STA457 Time Series Analysis
Assignment I

ChuanQi Sun(Team Lead) 1001369404
Guanlin Song(Team Member) 1003597930

Part A:

Question1:

In part A, we must find the optimal double moving average (MA) trading rules for all 30 DJ constituents stocks using monthly data. First, quantmod package is required to be installed. The data for the DJ constituents were downloaded from Yahoo finance. Between 1999-12-31 until 2018-12-31. Formulas are derived as below:

$$var(F_t) = var(\Sigma_{i=0}^{m-2} d_i X_{t-i}) = cov(\Sigma_{j=0}^{m-2} d_j X_{t-j}, \Sigma_{j=0}^{m-2} d_j X_{t-i})$$

$$= [d_0 \dots . d_{m-2}] \begin{bmatrix} \gamma_0 & \cdots & \gamma_{m-2} \\ \vdots & \ddots & \vdots \\ \gamma_{m-2} & \cdots & \gamma_0 \end{bmatrix} \begin{bmatrix} d_0 \\ \dots \\ d_{m-2} \end{bmatrix}$$

$$= \Sigma_{i=0}^{m-2}\Sigma_{j=0}^{m-2} d_i d_j \gamma_{i-j} = \Sigma_{i=0}^{m-2} d_i^2 \gamma_0 + \Sigma_{i\neq j} d_i d_j \gamma_{i-j}$$

$$= \Sigma_{j=0}^{m-2} d_j^2 \gamma_0 + 2\Sigma_{j=0}^{m-2} d_j d_{j+1} \gamma_1 + 2\Sigma_{j=0}^{m-3} d_j d_{j+2} \gamma_2 + \cdots + 2\Sigma_{j=0}^{0} d_j d_{j+m-2} \gamma_{m-2}$$

$$= \gamma_0 \left(\Sigma_{j=0}^{m-2} d_j^2 + 2\Sigma_{j=0}^{m-2}\Sigma_{j=0}^{\mathrm{k}} d_j d_{j+m-2-k\rho(m-2)-k}\right)$$

$$\mu_F = E(F_t) = E\left(\Sigma_{j=0}^{m-2} d_j X_{t-j}\right) = \mu_x * \Sigma_{j=0}^{m-2} d_j$$

Expected rule return: $E(R_t) = \sqrt{\frac{2}{\pi}}\sigma_X corr(X_t, F_{t-1}) e^{\frac{-\mu_F^2}{2\sigma_F^2}} + \mu_X \left(1 - 2\phi\left[\frac{-\mu_F}{\sigma_F}\right]\right)$

Where $\phi(h) = \int_{-\infty}^{h} \left(\sqrt{2\pi}\right)^{-1} \exp\{-x^2/2\}\, dx,\ \mu_x = E(X_t)$.

We also need the following results to compute the expectation of rule returns:

$$corr(X_t, F_{t-1}) = \frac{\gamma \sum_{i=0}^{m-2} d_i \rho_{i+1}}{\sqrt{\gamma var(F_t)}}$$

Using last year's code for daily trading rule, we obtain strategy for monthly data moving average trading rule. After downloading data, we use function to obtain optimal monthly MA trading rules which means

that these are such that maximizes the expected rule returns. Data is then summarized in a matrix with stock name and its m and r.

```
> #A
> #1
> #install.packages("quantmod")
> library(quantmod)
Loading required package: xts
Loading required package: zoo

Attaching package: ‘zoo’

The following objects are masked from ‘package:base’:

    as.Date, as.Date.numeric

Loading required package: TTR
Version 0.4-0 included new data defaults. See ?getSymbols.
Learn from a quantmod author: https://www.datacamp.com/courses/importing-and-
managing-financial-data-in-r
> stocks <-
c("MMM","AXP","AAPL","BA","CAT","CVX","CSCO","KO","DIS","DWDP","XOM","GS","HD","I
BM","INTC",
+
"JNJ","JPM","MCD","MRK","MSFT","NKE","PFE","PG","TRV","UTX","UNH","VZ","V","WMT",
"WBA")
> dj30 = new.env()
> getSymbols(stocks,src="yahoo", from="1999-12-31", to="2018-12-31")
‘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.


WARNING: There have been significant changes to Yahoo Finance data.
Please see the Warning section of ‘?getSymbols.yahoo’ for details.

This message is shown once per session and may be disabled by setting
options("getSymbols.yahoo.warning"=FALSE).
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
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
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
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
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
 [1] "MMM"  "AXP"  "AAPL" "BA"   "CAT"  "CVX"  "CSCO" "KO"   "DIS"  "DWDP" "XOM"
"GS"   "HD"   "IBM"  "INTC" "JNJ"  "JPM"  "MCD"  "MRK"  "MSFT" "NKE"  "PFE"  "PG"
"TRV"  "UTX"  "UNH"
[27] "VZ"   "V"    "WMT"  "WBA"
> #Loading functions:
> muF<-function(d,X){mean(X)*sum(d)}
> # Calculate the variance of forecaster using quadratic form
> # d: vector of dj coefficients (j=0, ..., m-2)
> # X: log returns
> varF<-function(d,X){
+   M<-length(d)-1
+   acfs<- acf(X, plot=F, type="covariance", lag.max=M)$acf
+   Gamma<-toeplitz(as.vector(acfs))
+   d%*%Gamma%*%as.vector(d)
+ }
> # Calculate ACF(1) of forecaster using matrix operation and outer function in r
> rhoF<-function(d,X){
+   M<-length(d)-1
+   acfs<- acf(X, plot = F, type = "covariance", lag.max=M+2)$acf#M+2
+   temp<-d%*%matrix(acfs[abs(outer(0:M,1:(M+1), "-")) +1,,1],
```

```
+                     M+1, M+1) %*% as.vector(d)
+   temp/varF(d,X)
+ }
> corXF<-function(d,X){
+   Mp<-length(d)
+   acfs<- acf(X, plot=F, type= "covariance", lag.max=Mp)$acf
+   sum(d*acfs[-1])/sqrt(acfs[1]*varF(d,X))
+ }
> Hold<-function(rho){pi/acos(rho)}
> # m > r >=1
> d<-function(m,r){ c((m-r)*((0:(r-1))+1), r*(m-(r: (m-1))-1))}
> # retX: log asset return
> # m: long-term MA
> # r: short-term MA
> ruleReturn<-function(retX, m, r){
+   vX<-sd(retX)
+   mX<-mean(retX)
+   mF<-muF(d(m,r),retX)
+   vF<-sqrt(varF(d(m,r),retX))
+   rXF<-corXF(d(m,r),retX)
+   rF<-rhoF(d(m,r),retX)
+   ER<-sqrt(2/pi)*vX*rXF*exp(-mF*mF/(2*vF*vF))+mX*(1-2*pnorm(-mF/vF))
+   H<-Hold(rF)
+   list("ER"=ER, "H"=H, "rhoF"=rF, "VF"=vF, "muF"=mF, "corXF"=rXF)
+ }
> all_mr_double_ma <-NULL
> #Q6 from 2018 assignment
> for(stock in stocks) {
+   #stockAdjusted = dj30[[stock]][,paste(stock, ".Adjusted",sep="")]
+   #monthlyData = coredata(diff(log(apply.monthly(stockAdjusted, last))))
+   #monthlyData=na.omit(monthlyData)
+   #https://www.rdocumentation.org/packages/quantmod/versions/0.4-
13/topics/periodReturn
+   monthlyData <- monthlyReturn(get(stock), type="log")
+
+   result<- numeric(0)
+   m <- numeric(0)
+   r <- numeric(0)
+   for (i in 2:11){
+     for(j in (i+1):12){
+       if (j>i){
+         result <- c(result, ruleReturn(monthlyData, m = j, r = i)[[1]])
+         m <- c(m,j)
+         r <- c(r,i)
+       }
```

```
+     }
+   }
+   m_optimal <- m[which.max(result)]
+   r_optimal <- r[which.max(result)]
+   #list(optimal_m = m_optimal, optimal_r = r_optimal)
+
+   #collect all m and r for double ma rule
+   all_mr_double_ma <- rbind(all_mr_double_ma, c(m_optimal, r_optimal))
+
+ }
> #add title for the result
> row.names(all_mr_double_ma) <- stocks
> colnames(all_mr_double_ma) <- c("m", "r")
> all_mr_double_ma
      m  r
MMM   9  8
AXP   5  4
AAPL  3  2
BA    9  8
CAT   5  4
CVX   9  6
CSCO  7  6
KO    7  6
DIS   5  4
DWDP  4  3
XOM  12 11
GS    4  3
HD   11 10
IBM  12 11
INTC  4  3
JNJ   9  8
JPM  12 11
MCD   7  5
MRK   9  7
MSFT  9  8
NKE   9  7
PFE   9  8
PG    5  3
TRV   9  8
UTX   5  4
UNH   9  8
VZ    9  8
V     5  4
WMT   9  8
WBA   6  5
```

Question2:

In this question, we aim to construct the equally weighted and risk-parity weighted portfolios using the 30 DJ constituents and summarize the performance of such EW and RP portfolio (expected returns, volatility, and Sharpe ratio). We use the 60-months window to obtain optimal trading rules. Using formula provided for calculating Sharpe ratio,

$$S_r = \frac{12 \times \bar{R} - r_f}{\sqrt{12} \times \hat{\sigma}_R}$$

As we are calculating in annulization, the time period would be 12 months.

We assume that the risk-free rate or treasury bond is 0.02 and that $\bar{R}$ is the monthly strategy returns. we get information for annualized expected return and annualized volatility.

Code and Result:

```
> #2)
> #cumulate returns
> returns_last5 <- NULL
> #In this question, I collect data from 5 years, 60 month period
> getSymbols(stocks, src="yahoo", from="2014-01-01", to="2018-12-31")
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
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
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
```

```
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
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] "MMM"  "AXP"  "AAPL" "BA"   "CAT"  "CVX"  "CSCO" "KO"   "DIS"  "DWDP" "XOM"
"GS"   "HD"   "IBM"  "INTC" "JNJ"  "JPM"  "MCD"  "MRK"  "MSFT" "NKE"  "PFE"  "PG"
"TRV"  "UTX"  "UNH"
[27] "VZ"   "V"    "WMT"  "WBA"
> for(stock in stocks){
+   #stockAdjusted = dj30[[stock]][,paste(stock, ".Adjusted",sep="")]
+   #monthlyData = coredata(diff(log(apply.monthly(stockAdjusted, last))))
+   #monthlyData=na.omit(monthlyData)
+   #na_fill = rep(NA,228-length(monthlyData))
+   #monthlyData=c(na_fill, monthlyData)
+   #monthlyData <- monthlyReturn(dj30[[stock]][,paste(stock,
".Adjusted",sep="")], type="log")
+   monthlyData<- monthlyReturn(get(stock), type="log")
+   returns_last5 <- cbind(returns_last5, monthlyData)
+ }
> #R^(EW)_1 = 1/30 \Sigma_i B_(i,t-1) *  r_(i,t)
> #The formula of R_t^EW expression, the average of sum rulereturn
> EW <- numeric(nrow(returns_last5))
> for (i in 1:nrow(returns_last5)){
+   #remove the na, for any exception.
+   EW[i] <- mean(returns_last5[i,], na.rm=TRUE)
+ }
> #show the result of EW
> mean(EW)
[1] 0.0057565
> var(EW)
[1] 0.001016225
> #R_t^(RP) = \Sigma _i ^30 w_(i,t-1) * B_(i,t-1) *  r_(i,t)
> #The formula of R_t^RP expression, the average of weighted ruleReturn
> RP <- numeric(nrow(returns_last5))
> #standard derivation of returns
> std_devs=apply(returns_last5, 2, sd, na.rm=TRUE)
> #The thing before B
> weights <- (1/std_devs)/(sum(1/std_devs))
> for(i in 1:nrow(returns_last5)){
+   RP[i] <- sum(returns_last5[i,] * weights, na.rm=TRUE)
```

```
+ }
> #Show the result of RP
> mean(RP)
[1] 0.005758299
> var(RP)
[1] 0.0009601758
> #Show performance: Sharpe ratio for Each of ew and rp
> perf_ew1 = ((12 * mean(EW) - 0.02) / (sqrt(12) * sqrt(var(EW))))
> perf_rp1 = ((12 * mean(RP) - 0.02) / (sqrt(12) * sqrt(var(RP))))
> perf_ew1
[1] 0.4444278
> perf_rp1
[1] 0.4574165
```

OBSERVATION AND EXPLANATION:

Background knowledge: The total risk of a portfolio is typically measured by its volatility of the return of the portfolio.

Sharpe ratio is measured by using the formula above. performance of equally weighted portfolio has a value of 0.444 which means that its Sharpe Ratio is 0.444, which is greater than delta= 0.2;

Furthermore, the risk parity portfolio has less volatility than the equally weighted portfolio. Due to the fact that risk parity portfolio is well-diversified, and it has low idiosyncratic risk than equally weighted portfolio where market risk still exists. this is expected as risk parity seeks equal risk exposure from all assets in the portfolio, therefore, less weight is allocated to riskier assets in comparison to equally weighted portfolio where equal weight of each asset is required.

Part B:

Question1:

In this part, we will compute the volatility estimate for the DJ constituents. We used the formula to calculate delta. However, the solution is zero, which does not make sense. According to the discussion board, we manually choose delta as 0.2.

$$\sum_{i=0}^{260}(1-\delta)\delta^{i}=1$$

The R code is implemented as follows to compute sigmat.

$$\sigma_{s,t}^{2}=12\sum_{i=0}^{11}(1-\delta)\delta^{i}\left(r_{s,t-1-i}-\bar{r}_{s,t}\right)^{2},$$

Furthermore, we have rule return:

$$\bar{r}_{s,t}=\sum_{i=0}^{11}(1-\delta)\delta^{i}\,r_{s,t-1-i}.$$

Thus, we get sigmat from above formula.

```
> #PART B QUESTION 1:
>   #find delta with the formula provided:
> #\Sigma(1-\delta)\delta^i=1
>   f <- function(s) {
+     sum((1 - s)* s^{0:260}) -1
+   }
> #Solve the delta
> res <- optim(0, f, lower = 0)
Warning message:
In optim(0, f, lower = 0) :
  bounds can only be used with method L-BFGS-B (or Brent)
> delta <- res$par
> delta
```

```
[1] 0
> #Restore the data
> stocks <-
c("MMM","AXP","AAPL","BA","CAT","CVX","CSCO","KO","DIS","DWDP","XOM","GS","HD","I
BM","INTC",
+
"JNJ","JPM","MCD","MRK","MSFT","NKE","PFE","PG","TRV","UTX","UNH","VZ","V","WMT",
"WBA")
> dj30_last5 = new.env()
> #In this question, I use the data for last 5 years
> getSymbols(stocks, src="yahoo", from="2014-01-01", to="2018-12-31")#last 5
years
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
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
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
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
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
 [1] "MMM"  "AXP"  "AAPL" "BA"   "CAT"  "CVX"  "CSCO" "KO"   "DIS"  "DWDP" "XOM"
"GS"   "HD"   "IBM"  "INTC" "JNJ"  "JPM"  "MCD"  "MRK"  "MSFT" "NKE"  "PFE"  "PG"
"TRV"  "UTX"  "UNH"
[27] "VZ"   "V"    "WMT"  "WBA"
> returns_last5=c()
> sigmat <- c()
> #Load the monthly return data to return matrix. Yes, it's a return
```

```
> for(stock in stocks){
+   #stockAdjusted = dj30_last5[[stock]][,paste(stock, ".Adjusted",sep="")]
+   #monthlyData = coredata(diff(log(apply.monthly(stockAdjusted, last))))
+   #monthlyData=na.omit(monthlyData)
+   monthlyData <- monthlyReturn(get(stock), type="log")
+   returns_last5 <- cbind(returns_last5, monthlyData)
+ }
> delta<-0.2
> #for all 30 stocks, compute its sigmat
> for(i in 1:30) {#iterate all 30 stocks
+   the_square <- c()
+
+   for(t in 13:nrow(returns_last5)) {#13 or 14?
+     #change to 12
+
+     #As the discussion from professor, we don't follow the formula on handout
here, we use
+     #12 \Sigma_(i=0)^11 (1-\delta)\delta^i (r_(t-i-1)-\bar(r))^2
+     #The r_t-i-1, the rule return for that period
+     r_t_i_1 = returns_last5[(t-1):(t-12),i]
+     #bar_r, the mean for all r
+     bar_r = sum((1 - delta)*delta^{0:11} * returns_last5[(t-1):(t-12),i])
+     #The square is \Sigma^2_(s,t)
+     the_square <- c(the_square, 12 * sum((1 - delta)*delta^{0:11}*( r_t_i_1 -
bar_r)^2))
+   }
+
+   #the square is a square, so we need square root!
+   sigmat <- cbind(sigmat, sqrt(the_square))
+ }
> sigmat
            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
[,7]       [,8]       [,9]      [,10]      [,11]      [,12]      [,13]
[,14]      [,15]      [,16]
 [1,] 0.16261825 0.17585574 0.21043740 0.14517148 0.01523878 0.19620310
0.02885421 0.132658204 0.194128752 0.054371192 0.17588531 0.117046816 0.16704516
0.14208696 0.09160817 0.102783125
 [2,] 0.05579610 0.11975083 0.04931825 0.07194976 0.02710043 0.02085088
0.04873131 0.027238767 0.158830573 0.091402687 0.04082451 0.043418925 0.13131718
0.03601459 0.04517609 0.033739787
 [3,] 0.02487146 0.04542645 0.09770000 0.08095153 0.05947566 0.04662082
0.02236352 0.055072070 0.041290132 0.044840859 0.04810687 0.024409117 0.05841409
0.06637620 0.02799874 0.056710245
```

 [4,] 0.01685702 0.10144313 0.05268879 0.06547741 0.11872107 0.09844561
0.04546948 0.072526575 0.084855622 0.035135869 0.08736657 0.054054788 0.01845639
0.10826656 0.05937410 0.041626857
 [5,] 0.03446436 0.07424900 0.05914266 0.14636404 0.12491972 0.10578941
0.07475032 0.066728796 0.055974430 0.076184360 0.02661208 0.061394504 0.03855645
0.09169247 0.13044147 0.048821069
 [6,] 0.03188159 0.14129156 0.02954456 0.07085724 0.17886698 0.10181303
0.01622909 0.141845724 0.030327292 0.037395967 0.02905422 0.020414501 0.08582890
0.09371106 0.07597393 0.095435927
 [7,] 0.05533431 0.12217875 0.06289211 0.13677905 0.19932817 0.04859903
0.03598330 0.177899433 0.056890823 0.073009939 0.04089689 0.009126937 0.18768789
0.09238974 0.09884359 0.107855980
 [8,] 0.06325854 0.05265918 0.11203232 0.07154734 0.22458741 0.10847794
0.04373585 0.066671707 0.070380709 0.105890469 0.08185212 0.019910148 0.21272693
0.09047032 0.05489107 0.021693163
 [9,] 0.12673139 0.07004508 0.12726176 0.05626870 0.15822095 0.11468206
0.08656931 0.064584856 0.047162228 0.051173710 0.10851728 0.030410352 0.10389024
0.16628228 0.06611367 0.029933693
[10,] 0.06538416 0.04848655 0.09701668 0.12278882 0.08992076 0.13371380
0.19094845 0.115019900 0.027713115 0.061004636 0.12031827 0.067757749 0.05601629
0.18068502 0.14757185 0.031969003
[11,] 0.03661114 0.10480496 0.21686066 0.13900049 0.13039751 0.16419867
0.17549553 0.168386411 0.045506572 0.070765684 0.11135349 0.088676070 0.05259284
0.02799668 0.18062316 0.055806723
[12,] 0.05212760 0.18960007 0.19143778 0.18566051 0.10211869 0.15311653
0.10570092 0.077941713 0.101714316 0.112348791 0.09820359 0.192471710 0.08397652
0.06258881 0.09064719 0.038261040
[13,] 0.06660558 0.20224184 0.07389989 0.12173902 0.22685613 0.17046014
0.21274260 0.101125655 0.215103038 0.124514271 0.08917389 0.275770685 0.12820183
0.12994634 0.13214647 0.085043229
[14,] 0.09168690 0.06980885 0.16269644 0.07202720 0.09781111 0.07924898
0.23264588 0.148401893 0.169995315 0.143517659 0.06988919 0.135955524 0.15672897
0.08375138 0.09278284 0.056806179
[15,] 0.04785700 0.06004231 0.06246035 0.05249469 0.14871789 0.10323863
0.15221765 0.092980325 0.037670323 0.112200950 0.08650920 0.074445719 0.07351463
0.09490401 0.14443178 0.019061049
[16,] 0.08957559 0.04979140 0.05529153 0.04412604 0.14040892 0.18089367
0.07669516 0.033035887 0.036247464 0.078649276 0.07431618 0.031713647 0.13257149
0.11312752 0.10734799 0.033904714
[17,] 0.06584282 0.07355870 0.10913597 0.04321850 0.04379504 0.02033656
0.10865796 0.073627070 0.059236639 0.081158163 0.01813918 0.069633096 0.05716826
0.04665284 0.23484403 0.052311709
[18,] 0.03097457 0.02135904 0.02738704 0.08820834 0.09791751 0.04108146
0.13133712 0.114549286 0.124344860 0.092536384 0.03924552 0.072334455 0.06974257
0.05890092 0.12440969 0.076084515

[19,] 0.06709915 0.04025615 0.06119311 0.17586775 0.09546329 0.07356064
0.16372456 0.114104374 0.271761259 0.084731398 0.04892092 0.094153204 0.07859110
0.11009447 0.08673979 0.118445247
[20,] 0.12244802 0.05738068 0.11032768 0.17884768 0.18980215 0.15683437
0.17165347 0.097448020 0.260221167 0.170239458 0.10545509 0.100589752 0.05065419
0.09659556 0.11376297 0.110508713
[21,] 0.13537906 0.04110856 0.13324599 0.15703176 0.34445076 0.21267505
0.11585582 0.044955541 0.135467342 0.291366679 0.15158181 0.204440810 0.10751009
0.02393989 0.08684244 0.108591141
[22,] 0.15713640 0.07118312 0.16387664 0.20504118 0.18106604 0.19684303
0.20428079 0.066178163 0.178321943 0.281281237 0.17396054 0.118546048 0.06233593
0.04529454 0.14470366 0.103474040
[23,] 0.04716478 0.14809358 0.13995985 0.10514318 0.09606215 0.04056075
0.09586107 0.009931164 0.105686975 0.136515520 0.04621834 0.110199437 0.13711169
0.06213233 0.09114175 0.017823150
[24,] 0.07181994 0.30533838 0.10209480 0.22944837 0.11124893 0.05464729
0.18154443 0.022166856 0.061451322 0.269181686 0.07488410 0.079449457 0.05446671
0.12602067 0.12596082 0.007121117
[25,] 0.05527867 0.43215945 0.14806684 0.26209196 0.24505608 0.10929463
0.30937212 0.046187841 0.136862344 0.460082919 0.04721008 0.109379863 0.08613171
0.22893953 0.13541291 0.015556846
[26,] 0.03748413 0.08077590 0.18973052 0.11886230 0.08011675 0.21957998
0.08150867 0.091659668 0.058726941 0.147682700 0.02389015 0.171119034 0.10835494
0.13677533 0.17561880 0.031170921
[27,] 0.07612053 0.08501980 0.34376543 0.08499717 0.17324409 0.12015745
0.15196217 0.141872707 0.050617488 0.048380795 0.03145066 0.057022239 0.10162842
0.23597598 0.19699698 0.021654513
[28,] 0.02172785 0.10993619 0.28024570 0.16026725 0.11292041 0.10475371
0.11660447 0.047786240 0.103622124 0.082482087 0.06849697 0.112403675 0.03749575
0.11496445 0.15008829 0.047172942
[29,] 0.04361562 0.11446431 0.13769106 0.13049045 0.16794968 0.06312722
0.08665087 0.034800251 0.034647925 0.063607265 0.07036522 0.086873715 0.06457022
0.08415222 0.01537312 0.088715106
[30,] 0.03795510 0.18111987 0.16767725 0.03785843 0.06842504 0.08081525
0.09622282 0.069916912 0.008285468 0.138808390 0.13650234 0.188289617 0.14416563
0.08817665 0.03410751 0.090729083
[31,] 0.03437669 0.07055623 0.08966019 0.07874702 0.12925738 0.03183219
0.05727337 0.043960587 0.013212555 0.114755913 0.05157904 0.073675477 0.15518636
0.09228170 0.05540658 0.103515872
[32,] 0.04993666 0.05762780 0.06079506 0.09100970 0.12614341 0.05991513
0.04685525 0.030259167 0.026050327 0.051729748 0.03843716 0.155070243 0.03301949
0.02712101 0.07410976 0.051119957
[33,] 0.07306883 0.09647866 0.08956208 0.07852879 0.18718855 0.02892038
0.05692301 0.041605124 0.056753117 0.098538582 0.07016932 0.238044652 0.06807308
0.05614047 0.16566898 0.025041626

```
          [,17]      [,18]      [,19]      [,20]      [,21]      [,22]
[,23]      [,24]      [,25]      [,26]      [,27]      [,28]      [,29]
[,30]
 [1,] 0.11839746 0.05921569 0.04535925 0.03432275 0.18093832 0.07324106
0.11820605 0.19022438 0.02259258 0.13986723 0.02262008 0.09869675 0.08172815
0.21431961
 [2,] 0.08663081 0.02819463 0.10134754 0.07332368 0.16958427 0.08626643
0.01514403 0.02955989 0.03751973 0.09458338 0.02454049 0.13194748 0.03884760
0.26086703
 [3,] 0.18881246 0.02435703 0.04516190 0.10722962 0.08754975 0.04581985
0.03171743 0.06114879 0.02440025 0.18775373 0.04859396 0.06975603 0.04780917
0.08698713
 [4,] 0.11534144 0.05337936 0.05783272 0.04162839 0.08312151 0.04096251
0.06441440 0.05795709 0.04857877 0.19629050 0.10800686 0.15554849 0.10621938
0.05176231
 [5,] 0.05579341 0.03934862 0.02476415 0.01725026 0.06556261 0.07383788
0.02798842 0.05677925 0.05089117 0.05723500 0.11328288 0.10290809 0.02802642
0.09849469
 [6,] 0.04517829 0.07204209 0.04475681 0.02749547 0.03871921 0.04791883
0.06254401 0.07958502 0.11369554 0.05383907 0.06450913 0.03510599 0.03197257
0.15907170
 [7,] 0.03789800 0.08112024 0.09793801 0.02535612 0.08500443 0.07587794
0.11280927 0.13680311 0.15736719 0.09946012 0.05495304 0.03516869 0.06278086
0.09149146
 [8,] 0.02918309 0.02845785 0.10063506 0.04688873 0.13835355 0.02366802
0.08363149 0.08275416 0.06203425 0.09492724 0.02357345 0.07314699 0.03960364
0.18613565
 [9,] 0.02634635 0.03493797 0.03994024 0.02593677 0.12053811 0.02194109
0.04627289 0.10135478 0.05512465 0.12973899 0.02348251 0.15369710 0.08080157
0.13822179
[10,] 0.04941314 0.05879132 0.08928739 0.05091106 0.06111031 0.03662009
0.03917303 0.06002090 0.02699169 0.08478966 0.05245933 0.10150964 0.17955699
0.03974968
[11,] 0.10909988 0.08605522 0.13328674 0.11191082 0.13572247 0.05569643
0.07583619 0.04716835 0.03057358 0.02106462 0.11064693 0.08491946 0.21543689
0.08235937
[12,] 0.23734815 0.06402221 0.15403566 0.16325609 0.05619816 0.05714939
0.10822254 0.05905176 0.06378899 0.04119662 0.11539155 0.06559199 0.01233401
0.17692078
[13,] 0.33552453 0.10300839 0.11909350 0.29132645 0.12304177 0.11519220
0.11227243 0.09455040 0.08466637 0.03897989 0.12950017 0.11581933 0.01399845
0.19283810
[14,] 0.16995824 0.11253845 0.04339634 0.21515022 0.04574897 0.12176542
0.06408349 0.08088082 0.13463896 0.08263153 0.12472662 0.12707934 0.02770418
0.14904546
```

[15,] 0.07434906 0.01195899 0.07294625 0.32251291 0.06121090 0.05240968
0.01995153 0.09458094 0.04403441 0.13089401 0.06925080 0.07351585 0.04416087
0.05548736
[16,] 0.01926518 0.01766264 0.06571072 0.30240013 0.07169285 0.06875185
0.02662694 0.09485712 0.08172787 0.17532107 0.08917268 0.05252690 0.02053113
0.08191322
[17,] 0.03645326 0.03758371 0.11816814 0.06760561 0.04915495 0.09092906
0.02574890 0.08585650 0.13135463 0.09436267 0.05308062 0.09731981 0.04310329
0.10058396
[18,] 0.06490801 0.08201455 0.13342775 0.15082266 0.05728207 0.14939495
0.05328191 0.17810452 0.06464600 0.04421308 0.08319973 0.17794831 0.08937120
0.20504315
[19,] 0.10441516 0.12793248 0.17230775 0.16514855 0.12397511 0.23838848
0.08736393 0.20400450 0.07112583 0.05605744 0.04816532 0.22582345 0.15654753
0.32148702
[20,] 0.07910490 0.14925633 0.11814458 0.17983550 0.16596718 0.15337978
0.14865390 0.13692918 0.13628680 0.07312238 0.07787941 0.10561333 0.14318957
0.11668757
[21,] 0.13692218 0.12115775 0.23851571 0.19842727 0.07645884 0.12662645
0.06079397 0.16019230 0.16474545 0.03560294 0.16625487 0.16623710 0.16364181
0.07768466
[22,] 0.06017409 0.15343666 0.17785640 0.20201821 0.09787719 0.14402622
0.10227789 0.16548153 0.17173223 0.07878264 0.13604401 0.13749223 0.21654891
0.04050810
[23,] 0.10318389 0.03501942 0.03732112 0.04139892 0.08375863 0.02827809
0.10315111 0.05463854 0.06315190 0.11213178 0.08747287 0.05785691 0.03583793
0.04710456
[24,] 0.12312443 0.05877542 0.05136634 0.07202357 0.06786197 0.05576250
0.05986569 0.05819739 0.13514811 0.08215693 0.07921071 0.03524820 0.05734057
0.10248166
[25,] 0.11794594 0.13098130 0.07088983 0.11344149 0.01250655 0.06543579
0.05924418 0.10613068 0.24666704 0.08953359 0.08746728 0.06052776 0.10576421
0.10302527
[26,] 0.14900621 0.15960159 0.08093548 0.20917302 0.02794304 0.08413488
0.05504089 0.10913229 0.08965429 0.05924364 0.08064130 0.10743863 0.04684523
0.10286146
[27,] 0.03593886 0.10149985 0.02684050 0.23430030 0.06149800 0.12883229
0.06606006 0.18058769 0.04582087 0.08084956 0.15968336 0.06635490 0.07377810
0.16490350
[28,] 0.07865970 0.05423484 0.01600767 0.21349135 0.04391426 0.07247362
0.06318774 0.13483013 0.10015128 0.02435854 0.11510550 0.04914205 0.10530671
0.09107539
[29,] 0.10370060 0.02837305 0.01430056 0.12683689 0.09328451 0.06503380
0.04130813 0.04007200 0.08525772 0.05445307 0.12070651 0.10987135 0.05050080
0.12567212

```
[30,] 0.11397593 0.01207451 0.03157343 0.17468475 0.03545680 0.06459855
0.04449197 0.08639367 0.04697836 0.07980747 0.15333343 0.15235453 0.05008395
0.15719516
[31,] 0.04471428 0.01694908 0.07052028 0.12454782 0.07575385 0.13949907
0.02109169 0.06013421 0.09222721 0.08512426 0.06561673 0.03184912 0.02820715
0.09061567
[32,] 0.08999307 0.02823147 0.11822772 0.02232264 0.16716149 0.04515754
0.03981359 0.08440019 0.05047360 0.11309164 0.07190630 0.03897595 0.04018783
0.02599681
[33,] 0.11569612 0.04843845 0.07830757 0.04747504 0.07966807 0.05835490
0.08785288 0.06698610 0.08748736 0.05915855 0.09892794 0.05948563 0.05008834
0.03725890
 [ reached getOption("max.print") -- omitted 15 rows ]
```

Question2:

In this part, we have the goal of determining the optimal h for both predictive regressions for all 30 DJ constituents.

We have:

$$\frac{r_{s,t}}{\sigma_{s,t-1}} = \alpha + \beta_h \cdot \frac{r_{s,t-h}}{\sigma_{s,t-h}} + \varepsilon_{s,t},$$

And we have

$$\frac{r_{s,t}}{\sigma_{s,t-1}} = a + b_h \cdot \mathrm{sign}\left(r_{s,t-h}\right) + \xi_{s,t}.$$

Where we have $r_{s,t}$ denoting the s th stock in the DJ. We want to determine the optimal h for all 30 stocks.

First, we obtain $\frac{r_{s,t}}{\sigma_{s,t-1}}$ with given rule return and calculated sigmat. Then for each h, we construct model that maps sign($r_{s,t-h}$) to $\frac{r_{s,t-h}}{\sigma_{s,t-1}}$. With the model we fitted, we predicted R_h and find maximum R_h to find the optimal h.

Code and Result:

```
> #b-QUESTION 2
> #Predictive regression
> #Determine the optimal h for both
> #predictive regressions for all 30 DJ constituents
> equation_left <- NULL
> for(i in 1:30) {
+   #As the formula: the one at the left of the equation is
+   #r_(s,t)/\sigma_(s,t-1)
```

```
+   #We bind them together as matrix
+   equation_left <- cbind(equation_left,
returns_last5[13:nrow(returns_last5),i]/sigmat[,i])
+
+ }
> #find optimal h for all 30 stocks
> optimal_h_30 <- numeric(30)
> for(i in 1:30) {
+
+   rh <- numeric(12)
+   for(h in 1:12) {
+     #Actually there is a mapping between y and x
+     #y is the equation left item
+     #x is actually the sign of a rule return: (r_(s,t-h))
+     #Still, make a matrix to calculate together
+     #We start from h+1 because we start from h+1 th month using the previous h
month data!
+     y <- equation_left[(h+1):nrow(equation_left) ,i]
+     x <- sign(returns_last5[(h+1):nrow(equation_left),i])
+     #We try to fit the model
+     model <- lm(y ~ x)
+     #And then get the R_h
+     rh[h] <- summary(model)$r.squared
+   }
+   #The optimal h is the one gets the highest R_h, isn't it?
+   optimal_h_30[i]<- (1:12)[which.max(rh)]
+ }
> #optimal h for 30 stocks with highest R-squared
> #Add title for the result
> names(optimal_h_30) <- stocks
> optimal_h_30
 MMM  AXP AAPL   BA  CAT  CVX CSCO   KO  DIS DWDP  XOM   GS   HD  IBM INTC  JNJ
JPM  MCD  MRK MSFT  NKE  PFE   PG  TRV  UTX  UNH   VZ    V  WMT  WBA
  11    2   12   11    1    1    6    2    9    1   11    3    8    7    5    2
2    9    9    8   12    4    8   11   10   11   10    1   10    7
```

Optimal h result obtained.

Question 3:

In this part, we will construct a time series momentum (TSMOM) portfolio and will summarize the performance.

Some assumptions will be made: First, the performance will be based on a 60-month window rolling window.

We have the time series momentum trading strategy specified by:

$$r_{t,t+1}^{TSMOM} = \frac{1}{30}\sum_{s=1}^{30} \underbrace{\text{sign}\left(r_{s,t-h_s:t}\right) \cdot \frac{40\%}{\sigma_{s,t}}}_{B_{s,t}} r_{s,t:t+1},$$

and we have

$$B_{s,t} = \text{sign}\left(r_{s,t-h_s:t}\right) \cdot \frac{40\%}{\sigma_{s,t}}$$

Which is our position for the s-th constituent at time t and $r_{s,t-h_s:t}$ denotes the h month lagged returns observed at time t. we will take h= 12 for all 30 stocks. $r_{s,t-h_s:t}$ and $\sigma_{s,t}$ are already known from previous question, then it's easy to find the TSMOM.

```
> #Question 3
> #Summarize he performance
> #TSMOM
> TSMOM <- numeric(nrow(returns_last5) - 12)
> #Start from the 13th month
> for(i in 13:(nrow(returns_last5))) {
+   #B_st = sign(return_(t-h:t)) * 40% / \sigma_t
+   B_st <- numeric(30)
+   for(j in 1:30) {
+   #assume hs = 12 for all stocks
+     B_st[j] <- sign(returns_last5[(i -12),j] )* 40/100 / sigmat[i-12,j]
```

```
+   }
+   #Still calculate TSMOM, with B * R. Remove the na
+   TSMOM[i] <-  1/30 * sum(B_st * returns_last5[i,], na.rm = TRUE)
+ }
> #performances mean and vavriance of TSMOM portfolio
> #You can see the mean is very small here
> mean(TSMOM)
[1] -0.0135907
> var(TSMOM)
[1] 0.01677537
> perf_tsmom = ((12 * mean(TSMOM) - 0.02) / (sqrt(12) * sqrt(var(TSMOM))))
> perf_tsmom
[1] -0.4080697
```

TSMOM's portfolio expected return is -0.0135907, TSMOM's portfolio expected return is 0.01677537, and the Shape Ratio of TSMOM's portfolio is -0.4080697.

Part C:

Question 1:

In this section, we suppose that our position to the trading rule is determined by the magnitude of the signal. First, we have to compute the expected h-period holding period return. The technical indicator F_t has been provided in the handout:

$$F_t = \sum_{h=0}^{m-2} d_i r_{t-j}.$$

Furthermore, the h-period holding period return is from the discussion from the prof:

$$\Sigma_i \Sigma_j d_i E(r_{t+i} r_{t+j}) = \Sigma_i \Sigma_j d_i [\gamma_{(i-j)} - E(r_t)^2]$$

When γ is the correlation of the difference between ith and jth. d_i is the vector from double MA rule. $E(r_t)$ is the expectation of rule return at time t.

This question just conducts a function called ERh, and it would be used in following question.

Code:

```
> #PARTC
> #Question changed:
> #The same as the 1st question.
> #Similar with the ruleReturn function, however, the ER expression is changed.
> ERh <- function(h, m, r, retX) {
+   M<-length(d(m,r))-1
+   acfs<- acf(retX, plot=F, type="covariance", lag.max=M)$acf
+   #Actually the E(r_t) in formula
+   mX<-mean(retX)
+   ds = d(m,r)
+   ER <- 0
+   rXF = corXF(ds, retX)
+   for (i in (1:length(ds))){
+     #The sum of di * [r(i-j) - E(r_t)^2]
+     ER <- ER + ds[i] * (rXF[m-r]- (mX^2))
+   }
+   #return ER
+   ER
+ }
```

Question 2:

We will find optimal double MA for all 30 DJ constituents that maximize the 12-period holding period return.

It's similar with the 7th question from last year, however, it uses expected rule return from previous question instead of the rule return.

Similar with the process of a-1, we find the optimal double-MA trading rule for expected rule return.

```
> all_mr2 <- NULL
> #Code from previous assignment, question 6/7
> #Loop for each stock
> for(stock in 1:30) {
+   #We generate the current ERH, from the previous m and r data
+   currERh = ERh(12,
all_mr_double_ma[stock,1],all_mr_double_ma[stock,2],returns_last5[,stock])
+   #Monthly return
+   monthlyData <-monthlyReturn(get(stocks[stock]), type="log")
+     #na.omit(ERh(12,
all_mr_double_ma[stock,1],all_mr_double_ma[stock,2],returns_last5[,stock]))
+
+   #choose the optimal m and r for monthly data
+   result <- numeric(0)
+   m <- numeric(0)
+   r <- numeric(0)
+   for (i in 2:11){
+     for(j in (i+1):12){
+       if(j>i){
+         #We calculate the current ERh
+         result <- c(result, ERh(12, m=j,r=i, monthlyData))
+         m <- c(m,j)
+         r <- c(r,i)
+
+       }
+     }
+   }
+   m_optimal <- m[which.max(result)]
+   r_optimal <- r[which.max(result)]
+
+   #combine them
+   all_mr2 <- rbind(all_mr2, c(m_optimal, r_optimal))
+ }
```

```
> row.names(all_mr2) <- stocks
> colnames(all_mr2) <- c("m", "r")
> #optimal m and r
> #From the result, some of the result is the same as the first question.
> #That's correct, as some of the m and r are truly the optimal ones, in both of
the questions!
> all_mr2
      m  r
MMM  11 10
AXP  12 11
AAPL 10  9
BA   10  9
CAT   9  8
CVX   9  8
CSCO  6  5
KO    7  6
DIS  10  9
DWDP 11 10
XOM  11 10
GS    9  8
HD   10  9
IBM   3  2
INTC  3  2
JNJ   4  3
JPM   9  8
MCD   7  6
MRK   7  6
MSFT  9  8
NKE   8  7
PFE   3  2
PG    4  3
TRV  11 10
UTX   3  2
UNH   4  3
VZ   12 11
V     6  5
WMT   3  2
WBA  11 10
>
```