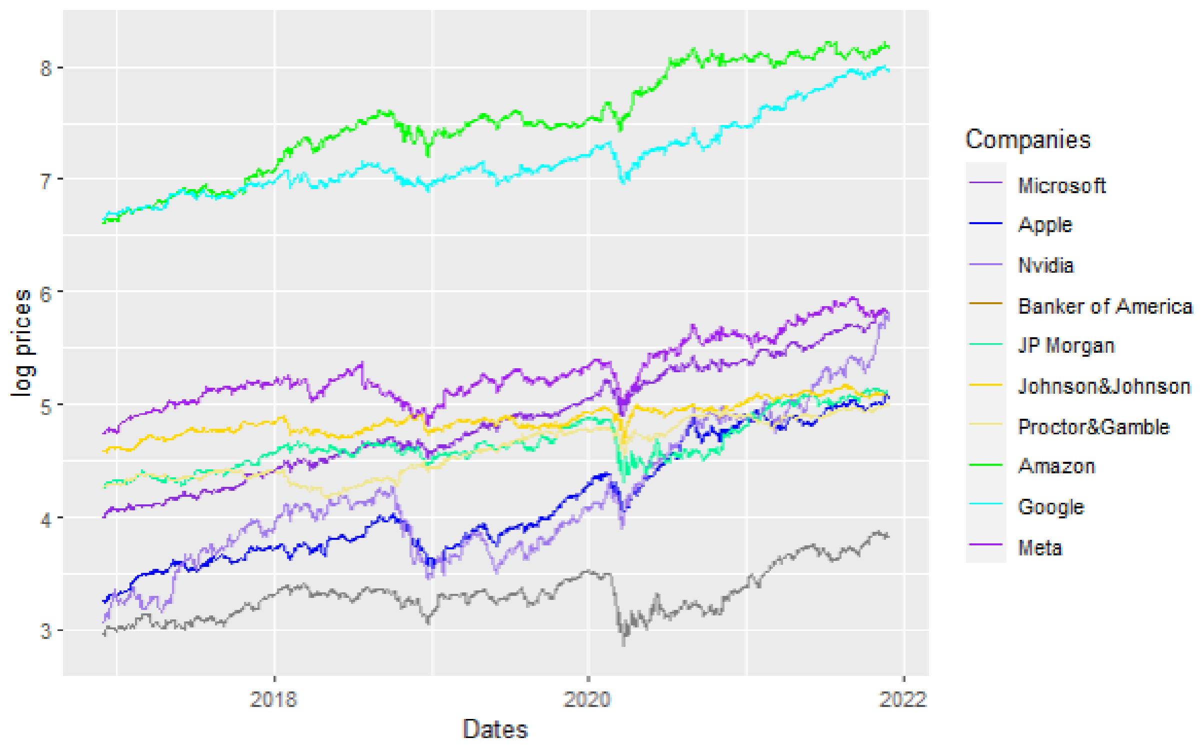
ST326 Assessed Coursework

Candidate No: 26617

Question 1



Results:

- Condition Number for 5 Years = 51.39345
- Condition Number for 3 Years = 76.12069
- Condition Number for 1 Year = 78.73177

The condition number represents how much small changes can affect the change in covariance. Here, we can see that the condition number decreases for longer periods showing that the covariance is less sensitive to changes in returns. If the covariance is too high, the covariance matrix would be considered as 'ill conditioned'.

Question 2

- Portfolio Weights Matrix:

The columns represent the weights for each stock respectively whereas the rows represent the n-th window.

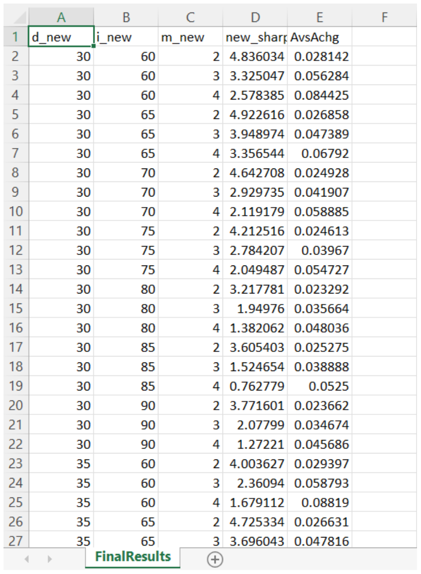
```
               [,1]          [,2]          [,3]          [,4]           [,5]          [,6]          [,7]          [,8]          [,9]         [,10]
 [1,]  0.0131046793  0.0166617761 -0.0022283697  0.0145271053 8  0.00778338505 -0.0038102723  0.0046111021 -0.0221237575  0.0091154315  0.0002565772
 [2,] -0.0003914665 -0.0120834804  0.0108734832  0.0126423391 9 -0.00968710731  0.0047779920 -0.0064526277  0.0146691413  0.0090304152 -0.0193024611
 [3,]  0.0164694143  0.0046582166 -0.0137498960 -0.0187417970 3  0.01573704059  0.0020497205  0.0000972153  0.0019352272 -0.0063791942  0.0033924681
 [4,] -0.0024621490 -0.0026932106  0.0062800610  0.0137575643 5 -0.01603820903  0.0056150051  0.0007560620  0.0131721083  0.0096960040 -0.0037720114
 [5,]  0.0023556845 -0.0075969254  0.0084832171  0.0000516192 4 -0.00069809797 -0.0006347625  0.0107042915  0.0042363307 -0.0080650240 -0.0102443549
 [6,]  0.0138689495  0.0111275631  0.0022102621 -0.0158896217 3  0.00047364622 -0.0027383091 -0.0203473666  0.0167611565  0.0069742439 -0.0271899957
 [7,]  0.0109972630  0.0050586555  0.0182213534 -0.0006413498 9  0.00917206790 -0.0103959654 -0.0058502149 -0.0072197465 -0.0160650373  0.0120019633
 [8,]  0.0121758685  0.0056930919  0.0045678253 -0.0152821150 1 -0.00303707222  0.0005045209 -0.0132870199  0.0181750139  0.0146789892  0.0071935016
 [9,]  0.0107163654 -0.0126169137 -0.0008441502  0.0094252174 6 -0.00968388433 -0.0019846793 -0.0213180898  0.0228190232  0.0001535751  0.0055528753
[10,]  0.0043459018 -0.0077661285 -0.0013003946  0.0042554718 5  0.00866276540 -0.0042445228  0.0161123246 -0.0230425790 -0.0043484078  0.0089579017
[11,]  0.0122536347  0.0082512733 -0.0078014118  0.0008482432 9  0.00311771214 -0.0012545619 -0.0114912506  0.0110702288  0.0076339934  0.0103049921
[12,]  0.0126670509  0.0048013310  0.0017038344 -0.0104960011 9  0.00265917790 -0.0081314342 -0.0111889473  0.0184841953 -0.0005045967 -0.0015460960
[13,]  0.0015766437  0.0060129381 -0.0165424710  0.0006885604 1 -0.00433923128  0.0036580561 -0.0087711344  0.0141322899 -0.0087125383  0.0091469255
[14,]  0.0098265563  0.0073719039 -0.0105940007 -0.0100251506 6 -0.00248695900  0.0025066402  0.0026977270  0.0057037928  0.0024891054 -0.0006804460
[15,]  0.0067616480  0.0029582324  0.0037117454  0.0000203746 7 -0.00183999733  0.0003510216 -0.0068787597 -0.0014852913  0.0141302283  0.0006676113
[16,]  0.0031956014 -0.0015902677  0.0025317787 -0.0042490041 2 -0.00176590092  0.0015509830 -0.0003593856 -0.0001032624  0.0025777747 -0.0019400927
[17,] -0.0025813810  0.0045851048  0.0014282673 -0.0012283137 1  0.00058217915  0.0007481487 -0.0006451363  0.0007844097 -0.0011284362 -0.0011235594
[18,] -0.0008239972  0.0003216216  0.0011283723 -0.0024475982 4  0.00005168046  0.0009906659 -0.0018550920  0.0018634173 -0.0029647138  0.0075187337
[19,] -0.0086146259 -0.0004832579  0.0023953993  0.0047702868 4 -0.00247500713  0.0028562852 -0.0030871974  0.0053329764 -0.0039537416  0.0044717823
[20,] -0.0003035721  0.0021496105  0.0027602380  0.0025396652 0 -0.00420746487 -0.0006530688 -0.0032824237  0.0039729836  0.0039610530 -0.0042105453
[21,]  0.0147300043 -0.0047113098 -0.0078152116 -0.0009130678 0  0.00502197974 -0.0007900742  0.0049684742  0.0017326769 -0.0045942088 -0.0006702737
[22,] -0.0030067506 -0.0011071030 -0.0042959405  0.0056079235 9 -0.00003920303  0.0096306055  0.0099472349 -0.0127661569  0.0074826119 -0.0082174948
[23,]  0.0063296783  0.0041969099 -0.0026542746  0.0084271665 4 -0.00397183019  0.0019338251 -0.0010128548  0.0014063002 -0.0126359354  0.0077464892
```

- Actual Portfolio Returns Vector:

```
 [1] 0.004302202 0.001676419 0.002790726 0.007936236 0.007451894 0.005214206 0.004590902 0.007079834 0.002402769 0.003051465 0.007030528 0.002319708
[13] 0.005911825 0.004853638 0.005210535 0.001088308 0.002376646 0.001618484 0.001475593 0.002376786 0.004216996 0.003102510 0.006084842
```

- Sharpe Ratio = 4.34264

Question 3



	A	B	C	D	E	F
1	d_new	i_new	m_new	new_sharp	AvsAchg	
2	30	60	2	4.836034	0.028142	
3	30	60	3	3.325047	0.056284	
4	30	60	4	2.578385	0.084425	
5	30	65	2	4.922616	0.026858	
6	30	65	3	3.948974	0.047389	
7	30	65	4	3.356544	0.06792	
8	30	70	2	4.642708	0.024928	
9	30	70	3	2.929735	0.041907	
10	30	70	4	2.119179	0.058885	
11	30	75	2	4.212516	0.024613	
12	30	75	3	2.784207	0.03967	
13	30	75	4	2.049487	0.054727	
14	30	80	2	3.217781	0.023292	
15	30	80	3	1.94976	0.035664	
16	30	80	4	1.382062	0.048036	
17	30	85	2	3.605403	0.025275	
18	30	85	3	1.524654	0.038888	
19	30	85	4	0.762779	0.0525	
20	30	90	2	3.771601	0.023662	
21	30	90	3	2.07799	0.034674	
22	30	90	4	1.27221	0.045686	
23	35	60	2	4.003627	0.029397	
24	35	60	3	2.36094	0.058793	
25	35	60	4	1.679112	0.08819	
26	35	65	2	4.725334	0.026631	
27	35	65	3	3.696043	0.047816	

FinalResults

d_new	i_new	m_new	new_sharpe_ratio	AvsAchg
30	60	2	4.836034191	0.028141756
30	60	3	3.325047102	0.056283512
30	60	4	2.57838546	0.084425268
30	65	2	4.922615779	0.026858037
30	65	3	3.948974285	0.047389018
30	65	4	3.356543986	0.06792
30	70	2	4.642708145	0.024928128
30	70	3	2.92973531	0.041906635
30	70	4	2.119179074	0.058885142
30	75	2	4.212515782	0.024613462
30	75	3	2.784206688	0.039670015
30	75	4	2.049486612	0.054726568
30	80	2	3.217780835	0.023292446
30	80	3	1.949759936	0.035664019
30	80	4	1.382062184	0.048035592
30	85	2	3.6054027	0.025275144
30	85	3	1.524653824	0.038887587
30	85	4	0.762779002	0.052500031
30	90	2	3.771601481	0.023662407
30	90	3	2.07799035	0.03467434
30	90	4	1.272210113	0.045686273

I have shown the first few combinations. I have 147 different combinations of parameters with the corresponding Sharpe ratio and average absolute change in portfolio weights which would be inconvenient to add to this document. However, I can provide the whole csv file and script upon request.

In my calculations for Sharpe ratio, I have used the actual returns as stated in the question sheet however I have seen Sharpe ratio calculated using excess return instead.

In practice, we want a smaller AvsAchg as this would mean we need to rebalance the portfolio less often as well as rebalance on a smaller scale. One of the reasons may be because rebalancing leads to costs through commissions and spreads involved when making trades.

Looking at the different combination of parameters and the corresponding values, I have noticed that lower m tends to lead to low AvsAchg whilst a greater value for both d and I leads to higher Sharpe ratio.

Since we want lower AvsAchg and higher Sharpe ratio, I believe the following achieves the best balance between Sharpe ratio and AvsAchg.

d = 55

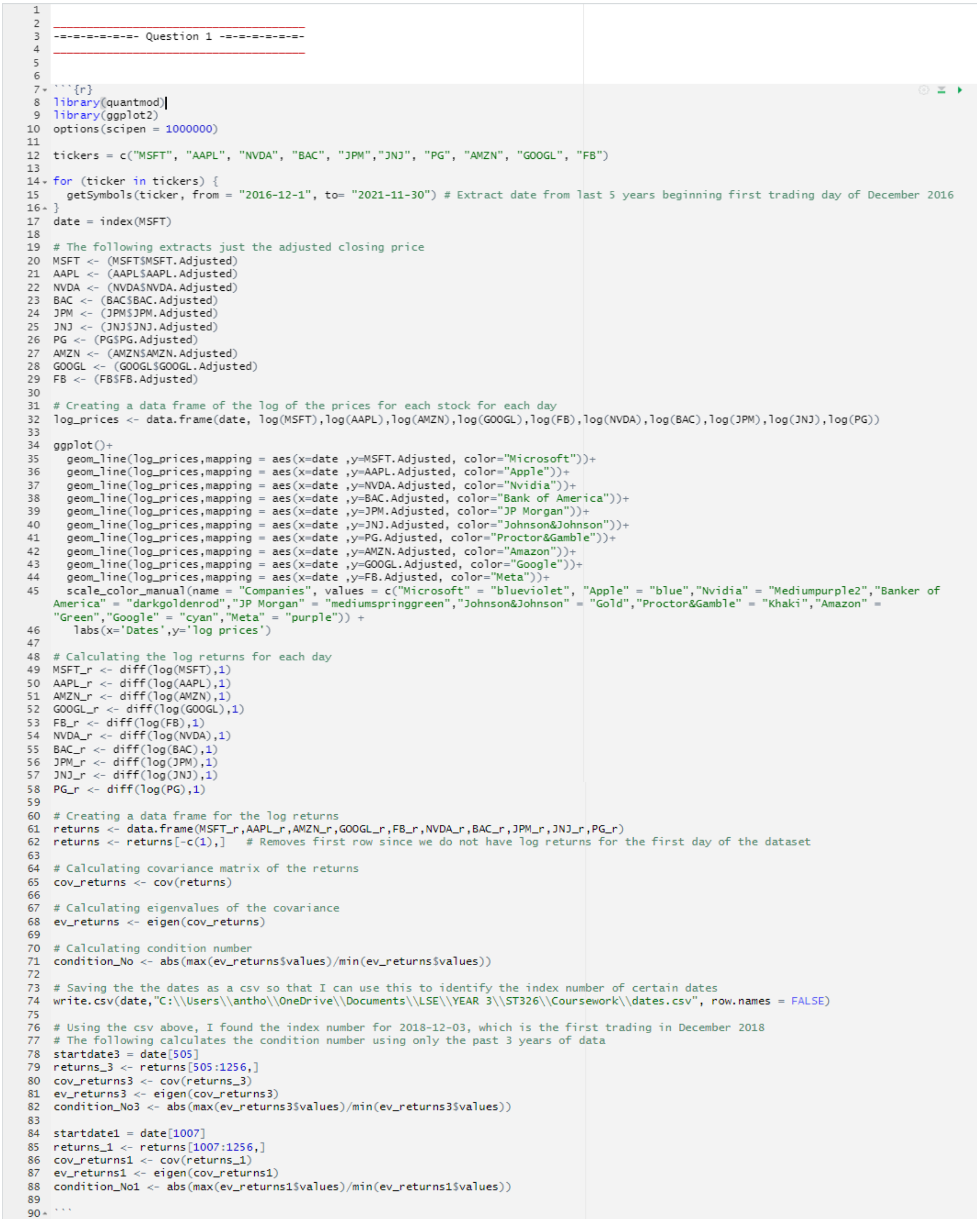
i = 85

m = 2

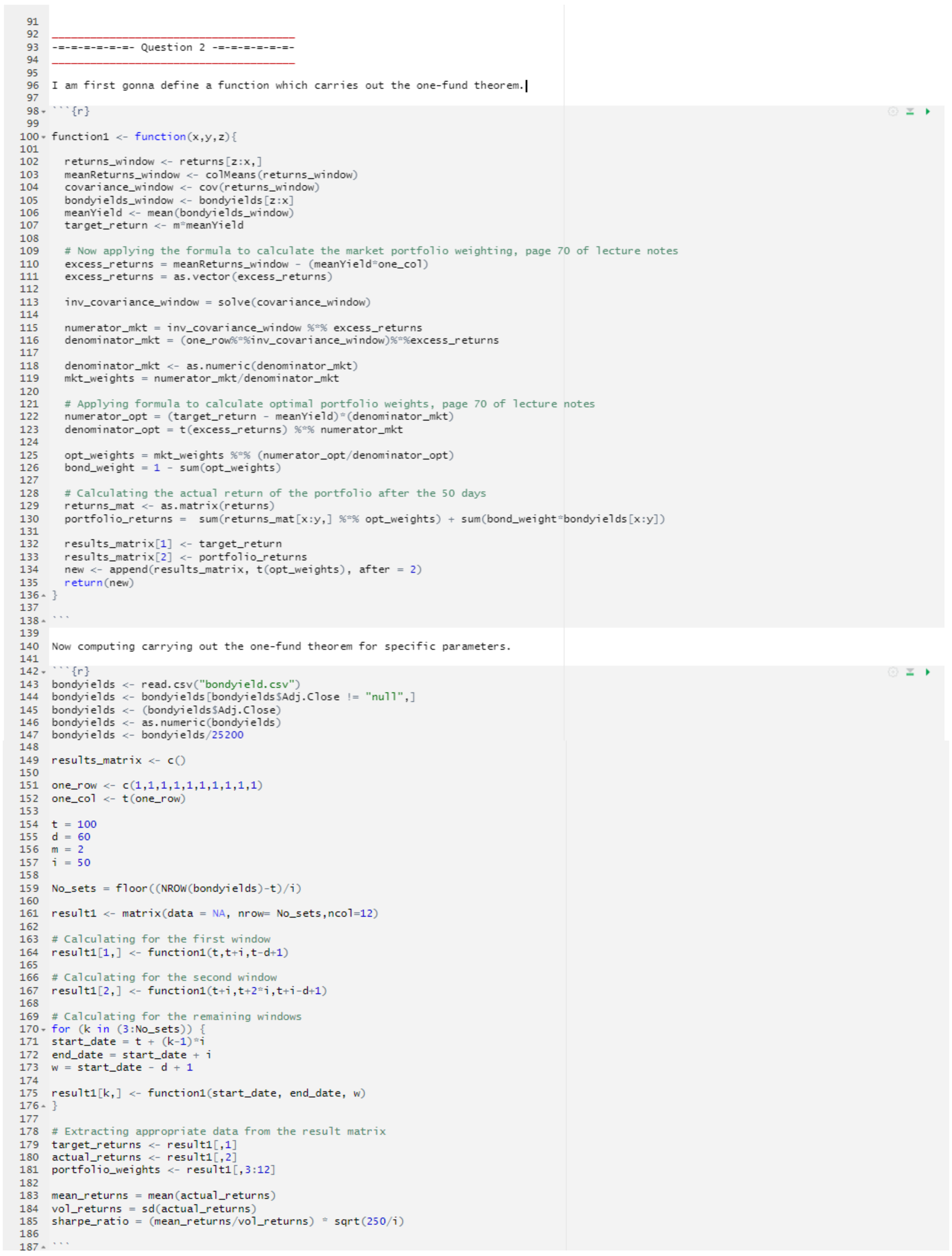
Below, I have included a short section of the table when Sharpe ratio is in order from largest to smallest.

d_new	i_new	m_new	new_sharpe_ratio	AvsAchg
55	85	2	5.808097	0.027898
60	85	2	5.640041	0.03033
50	85	2	5.397259	0.026789
30	65	2	4.922616	0.026858
30	60	2	4.836034	0.028142
35	65	2	4.725334	0.026631
30	70	2	4.642708	0.024928
35	75	2	4.557116	0.025915
40	75	2	4.546796	0.029327
55	70	2	4.513191	0.032819
60	75	2	4.493631	0.033459
40	70	2	4.437361	0.029559
30	75	2	4.212516	0.024613
45	85	2	4.148544	0.027984
55	75	2	4.109265	0.031198
35	70	2	4.103069	0.026472

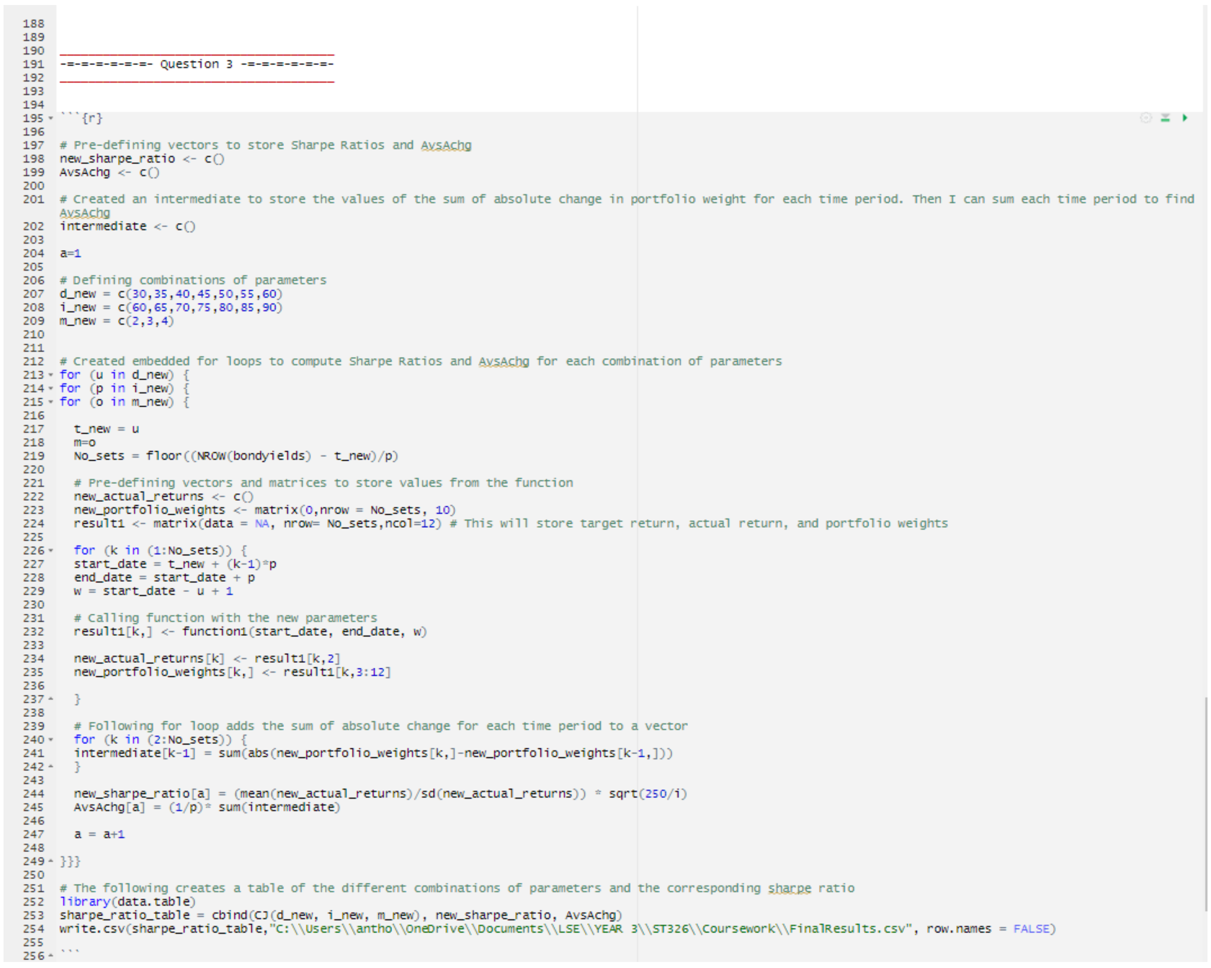
Appendix



~~~

-=-=-=-=-=-=- Question 1 -=-=-=-=-=-=-

```{r}
library(quantmod)
library(ggplot2)
options(scipen = 1000000)

tickers = c("MSFT", "AAPL", "NVDA", "BAC", "JPM","JNJ", "PG", "AMZN", "GOOGL", "FB")

for (ticker in tickers) {
  getSymbols(ticker, from = "2016-12-1", to= "2021-11-30") # Extract date from last 5 years beginning first trading day of December 2016
}
date = index(MSFT)

# The following extracts just the adjusted closing price
MSFT <- (MSFT$MSFT.Adjusted)
AAPL <- (AAPL$AAPL.Adjusted)
NVDA <- (NVDA$NVDA.Adjusted)
BAC <- (BAC$BAC.Adjusted)
JPM <- (JPM$JPM.Adjusted)
JNJ <- (JNJ$JNJ.Adjusted)
PG <- (PG$PG.Adjusted)
AMZN <- (AMZN$AMZN.Adjusted)
GOOGL <- (GOOGL$GOOGL.Adjusted)
FB <- (FB$FB.Adjusted)

# Creating a data frame of the log of the prices for each stock for each day
log_prices <- data.frame(date, log(MSFT),log(AAPL),log(AMZN),log(GOOGL),log(FB),log(NVDA),log(BAC),log(JPM),log(JNJ),log(PG))

ggplot()+
  geom_line(log_prices,mapping = aes(x=date ,y=MSFT.Adjusted, color="Microsoft"))+
  geom_line(log_prices,mapping = aes(x=date ,y=AAPL.Adjusted, color="Apple"))+
  geom_line(log_prices,mapping = aes(x=date ,y=NVDA.Adjusted, color="Nvidia"))+
  geom_line(log_prices,mapping = aes(x=date ,y=BAC.Adjusted, color="Bank of America"))+
  geom_line(log_prices,mapping = aes(x=date ,y=JPM.Adjusted, color="JP Morgan"))+
  geom_line(log_prices,mapping = aes(x=date ,y=JNJ.Adjusted, color="Johnson&Johnson"))+
  geom_line(log_prices,mapping = aes(x=date ,y=PG.Adjusted, color="Proctor&Gamble"))+
  geom_line(log_prices,mapping = aes(x=date ,y=AMZN.Adjusted, color="Amazon"))+
  geom_line(log_prices,mapping = aes(x=date ,y=GOOGL.Adjusted, color="Google"))+
  geom_line(log_prices,mapping = aes(x=date ,y=FB.Adjusted, color="Meta"))+
  scale_color_manual(name = "Companies", values = c("Microsoft" = "blueviolet", "Apple" = "blue","Nvidia" = "Mediumpurple2","Banker of
America" = "darkgoldenrod","JP Morgan" = "mediumspringgreen","Johnson&Johnson" = "Gold","Proctor&Gamble" = "Khaki","Amazon" =
"Green","Google" = "cyan","Meta" = "purple")) +
    labs(x='Dates',y='log prices')

# Calculating the log returns for each day
MSFT_r <- diff(log(MSFT),1)
AAPL_r <- diff(log(AAPL),1)
AMZN_r <- diff(log(AMZN),1)
GOOGL_r <- diff(log(GOOGL),1)
FB_r <- diff(log(FB),1)
NVDA_r <- diff(log(NVDA),1)
BAC_r <- diff(log(BAC),1)
JPM_r <- diff(log(JPM),1)
JNJ_r <- diff(log(JNJ),1)
PG_r <- diff(log(PG),1)

# Creating a data frame for the log returns
returns <- data.frame(MSFT_r,AAPL_r,AMZN_r,GOOGL_r,FB_r,NVDA_r,BAC_r,JPM_r,JNJ_r,PG_r)
returns <- returns[-c(1),]   # Removes first row since we do not have log returns for the first day of the dataset

# Calculating covariance matrix of the returns
cov_returns <- cov(returns)

# Calculating eigenvalues of the covariance
ev_returns <- eigen(cov_returns)

# Calculating condition number
condition_No <- abs(max(ev_returns$values)/min(ev_returns$values))

# Saving the the dates as a csv so that I can use this to identify the index number of certain dates
write.csv(date,"C:\\Users\\antho\\OneDrive\\Documents\\LSE\\YEAR 3\\ST326\\Coursework\\dates.csv", row.names = FALSE)

# Using the csv above, I found the index number for 2018-12-03, which is the first trading in December 2018
# The following calculates the condition number using only the past 3 years of data
startdate3 = date[505]
returns_3 <- returns[505:1256,]
cov_returns3 <- cov(returns_3)
ev_returns3 <- eigen(cov_returns3)
condition_No3 <- abs(max(ev_returns3$values)/min(ev_returns3$values))

startdate1 = date[1007]
returns_1 <- returns[1007:1256,]
cov_returns1 <- cov(returns_1)
ev_returns1 <- eigen(cov_returns1)
condition_No1 <- abs(max(ev_returns1$values)/min(ev_returns1$values))

```
~~~



~~~

-=-=-=-=-=-=- Question 2 -=-=-=-=-=-=-

I am first gonna define a function which carries out the one-fund theorem.

```{r}

function1 <- function(x,y,z){

  returns_window <- returns[z:x,]
  meanReturns_window <- colMeans(returns_window)
  covariance_window <- cov(returns_window)
  bondyields_window <- bondyields[z:x]
  meanYield <- mean(bondyields_window)
  target_return <- m*meanYield

  # Now applying the formula to calculate the market portfolio weighting, page 70 of lecture notes
  excess_returns = meanReturns_window - (meanYield*one_col)
  excess_returns = as.vector(excess_returns)

  inv_covariance_window = solve(covariance_window)

  numerator_mkt = inv_covariance_window %*% excess_returns
  denominator_mkt = (one_row%*%inv_covariance_window)%*%excess_returns

  denominator_mkt <- as.numeric(denominator_mkt)
  mkt_weights = numerator_mkt/denominator_mkt

  # Applying formula to calculate optimal portfolio weights, page 70 of lecture notes
  numerator_opt = (target_return - meanYield)*(denominator_mkt)
  denominator_opt = t(excess_returns) %*% numerator_mkt

  opt_weights = mkt_weights %*% (numerator_opt/denominator_opt)
  bond_weight = 1 - sum(opt_weights)

  # Calculating the actual return of the portfolio after the 50 days
  returns_mat <- as.matrix(returns)
  portfolio_returns =  sum(returns_mat[x:y,] %*% opt_weights) + sum(bond_weight*bondyields[x:y])

  results_matrix[1] <- target_return
  results_matrix[2] <- portfolio_returns
  new <- append(results_matrix, t(opt_weights), after = 2)
  return(new)
}

```

Now computing carrying out the one-fund theorem for specific parameters.

```{r}
bondyields <- read.csv("bondyield.csv")
bondyields <- bondyields[bondyields$Adj.Close != "null",]
bondyields <- (bondyields$Adj.Close)
bondyields <- as.numeric(bondyields)
bondyields <- bondyields/25200

results_matrix <- c()

one_row <- c(1,1,1,1,1,1,1,1,1,1)
one_col <- t(one_row)

t = 100
d = 60
m = 2
i = 50

No_sets = floor((NROW(bondyields)-t)/i)

result1 <- matrix(data = NA, nrow= No_sets,ncol=12)

# Calculating for the first window
result1[1,] <- function1(t,t+i,t-d+1)

# Calculating for the second window
result1[2,] <- function1(t+i,t+2*i,t+i-d+1)

# Calculating for the remaining windows
for (k in (3:No_sets)) {
start_date = t + (k-1)*i
end_date = start_date + i
w = start_date - d + 1

result1[k,] <- function1(start_date, end_date, w)
}

# Extracting appropriate data from the result matrix
target_returns <- result1[,1]
actual_returns <- result1[,2]
portfolio_weights <- result1[,3:12]

mean_returns = mean(actual_returns)
vol_returns = sd(actual_returns)
sharpe_ratio = (mean_returns/vol_returns) * sqrt(250/i)

```
~~~



~~~
-=-=-=-=-=-=- Question 3 -=-=-=-=-=-=-


```{r}

# Pre-defining vectors to store Sharpe Ratios and AvsAchg
new_sharpe_ratio <- c()
AvsAchg <- c()

# Created an intermediate to store the values of the sum of absolute change in portfolio weight for each time period. Then I can sum each time period to find
AvsAchg
intermediate <- c()

a=1

# Defining combinations of parameters
d_new = c(30,35,40,45,50,55,60)
i_new = c(60,65,70,75,80,85,90)
m_new = c(2,3,4)


# Created embedded for loops to compute Sharpe Ratios and AvsAchg for each combination of parameters
for (u in d_new) {
for (p in i_new) {
for (o in m_new) {

  t_new = u
  m=o
  No_sets = floor((NROW(bondyields) - t_new)/p)

  # Pre-defining vectors and matrices to store values from the function
  new_actual_returns <- c()
  new_portfolio_weights <- matrix(0,nrow = No_sets, 10)
  result1 <- matrix(data = NA, nrow= No_sets,ncol=12) # This will store target return, actual return, and portfolio weights

  for (k in (1:No_sets)) {
  start_date = t_new + (k-1)*p
  end_date = start_date + p
  w = start_date - u + 1

  # Calling function with the new parameters
  result1[k,] <- function1(start_date, end_date, w)

  new_actual_returns[k] <- result1[k,2]
  new_portfolio_weights[k,] <- result1[k,3:12]

  }

  # Following for loop adds the sum of absolute change for each time period to a vector
  for (k in (2:No_sets)) {
  intermediate[k-1] = sum(abs(new_portfolio_weights[k,]-new_portfolio_weights[k-1,]))
  }

  new_sharpe_ratio[a] = (mean(new_actual_returns)/sd(new_actual_returns)) * sqrt(250/i)
  AvsAchg[a] = (1/p)* sum(intermediate)

  a = a+1

}}}

# The following creates a table of the different combinations of parameters and the corresponding sharpe ratio
library(data.table)
sharpe_ratio_table = cbind(CJ(d_new, i_new, m_new), new_sharpe_ratio, AvsAchg)
write.csv(sharpe_ratio_table,"C:\\Users\\antho\\OneDrive\\Documents\\LSE\\YEAR 3\\ST326\\Coursework\\FinalResults.csv", row.names = FALSE)

```
~~~