20/21 Fall RMSC4002 Assignment 1

CHAN SZE YUEN SYNGIENE 1155127616

The code used in this assignment is written by R.

1(a)

1.

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.6.2
```

```
stock <- read_csv("C:\\Users\\Chan Sze Yuen\\Documents\\RDATASET\\RMSC4002DATA\\hkse50.csv")
```

```
## Parsed with column specification:
## cols(
##   code = col_double(),
##   name = col_character()
## )
```

```
set.seed(27616)
#random seed of my last 5 digits of student ID:1155127616
r <- sample(1:50, size=8)
#select 5 random integers
stock_q1a=stock[r,]
#Result:1044,144,267,388 and 941
stock_q1a
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 3.6.3
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
stock_num_1044=get.hist.quote(instrument = "1044.HK",
start="2018-01-01", end="2020-01-01",quote = c("Adjusted"),
provider = c("yahoo"), compression = "d",
retclass = c("zoo")) #stock 1044
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
stock_num_0144=get.hist.quote(instrument = "0144.HK",
start="2018-01-01", end="2020-01-01",quote = c("Adjusted"),
provider = c("yahoo"), compression = "d",
retclass = c("zoo")) #stock 0144

stock_num_0267=get.hist.quote(instrument = "0267.HK",
start="2018-01-01", end="2020-01-01",quote = c("Adjusted"),
provider = c("yahoo"), compression = "d",
retclass = c("zoo")) #stock 0267

stock_num_0388=get.hist.quote(instrument = "0388.HK",
start="2018-01-01", end="2020-01-01",quote = c("Adjusted"),
provider = c("yahoo"), compression = "d",
retclass = c("zoo")) #stock 0388

stock_num_0941=get.hist.quote(instrument = "0941.HK",
start="2018-01-01", end="2020-01-01",quote = c("Adjusted"),
provider = c("yahoo"), compression = "d",
retclass = c("zoo")) #stock 0941

stock_num_2319=get.hist.quote(instrument = "2319.HK",
start="2018-01-01", end="2020-01-01",quote = c("Adjusted"),
provider = c("yahoo"), compression = "d",
retclass = c("zoo")) #stock 2319

stock_num_0494=get.hist.quote(instrument = "0494.HK",
start="2018-01-01", end="2020-01-01",quote = c("Adjusted"),
provider = c("yahoo"), compression = "d",
retclass = c("zoo")) #stock 0494

stock_num_0151=get.hist.quote(instrument = "0151.HK",
start="2018-01-01", end="2020-01-01",quote = c("Adjusted"),
provider = c("yahoo"), compression = "d",
retclass = c("zoo")) #stock 0151
```

```
#checking
head(stock_num_0144)
```

```
##            Adjusted
## 2018-01-02 16.98554
## 2018-01-03 17.14925
## 2018-01-04 17.19019
## 2018-01-05 17.06740
## 2018-01-08 17.06740
## 2018-01-09 16.94461
```

```r
length(stock_num_0144)
```

```
## [1] 492
```

```r
#492 trading days data
```

2.

```r
pmatrix=matrix(c(c(as.numeric(stock_num_0151[,1])),
       c(as.numeric(stock_num_0144[,1])),
       c(as.numeric(stock_num_0267[,1])),
       c(as.numeric(stock_num_0388[,1])),
       c(as.numeric(stock_num_0941[,1])),
       c(as.numeric(stock_num_2319[,1])),
       c(as.numeric(stock_num_0494[,1])),
       c(as.numeric(stock_num_0151[,1]))),
       ncol=8,byrow=F)
head(pmatrix)
```

```
##          [,1]     [,2]      [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
## [1,] 6.111838 16.98554  9.972425 231.9618 68.44915 22.56340 3.538109 6.111838
## [2,] 6.007282 17.14925  9.954867 231.0347 67.54224 22.95580 3.554878 6.007282
## [3,] 5.940745 17.19019  9.989983 236.0411 67.49907 22.02383 3.487804 5.940745
## [4,] 6.083323 17.06740 10.007538 234.7432 67.41269 21.97478 3.487804 6.083323
## [5,] 6.054807 17.06740 10.165551 242.9017 67.36952 22.95580 3.588414 6.054807
## [6,] 6.178375 16.94461 10.112882 249.3914 67.28314 22.95580 3.605183 6.178375
```

```r
#matrix of the closing price of eight stock from 2018-01-01 to 2019-12-31
#also, as the matrix is sucessfully constructed,the length of the different stock vector is same.
price20191231=pmatrix[492,]
#vector of closing price of my eight stock
v0=5000*sum(price20191231)
#the value of my portfolio based on the closing price of the last day, 31/12/2019
v0
```

```
## [1] 1889874
```

3.

```r
t1=as.ts(pmatrix[,1])
t2=as.ts(pmatrix[,2])
t3=as.ts(pmatrix[,3])
t4=as.ts(pmatrix[,4])
t5=as.ts(pmatrix[,5])
t6=as.ts(pmatrix[,6])
t7=as.ts(pmatrix[,7])
t8=as.ts(pmatrix[,8])
u1=(lag(t1)-t1)/t1
u2=(lag(t2)-t2)/t2
u3=(lag(t3)-t3)/t3
```

```r
u4=(lag(t4)-t4)/t4
u5=(lag(t5)-t5)/t5
u6=(lag(t6)-t6)/t6
u7=(lag(t7)-t7)/t7
u8=(lag(t8)-t8)/t8
u=cbind(u1,u2,u3,u4,u5,u6,u7,u8)
n2=nrow(u) #no. of row in u
n1=n2-60+1 #60-th obs before n2
u60=u[n1:n2,]
set.seed(27616)
mu=apply(u60,2,mean) #compute column mean of u60
sigma=var(u60) #compute daily variance rate of u60
try(chol(sigma))
```

```
## Error in chol.default(sigma) :
##   the leading minor of order 8 is not positive definite
```

```r
#failed due to the semi-positive definite matrix
C=chol(sigma,pivot=T) #using pivot
```

```
## Warning in chol.default(sigma, pivot = T): the matrix is either rank-deficient
## or indefinite
```

```r
s=cbind(t1,t2,t3,t4,t5,t6,t7,t8)
s0=s[nrow(s),]
#vvvvv 1000 looping of simulation vvvvv#
port.v=rep(0,1000) #to store simulated portfolio
s1000=matrix(0,1000,ncol=8) #to store the simulated price
for (i in 1:1000){
  s0=s[nrow(pmatrix),] #Every simulation starts at the latest price
for (j in 1:10){# simulate price for future 10 days
z=rnorm(8) # generate normal random vector
v=mu+t(C)%*%z # transform to multivariate normal
s1=s0*(1+v) # new stock price
s0=s1 #update s0 with s1
}
s1000[i,]=s0
port.v[i]=5000*sum(s0)
}
```

4.

```r
port.v[1000]-v0 #profit and loss of last simulated result
```

```
## [1] 108341.7
```

```r
#or
sum(s1000[1000,])*5000-v0 #profit and loss of last simulated result
```

```
## [1] 108341.7
```

5.

```
pnl=port.v-v0
min(pnl)
```

```
## [1] -175972.3
```

```
max(pnl)
```

```
## [1] 236223.8
```

```
mean(pnl)
```

```
## [1] 27352.67
```

```
median(pnl)
```

```
## [1] 26585.17
```

```
sd(pnl)
```

```
## [1] 60653.34
```

```
quantile(pnl,0.01)
```

```
##        1%
## -106849.3
```

```
quantile(pnl,0.05)
```

```
##        5%
## -72370.47
```

1(b)

I choose 3M(MMM), American Express(AXP), Apple(AAPL), JPMorgan Chase(JPM), Coca-Cola(KO), Goldman Sachs(GS), Intel(INTC) and IBM(IBM) as eight of my shocks.

1

```
stocklist_1b=c("MMM","AXP","AAPL","JPM","KO","GS","INTC","IBM")
stock_1b=list()
for (i in 1:8){
stock_1b[[i]]=get.hist.quote(instrument = stocklist_1b[i],
start="2018-01-01", end="2020-01-01",quote = c("Adjusted"),
provider = c("yahoo"), compression = "d",
retclass = c("zoo"))
}
```

```
## time series starts 2018-01-02
## time series ends   2019-12-31
## time series starts 2018-01-02
## time series ends   2019-12-31
## time series starts 2018-01-02
## time series ends   2019-12-31
## time series starts 2018-01-02
## time series ends   2019-12-31
## time series starts 2018-01-02
## time series ends   2019-12-31
## time series starts 2018-01-02
## time series ends   2019-12-31
## time series starts 2018-01-02
## time series ends   2019-12-31
## time series starts 2018-01-02
## time series ends   2019-12-31
```

2

```
pmatrix_1b=matrix(c(as.numeric(stock_1b[[i]][,1]),
                    as.numeric(stock_1b[[2]][,1]),
                    as.numeric(stock_1b[[3]][,1]),
                    as.numeric(stock_1b[[4]][,1]),
                    as.numeric(stock_1b[[5]][,1]),
                    as.numeric(stock_1b[[6]][,1]),
                    as.numeric(stock_1b[[7]][,1]),
                    as.numeric(stock_1b[[8]][,1])),
                    byrow=F,ncol=8)
#matrix of the closing price of eight stock from 2018-01-01 to 2019-12-31
#also, as the matrix is sucessfully constructed,the length of the different stock vector is same.
price20191231_1b=pmatrix_1b[492,]
#vector of closing price of my eight stock
v0_1b=5000*sum(price20191231_1b)
#the value of my portfolio based on the closing price of the last day, 31/12/2019
v0_1b
```

```
## [1] 4561716
```

3

```
t1_1b=as.ts(pmatrix_1b[,1])
t2_1b=as.ts(pmatrix_1b[,2])
t3_1b=as.ts(pmatrix_1b[,3])
t4_1b=as.ts(pmatrix_1b[,4])
t5_1b=as.ts(pmatrix_1b[,5])
t6_1b=as.ts(pmatrix_1b[,6])
t7_1b=as.ts(pmatrix_1b[,7])
t8_1b=as.ts(pmatrix_1b[,8])
u1_1b=(lag(t1_1b)-t1_1b)/t1_1b
u2_1b=(lag(t2_1b)-t2_1b)/t2_1b
u3_1b=(lag(t3_1b)-t3_1b)/t3_1b
u4_1b=(lag(t4_1b)-t4_1b)/t4_1b
u5_1b=(lag(t5_1b)-t5_1b)/t5_1b
```

```r
u6_1b=(lag(t6_1b)-t6_1b)/t6_1b
u7_1b=(lag(t7_1b)-t7_1b)/t7_1b
u8_1b=(lag(t8_1b)-t8_1b)/t8_1b
u_1b=cbind(u1_1b,u2_1b,u3_1b,u4_1b,u5_1b,u6_1b,u7_1b,u8_1b)
n2_1b=nrow(u_1b) #no. of row in u
n1_1b=n2_1b-60+1 #60-th obs before n2
u60_1b=u_1b[(n1_1b):(n2_1b),]
set.seed(27616)
mu_1b=apply(u60_1b,2,mean) #compute column mean of u60
sigma_1b=var(u60_1b) #compute daily variance rate of u60
try(chol(sigma_1b))
```

```
## Error in chol.default(sigma_1b) :
##   the leading minor of order 8 is not positive definite
```

```r
#failed due to the semi-positive definite matrix
C_1b=chol(sigma_1b,pivot=T) #using pivot
```

```
## Warning in chol.default(sigma_1b, pivot = T): the matrix is either rank-
## deficient or indefinite
```

```r
s_1b=cbind(t1_1b,t2_1b,t3_1b,t4_1b,t5_1b,t6_1b,t7_1b,t8_1b)
s0_1b=s_1b[nrow(s_1b),]
#vvvvv 1000 looping of simulation vvvvv#
port.v_1b=rep(0,1000) #to store simulated portfolio
s1000_1b=matrix(0,1000,ncol=8) #to store the simulated price
for (i in 1:1000){
  s0_1b=s_1b[nrow(pmatrix_1b),] #Every simulation starts at the latest price
for (j in 1:10){# simulate price for future 10 days
z_1b=rnorm(8) # generate normal random vector
v_1b=mu+t(C_1b)%*%z_1b # transform to multivariate normal
s1_1b=s0_1b*(1+v_1b) # new stock price
s0_1b=s1_1b #update s0_1b with s1_1b
}
s1000_1b[i,]=s0_1b
port.v_1b[i]=5000*sum(s0_1b)
}
```

4

```r
port.v_1b[1000]-v0_1b #profit and loss of last simulated result
```

```
## [1] 279031.1
```

```r
#or
sum(s1000_1b[1000,])*5000-v0_1b #profit and loss of last simulated result
```

```
## [1] 279031.1
```

5

```r
pnl_1b=port.v_1b-v0_1b
min(pnl_1b)
```

```
## [1] -188527
```

```r
max(pnl_1b)
```

```
## [1] 503749.4
```

```r
mean(pnl_1b)
```

```
## [1] 145931.9
```

```r
median(pnl_1b)
```

```
## [1] 145457.4
```

```r
sd(pnl_1b)
```

```
## [1] 89348.07
```

```r
quantile(pnl_1b,0.01)
```

```
##        1%
## -69620.75
```

```r
quantile(pnl_1b,0.05)
```

```
##        5%
## -989.4879
```

2(a)

1

in this case, we want to minimize:

$$\sum_{i=1}^{n-19} |\sigma_i - s_i|$$

```r
pmatrix=matrix(c(c(as.numeric(stock_num_0151[,1])),
       c(as.numeric(stock_num_0144[,1])),
       c(as.numeric(stock_num_0267[,1])),
       c(as.numeric(stock_num_0388[,1])),
       c(as.numeric(stock_num_0941[,1])),
       c(as.numeric(stock_num_2319[,1])),
       c(as.numeric(stock_num_0494[,1])),
       c(as.numeric(stock_num_0151[,1]))),
       ncol=8,byrow=F)
head(pmatrix)
```

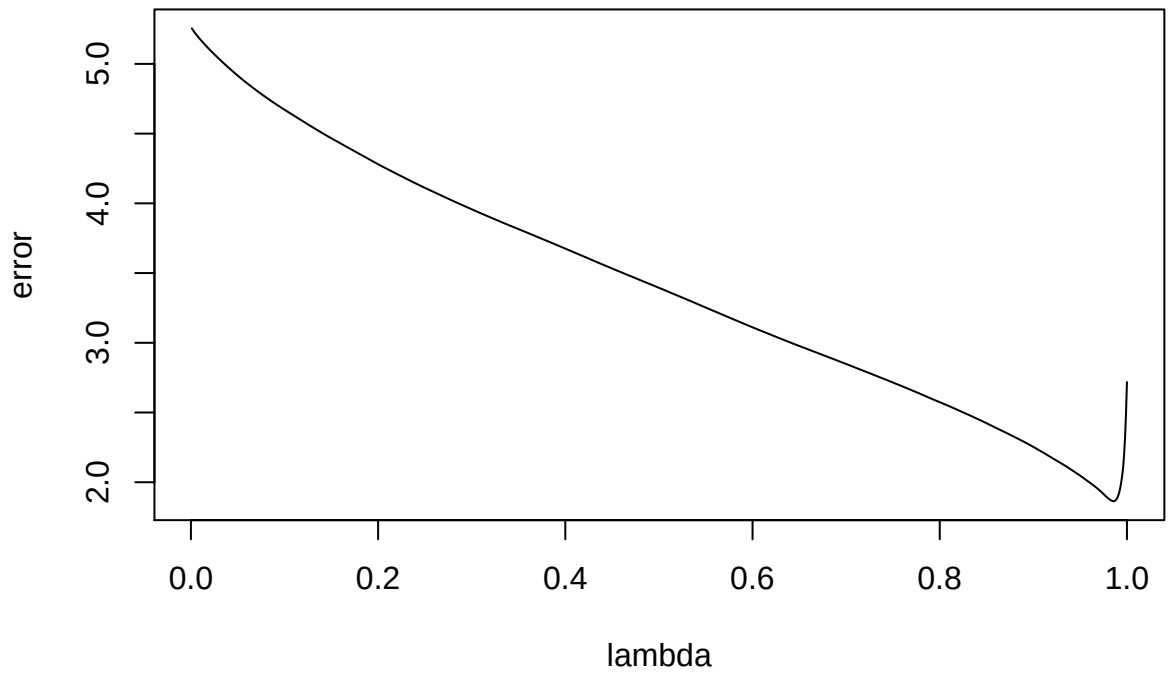
```
##          [,1]     [,2]      [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
## [1,] 6.111838 16.98554  9.972425 231.9618 68.44915 22.56340 3.538109 6.111838
## [2,] 6.007282 17.14925  9.954867 231.0347 67.54224 22.95580 3.554878 6.007282
## [3,] 5.940745 17.19019  9.989983 236.0411 67.49907 22.02383 3.487804 5.940745
## [4,] 6.083323 17.06740 10.007538 234.7432 67.41269 21.97478 3.487804 6.083323
## [5,] 6.054807 17.06740 10.165551 242.9017 67.36952 22.95580 3.588414 6.054807
## [6,] 6.178375 16.94461 10.112882 249.3914 67.28314 22.95580 3.605183 6.178375
```

```
t1=as.ts(pmatrix[,1]) # first stock in Q1.(a)
u1=(lag(t1)-t1)/t1
msd<-function(t,w) { # function to compute moving s.d.
n<-length(t)-w+1
out<-c() # initialize a null vector to store the output
for (i in 1:n) {
j<-i+w-1
s<-sd(window(t,i,j)) # compute the sd of t(i) to t(j)
out<-append(out,s) # append the result to out
}
out<-as.ts(out) # convert to time series
}
s1_20<-msd(u1,20)
#calculate the moving sd of relative change from i=n to i=n+19
#where n=1,2,...,length of u1
sigma_i=c() #save the value of sigma
sigma_i[1]=sd(u1[1:30]) #taking first 30 day return as the inital sd
  out=c()
#write a function to calculate the error terms with different lambda
errfun=function(lambda,x){
  for (i in 2:(length(x)-19)){
  sigma_i[i]=sqrt(lambda*sigma_i[i-1]^2+(1-lambda)*(x[i-1])^2)
  }
  append(out,sum(abs(sigma_i-s1_20)))
}
```

```
#gen lambda from 0 to 1 with d=0.001 and save it as error vector
error=c()
d=0.001
return=u1
for (i in seq(d,1,d)){
error=append(error,errfun(i,return))
}
#finding which lambda value give the minimum error
lambda_min=d*which(error==min(error))
lambda_min
```

```
## [1] 0.985
```

```
plot(error~seq(d,1,d),xlab="lambda",type="l")
```




```
#or using nlminb function
nlminb(0.4,errfun,x=u1)$par #approximatly 0.985
```

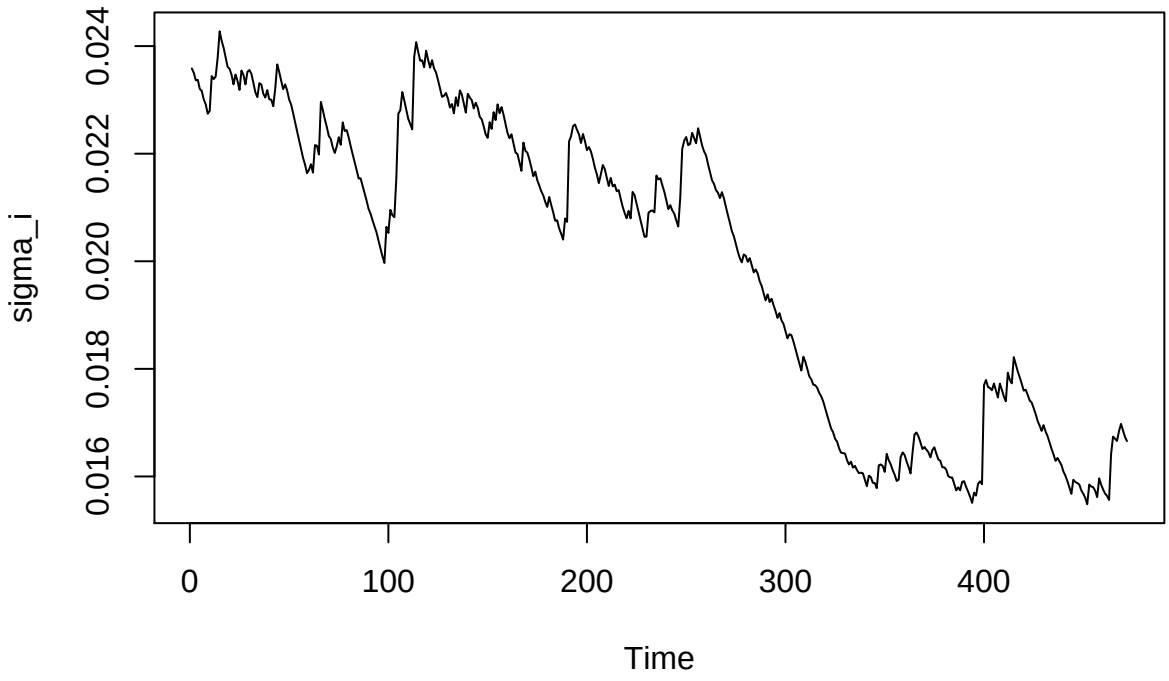
```
## [1] 0.9853057
```

Hence,we have lambda=0.985 minimizes the sum of the absolute error terms:

$$\sum_{i=1}^{n-19} |\sigma_i - s_i|$$

2

```
sigma_i=c()
sigma_i[1]=sd(u1[1:30])
lambda=lambda_min
  for (i in 2:(length(u1)-19)){
  sigma_i[i]=sqrt(lambda*sigma_i[i-1]^2+(1-lambda)*(u1[i-1])^2)
  }
ts.plot(sigma_i)
```



3

```
errfun(lambda_min,u1) #sum of absolute error if lambda=0.968
```

```
## [1] 1.863665
```

```
errfun(0.9,u1) #sum of absolute error if lambda=0.9
```

```
## [1] 2.253471
```

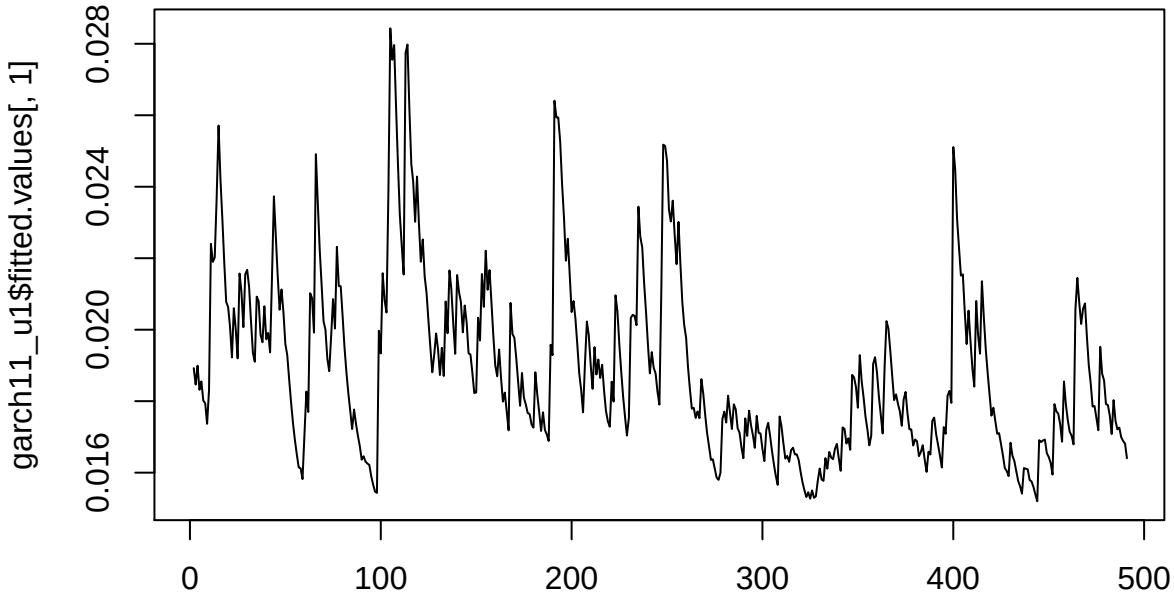
```
#the difference of error with two different lambda
abs(errfun(lambda_min,u1)-errfun(0.9,u1))
```

```
## [1] 0.3898058
```

2(b)

```
library(tseries)
garch11_u1<-garch(u1,order=c(1,1)) #fit the garch(1,1)
```

```
matplot(garch11_u1$fitted.values[,1],type='l')
```



3

(a)

```
mean(u1)
```

```
## [1] 0.0005063503
```

```
median(u1)
```

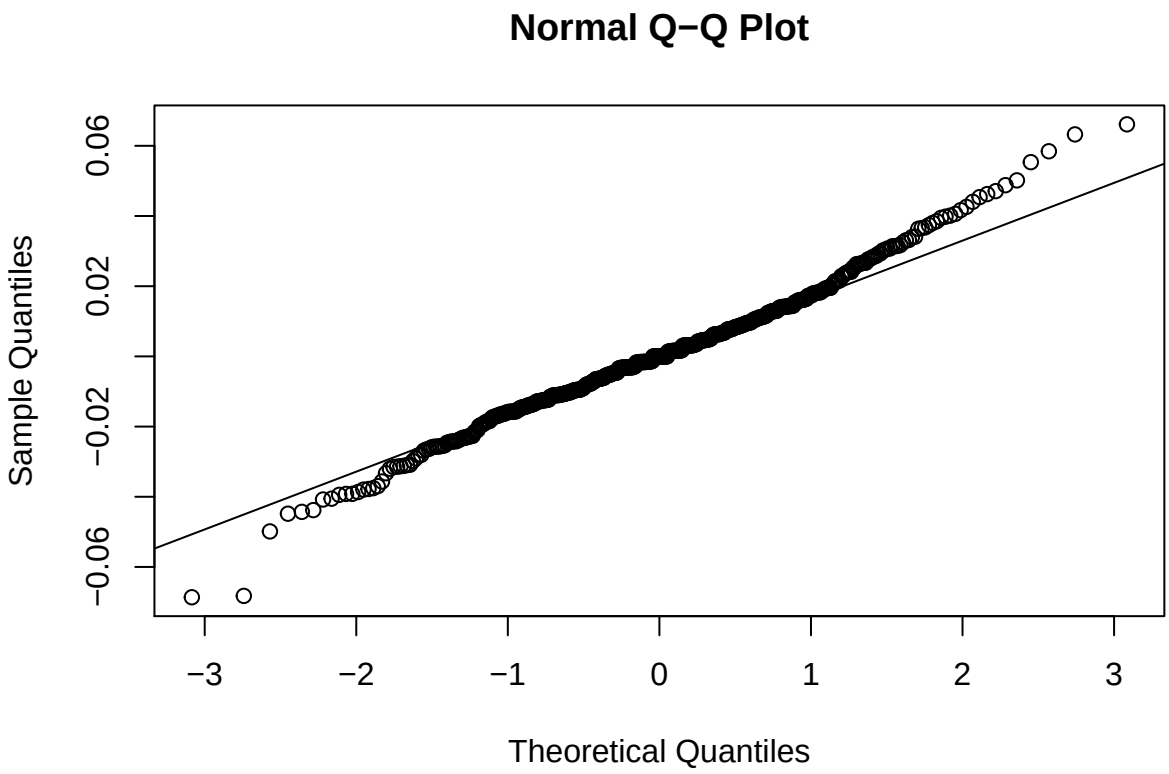
```
## [1] 0
```

```
sd(u1)
```

```
## [1] 0.01904476
```

(b)

```
qqnorm(u1)
qqline(u1)
```



The returns do not look normally distributed as the two tailed are quite differ form the normal quantile line, which is so called the "fat tailed distribution". The reason behind that maybe there is additional risk that may rise the actual loss compared to the loss under normal assumption. Also, higher risk imply higher profit and hence the actual profit is higher than the profit under normal assumption.

(c)

```
#test whether or not u1 is following normal distribution
ks.test(u1,rnorm)
```

```
## Warning in ks.test(u1, rnorm): ties should not be present for the Kolmogorov-
## Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  u1
## D = 4.3781, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
#p-value close to 0(p-value<0.05) hence the normality assumption is rejected
JB.test<-function(u) { # function for JB-test
n<-length(u) # sample size
s<-sd(u) # compute sd
sk<-sum(u^3)/(n*s^3) # compute skewness
```

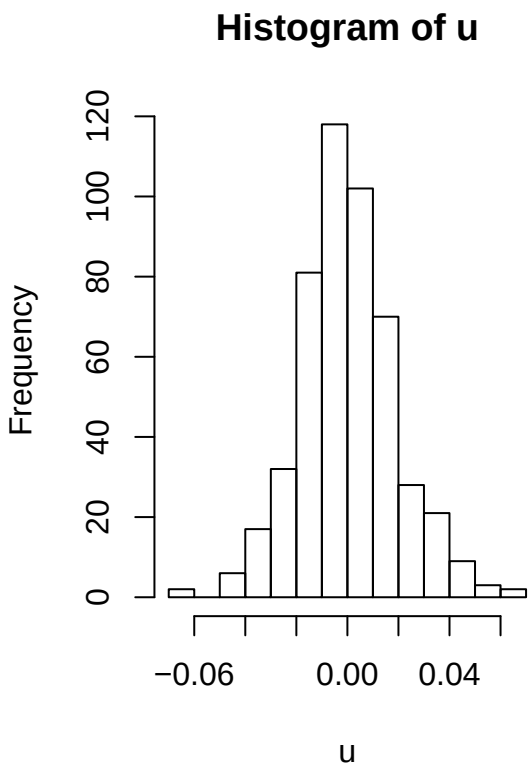
```
ku<-sum(u^4)/(n*s^4)-3 # excess kurtosis
JB<-n*(sk^2/6+ku^2/24) # JB test stat
p<-1-pchisq(JB,2) # p-value
cat("JB-stat:",JB," p-value:",p,"\n") # output
}
JB.test(u1)
```

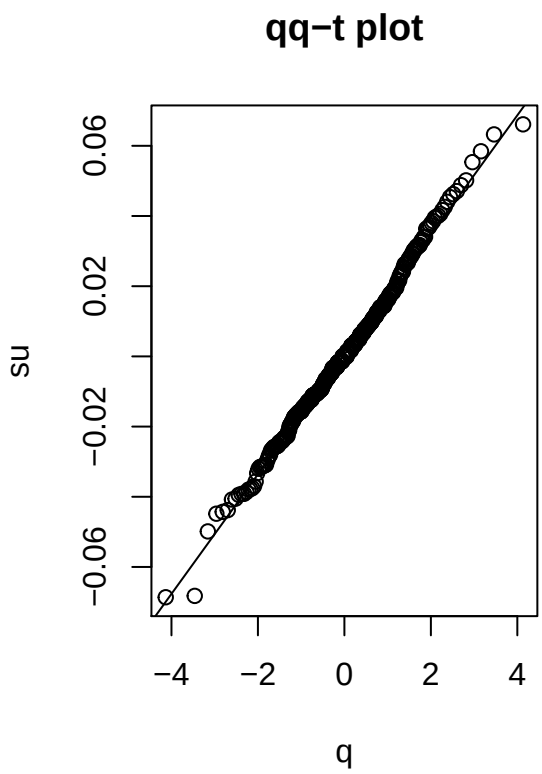
```
## JB-stat: 21.43968  p-value: 2.210202e-05
```

```
#p-value close to 0(p-value<0.05) hence the normality assumption is rejected
```

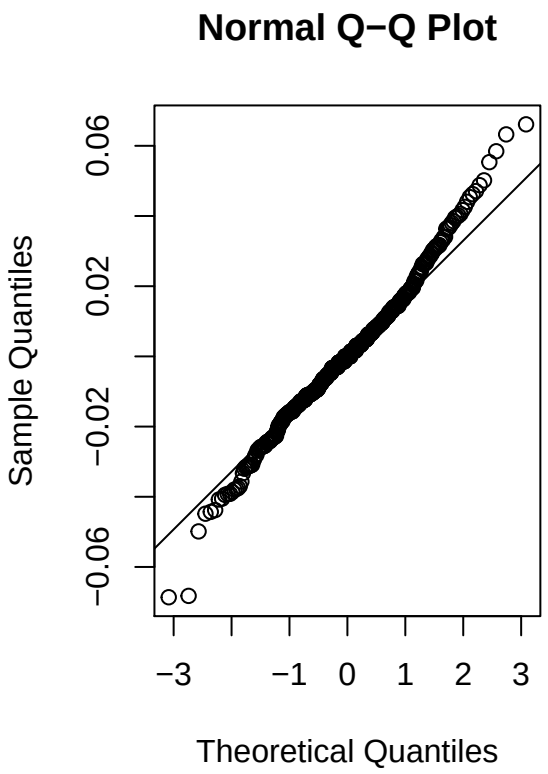
(d)

```
QQt.plot<-function(u) { # function for QQ-t plot
su<-sort(u) # sort u
n<-length(u) # sample size
s<-sd(u) # sd
ku<-sum(u^4)/(n*s^4)-3 # excess kurtosis
v<-round(6/ku+4) # estimate df, round to the nearest integer
i<-((1:n)-0.5)/n # create a vector of percentile
q<-qt(i,v) # percentile point from t(v)
hist(u) # histogram of u
plot(q,su,main="qq-t plot") # plot(q,su)
abline(lsfit(q,su)) # add reference line
v # output degree of freedom
}
par(mfrow=(c(1,2)))
u1_t=QQt.plot(u1)
```






```
#degree of freedom estimated is 10
qqnorm(u1)
qqline(u1)
```



Clearly, the t distribution with df=10 is more fitted to the return data especially in the two tailed (the line cross most of the dots).Which is because t distribution has fatter tails when df is small(typically <30) compared to the normal distribution.

(e)

```
ks.test(u1,pt,df=u1_t)
```

```
## Warning in ks.test(u1, pt, df = u1_t): ties should not be present for the
## Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  u1
## D = 0.47653, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

from the ks.test above which testing whether or not the return follows t distribution with df=10, and it is rejected as the p-value is still very small, which may imply that normal and t are not doing well in modeling the return as extreme return values are more likely at market downturns, and they are all positively skewed, while t-distribution does not. However, the t distribution with df=10 is more fitted to the return data especially in the two tailed (the line cross most of the dots).

4

(a)

ARCH(1)

(b)

AR(1)

(c)

0

(d)

$(0.3)^h$

(e)

$(0.6)^h$