FE590. Assignment #2.

2022-03-16

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<- "Darsh Kachhara"

CWID<- 10474181

Date<- 03/15/2022

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 = 10474181 #Place here your Campus wide ID number, this will personalize
#your results, but still maintain the reproduceable 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

Create a .csv file consisting of daily adjusted close prices for 10 different stocks and 2 ETF's. You should have at least two years of data for every asset and ETF and should include the date for your data to make sure that you are including everything appropriately. After creating the file, put it in your working directory (or move your working directory to where its stored). Read the data into R.

```
#insert r code here
#QQQ SPY JNJ NVDA TSLA BRK WMT TTCEHY TSM V BAC UNH
library(readr)
library(MASS)
WMT <- read_csv("WMT.csv")
BRK <- read_csv("BRK-B.csv")
JNJ <- read_csv("JNJ.csv")
NVDA <- read_csv("NVDA.csv")
QQQ <- read_csv("QQQ (1).csv")
SPY <- read_csv("SPY (1).csv")
TTCEHY <- read_csv("TCEHY.csv")
TSLA<- read_csv("TSLA.csv")
TSM <- read_csv("TSM.csv")
```

```
UNH <- read_csv("UNH.csv")
V <- read_csv("V.csv")
BAC <- read_csv("BAC.csv")

DATA <- data.frame(QQQ$`Adj Close`, SPY$`Adj Close`, JNJ$`Adj Close`, NVDA$`Adj Close`, TSLA$`Adj Close
data2 <- data.frame(DATA,DATA^2)
```

1. List the names of the variables in the data set.

```
#insert r code here
ls(DATA)
```

```
##  [1] "BAC..Adj.Close."    "BRK..Adj.Close."    "JNJ..Adj.Close."
##  [4] "NVDA..Adj.Close."   "QQQ..Adj.Close."    "SPY..Adj.Close."
##  [7] "TSLA..Adj.Close."   "TSM..Adj.Close."    "TTCEHY..Adj.Close."
## [10] "UNH..Adj.Close."    "V..Adj.Close."      "WMT..Adj.Close."
```

2. As the date will be unimportant, remove that field from your data frame

```
#insert r code here
# I didn't add the date in the data  frame in the first place
```

3. What is the range of each quantitative variable? Answer this question using the range() function with the sapply() function e.g., sapply(cars, range). Print a simple table of the ranges of the variables. The rows should correspond to the variables. The first column should be the lowest value of the corresponding variable, and the second column should be the maximum value of the variable. The columns should be suitably labeled.

```
#insert r code here
range_table<-sapply(DATA,range)
rownames(range_table)<-c("min","max")
range_table
```

```
##     QQQ..Adj.Close. SPY..Adj.Close. JNJ..Adj.Close. NVDA..Adj.Close.
## min        125.4581         213.727        105.5343         23.58766
## max        403.4800         477.710        176.1183        333.66229
##     TSLA..Adj.Close. BRK..Adj.Close. WMT..Adj.Close. TTCEHY..Adj.Close.
## min           35.794          161.26        63.36241           27.16547
## max         1229.910          327.74       150.84265           98.83556
##     TSM..Adj.Close. V..Adj.Close. BAC..Adj.Close. UNH..Adj.Close.
## min        26.69706      85.28028        17.30085        151.6781
## max       140.66000     249.74171        49.13974        504.0883
```

4. What is the mean and standard deviation of each variable? Create a simple table of the means and standard deviations.

```
#insert r code here
func <- function(x)
  {
  c(mean(x),sd(x))
    }
table2<- sapply(DATA,func)
rownames(table2)<-c("mean","standarddeviation")
table2
```

```
##                    QQQ..Adj.Close. SPY..Adj.Close. JNJ..Adj.Close.
## mean                     226.27212       308.87089       134.92864
## standarddeviation         82.03465        74.41577        18.46496
##                    NVDA..Adj.Close. TSLA..Adj.Close. BRK..Adj.Close.
## mean                        94.90440         283.9245       218.62717
## standarddeviation           71.53947         322.8682        39.08961
##                    WMT..Adj.Close. TTCEHY..Adj.Close. TSM..Adj.Close.
## mean                     109.01546           53.50922        62.47710
## standarddeviation         25.71426           14.43177        34.73007
##                    V..Adj.Close. BAC..Adj.Close. UNH..Adj.Close.
## mean                   165.48496       29.443289       280.50637
## standarddeviation       44.90173        7.293221        86.31716
```

5. Using the regsubsets function in the leaps library, regress one of your ETF's on the remaining assets.

```
#insert r code here
library(leaps)
regsubsets1<- regsubsets(DATA$QQQ..Adj.Close.~.,data=DATA)
summary(regsubsets1)
```

```
## Subset selection object
## Call: regsubsets.formula(DATA$QQQ..Adj.Close. ~ ., data = DATA)
## 11 Variables  (and intercept)
##                      Forced in Forced out
## SPY..Adj.Close.          FALSE      FALSE
## JNJ..Adj.Close.          FALSE      FALSE
## NVDA..Adj.Close.         FALSE      FALSE
## TSLA..Adj.Close.         FALSE      FALSE
## BRK..Adj.Close.          FALSE      FALSE
## WMT..Adj.Close.          FALSE      FALSE
## TTCEHY..Adj.Close.       FALSE      FALSE
## TSM..Adj.Close.          FALSE      FALSE
## V..Adj.Close.            FALSE      FALSE
## BAC..Adj.Close.          FALSE      FALSE
## UNH..Adj.Close.          FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
```

```
##          SPY..Adj.Close. JNJ..Adj.Close. NVDA..Adj.Close. TSLA..Adj.Close.
## 1  ( 1 ) "*"             " "             " "              " "
## 2  ( 1 ) "*"             " "             " "              " "
## 3  ( 1 ) "*"             " "             " "              "*"
## 4  ( 1 ) "*"             " "             "*"              " "
## 5  ( 1 ) "*"             " "             "*"              " "
## 6  ( 1 ) "*"             " "             "*"              " "
## 7  ( 1 ) "*"             " "             "*"              " "
## 8  ( 1 ) "*"             " "             "*"              " "
##          BRK..Adj.Close. WMT..Adj.Close. TTCEHY..Adj.Close. TSM..Adj.Close.
## 1  ( 1 ) " "             " "             " "                " "
## 2  ( 1 ) " "             " "             " "                " "
## 3  ( 1 ) " "             " "             " "                " "
## 4  ( 1 ) " "             " "             "*"                " "
## 5  ( 1 ) "*"             " "             "*"                " "
## 6  ( 1 ) "*"             " "             " "                "*"
## 7  ( 1 ) "*"             " "             "*"                "*"
## 8  ( 1 ) "*"             " "             "*"                "*"
##          V..Adj.Close. BAC..Adj.Close. UNH..Adj.Close.
## 1  ( 1 ) " "           " "             " "
## 2  ( 1 ) " "           "*"             " "
## 3  ( 1 ) " "           "*"             " "
## 4  ( 1 ) " "           "*"             " "
## 5  ( 1 ) " "           "*"             " "
## 6  ( 1 ) "*"           "*"             " "
## 7  ( 1 ) "*"           "*"             " "
## 8  ( 1 ) "*"           "*"             "*"
```

```
names(summary(regsubsets1))
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

```
# The validation part
cp=summary(regsubsets1)$cp
cp
```

```
## [1] 17530.96747  3897.24467  2262.75245   999.36678   561.87449   325.71385
## [7]   135.32675    54.41798
```

```
i=which.min(cp)
i
```

```
## [1] 8
```

```
bic=summary(regsubsets1)$bic
bic
```

```
## [1] -4633.817 -6255.908 -6730.160 -7285.141 -7551.835 -7722.436 -7879.798
## [8] -7950.765
```

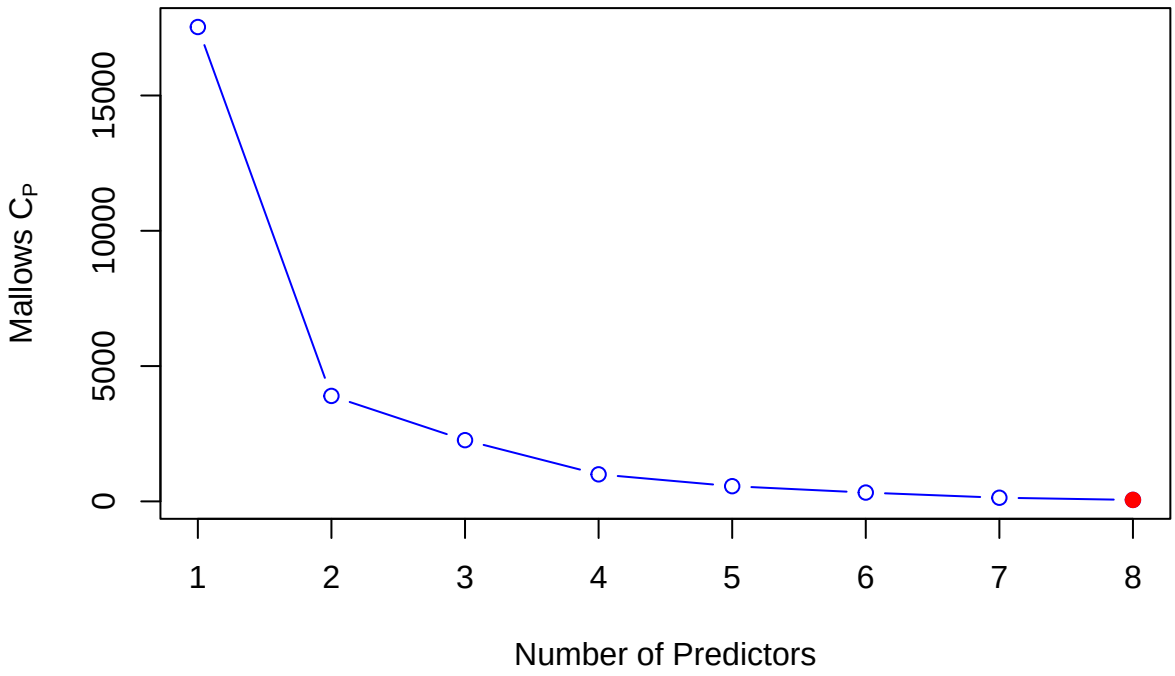
```
j<-which.min(bic)
j
```

```
## [1] 8
```

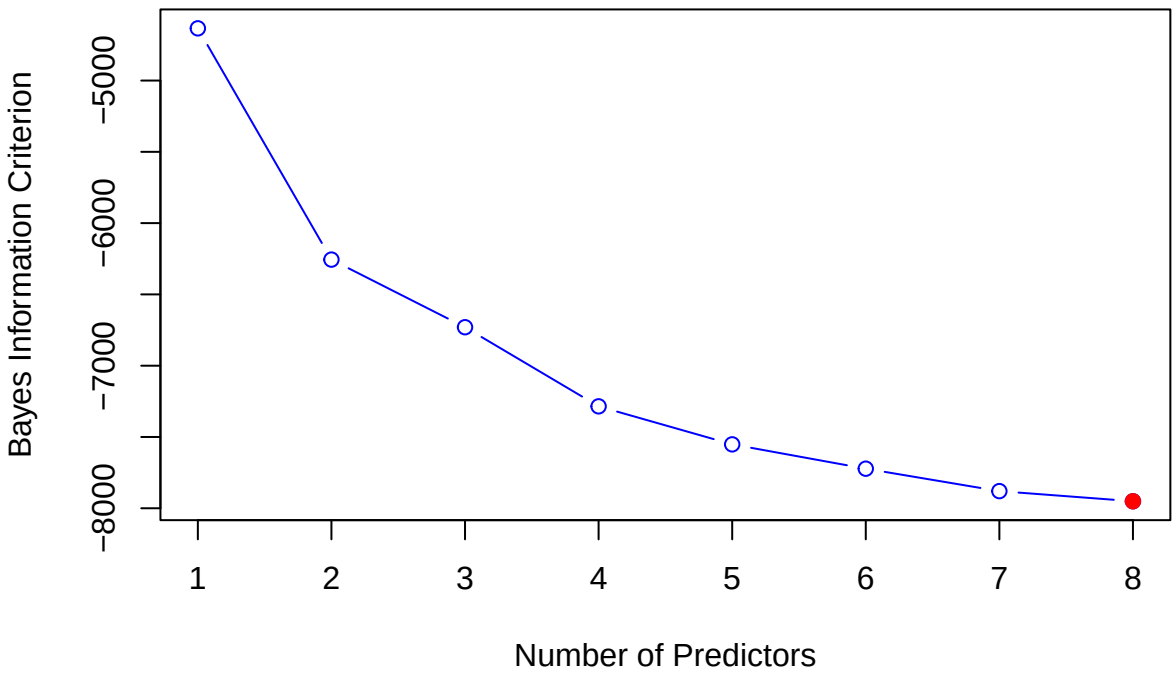
```
adjr2_1=summary(regsubsets1)$adjr2
k=which.max(adjr2_1)
c(i,j,k)
```

```
## [1] 8 8 8
```

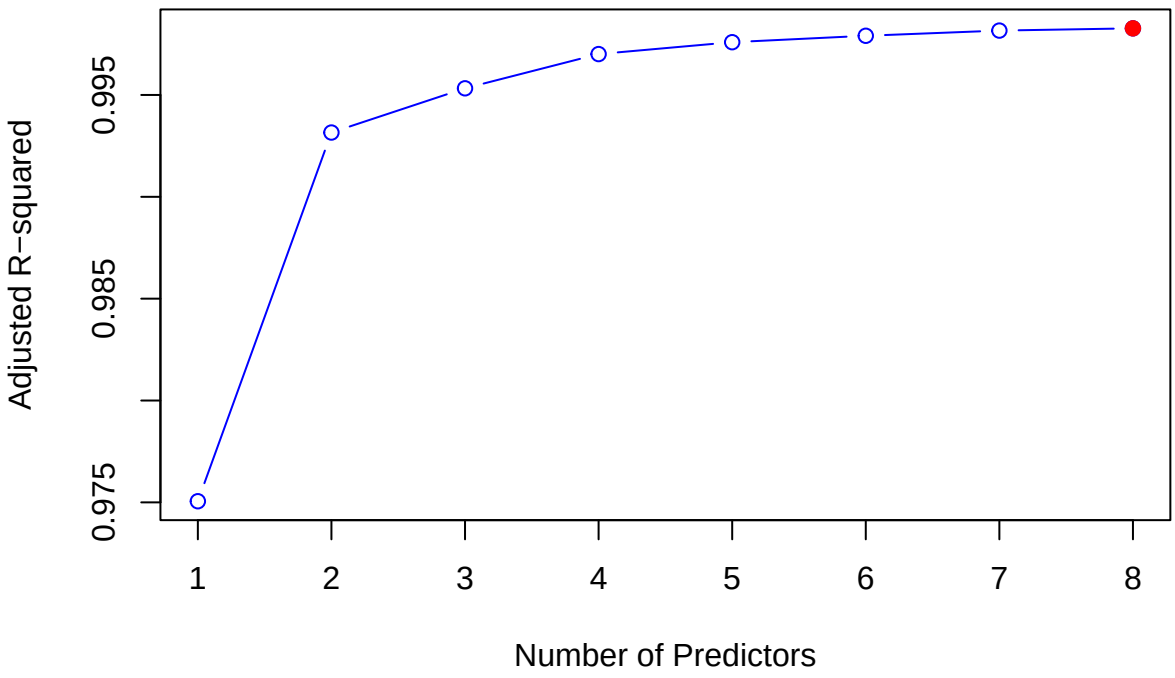
```
plot(cp,type='b',col="blue",xlab="Number of Predictors",ylab=expression("Mallows C"[P]))
points(i,cp[i],pch=19,col="red")
```




```
plot(bic,type='b',col="blue",xlab="Number of Predictors",ylab=expression("Bayes Information Criterion"))
points(j,bic[j],pch=19,col="red")
```




```
plot(adjr2_1,type='b',col="blue",xlab="Number of Predictors",ylab=expression("Adjusted R-squared"))
points(k,adjr2_1[k],pch=19,col="red")
```



a. Print a table showing what variables would be selected using best subset selection for all predictors up to order 2 (i.e. asset and asset^2). Determine the optimal model using BIC and output the model, including its coeffecients.

```
#insert r code here
regsubsets2<- regsubsets(data2$QQQ..Adj.Close.~.,data=data2)
names(summary(regsubsets2))
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

```
bic2=summary(regsubsets2)$bic
bic2
```

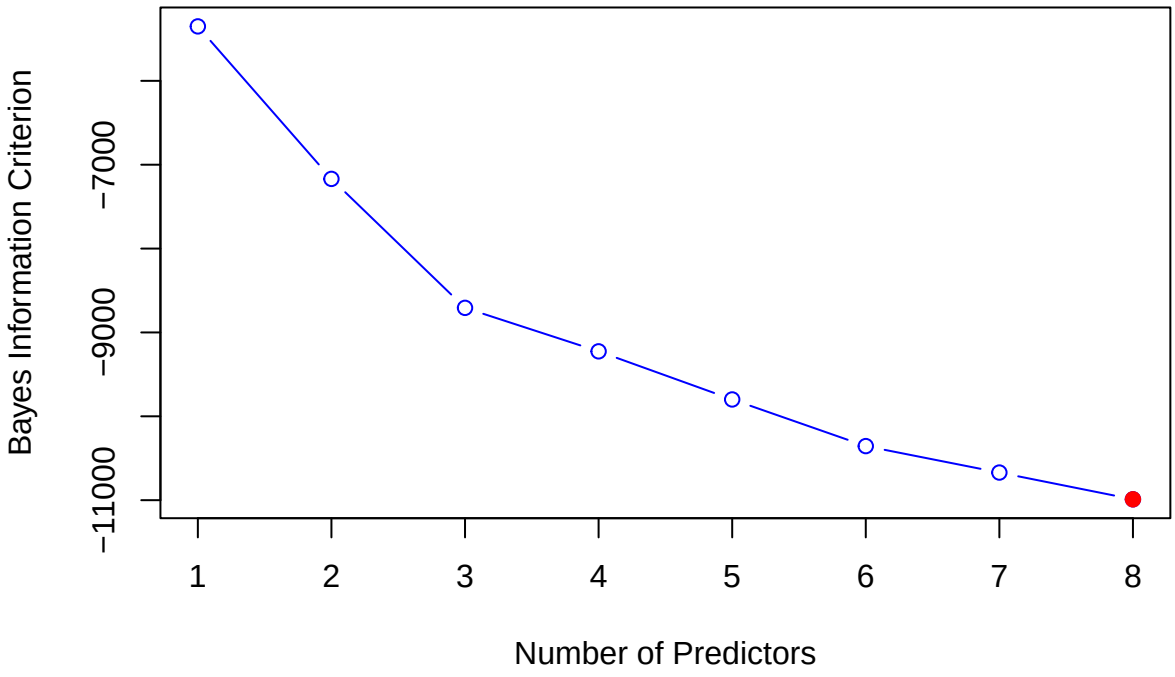
```
## [1]  -5350.882  -7169.088  -8706.756  -9225.416  -9799.033 -10355.227 -10670.493
## [8] -10989.318
```

```
j2=which.min(bic2)
coef(regsubsets2,j2)
```

```
##       (Intercept)   SPY..Adj.Close.   BRK..Adj.Close.     V..Adj.Close.
##     -3.534718e+01      1.354304e+00     -6.180839e-01      1.147654e-01
## QQQ..Adj.Close..1 SPY..Adj.Close..1 BRK..Adj.Close..1 BAC..Adj.Close..1
```

```
##      2.245102e-03     -2.490855e-03      1.411668e-03      7.000660e-03
## UNH..Adj.Close..1
##      5.515740e-05
```

```
plot(bic2,type='b',col="blue",xlab="Number of Predictors",ylab=expression("Bayes Information Criterion"))
points(j2,bic2[j2],pch=19,col="red")
```



b. Print a table showing what variables would be selected using forward subset selection for all predictors up to order 2 (i.e. asset and asset^2). Determine the optimal model using BIC and output the model, including its coeffecients.

```
#insert r code here
regsubsets3<- regsubsets(data2$QQQ..Adj.Close.~.,data=data2,method="forward")
names(summary(regsubsets3))
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

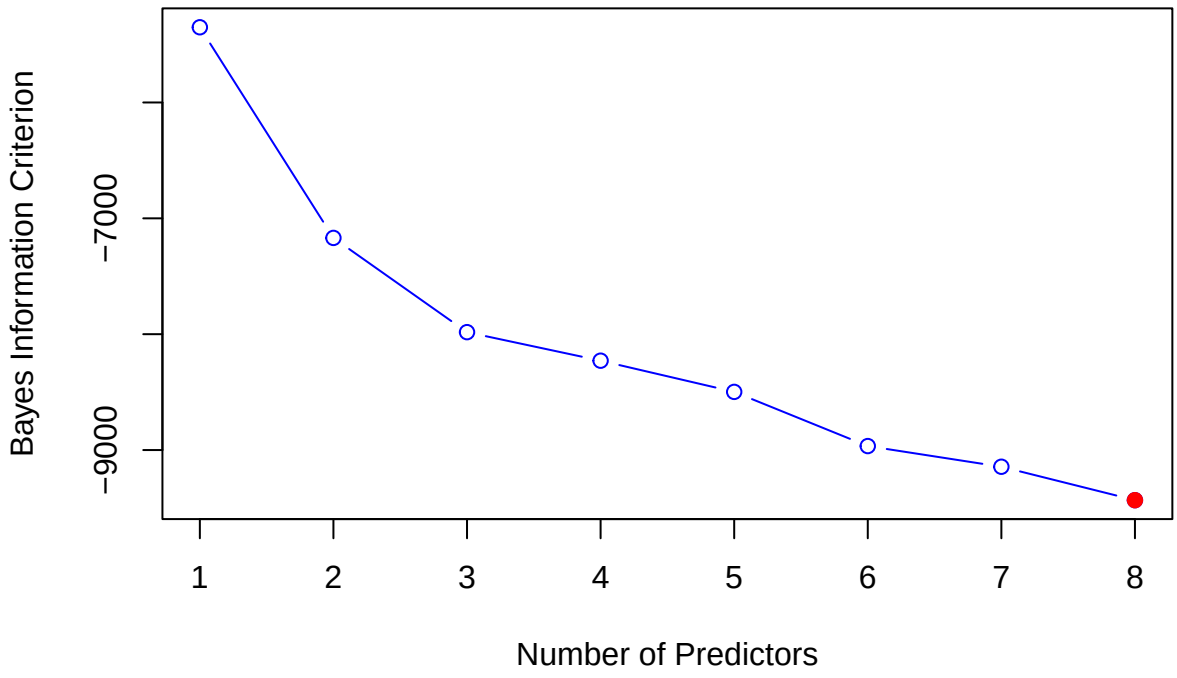
```
bic3=summary(regsubsets3)$bic
bic3
```

```
## [1] -5350.882 -7169.088 -7982.507 -8228.804 -8498.028 -8965.721 -9144.236
## [8] -9432.689
```

```
j3=which.min(bic3)
coef(regsubsets3,j3)
```

```
##        (Intercept)    SPY..Adj.Close.   NVDA..Adj.Close. TTCEHY..Adj.Close.
##      16.2277695618       0.2375575805       0.2306780605       0.1761758109
##      V..Adj.Close.    BAC..Adj.Close.  QQQ..Adj.Close..1 NVDA..Adj.Close..1
##       0.5860570775      -0.7824339320       0.0013108036      -0.0005365239
##    V..Adj.Close..1
##      -0.0012566737
```

```
plot(bic3,type='b',col="blue",xlab="Number of Predictors",ylab=expression("Bayes Information Criterion"))
points(j3,bic3[j3],pch=19,col="red")
```



c. Print a table showing what variables would be selected using backward subset selection for all predictors up to order 2 (i.e. asset and asset^2). Determine the optimal model using adjusted R^2 and output the model, including its coeffecients.

```
#insert r code here
regsubsets4<- regsubsets(data2$QQQ..Adj.Close.~.,data=data2,method="backward")
names(summary(regsubsets4))
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

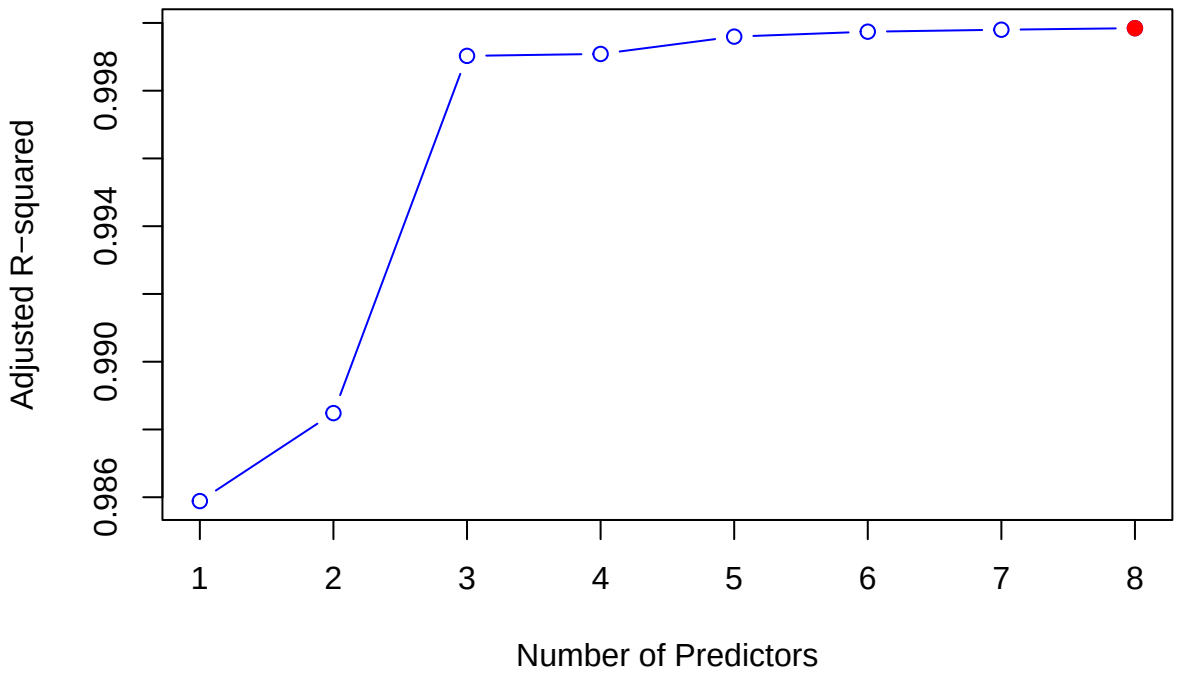
```
adjr4=summary(regsubsets4)$adjr2
adjr4
```

```
## [1] 0.9858869 0.9884833 0.9990278 0.9990842 0.9995956 0.9997413 0.9997996
## [8] 0.9998452
```

```
k4=which.max(adjr4)
coef(regsubsets4,k4)
```

```
##       (Intercept)   SPY..Adj.Close.   BRK..Adj.Close.     V..Adj.Close.
##     -3.534718e+01      1.354304e+00     -6.180839e-01      1.147654e-01
## QQQ..Adj.Close..1 SPY..Adj.Close..1 BRK..Adj.Close..1 BAC..Adj.Close..1
##      2.245102e-03     -2.490855e-03      1.411668e-03      7.000660e-03
## UNH..Adj.Close..1
##      5.515740e-05
```

```
plot(adjr4,type='b',col="blue",xlab="Number of Predictors",ylab=expression("Adjusted R-squared"))
points(k4,adjr4[k4],pch=19,col="red")
```



Question 2

Using the data set that you loaded for the first problem, choose the other ETF, and create a data frame consisting of simple lagged returns going up to 10 days back. Create another field in this data frame that

looks to the direction of the ETF moving one day into the future, this direction should be listed as a factor, not a number.

1. Split your data into a training set and a testing set and run LDA on the direction using all 10 different lags on the training set. How accurate is your model?

```
#insert r code here
#Q2) 1.
qqq<-QQQ$`Adj Close`
Direction_1=sign(log(qqq[12:1259]/qqq[11:1258]))
Direction_1[Direction_1==1]="Up"
Direction_1[Direction_1==0]="Same"
Direction_1[Direction_1==(-1)]="Down"
Direction_1=as.factor(Direction_1)
Log_Lag1=log(qqq[11:1258]/qqq[10:1257])
Log_Lag2=log(qqq[11:1258]/qqq[9:1256])
Log_Lag3=log(qqq[11:1258]/qqq[8:1255])
Log_Lag4=log(qqq[11:1258]/qqq[7:1254])
Log_Lag5=log(qqq[11:1258]/qqq[6:1253])
Log_Lag6=log(qqq[11:1258]/qqq[5:1252])
Log_Lag7=log(qqq[11:1258]/qqq[4:1251])
Log_Lag8=log(qqq[11:1258]/qqq[3:1250])
Log_Lag9=log(qqq[11:1258]/qqq[2:1249])
Log_Lag10=log(qqq[11:1258]/qqq[1:1248])
Log_Pred1=data.frame(Direction_1,Log_Lag1,Log_Lag2,Log_Lag3,Log_Lag4,Log_Lag5,Log_Lag6,Log_Lag7,Log_Lag8
train_1 <- sample(1259,800,replace=FALSE)
lda.fit_stocks=lda(Direction_1~.,data=Log_Pred1,subset=train_1)
lda.pred_stocks=predict(lda.fit_stocks,Log_Pred1[-train_1,])
lda.class_1=lda.pred_stocks$class
table(lda.class_1,Log_Pred1$Direction_1[-train_1])
```

```
##
## lda.class_1 Down Same  Up
##        Down   32    0  30
##        Same    0    0   0
##        Up    149    0 243
```

```
mean(lda.class_1==Log_Pred1$Direction_1[-train_1])#accuracy
```

```
## [1] 0.6057269
```

```
#2)
#splitting into 5 pieces
K<- 5
len<- length(Log_Pred1$Direction_1)
folds<- seq(1:K)
member<- sample(folds,len,replace=T)
cv.error=rep(0,K)
for (i in 1:K)
```

```
{
  train=Log_Pred1[member!=i,]
  test=Log_Pred1[member==i,]
  temp.mod=lda(Direction_1~.,data=train)
  temp.predict<- predict(temp.mod,test)
  cv.error[i]=mean((test$Direction_1==temp.predict$class)^2)
}
cv.error
```

```
## [1] 0.6192171 0.5520000 0.5791667 0.5637860 0.6196581
```

```
mean(cv.error)
```

```
## [1] 0.5867656
```

```
#LOOCV
K1=len
cv.error1=rep(0,K)
for (i in 1:K1)
{

  temp1.mod=lda(Direction_1~.,data=Log_Pred1[-i,])
  cv.error1[i]=(Log_Pred1$Direction_1[i]==predict(temp1.mod,Log_Pred1[i,])$class)^2
}
cv.error1
```

```
##    [1] 1 1 0 0 1 0 1 0 1 0 0 0 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1
##   [38] 1 1 1 1 1 1 1 0 1 1 0 0 1 1 0 0 1 0 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 1 0 1 0
##   [75] 1 1 1 1 1 1 0 1 0 1 0 0 0 1 1 0 1 1 0 0 0 1 1 1 1 0 0 0 1 0 0 0 1 1 1 1 0 0
##  [112] 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 0 0 1 0 1 0 0
##  [149] 1 1 1 0 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 0 1 0 0 1 1 1 1 1 0 1 0 1 1 0
##  [186] 0 1 0 0 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 0 0 1 0 0 1 0 0 0 1 0 0 1 1 0
##  [223] 0 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 1 1 1 1
##  [260] 0 1 1 0 1 1 1 0 0 0 1 1 1 1 0 1 0 1 1 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 0 1 0
##  [297] 1 1 1 1 0 1 1 1 0 1 0 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 0 1 0 0 1 0 0 0 0 1 1
##  [334] 1 0 0 0 1 1 1 1 1 1 0 0 0 0 1 0 1 1 0 1 1 0 1 1 1 1 0 1 0 0 0 0 1 1 0 1 0
##  [371] 0 1 0 1 0 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 1 1 1 0 0 0
##  [408] 1 1 0 0 1 0 1 0 0 0 1 0 1 1 1 0 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 1 1
##  [445] 1 1 1 0 1 1 0 0 1 1 1 1 1 0 1 1 1 0 0 1 0 0 1 1 0 0 1 0 1 1 1 1 1 0 0 1 1 1
##  [482] 0 0 1 1 1 0 0 0 1 1 1 0 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1
##  [519] 1 1 0 1 1 1 0 0 0 1 0 0 0 0 1 0 1 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 1 1 0 0
##  [556] 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 0 1 0 0 0 0 0
##  [593] 1 1 1 0 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0
##  [630] 0 1 0 0 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 0 1
##  [667] 1 0 0 1 1 1 1 1 0 0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 0 1 1 0
##  [704] 0 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 1 0 1 0 1 0 1 1 1 0 0 0 0 1 0 1 1 0 1 0 0
##  [741] 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 0 1 1 1 0 1 0 1 1 0 0 1 1 0 1 1 0 1
##  [778] 1 0 1 1 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 0
##  [815] 1 1 1 0 1 1 1 1 1 1 0 0 1 0 0 1 1 0 1 1 0 1 0 0 1 0 1 1 1 1 0 1 1 1 1 0 0 0
##  [852] 0 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 0 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1
##  [889] 1 1 1 1 1 0 0 1 0 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 0 1 1 1
##  [926] 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 0 1 0 1 1 1
```

```
##  [963] 1 1 0 1 0 1 1 0 1 1 1 1 0 1 1 0 0 0 0 0 0 1 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1
## [1000] 0 0 1 1 0 0 1 1 0 1 1 1 0 1 1 1 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 0 0 0 1 1 0
## [1037] 0 0 1 1 0 0 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 0 1 1 0 1
## [1074] 1 1 1 0 1 1 0 1 0 0 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 1 1 1 0 0 1 1 1 1
## [1111] 1 0 1 1 0 1 0 1 1 0 0 0 0 0 1 1 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0
## [1148] 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 0 1 0 1 1 1
## [1185] 1 1 0 0 1 0 0 0 1 0 1 1 0 0 1 0 1 0 0 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 1
## [1222] 1 0 1 1 1 1 1 0 0 1 1 1 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0
```

```
mean(cv.error1)
```

```
## [1] 0.5929487
```

2. Create code to determine the estimate for the expected test error using K=5 cross validation. Do this by actually splitting the date into five pieces and give the average of the test error, not just by using a command from a package (such as cv.glm).

3. Determine the LOOCV estimate of the expected test error of your model. How do your answers to each part of this question compare? Do you see any noticable differences between your answer? Why do you think that is?

Question 3

This question should be answered using the Weekly data set, which is part of the ISLR package. This data contains 1,089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

1. What does the data represent?

```
#insert r code here
library(ISLR)
# The data represents Weekly percentage returns for the S&P 500 stock index between 1990 and 2010
```

2. Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

```
#insert r code here
regression_weekly_1 <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,data = Weekly,family =
summary(regression_weekly_1)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

```
#only the intercept and lag 2 and found to be significant
```

3. Fit a logistic regression model using a training data period from 1990 to 2008, using the predictors from the previous problem that you determined were statistically significant. Test your model on the held out data (that is, the data from 2009 and 2010) and express its accuracy.

```
#insert r code here
training_weekly_1 <- Weekly[Weekly$Year <= 2008,]

testing_weekly_1 <- Weekly[!Weekly$Yea<= 2008,]

direction_Test_1 <- testing_weekly_1 $Direction

regression_weekly_2 <- glm(Direction ~ Lag2,data = training_weekly_1,family = binomial)

Predicting_weekly_regression_2 <- predict(regression_weekly_2, testing_weekly_1, type = "response")
showcase <- rep("Down", 104)

showcase[Predicting_weekly_regression_2 > 0.5] <- "Up"

table(showcase, direction_Test_1)
```

```
##         direction_Test_1
## showcase Down Up
##     Down    9  5
```

```
##     Up     34 56
```

```
rl <-mean(showcase==direction_Test_1)
```

4. Repeat Part 3 using LDA.

```
#insert r code here
regression_weekly_3 <- lda(Direction ~ Lag2,data = training_weekly_1 )
showcase_2 <- predict(regression_weekly_3 , testing_weekly_1)$class
table(showcase_2, direction_Test_1)
```

```
##           direction_Test_1
## showcase_2 Down Up
##       Down    9  5
##       Up     34 56
```

```
rlda<- mean(showcase_2== direction_Test_1)
```

5. Repeat Part 3 using QDA.

```
#insert r code here
regression_weekly_4 <- qda(Direction ~ Lag2,data = training_weekly_1 )
showcase_3 <- predict(regression_weekly_4, testing_weekly_1)$class
table(showcase_3, direction_Test_1)
```

```
##           direction_Test_1
## showcase_3 Down Up
##       Down    0  0
##       Up     43 61
```

```
rqda<- mean(showcase_3== direction_Test_1)
```

6. Repeat Part 3 using KNN with K = 1, 2, 3.

```
#insert r code here
library(FNN)
KNN_training <- matrix(training_weekly_1$Lag2)
KNN_testing <- matrix(testing_weekly_1$Lag2)
Direction_KNN <- training_weekly_1$Direction
KNN1 <- knn(KNN_training,KNN_testing,Direction_KNN ,k=1)
table(KNN1,direction_Test_1)
```

```
##       direction_Test_1
## KNN1   Down Up
##   Down   21 30
##   Up     22 31
```

```
r1   <- mean(KNN1 == direction_Test_1)

KNN2 <- knn(KNN_training,KNN_testing,Direction_KNN,k=2)
table(KNN1,direction_Test_1)
```

```
##       direction_Test_1
## KNN1   Down Up
##   Down   21 30
##   Up     22 31
```

```
r2   <- mean(KNN2 == direction_Test_1)

KNN3 <- knn(KNN_training,KNN_testing,Direction_KNN,k=3)
table(KNN3,direction_Test_1)
```

```
##       direction_Test_1
## KNN3   Down Up
##   Down   16 19
##   Up     27 42
```

```
r3   <- mean(KNN3 == direction_Test_1)
```

7. Which of these methods in Parts 3, 4, 5, and 6 appears to provide the best results on this data?

```
#insert r code here
result_total <- data.frame(rl,rlda,rqda,r3,r1,r2)
result_total
```

```
##      rl  rlda      rqda        r3  r1        r2
## 1 0.625 0.625 0.5865385 0.5576923 0.5 0.4423077
```

Question 4

Write a function that works in R to gives you the parameters from a linear regression on a data set of n predictors. You can assume all the predictors and the prediction is numeric. Include in the output the standard error of your variables. You cannot use the lm command in this function or any of the other built in regression models.

```
#insert r code here
# Simple Linear Regression

# Importing the dataset
dataset = data.frame(DATA$SPY..Adj.Close.,DATA$NVDA..Adj.Close.)
View(dataset)
```

```
RSS=function(b0,b1,y,x)
{
  yhat=b0+b1*x;
  err=yhat-y;
  return(sum(err^2));
}

beta0=seq(from=0,to=14,by=.005);
N0=length(beta0);
beta1=seq(from=.02, to=.08,by=.0005);
N1=length(beta1);
z=matrix(NA,N0,N1);
for(i in 1:N0)
{
  for(j in 1:N1)
  {
    z[i,j]=RSS(beta0[i],beta1[j],dataset$DATA.SPY..Adj.Close.,dataset$DATA.NVDA..Adj.Close.);
  }
}

which.min(z)
```

```
## [1] 338921
```

```
which.min(z)%%N0
```

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

```
ceiling(which.min(z)/N0)
```

```
## [1] 121
```

```
which(z==min(z), arr.ind = TRUE)
```

```
##       row col
## [1,] 2801 121
```

```
min(z)
```

```
## [1] 109901960
```

```
z[ 2801 ,121]
```

```
## [1] 109901960
```

Compare the output of your function to that of the lm command in R.

```r
#insert r code here
model_1<-lm(dataset$DATA.SPY..Adj.Close.~dataset$DATA.NVDA..Adj.Close.)
names(model_1)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```r
summary(model_1)
```

```
##
## Call:
## lm(formula = dataset$DATA.SPY..Adj.Close. ~ dataset$DATA.NVDA..Adj.Close.)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -82.056 -20.076  -5.767  21.074  58.216
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   2.159e+02  1.168e+00  184.83   <2e-16 ***
## dataset$DATA.NVDA..Adj.Close. 9.801e-01  9.828e-03   99.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.94 on 1257 degrees of freedom
## Multiple R-squared:  0.8878, Adjusted R-squared:  0.8877
## F-statistic:  9946 on 1 and 1257 DF,  p-value: < 2.2e-16
```

Question 5

As you have probably seen in this homework, just simply looking at the close prices and trying to run models on the variables is not terribly interesting. You've begun to see what types of techniques we will be studying in this class. Here is an exerpt from the final project/homework:

"In this assignment, you will be required to find a set of data to run regression on. This data set should be financial in nature, and of a type that will work with the models we have discussed this semester (hint: we didn't look at time series) You may not use any of the data sets in the ISLR package that we have been looking at all semester. Your data set that you choose should have both qualitative and quantitative variables. (or has variables that you can transform)

Provide a description of the data below, where you obtained it, what the variable names are and what it is describing."

You don't have to actually create the data set at this time, but what sort of problem are you looking to solve? What data set would you need to answer this question? Please provide what you are looking into and how you could approach the problem below.

#At this point I want find two models that could work long term in the market , regardless of what state the market is in. #There are fundamentals to the companies that are listed on the stock market. These are hardcore facts (mostly reliable) that show case the cash flow , asset base, business model, company history , information on board of directors etc. #I want to figure out a model that could act as a screener based on

these facts sampling the the listed companies on the market. The variable I would as my “y” variable that would predict a companies status/ public perception could be its credit evaluation/rating.

#Second model I want to work on would include the companies i select from the screener and their portfolio and try and predict their beta with respect to the broader market. Basically I would try and create a efficient portfolio of these fundamentally strong companies , to see if they could outperform the benchmarks in both a bull and a bear run.