QF301. Homework #5.

2021-11-22

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination. I further pledge that I have not copied any material from a book, article, the Internet or any other source except where I have expressly cited the source.

By filling out the following fields, you are signing this pledge. No assignment will get credit without being pledged.

Name: Siddharth Iyer

CWID: 10447455

Date: 11/15/2021

Instructions

In this assignment, you should use R markdown to answer the questions below. Simply type your R code into embedded chunks as shown above. When you have completed the assignment, knit the document into a PDF file, and upload both the .pdf and .Rmd files to Canvas.

```
CWID = 10447455 #Place here your Campus wide ID number, this will personalize
#your results, but still maintain the reproducible nature of using seeds.
#If you ever need to reset the seed in this assignment, use this as your seed
#Papers that use -1 as this CWID variable will earn 0's so make sure you change
#this value before you submit your work.
personal = CWID %% 10000
set.seed(personal)#You can reset the seed at any time in your code,
#but please always set it to this seed.
```

Question 1 (40pt)

Question 1.1

Use the quantmod package to obtain the daily adjusted close prices 15 different stocks. You should have at least 5 years of data for all assets. You should inspect the dates for your data to make sure you are including everything appropriately. Create a data frame of the daily log returns of all stocks. Print the first 6 lines of your data frame.

Solution:

```
library(quantmod)
stocks = c("MMM", "ABT", "AMD", "AAP", "AFL", "GOOG", "AMZN", "AAL", "AXP", "FDX", "AAPL", "T", "BBY",

for(stock in stocks){
  getSymbols(stock, from="2010-01-01", to="2019-12-31", src="yahoo")
}
```

```r
price_df = data.frame(MMM$MMM.Adjusted, ABT$ABT.Adjusted, AMD$AMD.Adjusted,
                      AAP$AAP.Adjusted, AFL$AFL.Adjusted, GOOG$GOOG.Adjusted,
                      AMZN$AMZN.Adjusted, AAL$AAL.Adjusted, AXP$AXP.Adjusted,
                      FDX$FDX.Adjusted, AAPL$AAPL.Adjusted, T$T.Adjusted,
                      BBY$BBY.Adjusted, BLK$BLK.Adjusted, BK$BK.Adjusted)
colnames(price_df) <- stocks

rets_df = data.frame(diff(log(MMM$MMM.Adjusted))[-1], diff(log(ABT$ABT.Adjusted))[-1], diff(log(AMD$AMD
                      diff(log(AAP$AAP.Adjusted))[-1], diff(log(AFL$AFL.Adjusted))[-1], diff(log(GOOG$GOO
                      diff(log(AMZN$AMZN.Adjusted))[-1], diff(log(AAL$AAL.Adjusted))[-1],
                      diff(log(AXP$AXP.Adjusted))[-1], diff(log(FDX$FDX.Adjusted))[-1], ... = diff(log(AA
                      diff(log(T$T.Adjusted))[-1], diff(log(BBY$BBY.Adjusted))[-1], diff(log(BLK$BLK.Adju

colnames(rets_df) <- stocks

head(rets_df)
```

```
##                      MMM          ABT          AMD           AAP          AFL
## 2010-01-05 -0.0062831590 -0.008112179  0.001030397 -0.0059610481  0.028596976
## 2010-01-06  0.0140825380  0.005537955 -0.014523077  0.0086816850  0.008746232
## 2010-01-07  0.0007169748  0.008250426 -0.010504298 -0.0002469077  0.010675486
## 2010-01-08  0.0070211566  0.005099049 -0.004232811  0.0039446398 -0.010068227
## 2010-01-11 -0.0040397530  0.005073279 -0.031235711 -0.0098913056  0.025969850
## 2010-01-12  0.0008328642 -0.002895886 -0.055101065 -0.0175482304 -0.005140018
##                    GOOG         AMZN         AAL           AXP          FDX
## 2010-01-05 -0.004413395  0.005882649  0.10724545 -0.0022014818  0.012976627
## 2010-01-06 -0.025531931 -0.018281786 -0.04231398  0.0160352184 -0.008314594
## 2010-01-07 -0.023554756 -0.017159620  0.02904362  0.0160888180 -0.010913498
## 2010-01-08  0.013243041  0.026716859 -0.01926818 -0.0007149556  0.024537071
## 2010-01-11 -0.001512745 -0.024335098 -0.01964696 -0.0115084688  0.026244121
## 2010-01-12 -0.017842125 -0.022977026  0.00790504  0.0131754886 -0.007708635
##                    AAPL             T           BBY          BLK           BK
## 2010-01-05  0.001727480 -0.0049108772  0.0250627941  0.004307533  0.010548796
## 2010-01-06 -0.016034380 -0.0147406243 -0.0077953392 -0.020832191 -0.015154394
## 2010-01-07 -0.001850180 -0.0112912442  0.0157713425  0.010934231  0.041389766
## 2010-01-08  0.006626417 -0.0073524933 -0.0400300142  0.007014269  0.006114086
## 2010-01-11 -0.008860611 -0.0048093338 -0.0171851858  0.016150564 -0.017421472
## 2010-01-12 -0.011440302 -0.0003705512  0.0007643574 -0.018035496  0.001721518
```

Question 1.2

Cluster these stocks based on the log returns.
Use K-Means Clustering with at least 20 attempts at clustering. Choose the number of clusters to use. Justify your choice in 1 paragraph (or less). Print the clusters.

Solution:

```r
model2 = kmeans(t(rets_df), nstart=20, centers = 2)
model3 = kmeans(t(rets_df), nstart=20, centers = 3)
model4 = kmeans(t(rets_df), nstart=20, centers = 4)
```

```r
model5 = kmeans(t(rets_df), nstart=20, centers = 5)
model6 = kmeans(t(rets_df), nstart=20, centers = 6)
model7 = kmeans(t(rets_df), nstart=20, centers = 7)
model8 = kmeans(t(rets_df), nstart=20, centers = 8)
model9 = kmeans(t(rets_df), nstart=20, centers = 9)
model10 = kmeans(t(rets_df), nstart=20, centers = 10)
model11 = kmeans(t(rets_df), nstart=20, centers = 11)
model12 = kmeans(t(rets_df), nstart=20, centers = 12)
model13 = kmeans(t(rets_df), nstart=20, centers = 13)
model14 = kmeans(t(rets_df), nstart=20, centers = 14)

print(sort(model2$cluster))
```

```
##  MMM  ABT  AAP  AFL GOOG AMZN  AAL  AXP  FDX AAPL    T  BBY  BLK   BK  AMD
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    2
```

```r
print(sort(model3$cluster))
```

```
##  AMD  MMM  ABT  AAP  AFL GOOG AMZN  AXP  FDX AAPL    T  BBY  BLK   BK  AAL
##    1    2    2    2    2    2    2    2    2    2    2    2    2    2    3
```

```r
print(sort(model4$cluster))
```

```
##  AMD  BBY  AAL  MMM  ABT  AAP  AFL GOOG AMZN  AXP  FDX AAPL    T  BLK   BK
##    1    2    3    4    4    4    4    4    4    4    4    4    4    4    4
```

```r
print(sort(model5$cluster))
```

```
##  BBY  AAP  AMD  MMM  ABT  AFL GOOG AMZN  AXP  FDX AAPL    T  BLK   BK  AAL
##    1    2    3    4    4    4    4    4    4    4    4    4    4    4    5
```

```r
print(sort(model6$cluster))
```

```
##  BBY  AAP GOOG AMZN AAPL  AAL  AMD  MMM  ABT  AFL  AXP  FDX    T  BLK   BK
##    1    2    3    3    3    4    5    6    6    6    6    6    6    6    6
```

```r
print(sort(model7$cluster))
```

```
##  AAP  MMM  ABT  AFL  AXP  FDX    T  BLK   BK GOOG AAPL  BBY  AMD  AAL AMZN
##    1    2    2    2    2    2    2    2    2    3    3    4    5    6    7
```

```r
print(sort(model8$cluster))
```

```
## GOOG AAPL  MMM  ABT    T AMZN  AAL  AAP  BBY  AFL  AXP  FDX  BLK   BK  AMD
##    1    1    2    2    2    3    4    5    6    7    7    7    7    7    8
```

```r
print(sort(model9$cluster))
```

```
## AMZN AAPL  AAP  AFL  AXP  FDX  BLK   BK  BBY  MMM  ABT    T GOOG  AAL  AMD
##    1    2    3    4    4    4    4    4    5    6    6    6    7    8    9
```

```r
print(sort(model10$cluster))
```

```
##  AAP  AFL  AXP  BLK   BK  FDX  MMM  ABT    T AMZN AAPL GOOG  AAL  AMD  BBY
##    1    2    2    2    2    3    4    4    4    5    6    7    8    9   10
```

```r
print(sort(model11$cluster))
```

```
##  AFL  BLK   BK  AAP  AMD GOOG  AXP AAPL  FDX  MMM  ABT    T  AAL  BBY AMZN
##    1    1    1    2    3    4    5    6    7    8    8    8    9   10   11
```

```r
print(sort(model12$cluster))
```

```
##  MMM  ABT    T  AMD  AAP  BLK   BK  AFL  BBY  AXP GOOG  AAL AAPL  FDX AMZN
##    1    1    1    2    3    4    4    5    6    7    8    9   10   11   12
```

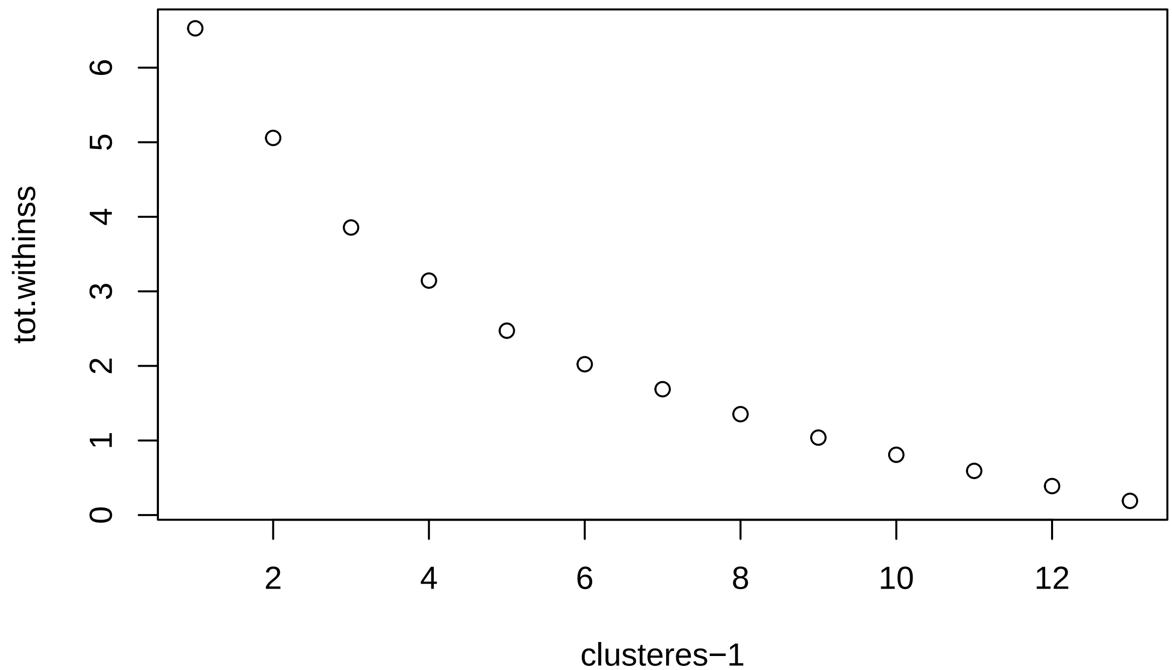
```r
print(sort(model13$cluster))
```

```
##  AAP  BLK   BK AMZN  MMM  ABT  AAL  AFL  FDX  AMD AAPL  BBY    T GOOG  AXP
##    1    2    2    3    4    4    5    6    7    8    9   10   11   12   13
```

```r
print(sort(model14$cluster))
```

```
## GOOG   BK AAPL  FDX  BLK  AAL    T  MMM  ABT AMZN  AXP  AFL  AMD  BBY  AAP
##    1    2    3    4    5    6    7    8    8    9   10   11   12   13   14
```

```r
plot(c(model2$tot.withinss, model3$tot.withinss,model4$tot.withinss, model5$tot.withinss, model6$tot.wi
```



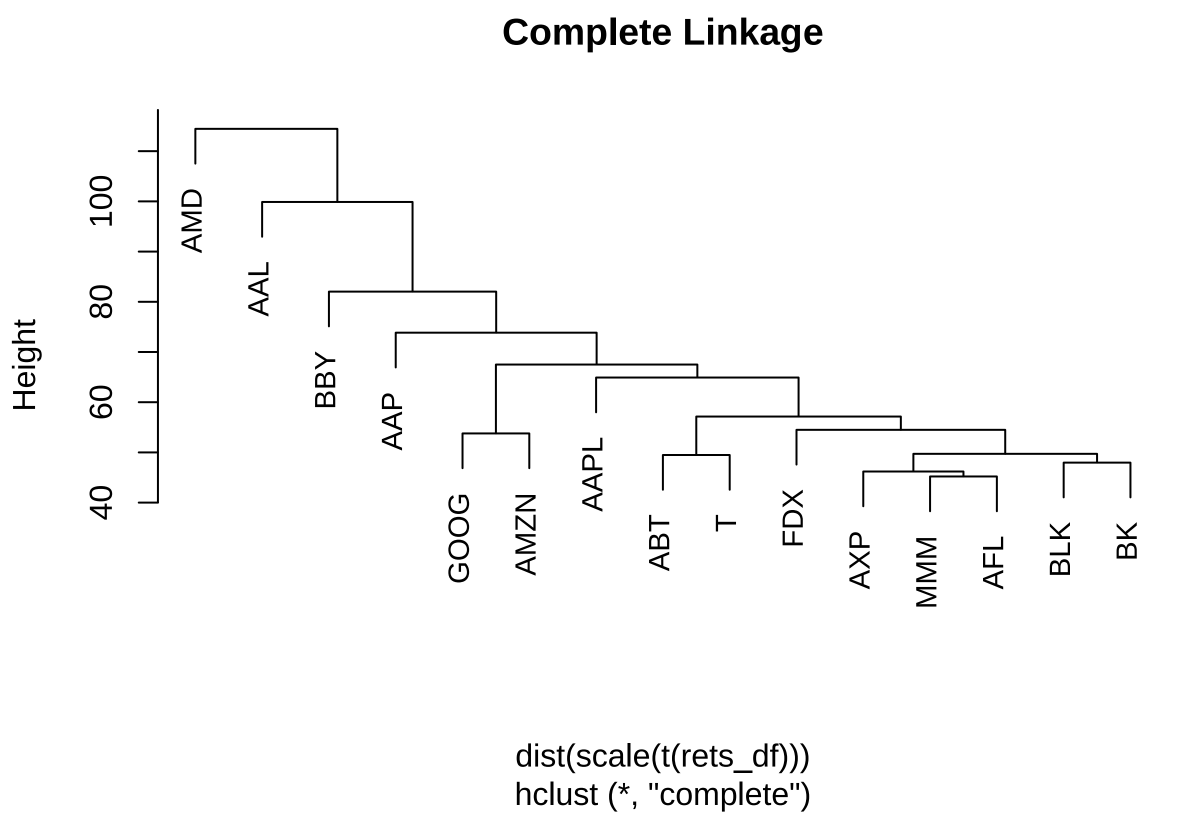
I would use 3 clusters (note: on the graph 2 represents 3 clusters)…the largest drop in tot.withinns occurs between 2 and 3 clusters. Obviously the withinss will keep falling as you add more clusters, but the largest drop occurs between 2 and 3.

Question 1.3

Cluster these stocks based on the log returns.
Use Hierarchical Clustering with complete distance metric and print the dendrogram. Choose the number of clusters to use. Justify your choice in 1 paragraph (or less). Print the clusters.

Solution:

```
hc.complete.rets = hclust(dist(scale(t(rets_df))), method="complete")
plot(hc.complete.rets,main="Complete Linkage",cex=.9)
```




```
sort(cutree(hc.complete.rets,6))
```

```
##  MMM  ABT  AFL  AXP  FDX AAPL    T  BLK   BK  AMD  AAP GOOG AMZN  AAL  BBY
##    1    1    1    1    1    1    1    1    1    2    3    4    4    5    6
```

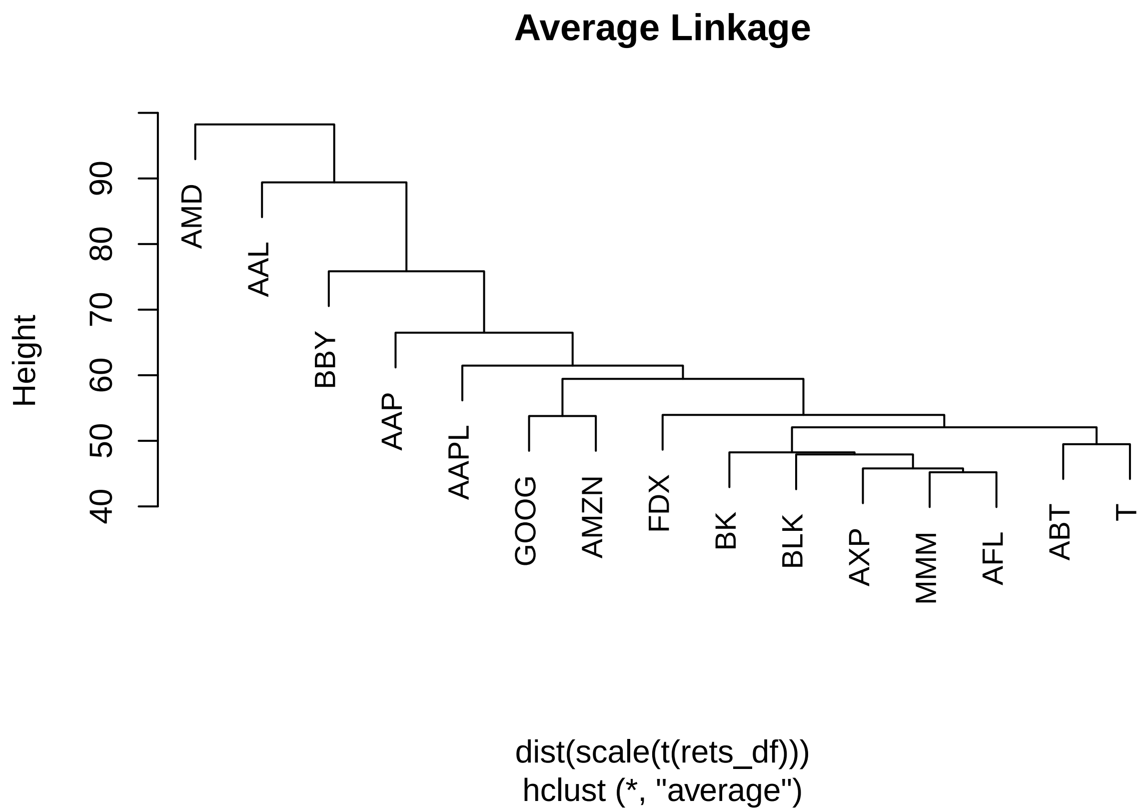
I would use about 3 clusters…the largest drop is in the 3rd level If you follow the children in the 3rd level, you have about 3 clusters.

Question 1.4

Cluster these stocks based on the log returns.
Use Hierarchical Clustering with average distance metric and print the dendrogram. Choose the number of clusters to use. Justify your choice in 1 paragraph (or less). Print the clusters.

Solution:

```
hc.average.rets = hclust(dist(scale(t(rets_df))), method="average")
plot(hc.average.rets,main="Average Linkage",cex=.9)
```




```
sort(cutree(hc.average.rets,3))
```

```
##  MMM  ABT  AAP  AFL GOOG AMZN  AXP  FDX AAPL    T  BBY  BLK   BK  AMD  AAL
##    1    1    1    1    1    1    1    1    1    1    1    1    1    2    3
```

I would choose 3 clusters. The largest drop is in the second level, which again means 3 clusters. It's almost identical to problem 1.

Question 1.5

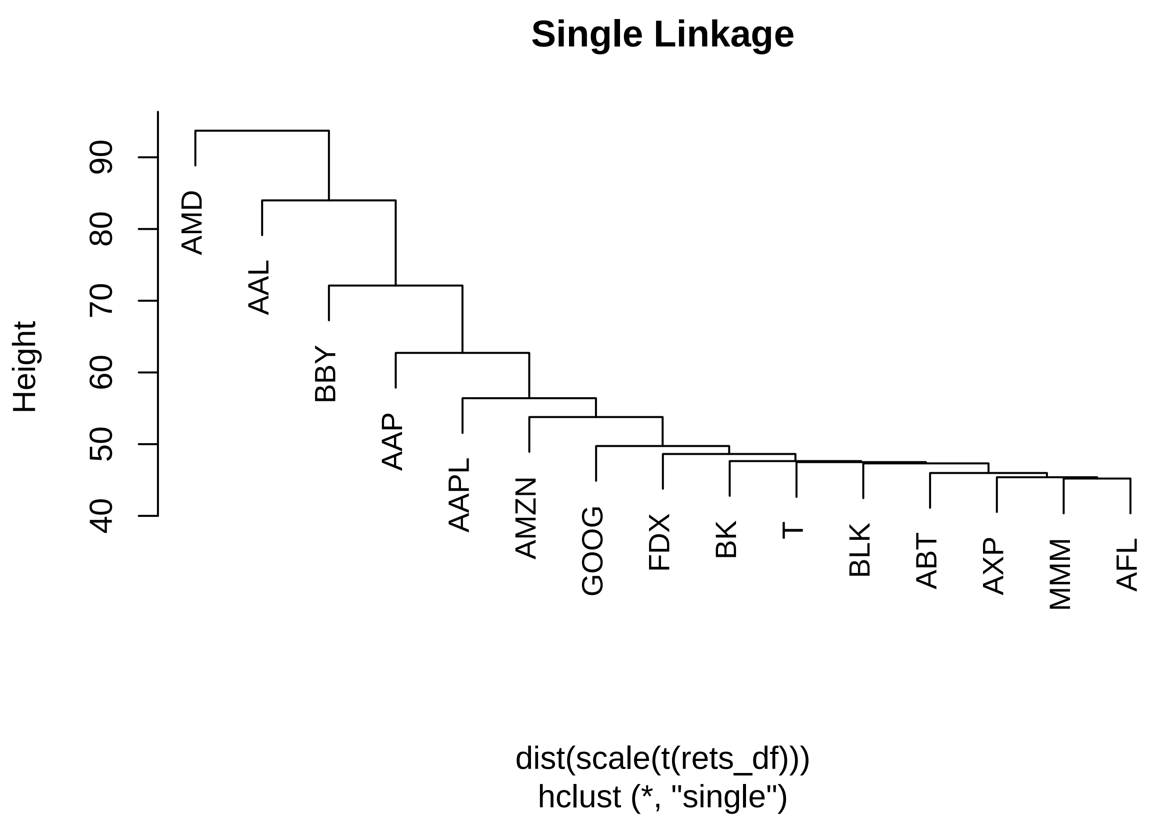
Cluster these stocks based on the log returns.
Use Hierarchical Clustering with single distance metric and print the dendrogram. Choose the number of clusters to use. Justify your choice in 1 paragraph (or less). Print the clusters.

Solution:

```
hc.single.rets = hclust(dist(scale(t(rets_df))), method="single")
cutree(hc.single.rets,2)
```

```
##  MMM  ABT  AMD  AAP  AFL GOOG AMZN  AAL  AXP  FDX AAPL    T  BBY  BLK   BK
##    1    1    2    1    1    1    1    1    1    1    1    1    1    1    1
```

```
plot(hc.single.rets,main="Single Linkage",cex=.9)
```



I would use 3 clusters because the largest drop occurs in the second level alongside BBY. This means AAL, AMD, the rest of the tree are 3 clusters.

Question 1.6

Of the clustering methods considered, which (if any) most closely matches your intuition. Explain briefly (1 paragraph) why you choose this fit.

Solution:

All the methods are quite bad…all of them produce clusters of uneven size and often group unrelated companies together. Also, the graphs are almost identical…so I guess the single is least computationally intensive.

Question 2 (20pt)

Question 2.1

Use the quantmod package to obtain the daily adjusted close prices for the SPY index and 15 different stocks. You should have at least 5 years of data for all assets. You should inspect the dates for your data to make sure you are including everything appropriately. You may use the same 15 stocks as in Question 1.

Create a data frame of the lagged daily log returns (single lag) of all stocks, lagged daily log returns (single lag) of the SPY index, and th (non-lagged) direction of the SPY index. Print the first 6 lines of your data frame.

Solution:

```
getSymbols("SPY", from="2010-01-01", to="2019-12-31", src="yahoo")
```

```
## [1] "SPY"
```

```
spy_prices = SPY$SPY.Adjusted
spy_rets = dailyReturn(spy_prices, type="log")[-1]
head(spy_rets)
```

```
##            daily.returns
## 2010-01-05  0.0026438162
## 2010-01-06  0.0007036038
## 2010-01-07  0.0042123913
## 2010-01-08  0.0033223391
## 2010-01-11  0.0013956942
## 2010-01-12 -0.0093700320
```

```
df = rets_df
df["SPY_lagged"] = spy_rets
new_r = rep(0, 16)
df = rbind(new_r, df)
df = df[-length(df),]
df["SPY"] = ((spy_rets > 0)+0)
df = df[-1,]
head(df)
```

```
##                      MMM          ABT          AMD           AAP          AFL
## 2010-01-05 -0.0062831590 -0.008112179  0.001030397 -0.0059610481  0.028596976
## 2010-01-06  0.0140825380  0.005537955 -0.014523077  0.0086816850  0.008746232
## 2010-01-07  0.0007169748  0.008250426 -0.010504298 -0.0002469077  0.010675486
## 2010-01-08  0.0070211566  0.005099049 -0.004232811  0.0039446398 -0.010068227
## 2010-01-11 -0.0040397530  0.005073279 -0.031235711 -0.0098913056  0.025969850
## 2010-01-12  0.0008328642 -0.002895886 -0.055101065 -0.0175482304 -0.005140018
##                    GOOG         AMZN         AAL           AXP          FDX
## 2010-01-05 -0.004413395  0.005882649  0.10724545 -0.0022014818  0.012976627
## 2010-01-06 -0.025531931 -0.018281786 -0.04231398  0.0160352184 -0.008314594
## 2010-01-07 -0.023554756 -0.017159620  0.02904362  0.0160888180 -0.010913498
## 2010-01-08  0.013243041  0.026716859 -0.01926818 -0.0007149556  0.024537071
## 2010-01-11 -0.001512745 -0.024335098 -0.01964696 -0.0115084688  0.026244121
## 2010-01-12 -0.017842125 -0.022977026  0.00790504  0.0131754886 -0.007708635
##                    AAPL             T           BBY          BLK           BK
## 2010-01-05  0.001727480 -0.0049108772  0.0250627941  0.004307533  0.010548796
## 2010-01-06 -0.016034380 -0.0147406243 -0.0077953392 -0.020832191 -0.015154394
## 2010-01-07 -0.001850180 -0.0112912442  0.0157713425  0.010934231  0.041389766
## 2010-01-08  0.006626417 -0.0073524933 -0.0400300142  0.007014269  0.006114086
```

```
## 2010-01-11 -0.008860611 -0.0048093338 -0.0171851858  0.016150564 -0.017421472
## 2010-01-12 -0.011440302 -0.0003705512  0.0007643574 -0.018035496  0.001721518
##               SPY_lagged SPY
## 2010-01-05  0.0026438162   1
## 2010-01-06  0.0007036038   1
## 2010-01-07  0.0042123913   1
## 2010-01-08  0.0033223391   1
## 2010-01-11  0.0013956942   0
## 2010-01-12 -0.0093700320   1
```

Question 2.2

Split your data into training and testing sets (80% training and 20% test).

Train a random forest classifier using 4 variables per tree and 500 trees in order to predict the direction of the SPY index. Print the summary of your classifier. Print the test accuracy and test confusion matrix.

Solution:

```
df = as.data.frame(df)
rownames(df) <- NULL

library(randomForest)
library(caret)

N = nrow(df)
train = sample(N, 4*N/5, replace = FALSE)
rf.model = randomForest(formula = as.factor(SPY) ~ ., data= df, proximity = TRUE, importance = TRUE, su

summary(rf.model)
```

```
##                 Length  Class  Mode
## call                  9 -none- call
## type                  1 -none- character
## predicted          2010 factor numeric
## err.rate           1500 -none- numeric
## confusion             6 -none- numeric
## votes              4020 matrix numeric
## oob.times          2010 -none- numeric
## classes               2 -none- character
## importance           64 -none- numeric
## importanceSD         48 -none- numeric
## localImportance       0 -none- NULL
## proximity       4040100 -none- numeric
## ntree                 1 -none- numeric
## mtry                  1 -none- numeric
## forest               14 -none- list
## y                  2010 factor numeric
## test                  0 -none- NULL
## inbag                 0 -none- NULL
## terms                 3 terms  call
```

```r
X_test = df[-train, -length(df)]
y_test = df[-train, length(df)]
y_test = as.vector(y_test)

y_pred = as.integer(as.vector(predict(rf.model, X_test)))

cat("Accuracy: ", 1 - sum(abs(y_test - y_pred))/length(y_pred), "\n")
```

```
## Accuracy:  0.9940358
```

```r
confusionMatrix(data = factor(y_pred), reference = factor(y_test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 245   2
##          1   1 255
##
##                Accuracy : 0.994
##                  95% CI : (0.9827, 0.9988)
##     No Information Rate : 0.5109
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9881
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9959
##             Specificity : 0.9922
##          Pos Pred Value : 0.9919
##          Neg Pred Value : 0.9961
##              Prevalence : 0.4891
##          Detection Rate : 0.4871
##    Detection Prevalence : 0.4911
##       Balanced Accuracy : 0.9941
##
##        'Positive' Class : 0
##
```

Question 2.3

Using the trained classifier from Question 2.2, run Mean Decrease in Impurity (MDI) analysis. Print the feature importance of all predictors.

Solution:

```r
rf.model$importance
```

```
##                      0           1 MeanDecreaseAccuracy MeanDecreaseGini
## MMM        0.0060015027 0.0034401092         0.0045609268        85.761739
## ABT        0.0016476099 0.0016016670         0.0016200498        16.299760
## AMD        0.0011795684 0.0009617911         0.0010584526         6.834876
## AAP        0.0006808287 0.0003151091         0.0004756834         5.216108
## AFL        0.0057285938 0.0010186796         0.0030741250        35.773238
## GOOG       0.0028906248 0.0030872266         0.0029836392        25.295696
## AMZN       0.0016427949 0.0026352152         0.0021906401        18.133784
## AAL        0.0004346437 0.0001251010         0.0002624159         4.283076
## AXP        0.0083221938 0.0037572876         0.0057404474        39.935167
## FDX        0.0022935053 0.0016288477         0.0019121904        27.718523
## AAPL       0.0009417614 0.0018921646         0.0014765685        12.101392
## T          0.0012657447 0.0014642809         0.0013743367         9.353492
## BBY        0.0006096544 0.0007598117         0.0006906401         5.293152
## BLK        0.0069033003 0.0035303157         0.0049958064        80.444682
## BK         0.0054005420 0.0039354655         0.0045728190        40.487581
## SPY_lagged 0.4650392921 0.3636478812         0.4072827880       573.797527
```

Question 2.4

Interpret the MDI feature importances (GINI) computed in Question 2.3.
Comment on the most and least important predictors (or if all predictors are of equal importance). Your response should be approximately 1 paragraph.

Solution:

The most important factor is the lagged SPY. The least important factor is AAP. This makes sense because more often than not, AAP is in a cluster all by itself, making it a significant features but less likely to change outcomes.

This suggests it's a strongly auto regressive time series.

Question 3 (20pt)

Question 3.1

Consider the same data and classification problem as in Question 2. Run Mean Decrease in Accuracy (MDA) analysis on a random forest classifier. Print the feature importance of each predictor.

Solution:

The code is already above

Question 3.2

Interpret the MDA feature importances computed in Question 3.1.
Comment on the most and least important predictors (or if all predictors are of equal importance). Does this match the MDI feature importances found in Question 2.3. Your response should be approximately 1 paragraph.

Solution:

. . . again the feature important shows that SPY_lagged is the most important predictor, BLK is second, and the weakest predictor is AAL (not AAP). So they aren't always consistent.

Question 4 (20pt)

Question 4.1

Consider the same data and classification problem as in Question 2. Run Principal Component Analysis (PCA) on the 16 predictors used for the classifer in Question 2 (and 3). Print the Proportion of Variance Explained (PCA) for each principal component. How many principal components are necessary to explain 80% of the variance?

Solution:

```
pca = prcomp(df, formula = as.factor(SPY) ~ .,scale = TRUE, subset = train)
pca$rotation
```

```
##                  PC1         PC2         PC3         PC4          PC5
## MMM        0.2743807 -0.07167570  0.15192601 -0.01992701 -0.040564259
## ABT        0.2359376  0.01016009  0.11205894  0.15051689 -0.106807810
## AMD        0.1769015  0.16185734 -0.18794080 -0.64766853 -0.199826835
## AAP        0.1523572 -0.20798860 -0.56287493  0.58970068  0.088197340
## AFL        0.2744764 -0.19700755  0.15265244 -0.01032412 -0.035558515
## GOOG       0.2365636  0.47245437  0.07468910  0.21780152  0.065161276
## AMZN       0.2130629  0.53982693 -0.04030615  0.17645507  0.121466316
## AAL        0.1903667 -0.12833767 -0.17921552 -0.24416058  0.759889374
## AXP        0.2715488 -0.12624528  0.06693964 -0.03208079  0.090683786
## FDX        0.2650273 -0.13145479 -0.01545112 -0.12027927  0.140835680
## AAPL       0.2130289  0.42757781 -0.09293170 -0.06155431 -0.056182066
## T          0.1998274 -0.20475937  0.21514283  0.13815003 -0.397496951
## BBY        0.1560201 -0.07893888 -0.67428982 -0.16593148 -0.372969402
## BLK        0.2942162 -0.15355043  0.11684305  0.00849321  0.007336543
## BK         0.2766549 -0.25497359  0.12546211 -0.05209030  0.027536028
## SPY_lagged 0.3477590  0.02084779  0.05782065  0.01831654 -0.039578610
## SPY        0.2528571  0.04659256  0.05041752  0.04905462 -0.104103715
##                   PC6          PC7          PC8          PC9         PC10
## MMM         0.096683113 -0.030532584  0.024098494  0.115477250  0.343427785
## ABT        -0.150601477  0.026841798 -0.822838785  0.252338282 -0.188448553
## AMD        -0.638941316  0.060681525  0.018846627 -0.091482805 -0.037059201
## AAP        -0.462583482  0.038746648  0.147638668  0.103633723  0.071973974
## AFL         0.029243085 -0.216071901  0.262469587  0.035558520 -0.178959472
## GOOG        0.093165238 -0.057776474  0.031152545 -0.125403829 -0.132319922
## AMZN       -0.027412976 -0.034727376  0.066886766 -0.493943020 -0.100805806
## AAL         0.161064170  0.413934276 -0.156463690 -0.043478864 -0.098744344
## AXP        -0.026926637 -0.186274169  0.088680901  0.042894739 -0.423820255
## FDX         0.029530019 -0.024375204  0.105773106 -0.067906046  0.488810818
## AAPL        0.149226797  0.192910296  0.255715202  0.728149393  0.043848356
## T           0.039029276  0.766124101  0.160275694 -0.213175305 -0.141051833
```

```
## BBY         0.531087209 -0.097527454 -0.129420872 -0.137001068 -0.122163546
## BLK         0.035401345 -0.177566193  0.042111042 -0.002484258  0.036240803
## BK         -0.014357903 -0.262168423  0.098540515 -0.030350250 -0.200107077
## SPY_lagged  0.006224228 -0.002459865  0.009151999  0.045127280  0.006738278
## SPY         0.023943122 -0.036858366 -0.250878106 -0.192850161  0.523297291
##                    PC11         PC12         PC13         PC14          PC15
## MMM        -0.321947624 -0.367177629  0.519094631  0.332383472  0.3074216124
## ABT        -0.222236449 -0.051114161 -0.148128294 -0.051541975 -0.0909063317
## AMD        -0.019797122  0.070271264  0.066712098  0.112572122  0.0113898207
## AAP         0.013054600  0.009671203  0.045866436  0.063714628  0.0229387314
## AFL        -0.032525379 -0.197246177 -0.239215536  0.443755996 -0.6272505640
## GOOG       -0.113919199  0.655350500  0.327890825  0.212339494 -0.1221662428
## AMZN       -0.108175782 -0.478502674 -0.273147400 -0.119701556  0.1344057036
## AAL         0.097061050 -0.002298173 -0.001959062  0.185912941  0.0004394102
## AXP         0.210565521 -0.181992744  0.517494509 -0.540937382 -0.1142756463
## FDX        -0.450733047  0.203195132 -0.153160426 -0.514127173 -0.2769689400
## AAPL        0.164095246 -0.063556200 -0.190438410 -0.115359596  0.0468045323
## T          -0.005919982  0.048860905 -0.011567688 -0.068773402  0.0186028632
## BBY        -0.066435399  0.007285661  0.021495723  0.025945081  0.0005552454
## BLK        -0.018654590  0.158731047 -0.236970393  0.045584151  0.4184304458
## BK          0.164726075  0.230666335 -0.276279660 -0.008000701  0.4219585738
## SPY_lagged  0.005329062 -0.028752283 -0.013975364  0.035789232 -0.0164854365
## SPY         0.710531392 -0.007566288  0.043504675  0.012630358 -0.1562898873
##                    PC16        PC17
## MMM         0.153242264 -0.12616095
## ABT         0.044077387 -0.10559512
## AMD        -0.031979900 -0.04636450
## AAP         0.011437208 -0.02979564
## AFL        -0.033296937 -0.15097749
## GOOG        0.021098201 -0.08311123
## AMZN        0.044952201 -0.07357203
## AAL        -0.012869844 -0.02368736
## AXP        -0.119497836 -0.08140704
## FDX         0.092763322 -0.06471523
## AAPL        0.037772799 -0.12271116
## T           0.009609212 -0.08686450
## BBY         0.004198411 -0.02218268
## BLK        -0.752192804 -0.12470144
## BK          0.615428185 -0.07802912
## SPY_lagged -0.021132215  0.93173162
## SPY         0.016320673 -0.10269619
```

```r
pca.var = pca$sdev^2
pve = pca.var/sum(pca.var)
print("PCA explained variance: ")
```

```
## [1] "PCA explained variance: "
```

```r
pve
```

```
##  [1] 0.455805216 0.061580382 0.055214807 0.048426507 0.045290405 0.043608142
##  [7] 0.040793674 0.035831351 0.035713246 0.032212328 0.031117674 0.024948693
## [13] 0.024080946 0.023162401 0.020868475 0.017598555 0.003747196
```

8 PCA components exlain slight over 80% of the variance in the data.

Question 4.2

Interpret the PCA computed in Question 4.1. Comment on the importance of different predictors. Does this match the MDI and MDA analysis from Questions 2 and 3? Your response should be approximately 1 paragraph.

Solution:

PCA1: SPY_lagged is the largest contributer PCA2: AMZN is the largest contributer in PCA2 PCA8: ABT is the largest controbuter in last PCA at 80% threshold

This doesn't match the MDI and MDA exactly but the PCA itself is a linear combination of factors, so makes senes that optimizing for variance will produce difference results than minimizing GINI.