QF301. Homework #2.

2021-10-01

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: 9/24/2021

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 (20pt)

Question 1.1

Use the quantmod package to obtain the daily adjusted close prices 2 different stocks. You should have at least two years of data for both 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 for both stocks. Print the first and last 6 lines of your data frame.

Solution:

```
library(quantmod)
stocks = c("AAPL", "AMD")
getSymbols(stocks)
```

```
## [1] "AAPL" "AMD"
```

```r
AAPL = AAPL$AAPL.Adjusted
AMD = AMD$AMD.Adjusted

aapl_log_rets = dailyReturn(AAPL, type="log")[2:nrow(AAPL)]
amd_log_rets = dailyReturn(AMD, type="log")[2:nrow(AMD)]

df = data.frame(aapl_log_rets, amd_log_rets)
colnames(df) <- c("aapl_rets", "amd_rets")

head(df)
```

```
##               aapl_rets     amd_rets
## 2007-01-04  0.021952707  0.013737229
## 2007-01-05 -0.007146817 -0.004050740
## 2007-01-08  0.004926025 -0.012251302
## 2007-01-09  0.079799840  0.009202570
## 2007-01-10  0.046746074  0.018154810
## 2007-01-11 -0.012448344  0.008459866
```

```r
tail(df)
```

```
##                aapl_rets     amd_rets
## 2021-09-23  0.0066967311  0.016815150
## 2021-09-24  0.0006127387 -0.003302659
## 2021-09-27 -0.0106060253  0.022061101
## 2021-09-28 -0.0240891000 -0.063355855
## 2021-09-29  0.0064620438 -0.011591739
## 2021-09-30 -0.0093554088  0.025093607
```

Question 1.2

List the names of the variables in the data set.

Solution:

Variable Names: aapl_rets, amd_rets

Question 1.3

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

Solution:

```r
row.names(df) <- NULL
head(df)
```

```
##      aapl_rets     amd_rets
## 1  0.021952707  0.013737229
## 2 -0.007146817 -0.004050740
## 3  0.004926025 -0.012251302
## 4  0.079799840  0.009202570
## 5  0.046746074  0.018154810
## 6 -0.012448344  0.008459866
```

Question 1.4

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

Solution:

```
summary_table = data.frame(c(mean(df$aapl_rets), sd(df$aapl_rets)), c(mean(df$amd_rets), sd(df$amd_rets
colnames(summary_table) <- c("AAPL", "AMD")
rownames(summary_table) <- c("Mean", "Sd")
summary_table
```

```
##             AAPL          AMD
## Mean 0.001079877 0.0004478228
## Sd   0.020343072 0.0371648571
```

Question 1.5

Regress one of your stock returns as a function of the other (simultaneous data). This should be of the form $r_1 = \beta_0 + \beta_1 r_2$. (No train/test split is required here.)

Solution:

```
m1 = glm(aapl_rets~amd_rets, data = df)
summary(m1)
```

```
##
## Call:
## glm(formula = aapl_rets ~ amd_rets, data = df)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.160349  -0.009367  -0.000124   0.009579   0.112820
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.0009875  0.0003093   3.192  0.00142 **
## amd_rets    0.2063820  0.0083234  24.795  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.000355105)
##
##     Null deviance: 1.5358  on 3711  degrees of freedom
## Residual deviance: 1.3174  on 3710  degrees of freedom
## AIC: -18947
##
## Number of Fisher Scoring iterations: 2
```

Question 2 (40pt)

Question 2.1

Using the data set that you loaded for the first problem, choose one of your stocks, and create a data frame consisting of lagged returns going up to 5 days back.

Solution:

```
r0 = df$aapl_rets[-5:-1]
r1 = as.numeric(lag(df$aapl_rets,k=1))[-5:-1]
r2 = as.numeric(lag(df$aapl_rets,k=2))[-5:-1]
r3 = as.numeric(lag(df$aapl_rets,k=3))[-5:-1]
r4 = as.numeric(lag(df$aapl_rets,k=4))[-5:-1]
r5 = as.numeric(lag(df$aapl_rets,k=5))[-5:-1]


aapl_rets_lagged = data.frame(r0,r1,r2,r3,r4,r5)
head(aapl_rets_lagged)
```

```
##             r0           r1           r2           r3           r4           r5
## 1 -0.012448344 -0.012448344 -0.012448344 -0.012448344 -0.012448344 -0.012448344
## 2 -0.012393969 -0.012393969 -0.012393969 -0.012393969 -0.012393969 -0.012393969
## 3  0.025872210  0.025872210  0.025872210  0.025872210  0.025872210  0.025872210
## 4 -0.022390783 -0.022390783 -0.022390783 -0.022390783 -0.022390783 -0.022390783
## 5 -0.063927710 -0.063927710 -0.063927710 -0.063927710 -0.063927710 -0.063927710
## 6 -0.006420034 -0.006420034 -0.006420034 -0.006420034 -0.006420034 -0.006420034
```

Question 2.2

Split your data into a training set and a testing set (50% in each set). Create an AR(1) model for your stock returns. Provide the test mean squared error.

Solution:

```r
train=sample(nrow(aapl_rets_lagged),nrow(aapl_rets_lagged)/2,replace=FALSE)
ar1=glm(r0~r1,data=aapl_rets_lagged,subset=train)

pred=predict(ar1,aapl_rets_lagged[-train,])
MSE=mean((pred-aapl_rets_lagged$r0[-train])^2)
cat("AR(1) RMSE: ", sqrt(MSE), "\n")
```

```
## AR(1) RMSE:  1.772264e-17
```

```r
#Compare with naive/constant predictor:
MSE0 = mean((mean(aapl_rets_lagged$r0[train])-aapl_rets_lagged$r0[-train])^2)
cat("Base RMSE: ", sqrt(MSE0), "\n")
```

```
## Base RMSE:  0.02006785
```

For this dataset, the AR(1) model seems very good compared to glm as it has a low MSE.

Question 2.3

Evaluate if your AR(1) model is weakly stationary or unit-root nonstationary.

(a) If your model is weakly stationary, provide the (long-run) average returns and autocovariances for your model. How do these compare with the empirical values? If your model is (unit-root) nonstationary, please interpret your model. Give as much detail as possible.

Solution:

```r
#summary(ar1)
b0 = summary(ar1)$coefficients[1,1]
b1 = summary(ar1)$coefficients[2,1]
b1.se = summary(ar1)$coefficients[2,2]
abs(b1) < 1 #If true then suspect weakly stationary
```

```
## [1] FALSE
```

```r
#Test for unit-root nonstationary
t = (b1 - 1)/b1.se
cat("t stat: ", t, "\n")
```

```
## t stat:  43.31534
```

```r
cat("p-value: ", pnorm(t), "\n") #Large enough data, treat as normal
```

```
## p-value:  1
```

```
#If small enough then reject the null hypothesis

cat("Long run average returns: ", b0/(1-b1), "\n")
```

```
## Long run average returns:  0.001270429
```

```
cat("Long run autocovariances: ", sd(pred-aapl_rets_lagged$r0[-train])^2 / (1-b1^2), "\n")
```

```
## Long run autocovariances:  -1.768987e-19
```

The long run average returns are very close to empirical numbers of .00107.

The model is unit root nonstationary because the weak stationarity condition in FALSE, and the p-value is too large.

The model is r0 = -1.612e-18 + 1.000*r1. The 1-lag is very significant according to the t-stat, and for some reason, none of the other variables contribute significantly to r0 because their coefficients are 0. I'm not sure if this is a mistake or AAPL stock returns just happen to be a simple drift from the previous day.

(b) Provide the formulas for the 1- and 2- step ahead forecasts. What are the variances for these forecast errors?

Solution:

1-step ahead forecast: $\hat{X_t(1)} = \beta_0 + \beta_1 X_t$

$Var(e_t(1)) = \sigma_\epsilon^2$

2-step ahead forecast: $X_t(2) = \beta_0 + \beta_1 \hat{X_t(1)}$

$Var(e_t(2)) = (1 + \beta_1^2)\sigma_\epsilon^2$

(c) Using the same train/test split as in Question 2.2. Create an AR(5) model for your stock returns. Provide the test mean squared error.

Solution:

```
y_var = c("r0")
x_var = c("r1", "r2", "r3", "r4", "r5")

X_train = as.matrix(aapl_rets_lagged[train,x_var])
Y_train = as.matrix(aapl_rets_lagged[train,names(aapl_rets_lagged) %in% y_var])

X_test = as.matrix(aapl_rets_lagged[-train,x_var])
Y_test = as.matrix(aapl_rets_lagged[-train,names(aapl_rets_lagged) %in% y_var])


ar5 = lm(r0~r1+r2+r3+r4+r5,data=aapl_rets_lagged,subset=train)

pred=predict(ar5,aapl_rets_lagged[-train,])
MSE = mean((pred-aapl_rets_lagged$r0[-train])^2) #Test MSE
cat("AR(5) RMSE: ", sqrt(MSE), "\n") #Root mean squared error
```

```
## AR(5) RMSE:  1.772264e-17
```

```
ar5
```

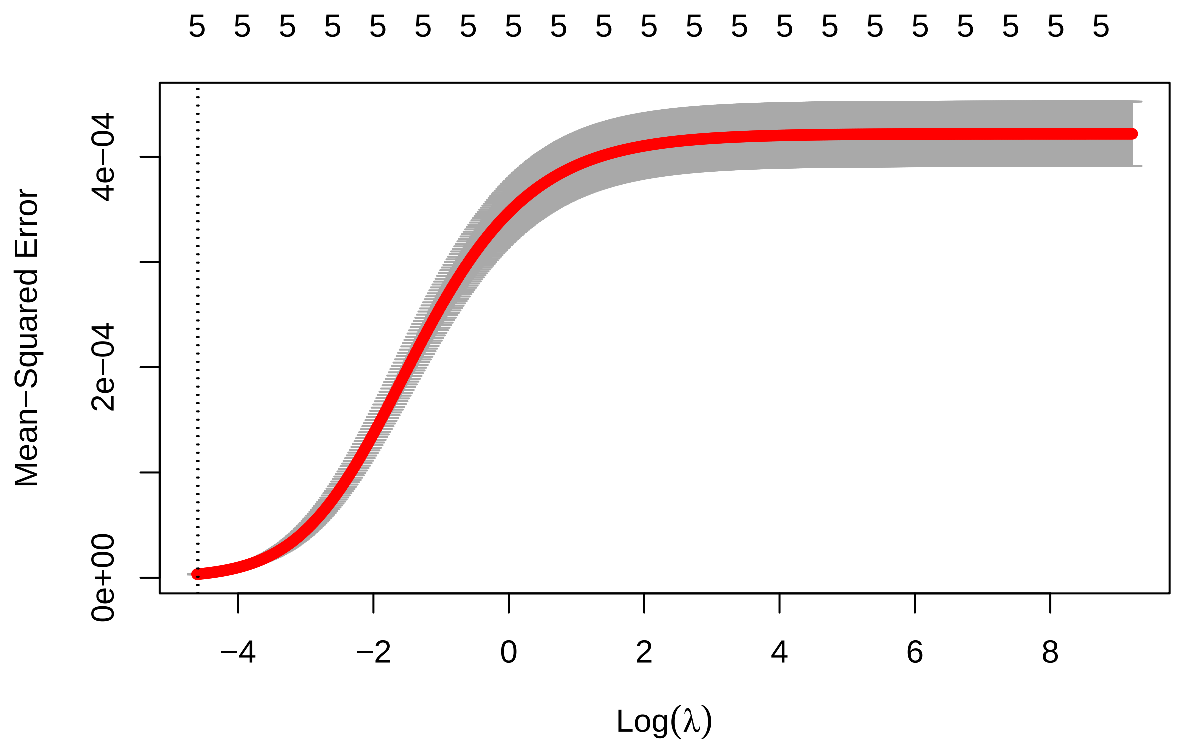
```
##
## Call:
## lm(formula = r0 ~ r1 + r2 + r3 + r4 + r5, data = aapl_rets_lagged,
##     subset = train)
##
## Coefficients:
## (Intercept)           r1           r2           r3           r4           r5
##  -1.128e-18    1.000e+00           NA           NA           NA           NA
```

Question 2.4

Using the same train/test split as in Question 2.2. Using Ridge regression, produce an AR(5) linear regression. Find the optimal tuning parameter for this regression. What is the test mean squared error?

Solution:

```
library(glmnet)
lambda=10^seq(-2,4,by=.01)
ridge.mod = glmnet(X_train, Y_train, alpha = 0, family="gaussian", lambda=lambda)
cv.out=cv.glmnet(X_train, Y_train, alpha=0,lambda=lambda)
plot(cv.out)
```



```
bestlam=cv.out$lambda.min
cat("Optimal Lambda (Ridge): ", bestlam, "\n")
```

```
## Optimal Lambda (Ridge):  0.01
```

```
bestridge.pred=predict(ridge.mod,s=bestlam,newx=X_test)
cat("Ridge Model MSE: ", mean((bestridge.pred-aapl_rets_lagged$r0[-train])^2), "\n") #Ridge MSE
```

```
## Ridge Model MSE:  3.168013e-06
```

```
ridgecoef=glmnet(aapl_rets_lagged[,x_var],aapl_rets_lagged[,names(aapl_rets_lagged) %in% y_var],alpha=0
predict(ridgecoef,type="coefficients",s=bestlam)
```

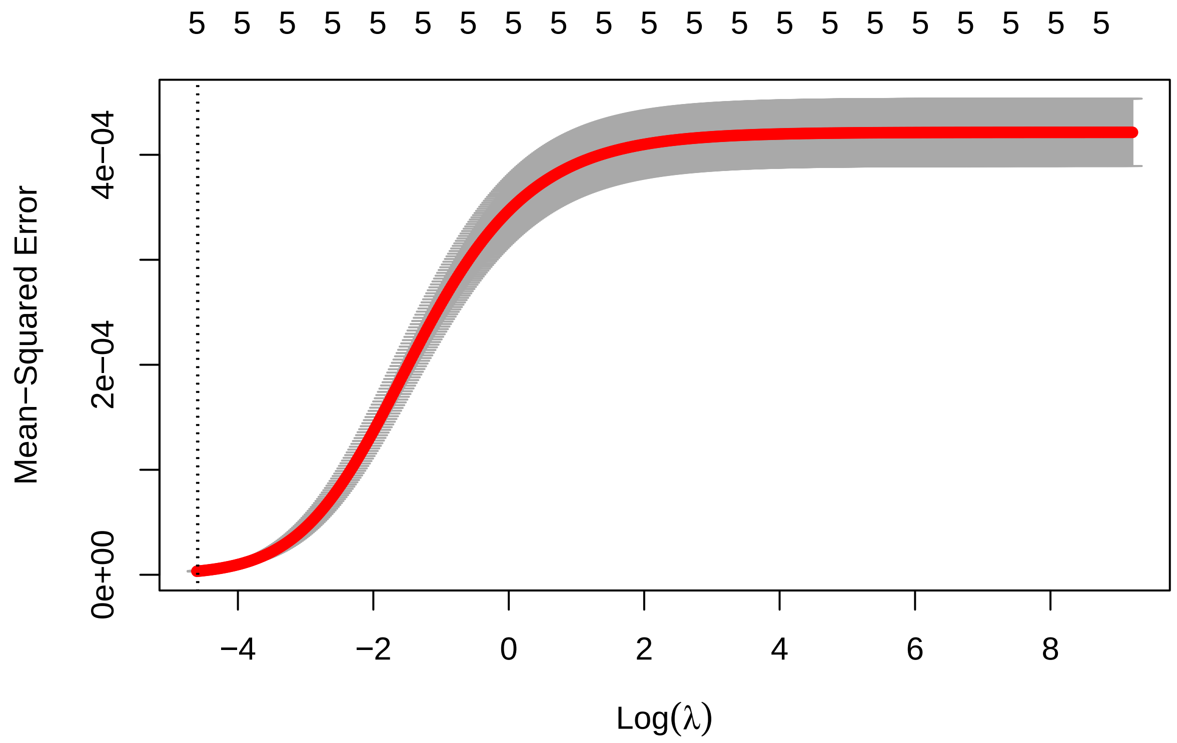
```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) 9.335306e-05
## r1          1.820913e-01
## r2          1.822399e-01
## r3          1.821839e-01
## r4          1.820328e-01
## r5          1.818509e-01
```

Question 2.5

Using the same train/test split as in Question 2.2. Using LASSO regression, produce an AR(5) linear regression. Find the optimal tuning parameter for this regression. What is the test mean squared error?

Solution:

```
lambda=10^seq(-2,4,by=.01)
lasso.mod = glmnet(X_train, Y_train, alpha = 1, family="gaussian", lambda=lambda)
cv.out=cv.glmnet(X_train, Y_train, alpha=0,lambda=lambda)
plot(cv.out)
```



```
bestlam=cv.out$lambda.min
cat("Optimal Lambda (Lasso): ", bestlam, "\n")
```

```
## Optimal Lambda (Lasso):  0.01
```

```
bestridge.pred=predict(lasso.mod,s=bestlam,newx=X_test)
cat("Lasso Model MSE: ", mean((bestridge.pred-aapl_rets_lagged$r0[-train])^2), "\n") #Lasso MSE
```

```
## Lasso Model MSE:  9.560932e-05
```

```
lassocoef=glmnet(aapl_rets_lagged[,x_var],aapl_rets_lagged[,names(aapl_rets_lagged) %in% y_var],alpha=1
predict(lassocoef,type="coefficients",s=bestlam)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) 5.133494e-04
## r1          5.072826e-01
## r2          1.503242e-13
## r3          .
## r4          .
## r5          .
```

Question 2.6

Briefly (1 paragraph or less) compare the coefficients for the 3 AR(5) regressions computed.

Solution:

Coefficients: (Intercept) r1 r2 r3 r4 r5
1.773e-18 1.000e+00 NA NA NA NA

Lambda (Ridge): 0.01 Ridge Model MSE: 3.751595e-06 6 x 1 sparse Matrix of class "dgCMatrix" s1 (Intercept) 9.335306e-05 r1 1.820913e-01 r2 1.822399e-01 r3 1.821839e-01 r4 1.820328e-01 r5 1.818509e-01

Optimal Lambda (Lasso): 0.01 Lasso Model MSE: 0.0001141452 6 x 1 sparse Matrix of class "dgCMatrix" s1 (Intercept) 0.0005133494 r1 0.5072825685 r2 .
r3 .
r4 .
r5 .

All of these models gave very different coefficients. For the unregularized linear model, r1 had a coefficient of 1, making me think that something was wrong with my implementation. But it seems that r0 and lagged r0 are very similar according to lm.

For ridge regression, all the coefficients are about .2, which is surprising because all of them have similar impact on the outcome. In contrast, Lasso was very different with r1 being the only feature important enough to weight. Against the coefficient is .507, which is quite large. I think this dataset isn't heavily dependent on lags and any regressions that are found are similar and can be aggregated into a single r1 coefficient.

Question 3 (15pt)

Question 3.1

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.

Solution:

For this example, I used Boston dataset available through MASS library. X, Y both passed in as matrices.

```r
library(MASS)
library(tibble)
library(matlib)

multi_lin_reg <- function(X_data, Y_data){
  beta0 = rep(1,length(Y_data))
  X_data = cbind(beta0, X_data)

  beta_hat = solve(t(X_data) %*% X_data) %*% t(X_data) %*% Y_data
  beta_hat
}

# for some reason, the lagged data is exactly singular. Not sure why, but I will use Boston data

# multi_lin_reg(as.matrix(aapl_rets_lagged[-1]), aapl_rets_lagged$r0)
Y <- Boston$medv
X <- as.matrix(Boston[-ncol(Boston)])

multi_lin_reg(X, Y)
```

```
##                  [,1]
## beta0    3.645949e+01
## crim    -1.080114e-01
## zn       4.642046e-02
## indus    2.055863e-02
## chas     2.686734e+00
## nox     -1.776661e+01
## rm       3.809865e+00
## age      6.922246e-04
## dis     -1.475567e+00
## rad      3.060495e-01
## tax     -1.233459e-02
## ptratio -9.527472e-01
## black    9.311683e-03
## lstat   -5.247584e-01
```

Question 3.2

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

Solution:

```r
lm(medv~., data = Boston)
```

```
##
## Call:
```

```
## lm(formula = medv ~ ., data = Boston)
##
## Coefficients:
## (Intercept)         crim           zn        indus         chas          nox
##   3.646e+01   -1.080e-01    4.642e-02    2.056e-02    2.687e+00   -1.777e+01
##          rm          age          dis          rad          tax      ptratio
##   3.810e+00    6.922e-04   -1.476e+00    3.060e-01   -1.233e-02   -9.527e-01
##       black        lstat
##   9.312e-03   -5.248e-01
```

Here we can see that the betas we get are exactly the same.