ }
+ grc<-getRegCoef(tM$R,tM$COV,tM$index,tM$n)
+ tM$returns<-tM$returns[,-index]
```

```
+ tM$R<-grc$R
+ tM$COV<-grc$COV
+ tM$RM<-grc$RM
+ tM$VM<-grc$VM
+ tM$alpha<-grc$alpha
+ tM$beta<-grc$beta
+ tM$vAlpha<-grc$vAlpha
+ tM$vBeta<-grc$vBeta
+ tM$MSE<-grc$MSE
+ }
+ if(tM$model=="CCM"){
+ tM$rho<-getCorr(tM$COV)
+ tM$COV[,]<-tM$rho
+ diag(tM$COV)<-1
+ tM$COV<-t(t(tM$COV*tM$sigma)*tM$sigma)
+ }
+ if(tM$model=="MGM"&&tM$shorts){
+ tM$rho<-getCorr(tM$COV,tM$industry)
+ theMatch<-match(tM$industry,unique(tM$industry))
+ tM$COV<-tM$rho[theMatch,theMatch]
+ diag(tM$COV)<-1
+ tM$COV<-t(t(tM$COV*tM$sigma)*tM$sigma)
+ colnames(tM$COV)<-tM$ticker
+ rownames(tM$COV)<-tM$ticker
+ }
+ elseif(tM$model=="MGM"){
+ }
+ return(tM)
+ }
>
>
> optimalPort<-function(model,Rf=NULL,shortSell=NULL,eps=10^(-4))
+ {
+ if(!is.null(Rf)){
+ model$Rf<-Rf
+ }
+ if(!is.null(shortSell)){
+ model$shorts<-ifelse(shortSell[1]%in%c("y","yes",
+ "Y","Yes","YES",TRUE),TRUE,FALSE)
+ }
+ if(!model$shorts&model$model=="none"){
+ warning("Shortsalesarealwayspermittedwhennomodelisspecified.")
+ model$shorts<-TRUE
+ }
+ if(model$Rf> -100){
+ temp<-optimalPort(model,Rf=-101,eps=eps)
```

```
+ if(model$Rf> =temp$R-eps){
+ errMess<-paste("Rfmustbelessthan",round(temp$R-
+ 0.005,4))
+ errMess<-paste(errMess,"\nRfmaynotbevalidforthisstockmodel.",
+ "\nNotethatthismessagedoesindicateNOTabug.",
+ "\nSeetheoptimalPorthelpfileformoreinfo.")
+ stop(errMess)
+ }
+ }
+ op<-list()
+ class(op)<-"optimalPortfolio"
+ op$model<-model
+ op$X<-NA
+ op$R<-NA
+ op$risk<-NA
+ if(model$model=="none"){
+ optimalPortUt<-function(model){
+ R<-model$R-model$Rf
+ Z<-solve(model$COV)%*%R
+ X<-as.numeric(Z/sum(Z))
+ names(X)<-rownames(Z)
+ ps<-portReturn(list(R=model$R,COV=model$COV),
+ X)
+ return(list(X=X,R=ps$R,VAR=ps$VAR))
+ }
+ minRiskPortUt<-function(model){
+ if(length(model$R)> 2){
+ MRPM<-minRiskPortMultiUt(model)
+ return(MRPM)
+ }
+ temp<-as.numeric(t(c(1,-1))%*%model$COV%*%
+ c(1,-1))
+ X<-model$COV[2:1,]%*%c(1,-1)*c(-1,1)/temp
+ port<-portReturn(model,X)
+ R<-sum(X*model$R)
+ V<-as.numeric(t(X)%*%model$COV%*%X)
+ return(list(X=X,R=port$R,VAR=V))
+ }
+ minRiskPortMultiUt<-function(model,curveInfo=FALSE){
+ maxRf<-optimalPortUt(model,-1000)$R
+ Rf<-maxRf-0.001*(1:2)
+ G1<-optimalPortUt(model,Rf[1])
+ G2<-optimalPortUt(model,Rf[2])
+ R.<-c(G1$R,G2$R)
+ V.<-matrix(NA,2,2)
+ V.[1,1]<-G1$VAR
```

```
+ V.[2,2]<-G2$VAR
+ V.[2,1]<-V.[1,2]<-as.numeric(t(G1$X)%*%model$COV%*%
+ G2$X)
+ MRP<-minRiskPortUt(list(R=R.,COV=V.))
+ X<-G1$X*MRP$X[1]+ G2$X*MRP$X[2]
+ if(!curveInfo){
+ return(list(R=MRP$R,VAR=MRP$VAR,X=X))
+ }
+ else{
+ return(list(R=MRP$R,VAR=MRP$VAR,X=X,
+ G1=G1))
+ }
+ }
+ OP<-optimalPortUt(model)
+ op$X<-OP$X
+ op$R<-OP$R
+ op$risk<-sqrt(OP$VAR)
+ }
+ elseif(model$model=="SIM"){
+ ratio<-(model$R-model$Rf)/model$beta
+ o<-order(-ratio)
+ alpha<-model$alpha[o]
+ beta<-model$beta[o]
+ R<-model$R[o]
+ MSE<-model$MSE[o]
+ ratio<-ratio[o]
+ c1<-(R-model$Rf)*beta/MSE
+ c2<-cumsum(c1)
+ c3<-beta^2/MSE
+ c4<-cumsum(c3)
+ Ci<-model$VM*c2/(1+ model$VM*c4)
+ cStar<-ifelse(model$shorts,rev(Ci)[1],max(Ci))
+ z<-(beta/MSE)*(ratio-cStar)
+ t<-ifelse(model$shorts,length(Ci),which.max(Ci)[1])
+ X<-z[1:t]/sum(z[1:t])
+ temp<-list(R=R[1:t],COV=model$COV[o[1:t],o[1:t]])
+ ps<-portReturn(temp,X)
+ VAR<-sum(beta[1:t]*X)^2*model$VM+ sum(MSE[1:t]*
+ X^2)
+ X<-X[match(model$ticker,names(X))]
+ names(X)<-model$ticker
+ X[is.na(X)]<-0
+ op$X<-X
+ op$R<-ps$R
+ op$risk<-sqrt(ps$VAR)
+ }
```

```
+ elseif(model$model=="CCM"){
+ ratio<-(model$R-model$Rf)/model$sigma
+ o<-order(-ratio)
+ ratio<-ratio[o]
+ R<-model$R[o]
+ rhoRatio<-model$rho/(1+ (1:length(model$R)-1)*
+ model$rho)
+ ratioSum<-cumsum(ratio)
+ Ci<-rhoRatio*ratioSum
+ cStar<-ifelse(model$shorts,rev(Ci)[1],max(Ci))
+ z<-(ratio-cStar)/((1-model$rho)*model$sigma[o])
+ t<-ifelse(model$shorts,length(Ci),which.max(Ci)[1])
+ X<-z[1:t]/sum(z[1:t])
+ temp<-list(R=R[1:t],COV=model$COV[o[1:t],o[1:t]])
+ ps<-portReturn(temp,X)
+ X<-X[match(model$ticker,names(X))]
+ names(X)<-model$ticker
+ X[is.na(X)]<-0
+ op$X<-X
+ op$R<-ps$R
+ op$risk<-sqrt(ps$VAR)
+ }
+ elseif(model$model=="MGM"&&model$shorts){
+ ind<-model$industry
+ indU<-unique(model$industry)
+ N<-rep(NA,length(indU))
+ for(iin1:length(indU)){
+ N[i]<-sum(ind==indU[i])
+ }
+ I3<-diag(rep(1,length(indU)))
+ A<-I3+ model$rho*N/(1-diag(model$rho))
+ temp<-diag(model$rho)==1
+ A[temp]<-(1+ model$rho*N/(1-diag(model$rho)))[temp]
+ C<-rep(NA,length(indU))
+ ratio<-(model$R-model$Rf)/model$sigma
+ for(iin1:length(indU)){
+ theI<-(ind==indU[i])
+ C[i]<-sum(ratio[theI]/(1-model$rho[i,i]))
+ if(model$rho[i,i]==1){
+ C[i]<-sum(ratio[theI])
+ }
+ }
+ PHI<-as.numeric(solve(A)%*%C)
+ names(PHI)<-indU
+ z<-rep(NA,length(ind))
+ for(iin1:length(ind)){
```

```
+ k<-which(indU==ind[i])
+ cStar<-sum(model$rho[k,]*PHI)
+ den<-model$sigma[i]*(1-model$rho[k,k])
+ if(model$rho[k,k]==1){
+ den<-model$sigma[i]
+ }
+ z[i]<-(ratio[i]-cStar)/den
+ }
+ X<-z/sum(z)
+ names(X)<-names(model$R)
+ ps<-portReturn(model,X)
+ op$X<-X
+ op$R<-ps$R
+ op$risk<-sqrt(ps$VAR)
+ }
+ elseif(model$model=="MGM"){
+ }
+ return(op)
+ }
>
>
> portPossCurve<-function(model,riskRange=2,detail=100,effFrontier=FALSE,
+ add=FALSE,type="l",xlab="Risk",ylab="ExpectedReturn",
+ doNotPlot=FALSE,...)
+ {
+ if(!model$shorts){
+ stop("Shortsellingmustbepermitted.\n")
+ }
+ if(!model$shorts&model$model%in%"none"){
+ model$shorts<-TRUE
+ warning("Shortsalesarealwaysallowedwhennomodelisprovided.\n")
+ }
+ G1<-optimalPort(model,Rf=-1000)
+ G2<-optimalPort(model,Rf=G1$R-0.01)
+ g1X<-G1$X[match(names(model$R),names(G1$X))]
+ g2X<-G2$X[match(names(model$R),names(G2$X))]
+ R<-c(G1$R,G2$R)
+ COV<-diag(c(G1$risk^2,G2$risk^2))
+ COV[1,2]<-as.numeric(t(g1X)%*%model$COV%*%g2X)
+ COV[2,1]<-COV[1,2]
+ meetRRF<-function(R,COV,X,detail,minRisk,RRF){
+ x<-X
+ X<-seq(X[1],X[2],length.out=detail)
+ r<-X*R[1]+ (1-X)*R[2]
+ v<-X^2*COV[1,1]+ (1-X)^2*COV[2,2]+ 2*X*
+ (1-X)*COV[1,2]
```

```
+ trim<-TRUE
+ if(sqrt(v[1])<RRF*minRisk){
+ x[1]<-2*x[1]
+ trim<-FALSE
+ }
+ if(sqrt(rev(v)[1])<RRF*minRisk){
+ x[2]<-2*x[2]
+ trim<-FALSE
+ }
+ if(trim){
+ these<-sqrt(v)<RRF*minRisk
+ if(sum(these)> detail/2){
+ out<-list()
+ out$X<-X[these]
+ out$R<-r[these]
+ out$V<-v[these]
+ }
+ else{
+ x[1]<-(x[1]-1)*0.75+ 1
+ x[2]<-(x[2]-1)*0.75+ 1
+ out<-meetRRF(R,COV,X=x,detail=detail,
+ minRisk=minRisk,RRF=RRF)
+ }
+ }
+ else{
+ out<-meetRRF(R,COV,X=x,detail=detail,minRisk=minRisk,
+ RRF=RRF)
+ }
+ return(list(R=out$R,V=out$V,X=out$X))
+ }
+ mRRF<-meetRRF(R,COV,X=c(-3,5),detail=detail,minRisk=G1$risk,
+ RRF=riskRange)
+ if(effFrontier){
+ these<-which(diff(mRRF$V)<0)
+ mRRF$R<-mRRF$R[these]
+ mRRF$V<-mRRF$V[these]
+ mRRF$X<-mRRF$X[these]
+ }
+ ports<-mRRF$X%*%t(g1X)+ (1-mRRF$X)%*%t(g2X)
+ toReturn<-list(R=mRRF$R,risk=sqrt(mRRF$V),ports=ports)
+ if(add&!doNotPlot){
+ lines(toReturn$risk,toReturn$R,type=type,...)
+ }
+ elseif(!doNotPlot){
+ plot(toReturn$risk,toReturn$R,type=type,xlab=xlab,
+ ylab=ylab,...)
```

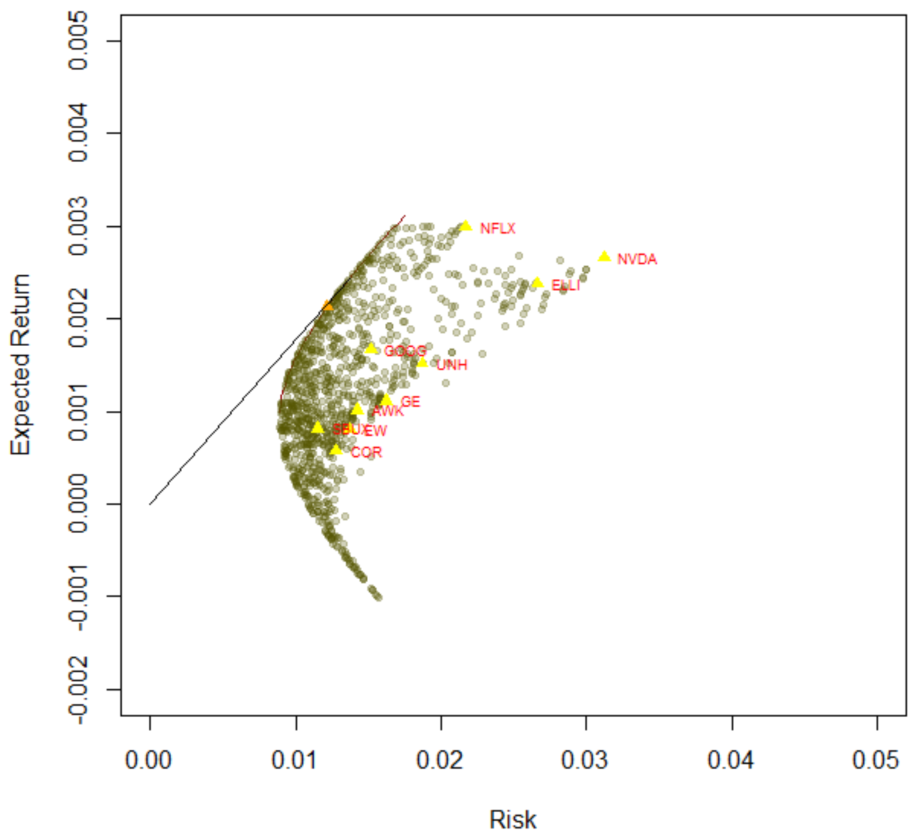
```
+ }
+ invisible(toReturn)
+ }
>
> portCloud<-function(model,riskRange=2,detail=25,N=3000,add=TRUE,
+ col=c("#55550044"),pch=20,subSamp=1000,xlim="default",
+ ylim="default",xlab="Risk",ylab="Return",...)
+ {
+ if(!model$shorts){
+ stop("Shortsellingmustbepermitted.\n")
+ }
+ n<-length(model$R)
+ ppc<-portPossCurve(model,riskRange=riskRange,detail=detail,
+ doNotPlot=TRUE)
+ PPP<-ceiling(N/n/dim(ppc$ports)[1])
+ N<-n*PPP*dim(ppc$ports)[1]
+ if(subSamp> N){
+ subSamp<-N
+ }
+ subSamp<-min(c(subSamp,N))
+ ports<-ppc$ports
+ portMat<-matrix(NA,subSamp,2)
+ steps<-seq(1/(PPP+ 1),1-1/(PPP+ 1),length.out=PPP)
+ x<-matrix(NA,subSamp,n)
+ r<-c()
+ v<-c()
+ m<-0
+ M<-0
+ if(subSamp==N){
+ subSamp<-1:N
+ }
+ else{
+ subSamp<-sample(N,subSamp)
+ }
+ for(iin1:n){
+ for(lin1:dim(ppc$ports)[1]){
+ for(jin1:PPP){
+ M<-M+ 1
+ if(M%in%subSamp){
+ m<-m+ 1
+ u<-c(rep(0,i-1),1-steps[j],rep(0,
+ n-i))
+ x[m,]<-steps[j]*ports[l,]+ u
+ r[m]<-sum(model$R*x[m,])
+ v[m]<-as.numeric(t(x[m,])%*%model$COV%*%
+ x[m,])
```

```
+ portMat[m,2]<-r[m]
+ portMat[m,1]<-sqrt(v[m])
+ }
+ }
+ }
+ }
+ if(add){
+ points(portMat,col=col,pch=pch,...)
+ }
+ else{
+ if(xlim[1]=="default"){
+ xMin<-min(portMat[,1])
+ xlim<-c(xMin,riskRange*xMin)
+ if(ylim[1]=="default"){
+ ylim<-range(portMat[portMat[,1]<xlim[2],
+ 2])
+ ylim<-ylim+ c(-1,1)*diff(ylim)/20
+ }
+ }
+ elseif(ylim[1]=="default"){
+ ylim<-range(portMat[,2])
+ }
+ plot(portMat,col=col,pch=pch,xlim=xlim,ylim=ylim,
+ xlab=xlab,ylab=ylab,...)
+ }
+ invisible(list(ports=x,R=r,risk=sqrt(v)))
+ }
>
>
> portReturn<-function(model,X)
+ {
+ if(is.null(names(model$R))|is.null(names(X))){
+ R<-sum(model$R*X)
+ V<-as.numeric(t(X)%*%model$COV%*%X)
+ }
+ else{
+ these<-match(names(X),names(model$R))
+ R<-sum(model$R[these]*X)
+ V<-as.numeric(t(X)%*%model$COV[these,these]%*%
+ X)
+ }
+ portSum<-list(R=R,VAR=V,X=X,ticker=model$ticker,
+ model=model)
+ class(portSum)<-"portReturn"
+ return(portSum)
+ }
```

```
>
> #########sjy######
>
> stocksname<-c("GE","SBUX","NVDA","ELLI","NFLX","AWK","EW",
+ "COR","GOOG","UNH")
> mydata<-get.hist.quote(stocksname[1],start="2015-01-05",end="2016-10-21",
quote="AdjClose",compression="d")
timeseriesends2016-10-20
> for(iin2:length(stocksname)){
+ mystocks<-get.hist.quote(stocksname[i],start="2015-01-05",end="2016-10-21",
quote="AdjClose",compression="d");mystocks<-na.approx(mystocks)
+ mydata<-cbind.data.frame(mydata,mystocks)
+ }
timeseriesends 2016-10-20
timeseriesends 2016-10-20
timeseriesends 2016-10-20
timeseriesends 2016-10-20
timeseriesends 2016-10-20
timeseriesends 2016-10-20
timeseriesends 2016-10-20
timeseriesends 2016-10-20
timeseriesends 2016-10-20
> colnames(mydata)<-stocksname#keepstocknames
> mydata.returns<-apply(mydata,2,Delt)[-1,]
> rownames(mydata.returns)<-rownames(mydata)[-1]
> mmm1<-stockModel(mydata.returns,model="none",Rf=-0.0049)
>
>
> quartz()
Errorinquartz():couldnotfindfunction"quartz"
> (all.these.stupid.stocks<-portPossCurve(mmm1,effFrontier=TRUE,
xlim=c(0,0.05),ylim=c(-0.002,0.005),col="darkred"))
$R
 [1] 0.0031125645 0.0030367263 0.0029608881 0.0028850499 0.0028092117
 [6] 0.0027333735 0.0026575353 0.0025816971 0.0025058589 0.0024300207
[11] 0.0023541825 0.0022783443 0.0022025061 0.0021266679 0.0020508298
[16] 0.0019749916 0.0018991534 0.0018233152 0.0017474770 0.0016716388
[21] 0.0015958006 0.0015199624 0.0014441242 0.0013682860 0.0012924478
[26] 0.0012166096 0.0011407714 0.0010649332 0.0009890951

$risk
 [1] 0.017463897 0.017022576 0.016585472 0.016152926 0.015725316 0.015303054
 [7] 0.014886596 0.014476443 0.014073145 0.013677311 0.013289606 0.012910762
[13] 0.012541584 0.012182949 0.011835816 0.011501227 0.011180307 0.010874268
[19] 0.010584399 0.010312065 0.010058690 0.009825741 0.009614702 0.009427046
[25] 0.009264193 0.009127470 0.009018067 0.008936987 0.008885005
```

```
$ports
              GE          SBUX       NVDA        ELLI        NFLX
 [1,] -0.349238523 -0.3665554575 0.52893037 0.165452786  0.18834967
 [2,] -0.329659020 -0.3512990147 0.51145220 0.159711465  0.18124961
 [3,] -0.310079517 -0.3360425719 0.49397402 0.153970144  0.17414955
 [4,] -0.290500013 -0.3207861291 0.47649584 0.148228823  0.16704949
 [5,] -0.270920510 -0.3055296863 0.45901767 0.142487502  0.15994943
 [6,] -0.251341007 -0.2902732435 0.44153949 0.136746180  0.15284937
 [7,] -0.231761504 -0.2750168007 0.42406132 0.131004859  0.14574931
 [8,] -0.212182001 -0.2597603579 0.40658314 0.125263538  0.13864925
 [9,] -0.192602497 -0.2445039151 0.38910497 0.119522217  0.13154919
[10,] -0.173022994 -0.2292474723 0.37162679 0.113780895  0.12444913
[11,] -0.153443491 -0.2139910295 0.35414862 0.108039574  0.11734907
[12,] -0.133863988 -0.1987345867 0.33667044 0.102298253  0.11024901
[13,] -0.114284485 -0.1834781440 0.31919227 0.096556932  0.10314895
[14,] -0.094704982 -0.1682217012 0.30171409 0.090815611  0.09604889
[15,] -0.075125478 -0.1529652584 0.28423592 0.085074289  0.08894883
[16,] -0.055545975 -0.1377088156 0.26675774 0.079332968  0.08184878
[17,] -0.035966472 -0.1224523728 0.24927957 0.073591647  0.07474872
[18,] -0.016386969 -0.1071959300 0.23180139 0.067850326  0.06764866
[19,]  0.003192534 -0.0919394872 0.21432322 0.062109005  0.06054860
[20,]  0.022772038 -0.0766830444 0.19684504 0.056367683  0.05344854
[21,]  0.042351541 -0.0614266016 0.17936687 0.050626362  0.04634848
[22,]  0.061931044 -0.0461701588 0.16188869 0.044885041  0.03924842
[23,]  0.081510547 -0.0309137160 0.14441052 0.039143720  0.03214836
[24,]  0.101090050 -0.0156572732 0.12693234 0.033402398  0.02504830
[25,]  0.120669554 -0.0004008305 0.10945417 0.027661077  0.01794824
[26,]  0.140249057  0.0148556123 0.09197599 0.021919756  0.01084818
[27,]  0.159828560  0.0301120551 0.07449782 0.016178435  0.00374812
[28,]  0.179408063  0.0453684979 0.05701964 0.010437114 -0.00335194
[29,]  0.198987566  0.0606249407 0.03954147 0.004695792 -0.01045200
            AWK         EW        COR       GOOG          UNH
 [1,] 0.3058451 0.14603983 0.40385674 0.10289810 -0.125578599
 [2,] 0.3091776 0.14218582 0.39260493 0.10212979 -0.117553323
 [3,] 0.3125100 0.13833180 0.38135312 0.10136148 -0.109528047
 [4,] 0.3158425 0.13447778 0.37010130 0.10059318 -0.101502771
 [5,] 0.3191750 0.13062376 0.35884949 0.09982487 -0.093477495
 [6,] 0.3225074 0.12676974 0.34759768 0.09905656 -0.085452219
 [7,] 0.3258399 0.12291572 0.33634587 0.09828826 -0.077426942
 [8,] 0.3291724 0.11906171 0.32509406 0.09751995 -0.069401666
 [9,] 0.3325048 0.11520769 0.31384225 0.09675164 -0.061376390
[10,] 0.3358373 0.11135367 0.30259043 0.09598334 -0.053351114
[11,] 0.3391698 0.10749965 0.29133862 0.09521503 -0.045325838
[12,] 0.3425023 0.10364563 0.28008681 0.09444672 -0.037300562
[13,] 0.3458347 0.09979162 0.26883500 0.09367842 -0.029275285
```

```
[14,] 0.3491672 0.09593760 0.25758319 0.09291011 -0.021250009
[15,] 0.3524997 0.09208358 0.24633138 0.09214180 -0.013224733
[16,] 0.3558321 0.08822956 0.23507956 0.09137350 -0.005199457
[17,] 0.3591646 0.08437554 0.22382775 0.09060519  0.002825819
[18,] 0.3624971 0.08052152 0.21257594 0.08983689  0.010851095
[19,] 0.3658295 0.07666751 0.20132413 0.08906858  0.018876372
[20,] 0.3691620 0.07281349 0.19007232 0.08830027  0.026901648
[21,] 0.3724945 0.06895947 0.17882051 0.08753197  0.034926924
[22,] 0.3758270 0.06510545 0.16756869 0.08676366  0.042952200
[23,] 0.3791594 0.06125143 0.15631688 0.08599535  0.050977476
[24,] 0.3824919 0.05739741 0.14506507 0.08522705  0.059002752
[25,] 0.3858244 0.05354340 0.13381326 0.08445874  0.067028029
[26,] 0.3891568 0.04968938 0.12256145 0.08369043  0.075053305
[27,] 0.3924893 0.04583536 0.11130964 0.08292213  0.083078581
[28,] 0.3958218 0.04198134 0.10005782 0.08215382  0.091103857
[29,] 0.3991542 0.03812732 0.08880601 0.08138551  0.099129133

>
> #Add a cloud of many portfolios:
> portCloud(mmm1, add=TRUE)
>
> standard.dev <- sd
> e.r.data <- meanReturns
>
> #Add the five stocks plus the point of tangency:
> points(standard.dev, e.r.data, pch=17, col = "yellow")
> points(standard.dev, e.r.data, pch=17, col="yellow")
> points(all.these.stupid.stocks$risk[14], all.these.stupid.stocks$R[14],
col = "orange", pch = 17)
>
> #Add the tangent (the following will draw the line only up to G):
> riskfree <- 0.001/365 # gotta make same frequency
>
> #Choose [10] because we actually saw the point and it looked most tangeant
>
> tanx <- all.these.stupid.stocks$risk[10]
> tany <- all.these.stupid.stocks$R[10]
> #segments(0,riskfree,op$risk,op$R)
> segments(0,riskfree,tanx,tany)
>
>
> text(standard.dev, e.r.data, stocksname, cex=0.6, pos=4, col="red")
```



Note: This graph looks different in R(with green shadow).
(d)
Based on MPT (Modern Portfolio Theory), compute the optimal weights (i.e., how much to invest in each stock), and identify your tangency portfolio.

```
> getSymbols("GE")
[1] "GE"
Warning message:
semi-transparency is not supported on this device: reported only once per page
> getSymbols("SBUX")
[1] "SBUX"
> getSymbols("UNH")
[1] "UNH"
> getSymbols("NVDA")
[1] "NVDA"
> getSymbols("ELLI")
[1] "ELLI"
> getSymbols("NFLX")
[1] "NFLX"
```

```
> getSymbols("AWK")
[1] "AWK"
> getSymbols("GOOG")
[1] "GOOG"
> getSymbols("EW")
[1] "EW"
> getSymbols("COR")
[1] "COR"
> ge1 <- Ad(GE)['2016-01-05::2017-10-21']
> sbux1 <- Ad(SBUX)['2016-01-05::2017-10-21']
> unh1<- Ad(UNH)['2016-01-05::2017-10-21']
> nvda1 <- Ad(NVDA)['2016-01-05::2017-10-21']
> elli1 <- Ad(ELLI)['2016-01-05::2017-10-21']
> nflx1 <- Ad(NFLX)['2016-01-05::2017-10-21']
> awk1 <- Ad(AWK)['2016-01-05::2017-10-21']
> google1 <- Ad(GOOG)['2016-01-05::2017-10-21']
> ew1 <- Ad(EW)['2016-01-05::2017-10-21']
> cor1 <- Ad(COR)['2016-01-05::2017-10-21']
> newdata <- data.frame(ge1,sbux1,unh1,nvda1,elli1,nflx1,awk1,google1,
ew1,cor1)
> newdata1 <- apply(newdata, 2, Delt)[-1,]
> port <- portfolio.optim(newdata1, rf = 0.01); port
$pw
 [1]  5.359215e-02  3.416356e-02  3.147416e-01  6.195096e-02 -1.034984e-18
 [6]  0.000000e+00  2.914297e-01  1.262573e-01  1.444490e-02  1.034198e-01

$px
  [1] -5.189911e-03 -2.203661e-02 -9.825066e-03  2.844339e-03  1.226755e-02
  [6] -1.912100e-02  1.373892e-02 -1.513469e-02  1.295020e-02  2.644584e-03
 [11]  1.851477e-03  2.203707e-02 -9.248332e-03  1.045383e-02 -1.071511e-02
 [16]  8.576741e-03  2.878847e-02  5.586818e-03 -7.553777e-03 -7.081822e-03
 [21] -5.941255e-03 -2.251486e-02 -1.547760e-02  1.047738e-02  1.204330e-02
 [26] -5.018132e-03  2.219283e-03  1.560913e-02  1.310252e-02  7.339934e-03
 [31]  4.731003e-03  1.848828e-02 -8.966869e-04  5.507186e-06  1.308299e-02
 [36] -1.352253e-02 -1.152037e-02  1.894849e-02  1.415276e-02 -4.551796e-03
 [41]  4.482793e-04 -6.471300e-03  1.727945e-03  4.051411e-03  3.489727e-03
 [46]  1.326359e-02  7.780406e-04 -1.829534e-04  5.450601e-03  5.046398e-03
 [51]  6.306652e-04 -8.027302e-04  3.404423e-03  4.169051e-03 -2.480299e-03
 [56]  4.253758e-04  1.562003e-02  6.765195e-04 -8.263967e-04  7.857545e-03
 [61]  3.659399e-05 -1.247407e-02  8.871569e-03 -3.814570e-03 -2.646797e-04
 [66]  4.536815e-04  2.715605e-03  5.481217e-03  1.758266e-03  4.181864e-03
 [71]  6.848207e-03  4.911439e-03  2.043662e-03 -1.408730e-02 -9.007321e-04
 [76]  7.560029e-03  2.069283e-03  4.179241e-04 -1.564448e-03 -1.287391e-03
 [81]  1.139447e-02 -1.455226e-03 -2.225418e-03 -9.845375e-04  6.837232e-03
 [86]  2.861507e-03  5.676037e-03 -9.624445e-03 -1.406347e-03  4.589609e-03
 [91]  9.357504e-03 -1.517042e-02  2.181106e-04 -7.909847e-04  6.716633e-03
```

```
 [96] -2.781209e-03  1.510994e-02 -3.168050e-04  1.698459e-03  6.311709e-03
[101]  3.884310e-04  6.503248e-03  4.952458e-03  5.810298e-03  2.827188e-03
[106]  5.483448e-04  1.368487e-02  5.361067e-03 -7.266415e-03 -2.337472e-03
[111]  1.988812e-03  1.235834e-04  7.573860e-03 -8.457386e-03  2.326901e-03
[116]  1.812228e-03 -9.438931e-04  1.019334e-02 -1.306630e-02  3.684148e-03
[121]  1.223699e-02  8.723931e-03  1.519219e-02 -2.880896e-03  6.326090e-03
[126]  2.068492e-03 -7.479298e-03  1.049228e-02 -5.048139e-03 -4.121974e-03
[131]  3.293490e-03 -8.104872e-04 -1.418408e-03  1.813020e-03  5.998066e-03
[136]  3.050733e-03 -4.677165e-03  1.057372e-02 -4.542643e-03 -1.773306e-03
[141] -2.704099e-03  2.465310e-03  7.931401e-03  2.730424e-03 -8.670016e-03
[146] -4.367688e-03 -1.707587e-03 -3.090632e-03 -6.875024e-03  2.004849e-04
[151]  3.715707e-03  1.810267e-03  6.297140e-03 -8.500884e-03 -6.582951e-03
[156]  6.179175e-03  3.915073e-03 -3.735079e-03 -8.573445e-04 -3.127806e-04
[161] -8.334393e-03 -7.289026e-03 -6.806966e-03  5.466997e-03 -8.676873e-03
[166] -2.576632e-03 -4.915020e-05  8.746193e-03  6.263015e-03 -7.052857e-04
[171] -1.432137e-03 -2.746899e-02  1.361554e-02 -1.382396e-02 -5.303693e-04
[176]  1.133877e-02  9.128296e-03  2.007287e-03 -9.084140e-04  1.614662e-02
[181]  9.178278e-03 -5.283337e-03 -4.702715e-03  4.310145e-03 -9.692634e-04
[186] -1.278631e-02  3.414523e-03 -7.574471e-03 -8.655607e-03 -7.485901e-03
[191] -4.080514e-03 -2.447165e-03  9.962376e-03 -1.244998e-02  7.231243e-03
[196] -5.525721e-04  3.056537e-04 -3.271189e-04  2.950225e-02  1.297915e-03
[201]  1.732804e-03  5.811009e-05  6.861188e-03 -1.384159e-03 -8.978645e-03
[206] -1.068829e-02  9.717666e-05  7.695579e-03 -1.352108e-02 -8.286877e-03
[211] -7.832322e-03 -5.898542e-03  2.609207e-02  8.336588e-03 -6.890800e-03
[216] -4.667372e-03  1.845133e-02  3.711215e-03  1.012309e-02  5.352866e-03
[221] -2.618039e-03  1.552294e-03  6.547523e-03  1.357744e-02 -5.323754e-03
[226]  7.325845e-03  1.380668e-03  1.697189e-02 -1.536957e-02 -5.340789e-03
[231]  4.221324e-03  7.363894e-04 -7.926037e-04  1.434127e-02  4.370983e-03
[236]  3.649901e-03  1.545849e-03  8.343017e-03 -9.802957e-03  3.830165e-03
[241]  1.198619e-02  3.050382e-04  2.342387e-03 -2.862600e-03  3.426011e-04
[246]  7.548444e-03  3.452262e-03 -1.481238e-02  5.749344e-03 -8.576905e-03
[251]  3.871616e-03  8.291313e-03  1.427608e-03  1.973533e-03 -4.423928e-03
[256] -4.077878e-03  6.232744e-04  4.226036e-04  1.765504e-05  1.517975e-03
[261] -1.139661e-03  4.063035e-05 -9.647713e-04  2.723901e-03  7.655741e-03
[266]  3.147253e-03  5.869593e-03 -2.426375e-03 -8.494495e-03  4.021836e-03
[271] -3.874364e-03  1.731995e-03  3.348852e-03 -2.607210e-03  1.801123e-03
[276]  1.936172e-03 -5.219352e-04  3.638157e-03 -1.113273e-03  5.686408e-04
[281]  3.787560e-03  1.648698e-03 -9.733167e-03  1.381620e-02  2.623730e-03
[286]  1.092829e-03  5.311851e-03  6.155182e-03  2.949899e-03  7.156878e-03
[291] -3.415409e-03 -1.939120e-03 -1.230386e-03 -2.124467e-03 -8.173648e-03
[296]  7.050514e-04  8.528315e-03  4.137207e-03 -4.299667e-03  1.282390e-02
[301] -6.488367e-03  2.872882e-03 -4.019553e-03 -4.327367e-03  3.473655e-03
[306] -7.042978e-03  2.061922e-03 -1.926127e-03  2.975937e-03 -6.766043e-04
[311]  4.737759e-03  2.005256e-03  9.705661e-04 -2.448828e-03  2.349175e-03
[316] -5.177325e-04  7.550114e-04 -2.202942e-03  2.745104e-03 -3.987187e-04
[321] -4.291734e-03  1.306505e-02  4.072130e-03  3.170589e-03  6.993593e-03
```

```
[326]  2.356745e-03  6.695244e-03  6.384102e-03 -9.615261e-04  6.919729e-03
[331]  1.616100e-03  8.420366e-04 -2.737481e-03 -6.024438e-03 -3.491708e-03
[336]  5.182528e-03 -2.078266e-04 -7.508592e-03  1.152999e-02  2.248073e-03
[341]  1.124610e-04  6.551026e-03 -8.307827e-03 -8.673668e-03  7.769482e-03
[346]  9.565041e-03  1.043172e-02  2.199777e-03  5.037030e-03  8.868731e-03
[351] -5.246577e-04  2.257591e-03 -1.235314e-03  1.156429e-02  3.274588e-03
[356]  8.531712e-04  5.294082e-04  7.303484e-03 -8.917568e-04 -5.107044e-03
[361] -3.203429e-03  6.985819e-03  8.244551e-03  2.090884e-04  4.953220e-03
[366]  3.450306e-03 -2.461500e-03  4.133110e-03  3.014329e-03 -3.605330e-03
[371] -3.685917e-03 -1.517118e-02  5.040272e-03 -1.489392e-02 -1.788648e-03
[376] -1.067724e-03  5.226608e-03 -9.292665e-03  8.946090e-03  2.825508e-03
[381] -1.129398e-03  1.164663e-02 -5.095816e-03  8.507976e-03  7.202108e-04
[386]  5.244124e-03  5.867668e-03  7.014325e-03  3.541340e-03 -4.840043e-03
[391] -5.515083e-03  3.951641e-03 -7.788907e-03  4.126996e-03 -1.746103e-04
[396]  4.434861e-03  3.860084e-03 -2.430170e-03  1.468247e-03  4.326918e-03
[401] -2.105937e-03 -2.217649e-04 -1.152237e-02 -3.186173e-03  1.297630e-02
[406]  2.109756e-03 -2.908318e-04 -9.199773e-03 -3.202050e-03  4.737214e-03
[411]  1.234996e-02 -7.128478e-04 -1.139781e-03  1.642409e-04  2.293433e-03
[416]  2.111056e-03 -1.052411e-03  8.880888e-03  1.928102e-03 -3.735567e-03
[421] -3.464884e-03  2.485087e-03 -1.466414e-03  1.090676e-02 -1.051981e-02
[426] -2.229749e-03  6.745073e-04  5.275205e-03  3.040555e-03 -4.586831e-03
[431]  1.122350e-03 -3.985657e-03 -5.186789e-03 -3.801955e-03  1.897589e-03
[436]  1.737029e-03  4.013023e-03  2.483660e-03  5.889195e-03  1.031659e-03
[441]  5.756535e-03  1.634022e-03  2.084066e-03 -3.207751e-03  2.280958e-03
[446]  4.515599e-03 -1.123696e-03 -1.460848e-03  3.059746e-03  2.199032e-02
[451]  3.755712e-03  7.176235e-04  7.658115e-03

$pm
[1] 0.001250053

$ps
[1] 0.00729029
```

As we can see from the results, the optimal weight for 10 stocks is as follows:
General Electric Company : 0.0536
Starbucks: 0.03416
UnitedHealth Group:0.314
Nvidia:0.06195
Elli:$\approx$ 0 (not legibile)
Netfelix:0
AmericanWaters:0.2914
Google:0.1262
Edwards Lifesciences Corporation:0.0144
CoreSite Realty Corporation:0.1034

Pie Chart For Final Portfolio

```
> slices<-c(28.75,9.42,1.36,5.05,12.82,6.54,3.44,32.8)
> lbls<-c("SBUX","UNH","AWK","GE","GOOG","NVDA","EW","COR")
> pct<-round(slices/sum(slices)*100)
> lbls<-paste(lbls,pct) # add percents to labels
> lbls<-paste(lbls,"%",sep="") # ad % to labels
> pie(slices,labels=lbls,col=rainbow(length(lbls)),
+ main="Pie Chart of Portfolio")
```

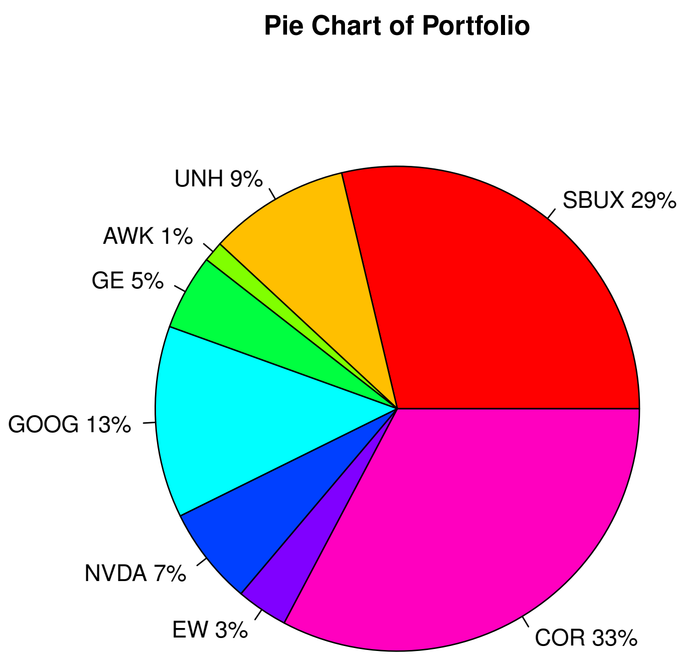


Table 1: My portfolio

	Wednesday	Thursday	Friday	Monday	Tuesday	Final Weight
AWK	239.05	267.89	198.04	237.12	215.84	28.75%
COR	340.23	238.90	123.37	152.00	13.68	9.42%
EW	-45.89	23.89	12.09	16.68	-9.24	1.36%
GE	-125.90	-200.07	-178.00	-383.04	-243.58	5.05%
GOOG	-27.89	34.67	-24.67	-27.84	67.44	12.82%
NVDA	-35.89	-23.67	-78.09	-90.19	7.25	6.54%
SBUX	29.36	25.89	36.90	23.20	-214.6	3.44%
UNH	240.56	230.82	289.06	235.32	313.53	32.8%
Total P/L	613.63	598.32	378.7	201.27	150.5	

As we can see from the results above, in the final portfolio:
UNH(0.09),
SBUX(0.29),
COR(0.33),
EW(0.03),
NVDA(0.07),
GOOG(0.13),
GE(0.05),
AWK(0.01),
UNH(0.09);
compared to the optiaml weights we get in part(d):
General Electric Company : 0.0536
Starbucks: 0.03416
UnitedHealth Group:0.314
Nvidia:0.06195
Elli:$\approx$ 0 (not legibile)
Netfelix:0
AmericanWaters:0.2914
Google:0.1262
Edwards Lifesciences Corporation:0.0144
CoreSite Realty Corporation:0.1034

AWK(AmericanWatersWork) and UNH(UnitedHealthGroup) have better market performance compared to other stocks but their weight is small: AWK is only 0.01 and UNH is only 0.09, which results in small profit in our portfilio. Average Daily Return for UNH and AWK are 261.8 dollars and 231.5 dollars respectivley. But as we can see from the risk-return plots, the stocks such as NVDA don't perform quite well as expected: in the table, its average daily return is only around -27.8(whichi means it loses profit everyday).

2 Question 2

(a)

```
setwd("D:/Econ 403A/Project 2")
wages<-read.csv("wages.csv")
#t test for bivariate
t.test(wages$women,wages$men,alternative = "two.sided",conf.level
= 0.95)
##
##     Welch Two Sample t-test
##
## data:  wages$women and wages$men
## t = -1.7702, df = 117.84, p-value = 0.07928
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -6522.5198   365.3984
## sample estimates:
## mean of x mean of y
##  51848.88  54927.44
```

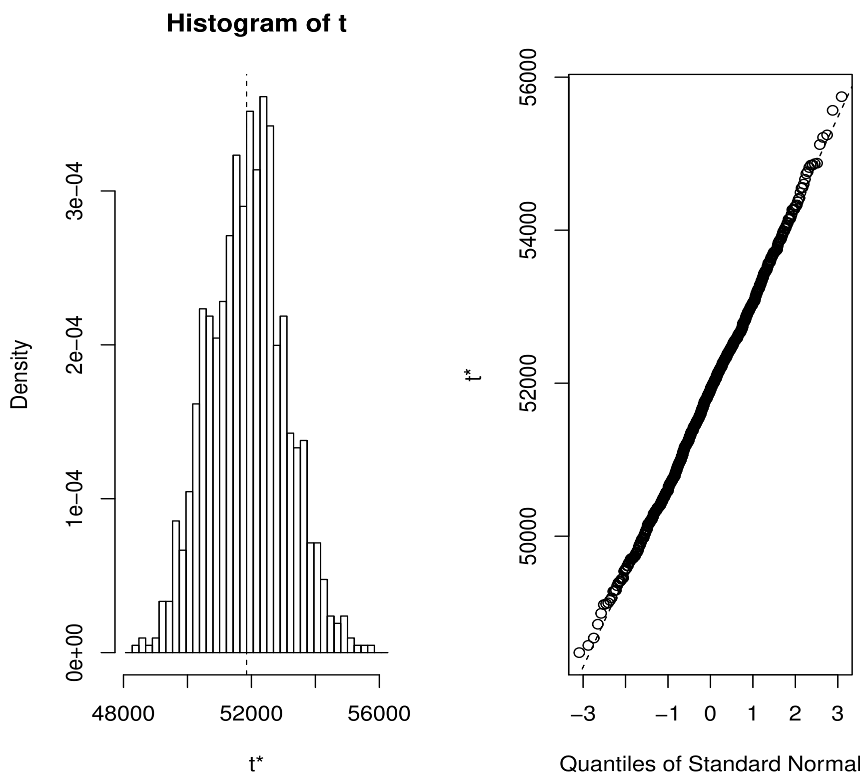
Comment: After t-test, the p-value=0.07928 ¿ significant level=0.05, so we cannot reject Null Hypothesis, so that there is no significant difference of mean wages between women and men.

(b)

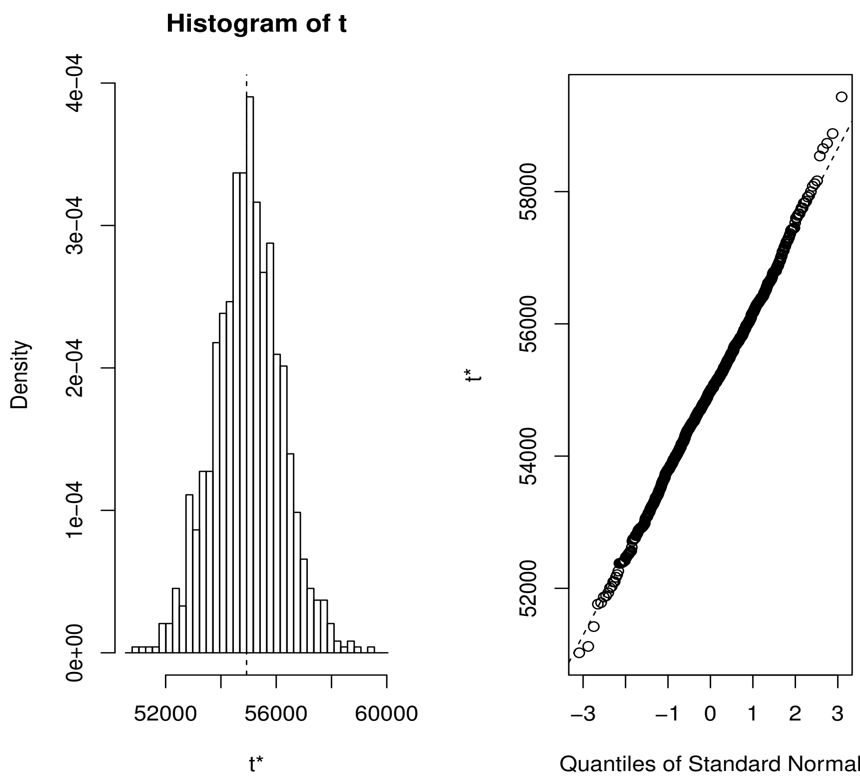
```
library(boot)
women_wages<-c(wages$women)
men_wages<-c(wages$men)
# bootstrap
bs <- function(data, indices) {
d <- data[indices]
return(mean(d))
}
M_BSF<-boot(data=women_wages,statistic = bs,R=1000)
M_BSM<-boot(data=men_wages,statistic = bs,R=1000)
M_BS<-c(M_BSF,M_BSM)
```

(c)

```
# plot M_BSF
plot(M_BSF)
```



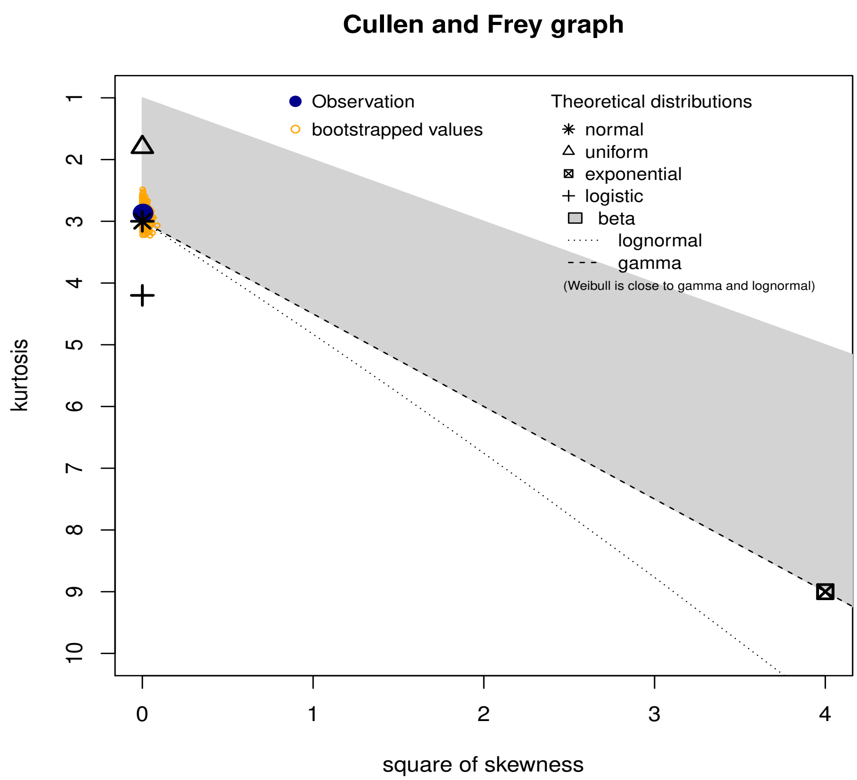
```
# plot M_BSM
plot(M_BSM)
```



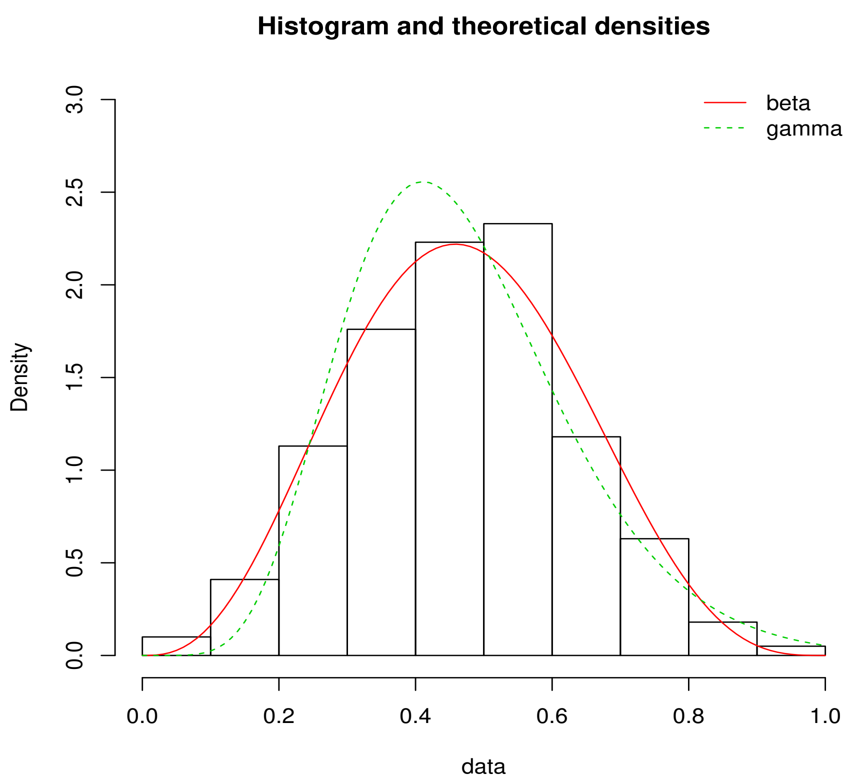
```
library(fitdistrplus)
# fit distribution of women
descdist(M_BSF$t[,1], boot = 1000)
```

```
## summary statistics
## ------
## min:  48477.95   max:  55745.17
## median:  51910.5
## mean:  51883.37
## estimated sd:  1197.922
## estimated skewness:  0.06308384
## estimated kurtosis:  2.867748
```

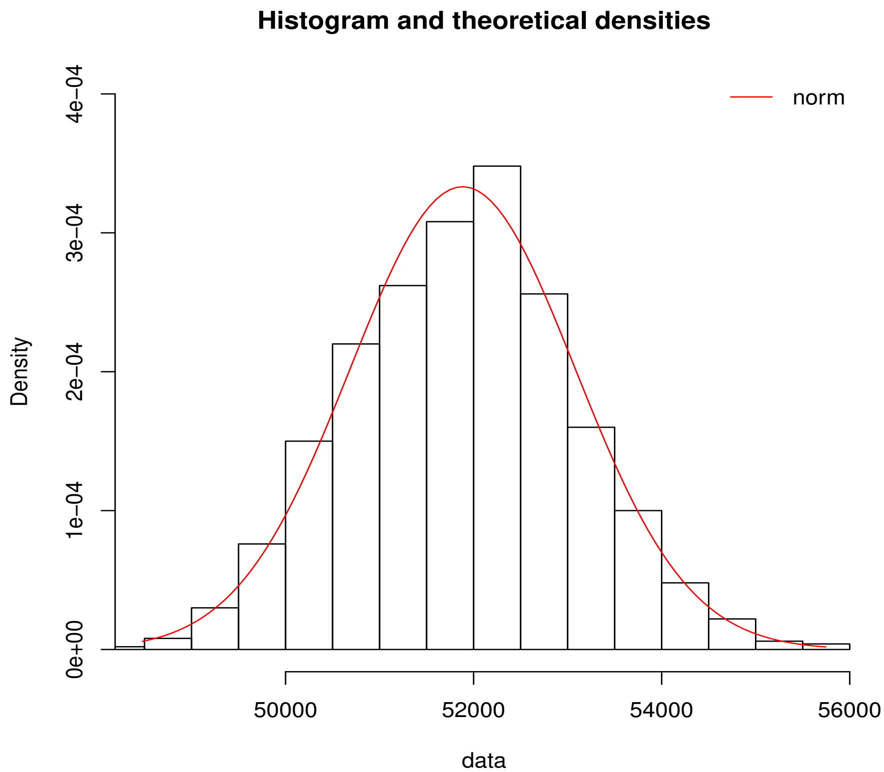
```
R_wage_boot_1<-(M_BSF$t[,1]-min(M_BSF$t[,1]))/(max(M_BSF$t[,1])
-min(M_BSF$t[,1]))
```



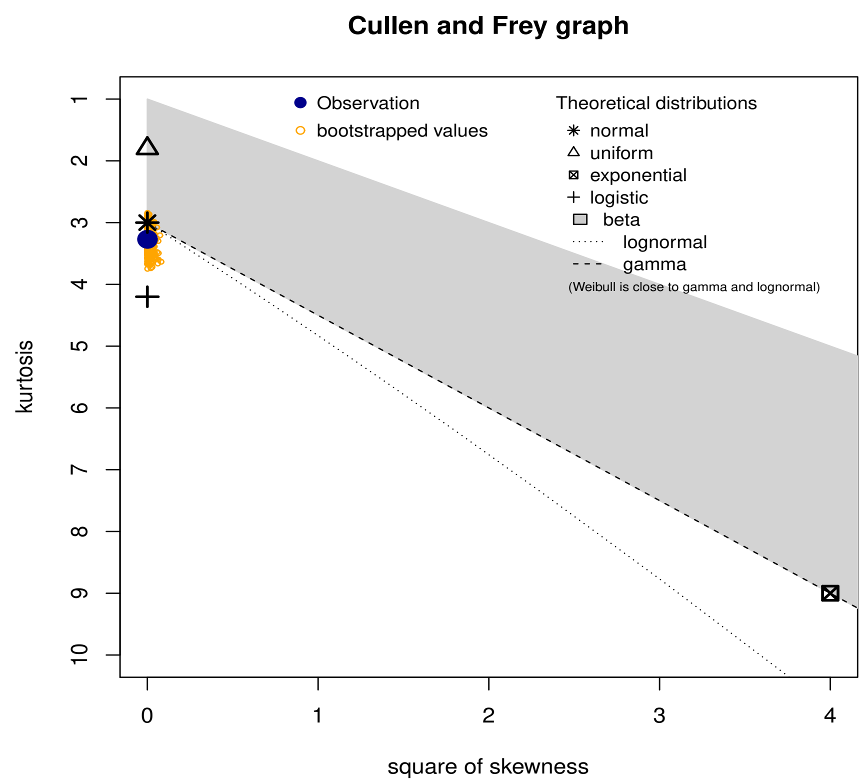
```
# fit beta distribution
betafit_women <- fitdist(R_wage_boot_1, "beta", method = "mme")
# fit gamma distribution
gammafit_women <- fitdist(R_wage_boot_1, "gamma", method = "mme")
# fit lnormal distribution
normalfit_women <- fitdist(M_BSF$t[,1], "norm", method = "mme")
# plot density
denscomp(list(betafit_women, gammafit_women), ylim = c(0, 3),
lwd=3, legendtext = c("beta", "gamma"))
```



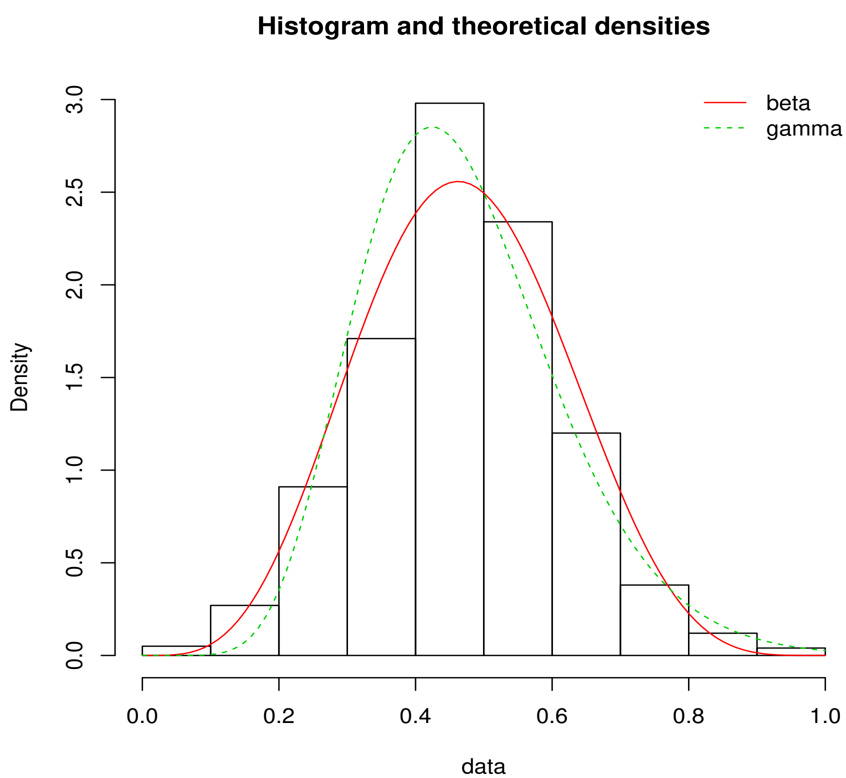
```
denscomp(normalfit_women,ylim = c(0, 0.0004),legendtext = c("norm"))
```



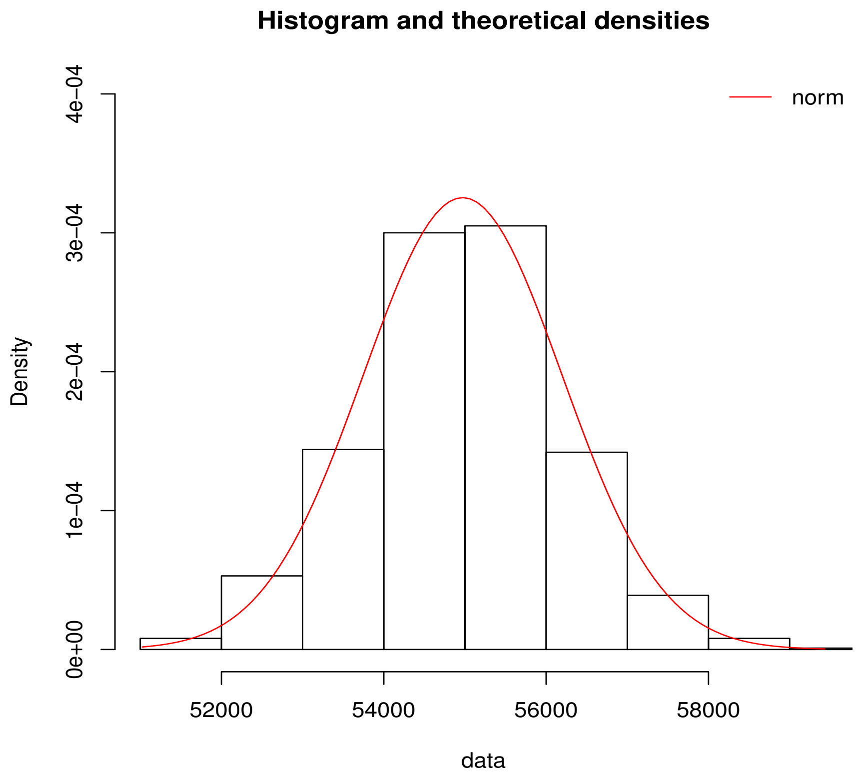
```
descdist(M_BSM$t[,1], boot = 1000)
## summary statistics
## ------
## min:  51024.83   max:  59433.16
## median:  54984.23
## mean:  54971.44
## estimated sd:  1226.828
## skewness:  0.02777143
## estimated kurtosis:  3.270396
R_wage_boot_2<-(M_BSM$t[,1]-min(M_BSM$t[,1]))/(max(M_BSM$t[,1])
-min(M_BSM$t[,1]))
```



```
# fit beta distribution
betafit_men <- fitdist(R_wage_boot_2, "beta", method = "mme")
# fit gamma distribution
gammafit_men <- fitdist(R_wage_boot_2, "gamma", method = "mme")
# fit lnormal distribution
normalfit_men <- fitdist(M_BSM$t[,1], "norm", method = "mme")
# plot density
denscomp(list(betafit_men, gammafit_men),
ylim = c(0, 3),legendtext = c("beta", "gamma"))
```



```
denscomp(normalfit_men,ylim = c(0, 0.0004),legendtext = c("norm"))
```

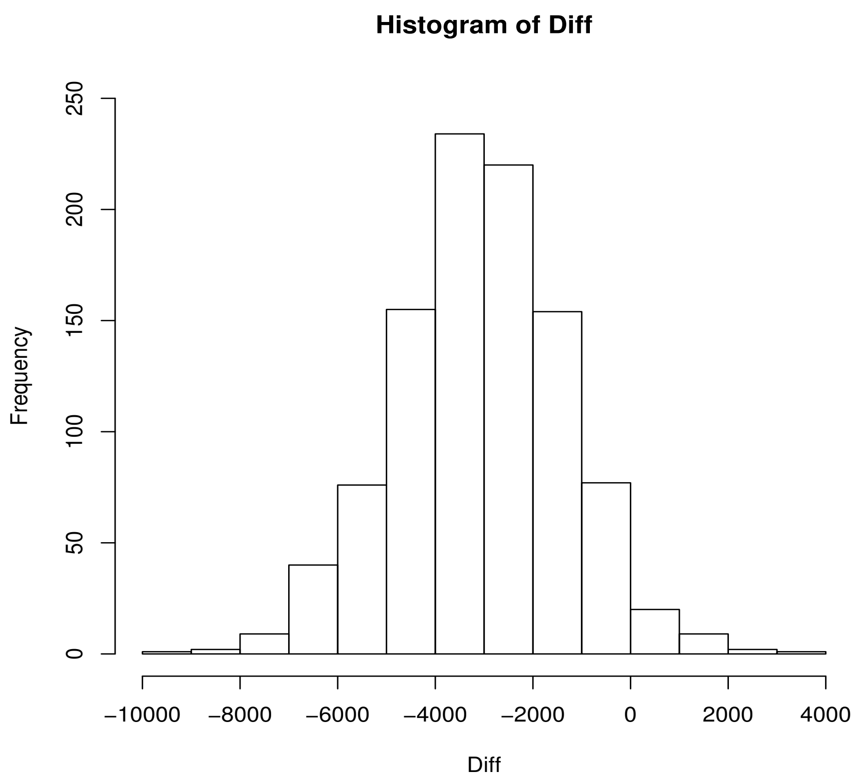



```
# ks.test
ks.test(M_BSF$t[,1], M_BSM$t[,1])
##
##      Two-sample Kolmogorov-Smirnov test
##
## data:  M_BSF$t[, 1] and M_BSM$t[, 1]
## D = 0.796, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

Conclusion: So from the Cullen and Frey Graphs, we should choose normal distribution. And we can see from the histograms, it shows evenly distribution. Therefore, we choose normal distribution.Therefore,after ts-test, there shows a difference between wages of women and men.

(d)

```
E_M_BSF<-M_BSF$t
E_M_BSM<-M_BSM$t
# calculate difference
diff<-E_M_BSF-E_M_BSM
Diff<-as.vector(diff)
# plot histogram
hist(Diff, ylim = c(0, 250))
```




```
library(tseries)
# Jarque Bera Test
jarque.bera.test(Diff)
##
##     Jarque Bera Test
##
## data:  Diff
## X-squared = 3.0923, df = 2, p-value = 0.2131
```

Comment: Jarque-Bera Test tests whether the data are from normal distribution. After the test, we found the p-value is 0.9757. We cannot reject the null hypothesis, so that the data is from normal distribution.

(e)

```
# t-test
t.test(E_M_BSF,E_M_BSM,alternative = "two.sided",conf.level = 0.95)

##     Welch Two Sample t-test
##
## data:  E_M_BSF and E_M_BSM
## t = -56.951, df = 1996.9, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3194.403 -2981.724
## sample estimates:
## mean of x mean of y
##  51883.37  54971.44
```

The p-value in t-tset is extremely small,So we could conclude that if we expand sample size to 1000, we should reject Null Hypothesis, and it shows that there is a difference of wages between women and men.

(f) General Conclusion: At first we set the sample size to 60, and find that there is no difference of wages between women and men. After we bootstrap the sample size to 1000, it shows the opposite result. Therefore, the sample size we choose at first is too small and unrepresentitive to interpret the fact. In the other statistical researches, we should choose large and representitive samples to fit the model and interpret them.

3 Question 3

(a)

```
setwd("D:/Econ 403A/Project 2")
capm<-read.csv("capm4.csv")
y_dis<-capm$dis-capm$riskfree
x<-capm$mkt-capm$riskfree
dis.lm<-lm(y_dis~x)
coeffs=coefficients(dis.lm)
coeffs
##  (Intercept)            x
## -0.003659946  0.914594877
# to see the standard error
summary(dis.lm)
##
## Call:
## lm(formula = y_dis ~ x)
##
## Residuals:
##      Min        1Q    Median        3Q       Max
## -0.182443 -0.028738 -0.007054  0.027853  0.276871
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.00366    0.00694  -0.527    0.599
## x            0.91460    0.12015   7.612 4.87e-12 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 0.06848 on 130 degrees of freedom
## Multiple R-squared:  0.3083,    Adjusted R-squared:  0.303
## F-statistic: 57.94 on 1 and 130 DF,  p-value: 4.866e-12
```

so the intercept alpha is -0.00366 with error 0.00694
the coefficient of x_dis beta is 0.91460 with error 0.12015

```
y_ge<-capm$ge-capm$riskfree
ge.lm<-lm(y_ge~x)
coeffs=coefficients(ge.lm)
coeffs
## (Intercept)           x
## -0.00532363  0.85897377
# to see the standard error
summary(ge.lm)
##
## Call:
```

```
## lm(formula = y_ge ~ x)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.156837 -0.036767 -0.004774  0.034106  0.181055
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.005324   0.005518  -0.965    0.336
## x            0.858974   0.095525   8.992 2.48e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05444 on 130 degrees of freedom
## Multiple R-squared:  0.3835,    Adjusted R-squared:  0.3787
## F-statistic: 80.86 on 1 and 130 DF,  p-value: 2.477e-15
```

so the intercept alpha is -0.00532363 with error 0.005518
the coefficient of x_ge i.e. beta is 0.85897377 with error 0.095525

```
y_gm<-capm$gm-capm$riskfree
gm.lm<-lm(y_gm~x)
coeffs=coefficients(gm.lm)
coeffs
##  (Intercept)            x
## -0.007247694  1.146837699
# to see the standard error
summary(gm.lm)
##
## Call:
## lm(formula = y_gm ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.40666 -0.06120 -0.00273  0.06278  0.29125
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.007248   0.011393  -0.636    0.526
## x            1.146838   0.197242   5.814 4.46e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1124 on 130 degrees of freedom
## Multiple R-squared:  0.2064,    Adjusted R-squared:  0.2003
## F-statistic: 33.81 on 1 and 130 DF,  p-value: 4.464e-08
```

so the intercept alpha is -0.007247694 with error 0.011393
the coefficient of x_gm i.e. beta is 1.146837699 with error 0.197242

```
y_ibm<-capm$ibm-capm$riskfree
ibm.lm<-lm(y_ibm~x)
coeffs=coefficients(ibm.lm)
coeffs
## (Intercept)           x
##  0.01020723  1.14824488
# to see the standard error
summary(ibm.lm)
##
## Call:
## lm(formula = y_ibm ~ x)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.262998 -0.039921 -0.002788  0.038935  0.269202
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.010207   0.007114   1.435    0.154
## x           1.148245   0.123152   9.324 3.83e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07019 on 130 degrees of freedom
## Multiple R-squared:  0.4007,    Adjusted R-squared:  0.3961
## F-statistic: 86.93 on 1 and 130 DF,  p-value: 3.829e-16
```

so the intercept alpha is 0.01020723 with error 0.007114
the coefficient of x_ibm i.e. beta is 1.14824488 with error 0.123152

```
y_msft<-capm$msft-capm$riskfree
msft.lm<-lm(y_msft~x)
coeffs=coefficients(msft.lm)
coeffs
## (Intercept)           x
##  0.01373669  1.25991925
# to see the standard error
summary(msft.lm)
##
## Call:
## lm(formula = y_msft ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -0.26864 -0.05569 -0.00845  0.04261  0.35678
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.013737   0.009061   1.516    0.132
## x           1.259919   0.156861   8.032 5.03e-13 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 0.0894 on 130 degrees of freedom
## Multiple R-squared:  0.3317,    Adjusted R-squared:  0.3265
## F-statistic: 64.51 on 1 and 130 DF,  p-value: 5.034e-13
```

so the intercept alpha is 0.01373669 with error 0.009061
the coefficient of x_msft i.e. beta is 1.25991925 with error 0.156861

```
y_xom<-capm$xom-capm$riskfree
xom.lm<-lm(y_xom~x)
coeffs=coefficients(xom.lm)
coeffs
##  (Intercept)           x
## -0.007966102  0.461257641
# to see the standard error
summary(xom.lm)
##
## Call:
## lm(formula = y_xom ~ x)
##
## Residuals:
##      Min        1Q    Median        3Q       Max
## -0.127422 -0.032706 -0.002982  0.027316  0.216216
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.007966   0.005118  -1.556    0.122
## x            0.461258   0.088607   5.206 7.35e-07 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 0.0505 on 130 degrees of freedom
## Multiple R-squared:  0.1725,    Adjusted R-squared:  0.1661
## F-statistic:  27.1 on 1 and 130 DF,  p-value: 7.349e-07
```

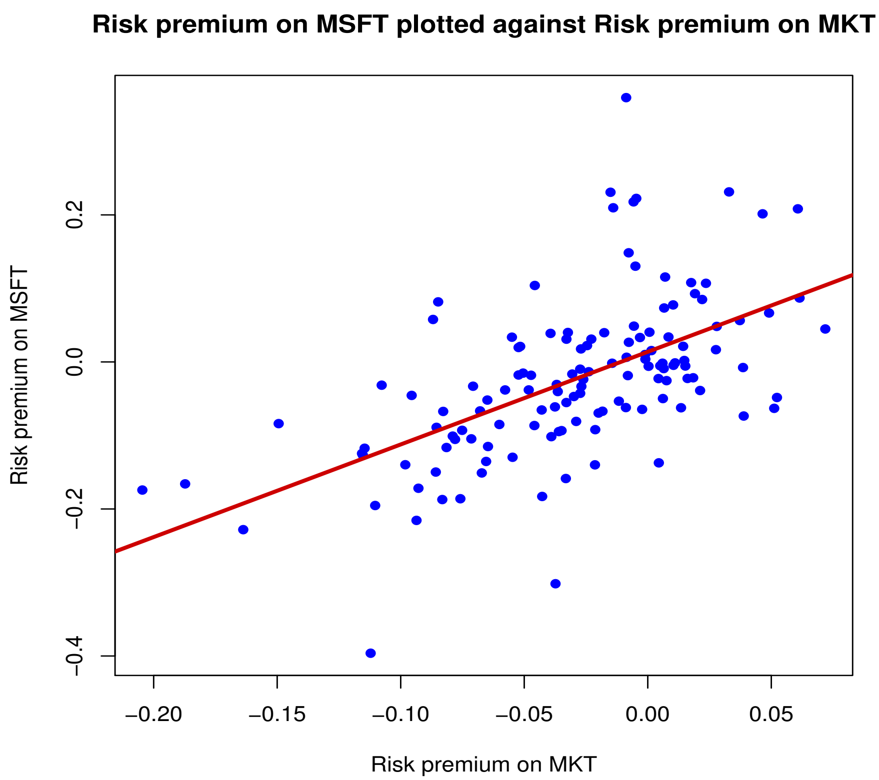
so the intercept alpha is -0.007966102 with error 0.005118
the coefficient of x_xom i.e. beta is 0.461257641 with error 0.088607
Conclusion: the values of beta of dis, ge and xom are "defensive", however, the values of beta of gm, ibm and msft are "aggressive", in which xom is the most

"defensive" and msft is the most "aggressive".

(b)

It seems correct.

```
# Regession curve
plot(x,y_msft,pch=16,cex=1,col="blue",main="Risk premium on MSFT
plotted against Risk premium on MKT",xlab="Risk premium on MKT",
ylab="Risk premium on MSFT")
abline(lm(y_msft~x),lwd=3,col="red3")
```



(c)

When α is setted to 0, we can use the function $y \sim -1 + x$
PS: I'm not sure which confidence interval should I use, so I choose the prediction CI in this part and the regression parameters' CIs in part (d) and (e).

```
dis.lm_0<-lm(y_dis~-1+x)
```

```
coeffs=coefficients(dis.lm_0)
coeffs
# to see the standard error
summary(dis.lm_0)
# the coefficient of x_dis beta is 0.9470557 with error 0.1029

# predict with confidence interval
new<-data.frame(x=0.5)
dis.lm_pred<-predict(dis.lm_0,new,interval="prediction",level=0.95)
dis.lm_pred
##         fit       lwr       upr
## 1 0.4735278 0.3043843 0.6426713
```

When x_0=0.5, and the prediction interval is 0.95, the value of y_0 is 0.473527, and the prediction interval is [0.3043843, 0.6426713].

```
ge.lm_0<-lm(y_ge~-1+x)
coeffs=coefficients(ge.lm_0)
coeffs
# to see the standard error
summary(ge.lm_0)
# the coefficient of x_dis i.e. beta is 0.9061901 with error 0.08202

# predict with confidence interval
ge.lm_pred<-predict(ge.lm_0,new,interval="prediction",level=0.95)
ge.lm_pred
##         fit       lwr       upr
## 1 0.4530951 0.3182832 0.5879069
```

When x_0=0.5, and the prediction interval is 0.95, the value of y_0 is 0.4530951, and the prediction interval is [0.3182832, 0.5879069].

```
gm.lm_0<-lm(y_gm~-1+x)
coeffs=coefficients(gm.lm_0)
coeffs
# to see the standard error
summary(gm.lm_0)
# the coefficient of x_dis i.e. beta is 1.211119 with error 0.169

# predict with confidence interval
gm.lm_pred<-predict(gm.lm_0,new,interval="prediction",level=0.95)
gm.lm_pred
##         fit       lwr       upr
## 1 0.6055595 0.3277554 0.8833636
```

When x_0=0.5, and the prediction interval is 0.95, the value of y_0 is 0.6055595, and the prediction interval is [0.3277554, 0.8833636].

```
ibm.lm_0<-lm(y_ibm~-1+x)
coeffs=coefficients(ibm.lm_0)
coeffs
# to see the standard error
summary(ibm.lm_0)
# the coefficient of x_dis i.e. beta is 1.057715 with error 0.1062

# predict with confidence interval
ibm.lm_pred<-predict(ibm.lm_0,new,interval="prediction",level=0.95)
ibm.lm_pred
##         fit       lwr       upr
## 1 0.5288575 0.3543083 0.7034066
```

When x_0=0.5, and the prediction interval is 0.95, the value of y_0 is 0.5288575, and the prediction interval is [0.3543083, 0.7034066].

```
msft.lm_0<-lm(y_msft~-1+x)
coeffs=coefficients(msft.lm_0)
coeffs
# to see the standard error
summary(msft.lm_0)
# the coefficient of x_dis i.e. beta is 1.138086 with error 0.1354

# predict with confidence interval
msft.lm_pred<-predict(msft.lm_0,new,interval="prediction",level=0.95)
msft.lm_pred
##         fit      lwr       upr
## 1 0.5690429 0.346515 0.7915708
```

When x_0=0.5, and the prediction interval is 0.95, the value of y_0 is 0.5690429, and the prediction interval is [0.346515, 0.7915708].

```
xom.lm_0<-lm(y_xom~-1+x)
coeffs=coefficients(xom.lm_0)
coeffs
# to see the standard error
summary(xom.lm_0)
# the coefficient of x_dis i.e. beta is 0.5319106 with error 0.07651

# predict with confidence interval
xom.lm_pred<-predict(xom.lm_0,new,interval="prediction",level=0.95)
xom.lm_pred
##         fit       lwr       upr
## 1 0.2659553 0.1401957 0.3917149
```

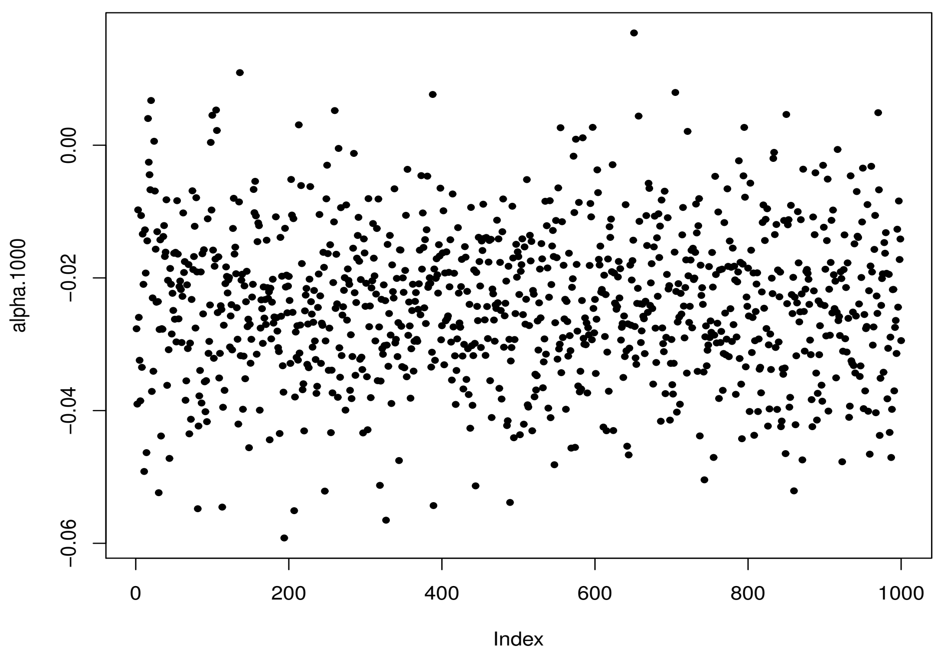
When x_0=0.5, and the prediction interval is 0.95, the value of y_0 is 0.2659553, and the prediction interval is [0.1401957, 0.3917149].

Conclusion: The restriction $\alpha_j = 0$ has led to small changes in the β, but it has not changed the aggressive or defensive nature of the stock.

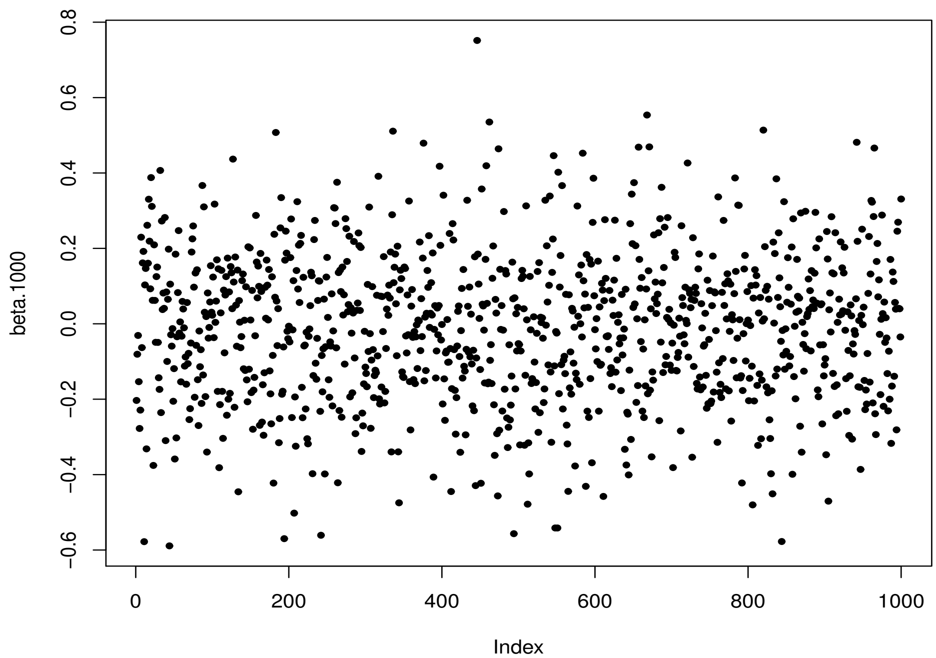
(d)

```
# bootstrap 1000 samples
msft.boot.sample<-list()
for(i in 1:1000){
msft.boot.sample[[i]]<-sample(y_msft,size=132,replace=TRUE)
}
x.boot.sample<-list()
for(i in 1:1000){
x.boot.sample[[i]]<-sample(x,size=132,replace=TRUE)
}

# linear regression
n<-1000
my_lms<-lapply(1:n,function(i) lm(msft.boot.sample[[i]]~x.boot.sample[[i]]))
sapply(my_lms,coef)
summaries<-lapply(my_lms,summary)

#Two ways to get alphas and betas
alpha.1000=numeric(1000)
for(i in 1:1000){
alpha.1000[i]<-as.numeric(coef(my_lms[[i]])["(Intercept)"])
}
alpha.1000

beta.1000=numeric(1000)
for(i in 1:1000){
beta.1000[i]<-as.numeric(coef(my_lms[[i]])["x.boot.sample[[i]]"])
}
beta.1000

# Or use sapply
coef_list=sapply(my_lms,coef)
alpha.1000<-coef_list[1,]
beta.1000<-coef_list[2,]
```

```
# plot
plot(alpha.1000,pch=20)
```



```
plot(beta.1000,pch=20)
```



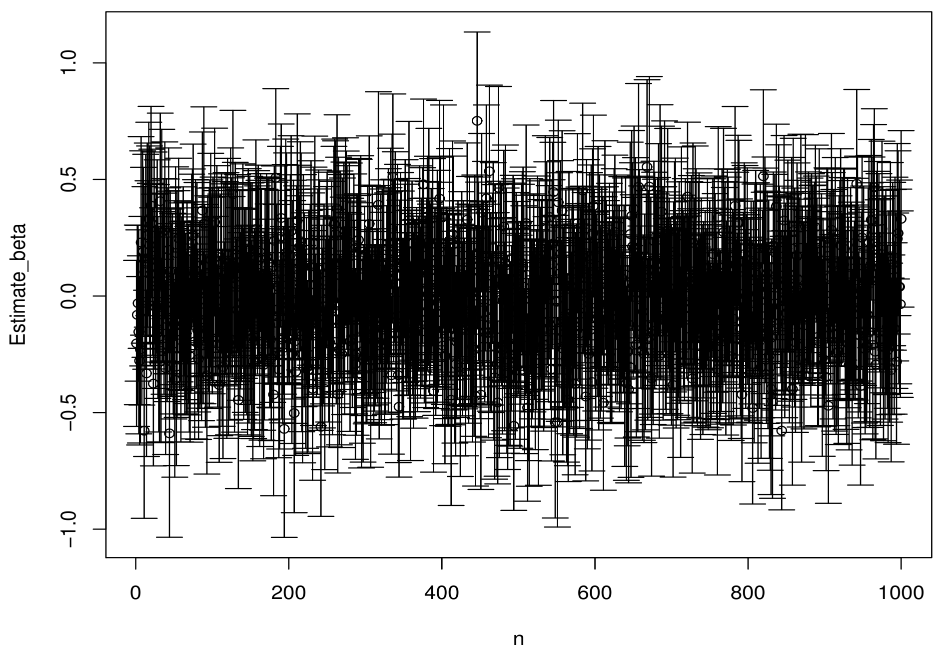
To compute the parameters' 95% confidence intervals

```
# d_beta
d_beta.1=numeric(1000)
d_beta.2=numeric(1000)
for(i in 1:1000){
d_beta.1[i]<-as.numeric(confint(lm(msft.boot.sample[[i]]
~x.boot.sample[[i]]),'x.boot.sample[[i]]',level=0.95)[1])
}
for(i in 1:1000){
d_beta.2[i]<-as.numeric(confint(lm(msft.boot.sample[[i]]
~x.boot.sample[[i]]),'x.boot.sample[[i]]',level=0.95)[2])
}

# d_alpha
d_alpha.1=numeric(1000)
d_alpha.2=numeric(1000)
for(i in 1:1000){
```

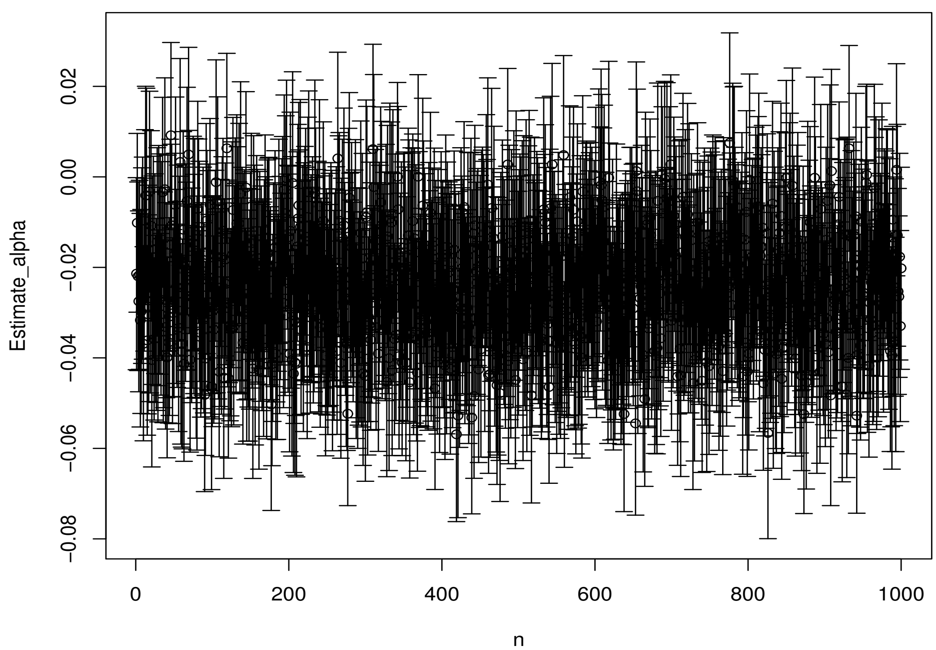
```
d_alpha.1[i]<-as.numeric(confint(lm(msft.boot.sample[[i]]
~x.boot.sample[[i]]),'(Intercept)',level=0.95)[1])
}
for(i in 1:1000){
d_alpha.2[i]<-as.numeric(confint(lm(msft.boot.sample[[i]]
~x.boot.sample[[i]]),'(Intercept)',level=0.95)[2])
}
```

Define that the length of confidence intervals is the upper bound minus lower bound. Plot the length of confidence intervals:

```
# d_beta
n<-1:1000
Estimate_beta<-c(beta.1000)
L<-c(d_beta.1)
U<-c(d_beta.2)
require(plotrix)
plotCI(n,Estimate_beta,ui=U,li=L)
```



```
# d_alpha
n<-1:1000
Estimate_alpha<-c(alpha.1000)
L<-c(d_alpha.1)
U<-c(d_alpha.2)
require(plotrix)
plotCI(n,Estimate_alpha,ui=U,li=L)
```



Note: Because there are 1000 confidence intervals, so the image is too dense to see each length of cis clearly.

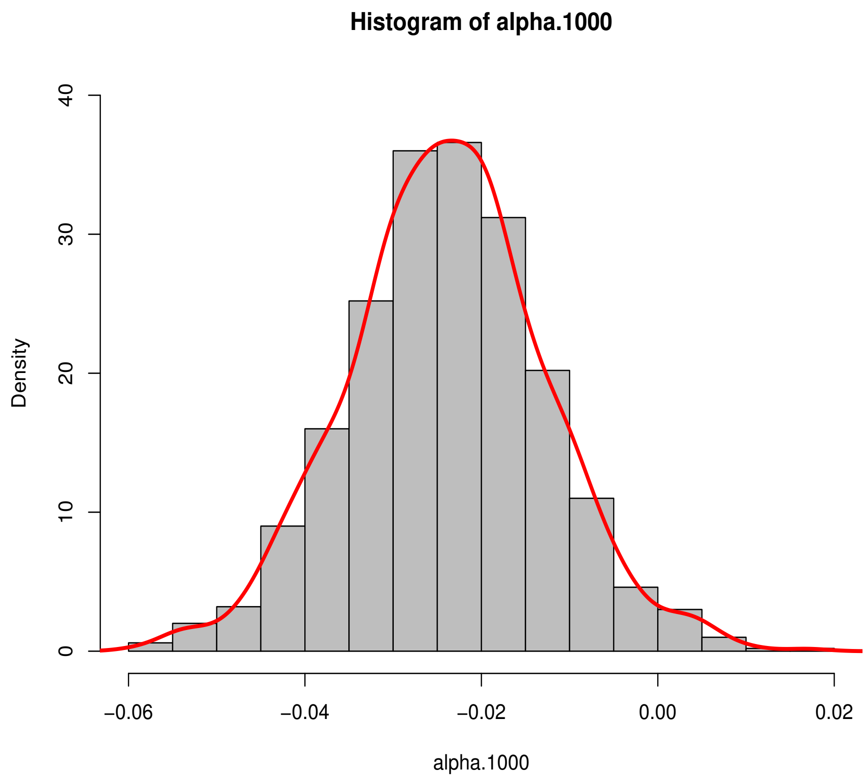
(e)

```
# Compute the mean
# alpha
mean(alpha.1000)
[1] -0.02392545

# beta
mean(beta.1000)
```

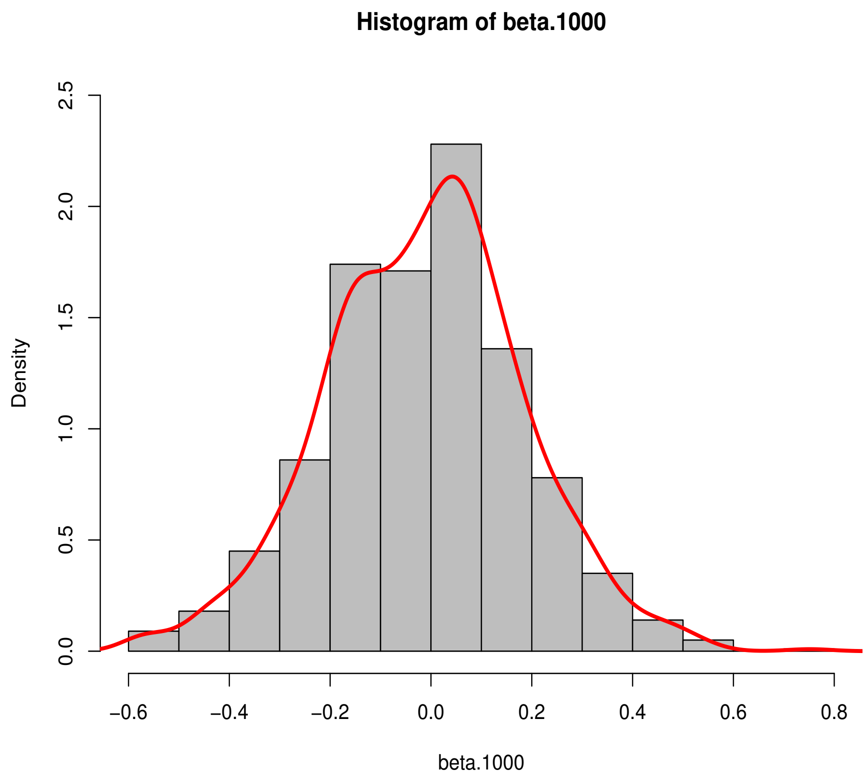
```
[1] -0.01213128

# d_alpha
d_alpha<-d_alpha.2-d_alpha.1
mean(d_alpha)
[1] 0.04353836

# d_beta
d_beta<-d_beta.2-d_beta.1
mean(d_beta)
[1] 0.7591905

# Compute the volatility (S.D./mean)
vol_alpha<-sd(alpha.1000)/mean(alpha.1000)
vol_alpha
[1] -0.4669488

vol_beta<-sd(beta.1000)/mean(beta.1000)
vol_beta
[1] -16.14764

vol_d_alpha<-sd(d_alpha)/mean(d_alpha)
vol_d_alpha
[1] 0.09240299

vol_d_beta<-sd(d_beta)/mean(d_beta)
vol_d_beta
[1] 0.1159136
```

Hist with density curves.

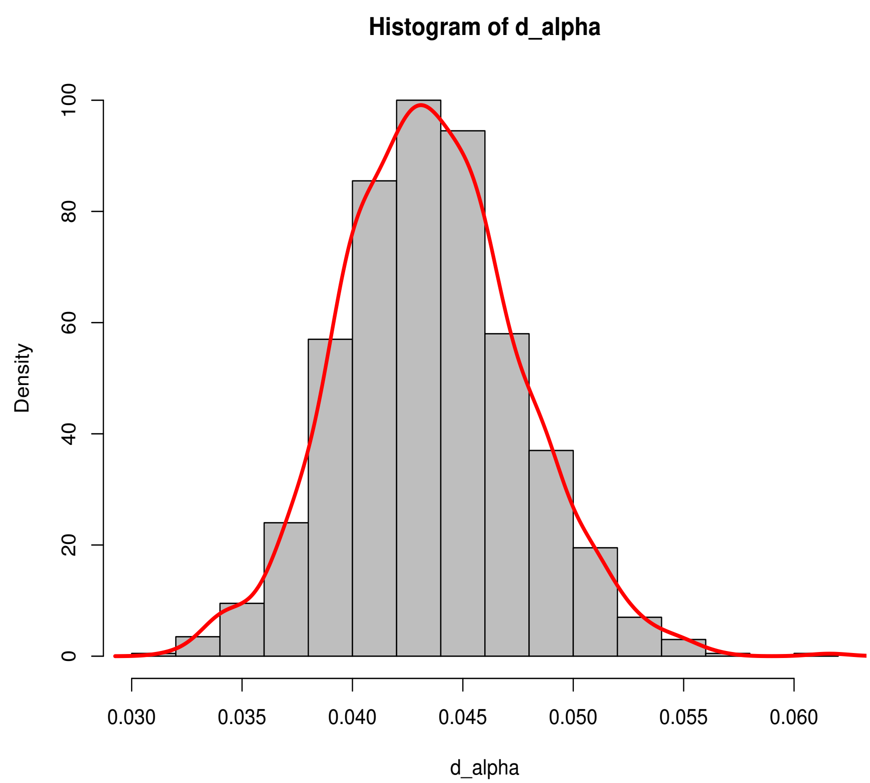
```
# alpha
hist(alpha.1000,col=8,ylim=c(0,35),prob=TRUE)
lines(density(alpha.1000),col="red",lwd=3)
```



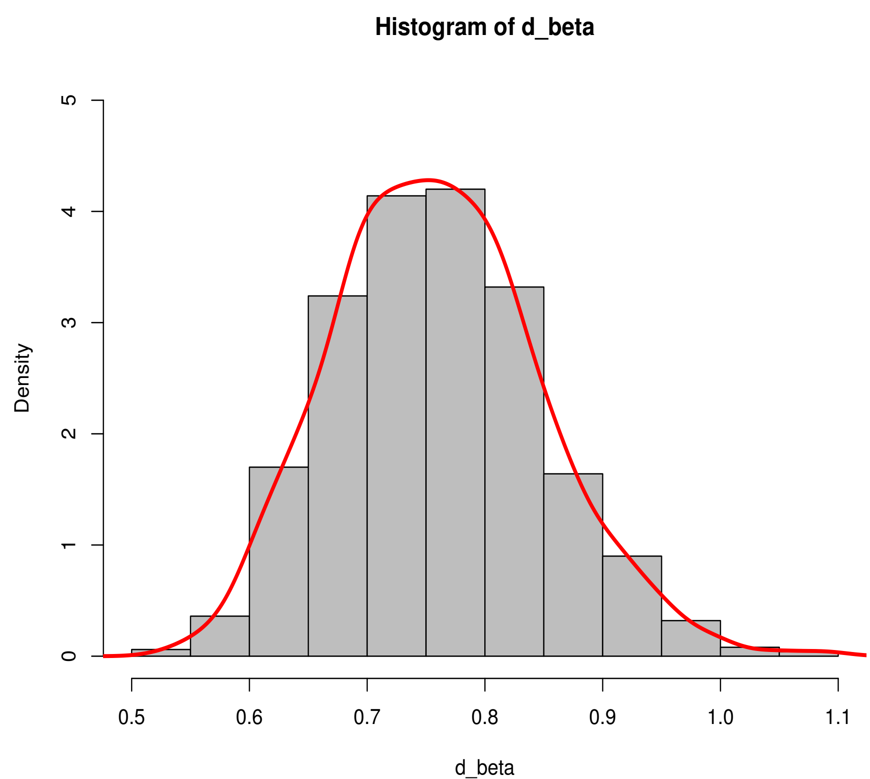
```
# beta
hist(beta.1000,col=8,ylim=c(0,2.5),prob=TRUE)
lines(density(beta.1000),col="red",lwd=3)
```



```
# d_alpha
hist(d_alpha,col=8,ylim=c(0,100),prob=TRUE)
lines(density(d_alpha),col="red",lwd=3)
```



```
# d_beta
hist(d_beta,col=8,ylim=c(0,5),prob=TRUE)
lines(density(d_beta),col="red",lwd=3)
```



(e)

I trust the bootstrap samples more because it has more observations.