STA457H1F Assignment 1

Ahmed Nawaz Amanullah, St.# 1001325773
Ling Long, St.# ####

February 24, 2019

Part A

Question 1

In this section, we will find the optimal double moving average (MA) trading rules for all 30 DJ constituents using monthly data. As hinted by the assignment, we will refer to the Fall 2018 assignment to implement this. We will proceed in chronological order of said assignment in order to gain understanding of this question.

We will predict the direction of the trend of asset prices using a function of past asset prices F_t, which will be converted to buy and sell trading signals B_t, with buy corresponding to +1 and sell to -1. F_t will be based on a moving-average technical indicator, which can be expressed as a function of log returns:

$$F_t = \delta + \sum_{j=0}^{m-2} d_j X_{t-j}$$

where $X_t = ln(P_t/P_{t-1})$, while δ and d_j are defined by a given trading rule. For the assignment, we will assume $\delta = 0$.

The function is then, as mentioned earlier, converted to trading signals as follows:

$$\begin{cases} \text{"}Sell\text{"} \iff B_t = -1 \iff F_t < 0 \\ \text{"}Buy\text{"} \iff B_t = +1 \iff F_t > 0 \end{cases}$$

We will also be assessing the results and performance of trading decisions, and the returns at time t will be obtained using "ruled returns", which will be denoted as R_t. So if for a period $[t-1, t)$, a trader establishes a position at time $t-1$, represented by B_{t-1}, the "ruled returns" expression is as follows:

$$R_t = B_{t-1} X_t \iff \begin{cases} R_t = -X_t & B_{t-1} = -1 \\ R_t = +X_t & B_{t-1} = +1 \end{cases}$$

where $X_t = ln(P_t/P_{t-1})$ denote the logarithm return over this period (so we will be assuming no dividend payouts during the time period).

The realized returns will thus be determined using:

$$\tilde{R}_t = \sum_{D=1}^{n} R_{t+D}$$

where D represents the stochastic duration of the position lasting n days provided that:

$$\{D = n\} \iff \{B_{t-1} \neq B_t, B_t = ... = B_{t+n-1}, B_{t+n-1} \neq B_{t+n}\}$$

Step 1: Derive variance of predictor F_t

The variance of the predictor is derived as follows:

$$\sigma_F^2 = var(\sum_{i=0}^{m-2} d_i X_{t-i})$$
$$= cov(\sum_{i=0}^{m-2} d_i X_{t-i}, \sum_{i=0}^{m-2} d_i X_{t-i})$$
$$= d_0 cov(X_t, \sum_{i=0}^{m-2} d_i X_{t-i}) + d_1 cov(X_{t-1}, \sum_{i=0}^{m-2} d_i X_{t-i}) + ... + d_{m-2} cov(X_{t-m+2}, \sum_{i=0}^{m-2} d_i X_{t-i})$$
$$= d_0(d_0\gamma_0 + d_1\gamma_1 + ... + d_{m-2}\gamma_{m-2}) + d_1(d_0\gamma_1 + d_1\gamma_0 + d_2\gamma_1 + ... + d_{m-2}\gamma_{m-3}) + ... + d_{m-2}(d_0\gamma_{m-2} + d_1\gamma_{m-3} + ... + d_{m-2}\gamma_0)$$
$$= \gamma_0 \sum_{i=0}^{m-2} d_i^2 + 2\sum_{i=1}^{m-2}\sum_{j=0}^{m-2-i} d_j d_{j+i}\gamma_i$$
$$= \gamma_0(\sum_{i=0}^{m-2} d_i^2 + 2\sum_{i=1}^{m-2}\sum_{j=0}^{m-2-i} d_j d_{j+i}\rho_i)$$
$$= \gamma_0(\sum_{i=0}^{m-2} d_i^2 + 2\sum_{j=0}^{m-2}\sum_{i=j+1}^{m-3} d_j d_i \rho_{i-j})$$

Provided below is the R code for computing the variance of the predictor:

```
#Step 1: Get the variance of the predictor
# d: vector of dj coefficients (j = 0, ..., m-2)
# X: log returns
#Will use quadratic form...
varF <- function(d, X){
    #maximum lag
    M <- length(d)-1
    #get auto-covariance
    acfs <- acf(X, plot = F, type="covariance", lag.max=M)$acf
    #get toeplitz matrix of acfs
    #name used to refer to:
    #[[\gamma_0, ..., \gamma_{m-2}],
    # [\gamma_1, \gamma_0, ..., \gamma_{m-3}], ...
    # [\gamma_{m-2}, ..., \gamma_0]]
    Gamma <- toeplitz(as.vector(acfs))
    #quadratic form
    varF <- d%*%Gamma%*%as.vector(d)
    varF
}
```

Note that the implementation above uses the quadratic form of the expression:

$$\sigma_F^2 = \begin{bmatrix} d_0 & ... & d_{m-2} \end{bmatrix} \begin{bmatrix} \gamma_0 & ... & \gamma_{m-2} \\ ... & ... & ... \\ \gamma_{m-2} & ... & \gamma_0 \end{bmatrix} \begin{bmatrix} d_0 \\ ... \\ d_{m-2} \end{bmatrix}$$

Step 2: Derive expectation of predictor F_t

The expectation of the predictor is derived as follows:

$$\mu_F = E(\sum_{i=0}^{m-2} d_i X_{t-i})$$
$$= \sum_{i=0}^{m-2} d_i E(X_{t-i})$$
$$= \mu_X \sum_{i=0}^{m-2} d_i$$

Provided below is the R code for computing the expectation of the predictor:

```r
#Step 2: Get the expectation of the predictor
# d: vector of dj coefficients (j = 0, ..., m-2)
# X: log returns
muF <- function(d, X){
    muF <- mean(X)*sum(d)
    muF
}
```

Step 3: Derive autocorrelation function at lag one for predictor

The expression for the autocorrelation function for lag 1 is derived as follows:

$$\rho_F(1) = corr(F_t, F_{t-1}) = \frac{cov(F_t, F_{t-1})}{var(F_t)}$$

First the covariance expression,

$$cov(F_t, F_{t-1}) = corr(\sum_{i=0}^{m-2} d_i X_{t-i}, \sum_{i=0}^{m-2} d_i X_{t-1-i})$$
$$= d_0 cov(X_t, \sum_{i=0}^{m-2} d_i X_{t-1-i}) + d_1 cov(X_{t-1}, \sum_{i=0}^{m-2} d_i X_{t-1-i}) + ... + d_{m-2} cov(X_{t-m+2}, \sum_{i=0}^{m-2} d_i X_{t-1-i})$$
$$= d_0(d_0\gamma_1 + d_1\gamma_2 + ... + d_{m-2}\gamma_{m-1}) + d_1(d_0\gamma_0 + d_1\gamma_1 + d_2\gamma_2 + ... + d_{m-2}\gamma_{m-2}) + ... + d_{m-2}(d_0\gamma_{m-3} + d_1\gamma_{m-4} + ... + d_{m-2}\gamma_1)$$
$$= \gamma_0 \left\{ d_0(d_0\rho_1 + d_1\rho_2 + ... + d_{m-2}\rho_{m-1}) + ... + d_{m-2}(d_0\rho_{m-3} + d_1\rho_{m-4} + ... + d_{m-2}\rho_1) \right\}$$
$$= \gamma_0 \sum_{i=0}^{m-2} \sum_{j=0}^{m-2} d_i d_j \rho_{j-1-i}$$

Then the autocorrelation function:

$$\rho_F(1) = \frac{cov(F_t, F_{t-1})}{var(F_t)}$$
$$= \frac{\gamma_0 \sum_{i=0}^{m-2} \sum_{j=0}^{m-2} d_i d_j \rho_{j-1-i}}{var(F_t)}$$

Provided below is the R code for computing the lag one autocorrelation of the predictor:

```
#Step 3: Computing ACF(1) of forecaster
#will use a quadratic form
# d: vector of dj coefficients (j = 0, ..., m-2)
# X: log returns
rhoF <- function(d, X){
    #max lag
    M <- length(d)-1
    #get acfs
    acfs <- acf(X, plot = F, type = "covariance", lag.max = M+2)$acf
    #get the covariance
    temp <- d%*%matrix(acfs[as.vector(
    abs(outer(0:M,1:(M+1), "-"))+1)], M+1, M+1)%*%as.vector(d)
    rhoF <- temp/varF(d, X)
    rhoF
}
```

Additionally note that as with when computing the variance of the predictor, a similar approach utilizing matrix multiplication is being used above:

$$cov(F_t, F_{t-1}) = \begin{bmatrix} d_0 & ... & d_{m-2} \end{bmatrix} \begin{bmatrix} \gamma_1 & ... & ... & \gamma_{m-1} \\ ... & ... & ... & ... \\ ... & ... & ... & ... \\ \gamma_{m-3} & ... & \gamma_0 & \gamma_1 \end{bmatrix} \begin{bmatrix} d_0 \\ ... \\ d_{m-2} \end{bmatrix}$$

Step 4: Computing the expected ruled returns and expected holding period length

Under the assumption that X_t follows a stationary Gaussian process, the expected ruled returns has been provided as follows:

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

Most of the expressions needed for this computation has been derived earlier. We just need the expression:

$$corr(X_t, F_{t-1}) = \frac{cov(X_t, F_{t-1})}{\sqrt{\gamma_0 var(F_t)}}$$

$$= \frac{(d_0\gamma_1 + d_1\gamma_2 + ... + d_{m-2}\gamma_{m-1})}{\sqrt{\gamma_0 var(F_t)}}$$

$$= \frac{\gamma_0 \sum_{i=0}^{m-2} d_i \rho_{i+1}}{\sqrt{\gamma_0 var(F_t)}}$$

The R code for implementing this computation is as follows:

```
#Step 4 Intermediate: correlation between X_t and F_{t-1}
#will use a quadratic form
# d: vector of dj coefficients (j = 0, ..., m-2)
# X: log returns

corXF <- function(d, X){
    Mp <- length(d)
```

```
    acfs <- acf(X, plot = F, type = "covariance", lag.max = Mp)$acf
    corXF <- sum(d*acfs[-1])/sqrt(acfs[1]*varF(d, X))
    corXF
}
```

Additionally, the unconditional variance of the ruled returns has been provided as:

$$var(R_t) = E(X_t^2) - E(R_t)^2 = \sigma_X^2 + \mu_X^2 - E(R_t)^2$$

Furthermore, as provided, Kedem (1986) shows that the expected zero-crossing rate for a discrete-time, zero-mean, stationary Gaussian sequence Z_t is given by:

$$\frac{1}{\pi} cos^{-1} \rho_Z(1)$$

where $\rho_Z(1)$ denotes the autocorrelation function of $\{Z_t\}$ at lag one. Using this result, we may approximate the expected length of the holding period to be:

$$H = \frac{\pi}{cos^{-1} \rho_F(1)}$$

Provided below is the R code for the computation of the expected holding period, the double MA co-efficients and the expected ruled returns:

```
#Step 4: Expected holding period and ruled returns
#expected holding period
#rho is lag one autocorrelation
#provided by step 3
Hold <- function(rho){
    Hold <- pi/acos(rho)
}

#next is the double MA co-efficients
d <- function(m, r){
    d <- c((m-r)*((0:(r-1))+1), r*(m-(r: (m-1))-1))
    d
}

# retX: log asset return
# m: long-term MA
# r: short-term MA

ruleReturn <- function(retX, m, r){
    #returns variance
    vX <- sd(retX)
    #returns mean
    mX <- mean(retX)
    #predictor mean
    mF <- muF(d(m, r), retX)
    #predictor sd
    vF <- sqrt(varF(d(m, r), retX))
    #lag 1 correlation (rho) between X_t and F_{t-1}
    rXF <- corXF(d(m, r), retX)
```

```
    #lag 1 correlation of predictor
    rF <- rhoF(d(m, r), retX)
    #expected return
    ER <- sqrt(2/pi)*vX*rXF*exp(-mF*mF/(vF*vF)) +
    mX*(1-2*pnorm(-mF/vF))
    #expected hold duration
    H <- Hold(rF)
    #return results
    list("ER" = ER, "H" = H, "rhoF" = rF, "VF"= vF, "muF"=mF,
    "corXF"=rXF)
}
```

Step 5: Downloading the monthly data of the constituents

The data for the DJ constituents were downloaded from Yahoo Finance in csv files and stored in a folder. The R code for retrieving this data from the csv files is as follows:

```
#This function is to download data from folder...
#idea: return three arrays - one containing the stock tickers...
#   another the adjusted close (using hint from prev assn)
#   and finally one containing the log returns

# dataDir: directory containing csv files containing data
# currDir: current working directory (to change back to after
#   loading data)
getAdC <- function(dataDir, currDir){
    #switch to directory containing price data
    setwd(dataDir)
    #get list of files in directory
    priceFiles <- list.files(dataDir, pattern="*.csv",
    full.names = F, recursive = F)
    #to hold tickers
    tickers <- c()
    #to hold prices
    prices <- c()
    #to hold log returns
    logrets <- c()
    #iterate through list above
    for (file in seq(1, length(priceFiles))){
        tickers <- c(tickers, substr(priceFiles[file],
        1,nchar(priceFiles[file])-4))
        #use rbind for the adjusted close data...
        #first read the file
        stockdat <- read.csv(file = priceFiles[file],
         header = T, sep=",")
        #then get adjusted close prices from file
        price <- stockdat[1:dim(stockdat)[1], 6]
        prices <- rbind(prices, price)
        #also the log returns
        logrets <- rbind(logrets,
        log(price[2:length(price)]/price[1:(length(price)-1)]))
    }
    #return to original directory
```

```
    setwd(currDir)

    #return tickers, prices and log returns
    list("tickers"=tickers, "prices"=prices, "logrets" = logrets)
}
```

Step 6: Writing R function to find optimal monthly double MA trading rules

Provided below are the R functions utilized to find the optimal double MA trading rules using monthly data:

```
#Function to choose the optimal daily and monthly MA trading rules
#   (i.e. that maximize expected rule returns)
#will test function against the one in prev assn after
#   implementation...

#first optimal monthly
#   retX: vector of log returns
monthlyoptimal_dma <- function(retX){
    #to hold optimal m and r
    optimal_m <- 2
    optimal_r <- 1
    #get ruleReturn for this setting
    currER <- ruleReturn(retX, optimal_m, optimal_r)$ER
    #iterate up to 12 max as we are doing monthly
    #loop through r
    for (i in seq(1, 11)){
        for (j in seq(i+1, 12)){
            ERij <- ruleReturn(retX, j, i)$ER
            if (ERij > currER){
                optimal_m <- j
                optimal_r <- i
                currER <- ERij
            }
        }
    }
    #return optimal double MA trading rules
    list("monthlyoptimal_m"=optimal_m, "monthlyoptimal_r"=optimal_r)
}

#functions that will implement the above for all constituents...

#   retsX: the log returns of the constituents

monthlyoptimals_dma <- function(retsX){
    #to hold the optimal trading rules
    m <- c()
    r <- c()
    #amount of tickers to go through
    numstocks <- dim(retsX)[1]
    #number of periods/ months
    months <- dim(retsX)[2]
    for (i in seq(1, numstocks)){
        optimals <- monthlyoptimal_dma(retsX[i, 1:months])
```

```
        #add optimals to list
        m <- c(m, optimals$monthlyoptimal_m)
        r <- c(r, optimals$monthlyoptimal_r)
    }

    #return optimal rules
    list("monthlyoptimals_m"=m, "monthlyoptimals_r"=r)
}
```

Disclosed below are the optimal trading rules found for the constituents:

```
## Warning in rbind(prices, price): number of columns of result is not a
## multiple of vector length (arg 2)
```

```
## Warning in rbind(logrets, log(price[2:length(price)]/price[1:(length(price)
## - : number of columns of result is not a multiple of vector length (arg 2)
```

```
## Warning: package 'knitr' was built under R version 3.2.5
```

Table 1: DJ double MA optimal trading rules

tickers	m	r
AAPL	3	2
AXP	2	1
BA	9	8
CAT	5	4
CSCO	7	6
CVX	8	6
DIS	5	4
DWDP	2	1
GS	2	1
HD	11	10
IBM	12	11
INTC	4	3
JNJ	9	8
JPM	12	11
KO	7	6
MCD	7	1
MMM	9	8
MRK	12	11
MSFT	9	8
NKE	10	7
PFE	10	8
PG	2	1
TRV	9	8
UNH	9	8
UTX	2	1
V	12	2
VZ	9	8
WBA	11	10
WMT	9	8
XOM	12	11

Step 7: Writing R function to compute in-sample trading statistics and compare with theoretical results

Please find below the functions implemented in R to generate the in-sample as well as theoretical trade statistics (cumulative return and holding time) for the optimal trade rules derived earlier.

```
#functions for computing in-sample trading statistics
#   (i.e. cumulative return and holding time)
#   to later compare with theoretical results
#first function to compute the predictor

#   d: d is d(m ,r) - presumably the optimal ones found
#   retX: the log returns
#   t: get signal at time t

f <- function(d, retX, t){
    M <- length(d) - 1
    if (t >= M){
        output <- sum(d*rev(retX[(t-M):t]))
        output
    }
    else {print('t is smaller than M')}
}

#next the realized return
#   d: d is d(m ,r) - presumably the optimal ones found
#   retX: the log returns
realized_ret <- function(d, retX){
    re <- c()
    M <- length(d)
    for (t in seq(M + 1,length(retX))){
        re <- c(re, sign(f(d, retX, t-1))*retX[t])
    }
    realized_ret <- sum(re)/length(M:length(retX))
    realized_ret
}

#next is the holding period
holding_period <- function(d, retX){
    num_change <- 0
    M <- length(d)
    for (t in seq(M+1, length(retX))){
        if (sign(f(d, retX,t)) != sign(f(d, retX,t-1))){
            num_change <- num_change + 1
        }
    }
    hold <- length(M:length(retX))/num_change
    hold
}

#function that will take in the array of log returns
#   and the optimal trading decisions
in_sample_estimate <- function(retX, m, r){
    #number of tickers
```

```r
    num_stocks <- dim(retX)[1]
    #number of periods/ months
    #to hold in_sample and theoretical values
    djER <- c()
    djH <- c()
    djISR <- c()
    djISH <- c()
    months <- dim(retX)[2]
    for (i in seq(1, num_stocks)){
        ruleReturn_i <- ruleReturn(retX[i, 1:months], m[i], r[i])
        djER <- c(djER, ruleReturn_i$ER)
        djH <- c(djH, ruleReturn_i$H)
        djISR <- c(djISR,
        realized_ret(d(m[i],r[i]),retX[i, 1:months]))
        djISH <- c(djISH,
        holding_period(d(m[i],r[i]),retX[i, 1:months]))
    }
    #return estimated and theoretical values
    list("djER"=djER ,"djH"=djH, "djISR"=djISR, "djISH"=djISH)
}
```

The results are disclosed below as well:

Table 2: DJ double MA optimal trading rules statistics

tickers	theoretical return	expected return	theoretical hold	expected hold
AAPL	0.0141	0.0062	2.932	2.861
AXP	0.0089	0.0107	2.178	2.225
BA	0.0082	0.0077	5.933	7.333
CAT	0.0138	0.0108	3.865	4.48
CSCO	0.0112	0.0065	4.671	4.826
CVX	0.0043	0.005	6.016	6.697
DIS	0.0054	0.0117	4.012	4.87
DWDP	0.0086	0.0031	2.144	2.102
GS	0.0074	0.0049	2.136	2.248
HD	0.0047	0.0044	5.928	7.517
IBM	0	-0.0028	4.796	4.717
INTC	0.0088	-0.0017	3.239	3.629
JNJ	0.0021	0.0038	4.473	6.471
JPM	-8e-04	0.0031	4.986	7.233
KO	0.0026	0.0068	4.311	5.163
MCD	0.0052	0.0046	4.961	4.723
MMM	0.0028	0.0035	4.84	6.286
MRK	0.003	0.005	5.85	7
MSFT	0.0043	2e-04	4.444	6.286
NKE	0.0079	0.0107	7.02	13.688
PFE	0.0019	0.0048	7.23	7.3
PG	0.0045	0.0056	2.129	2.064
TRV	-2e-04	0.0046	4.256	5.641
UNH	0.013	0.0142	5.6	8.462
UTX	0.0011	0.002	2.012	2.142
V	0.0095	0.0155	7.499	31
VZ	0.0032	0.0053	4.034	4.49

tickers	theoretical return	expected return	theoretical hold	expected hold
WBA	-0.005	-0.0382	2.228	6.412
WMT	0.0016	0.0012	4.334	3.607
XOM	0.002	0.0048	5.451	7.483

Question 2

In this section, we will construct the equally weighted (EW) and risk-parity (RP) weighted portfolios using the 30 DJ constituents and summarize the performance.

As per general instruction, performance will be based on a 60-month rolling window and the portfolio will be rebalanced monthly. The parameters (σ) will be calibrated/ estimated at the end of each year.

The following assumptions are going to be made:

1. Will use the 60-month window to get optimal trading rules.
2. Will use these optimal settings to predict the signals for next month
3. For risk parity, will use the last 12 months of window to calibrate the asset volatilities (see section B) as brought up earlier.
4. Use predicted signal and portfolio weights to get rule return of portfolio (see hint).
5. Check performance of next 12 months using predicted signals. Since we recalibrate parameters annually (i.e. at end of each year), we will re-calculate the ex-ante volatilities every 12 months.
6. The 60-month window moves 12 months ahead and we repeat (1-5). Process terminates when we reach the end of our data.

Please find the implementation of these steps for the equally weighted portfolio as follows:

```
#Question 2: summarizing performance of the equally weighted
#   portfolio
#   retX: 30X(periods) dim matrix of monthly log returns

ew_performance <- function(retX){
    #number of stocks provided
    num_stocks <- dim(retX)[1]
    #number of periods of data provided
    months <- dim(retX)[2]
    #to hold the ruled returns
    re <- c()
    #to hold the cumulative return up to time t
    cre <- c()
    #to hold expected return at time t
    ere <- c()
    #to hold the ex-ante volatility at time t
    ea_sd <- c()
    #to hold sharpe ratio at time t
    sharpes <- c()
    #only go up to as long as window can be fit
    for (i in seq(1, months - 60, 12)){
        #get optimal dma within 60 month window for all stocks
```

```
    optimal_dma <- monthlyoptimals_dma(
    retX[1:num_stocks, i:(i+60-1)])
    m <- optimal_dma$monthlyoptimals_m
    r <- optimal_dma$monthlyoptimals_r
    #need to get the ex-ante volatility every 12 months
    ea_sd_i <- ea_volatility(
    retX[1:num_stocks,1:(i+60-1)])
    #Ok, now we use the optimal trading rules to find the
    #   ruled return for each stock
    #to hold portfolio weights
    #this is an ew portfolio so all weights are the same
    weights <- rep(1/num_stocks, num_stocks)

    #this is the portfolio variance
    var_t <- 0
    for (s in seq(1, num_stocks)){
        var_t <- var_t + (weights[s]^2)*ea_sd_i[s]^2
    }

    ea_sd <- c(ea_sd, sqrt(var_t))

    #going back to old method
    for (t in seq(i + 60, i + 72 -1)){
        if (t <= months){
            re_t <- 0
            for (s in seq(1, num_stocks)){
                s_d <- d(m[s], r[s])
                re_t <- re_t + weights[s]*
                sign(f(s_d, retX[s, 1:months], t-1))*retX[s,t]
            }
            #now add the portfolio ruled return
            re <- c(re, re_t)
            if (length(re) > 1){
                cre <- c(cre, cre[length(cre)] + re_t)
            }else{
                cre <- c(cre, re_t)
            }
            #expected (annual) return at time t
            ere <- c(ere, (12*cre[length(cre)]/length(cre)))
            #as well the sharpe
            sharpe_t <- (ere[length(ere)]-0.02)/
            ea_sd[length(ea_sd)]
            sharpes <- c(sharpes, sharpe_t)
            #note that since this is a EW portfolio, there is
            #   no rebalancing step as weights remain the same
        }
    }


}

#return results
list("return"=ere[length(ere)],
```

```
    "volatility"=ea_sd[length(ea_sd)],
    "sharpe"=sharpes[length(sharpes)], "cumul_return"=cre,
    "e_return"=ere, "ea_volatilities"=ea_sd, "sharpes"=sharpes)
}

#risk parity function
rp_performance <- function(retX){
    #number of stocks provided
    num_stocks <- dim(retX)[1]
    #number of periods of data provided
    months <- dim(retX)[2]
    #to hold the ruled returns
    re <- c()
    #to hold the cumulative return up to time t
    cre <- c()
    #to hold expected return at time t
    ere <- c()
    #to hold the ex-ante volatility at time t
    ea_sd <- c()
    #to hold sharpe ratio at time t
    sharpes <- c()
    #only go up to as long as window can be fit
    for (i in seq(1, months - 60, 12)){
        #get optimal dma within 60 month window for all stocks
        optimal_dma <- monthlyoptimals_dma(
        retX[1:num_stocks, i:(i+60-1)])
        m <- optimal_dma$monthlyoptimals_m
        r <- optimal_dma$monthlyoptimals_r
        #need to get the ex-ante volatility every 12 months
        ea_sd_i <- ea_volatility(
        retX[1:num_stocks,1:(i+60-1)])
        #Ok, now we use the optimal trading rules to find the
        #   ruled return for each stock
        #to hold portfolio weights
        #this is an ew portfolio so all weights are the same
        temp <- 1/ea_sd_i
        weights <- temp/sum(temp)

        #this is the portfolio variance
        var_t <- 0
        for (s in seq(1, num_stocks)){
            var_t <- var_t + (weights[s]^2)*ea_sd_i[s]^2
        }

        ea_sd <- c(ea_sd, sqrt(var_t))

        #going back to old method
        for (t in seq(i + 60, i + 72 -1)){
            if (t <= months){
                re_t <- 0
                for (s in seq(1, num_stocks)){
                    s_d <- d(m[s], r[s])
                    re_t <- re_t + weights[s]*
```

```
                    sign(f(s_d, retX[s, 1:months], t-1))*retX[s,t]
                }
                #now add the portfolio ruled return
                re <- c(re, re_t)
                if (length(re) > 1){
                    cre <- c(cre, cre[length(cre)] + re_t)
                }else{
                    cre <- c(cre, re_t)
                }
                #expected (annual) return at time t
                ere <- c(ere, (12*cre[length(cre)]/length(cre)))
                #as well the sharpe
                sharpe_t <- (ere[length(ere)]-0.02)/
                ea_sd[length(ea_sd)]
                sharpes <- c(sharpes, sharpe_t)
                #note that since this is a EW portfolio, there is
                #   no rebalancing step as weights remain the same
            }
        }


    }

    #return results
    list("return"=ere[length(ere)],
    "volatility"=ea_sd[length(ea_sd)],
    "sharpe"=sharpes[length(sharpes)], "cumul_return"=cre,
    "e_return"=ere, "ea_volatilities"=ea_sd, "sharpes"=sharpes)
}
```

The performance results obtained implementing the function above is as follows:

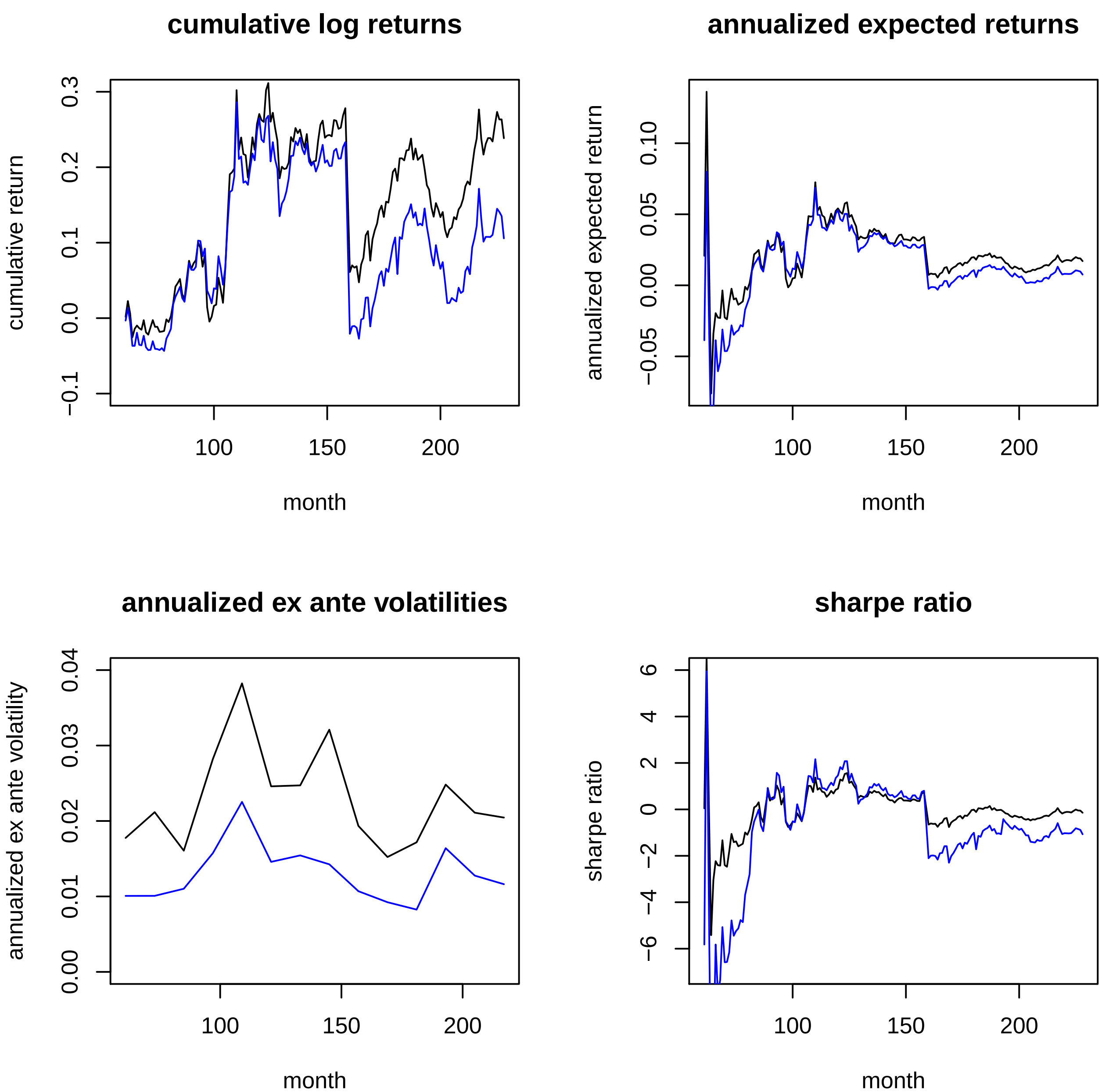

Table 3: EW performance metrics

Annualized expected return	Annualized volatility	Annualized sharpe ratio
0.017	0.0205	-0.1448

Table 4: RP performance metrics

Annualized expected return	Annualized volatility	Annualized sharpe ratio
0.0076	0.0116	-1.0707

Part B

Question 1

In this section, we will cover the computation of the ex-ante volatility estimate σ_t for the DJ constituents. The formula used is:

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

Where the weights $\delta^i(1-\delta)$ add up to one, and $\bar{r}_{s,t}$ is the exponentially weighted average return computed similarly:

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

For this assignment, as instructed, we will use $\delta = 0.2$.

The code for implementing this computation in R is as follows:

```
ea_volatility <- function(retX){
    #need to solve for delta
    #according to paper delta has been solved for as
    delta <- 0.2
    #number of stocks
    num_stocks <- dim(retX)[1]
    #number of periods provided
    t <- dim(retX)[2]
    #vector to hold the ex ante volatilities
    ea_sd <- c()
    #need to iterate over the stocks
    for (s in seq(1, num_stocks)){
        #compute the exponentially weighted returns
        r_bar <- sum((1-delta)*(delta^c(0:11))*
        rev(retX[s, (t-12):(t-1)]))
        #next the ex-ante volatility (annualized)
        #first the variance
        var <- 12*sum((1-delta)*(delta^c(0:11))*
        (rev(retX[s, (t-12):(t-1)])-r_bar)^2)
        #then add the ex-ante volatility of stock s to list
        ea_sd <- c(ea_sd, sqrt(var))
    }

    #return volatilities
    ea_sd
}
```

Part C

Question 1

In this section, we will make it so that our position to the trading rule is determined by the magnitude of the signal. First, we will need to compute the expected h-period holding period return. We have been provided the technical indicator F_t:

$$F_t = \sum_{i=0}^{m-2} d_i r_{t-i}$$

As well as the h-period holding return expression:

$$R_{t:t+h} = \sum_{j=0}^{h-1} F_{t+j} r_{t+j+1} = \sum_{j=0}^{h-1} \left(\sum_{i=0}^{m-2} d_i r_{t+j-i} \right) r_{t+j+1}$$

The expected h-period holding period return is computed as follows:

$$E(R_{t:t+h}) = E\left(\sum_{j=0}^{h-1} \left(\sum_{i=0}^{m-2} d_i r_{t+j-i} \right) r_{t+j+1} \right)$$

$$= E(r_{t+1}(d_0 r_t + d_1 r_{t-1} + ... + d_{m-2} r_{t-m+2}) + ... + r_{t+h}(d_0 r_{t+h-1} + ... + d_{m-2} r_{t+h-m+1}))$$

$$= d_0(h\gamma_1) + d_1(h\gamma_2) + ... + d_{m-2}(h\gamma_{m-1})$$

$$= h \sum_{i=0}^{m-2} d_i \gamma_{i+1}$$

The R-code for the computation of the expected h-period holding period return is as follows:

```
e_hperiod_holdperiodreturn <- function(retX, m, r, h){
    #get auto-covariances
    acfs <- acf(retX, plot = F, type = "covariance", lag.max = m)$acf
    #get expected holding period return
    ehphpr <- h*sum(d(m,r)*acfs[2:length(acfs)])
    #return value
    ehphpr
}
```

Question 2

In this section we will find the optimal double MA for all 30 DJ constituents that maximize the 12-period holding period return. Please find enclosed below the R-function that was written to implement this:

```
#Ques 2: optimal double MA for 30 DJ constituents maximizing
#   12-period holding period return.
monthlyoptimalEHR_dma <- function(retX){
    #to hold optimal m and r
    optimal_m <- 2
    optimal_r <- 1
```

```
    #get ruleReturn for this setting
    currEHR <- e_hperiod_holdperiodreturn(retX,
    optimal_m, optimal_r, 12)
    #iterate up to 12 max as we are doing monthly
    #loop through r
    for (i in seq(1, 11)){
        for (j in seq(i+1, 12)){
            EHRij <- e_hperiod_holdperiodreturn(retX,
            j, i, 12)
            if (EHRij > currEHR){
                optimal_m <- j
                optimal_r <- i
                currEHR <- EHRij
            }
        }
    }
    #return optimal double MA trading rules
    list("monthlyoptimal_m"=optimal_m, "monthlyoptimal_r"=optimal_r)
}

monthlyoptimalsEHR_dma <- function(retsX){
    #to hold the optimal trading rules
    m <- c()
    r <- c()
    #amount of tickers to go through
    numstocks <- dim(retsX)[1]
    #number of periods/ months
    months <- dim(retsX)[2]
    for (i in seq(1, numstocks)){
        optimals <- monthlyoptimalEHR_dma(retsX[i, 1:months])
        #add optimals to list
        m <- c(m, optimals$monthlyoptimal_m)
        r <- c(r, optimals$monthlyoptimal_r)
    }

    #return optimal rules
    list("monthlyoptimals_m"=m, "monthlyoptimals_r"=r)
}
```

Table 5: DJ double MA trading rules for optimal expected 12-period holding period return

tickers	m	r
AAPL	12	10
AXP	10	3
BA	12	7
CAT	9	4
CSCO	12	5
CVX	12	6
DIS	5	3
DWDP	5	3
GS	4	1
HD	12	7

tickers	m	r
IBM	12	11
INTC	8	3
JNJ	2	1
JPM	2	1
KO	7	6
MCD	9	5
MMM	9	8
MRK	12	6
MSFT	12	8
NKE	8	7
PFE	12	8
PG	5	1
TRV	2	1
UNH	12	7
UTX	2	1
V	5	4
VZ	9	7
WBA	11	10
WMT	9	8
XOM	12	11