QF301. Lecture 5 In-Class Assignment.

2021-09-20

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: 9/17/2021

Question 1 (100pt)

Question 1.1

```
CWID = 10447455 #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)
```

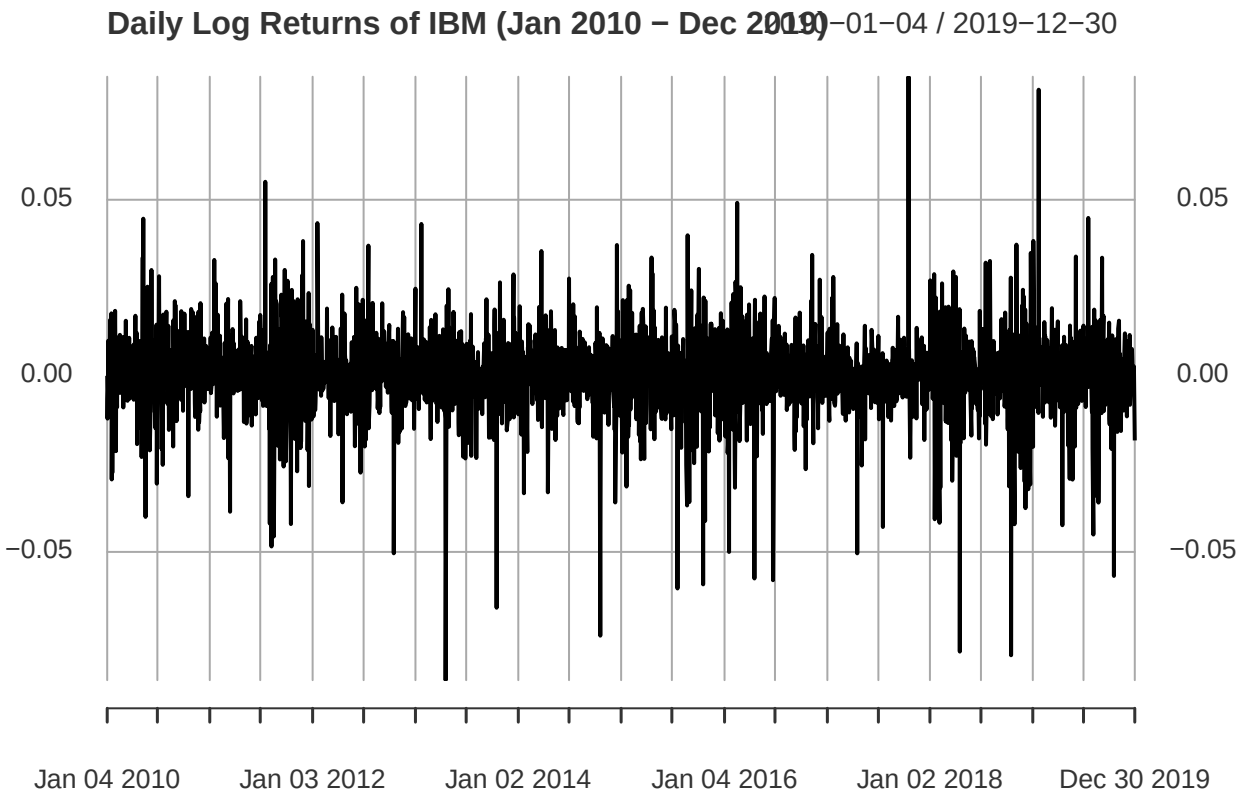
Obtain the daily log returns for IBM from January 1, 2010 until December 31, 2019. Plot these returns.

Solution:

```
library("quantmod")
getSymbols("IBM", from="2010-01-01", to="2019-12-31", warnings=FALSE)
```

```
## [1] "IBM"
```

```
dly_rets = dailyReturn(IBM$IBM.Adjusted,type = "log")
plot(dly_rets, main="Daily Log Returns of IBM (Jan 2010 - Dec 2019)")
```



Question 1.2

Calculate the mean and standard deviation of these daily returns. Do your answers make sense?

Solution:

```
print(mean(dly_rets))
```

```
## [1] 0.0001180961
```

```
print(sd(dly_rets))
```

```
## [1] 0.01236904
```

It makes sense that the daily log returns are very close to zero. . . annualized, they are 2.97%. This small number is consistent with returns that we generally assume are standard normal with a light positive mean shift.

The daily variance is .01237. Annualized, this is 19.637%. This is consistent with volatility seen in other technology equities.

Question 1.3

Using multiple linear regressions, produce an AR(10) linear regression. Use 50% of your data for training and 50% for testing performance. What is the test mean squared error?

Solution:

```
x1 = as.numeric(lag(dly_rets,k=1))[-(1:10)]
x2 = as.numeric(lag(dly_rets,k=2))[-(1:10)]
x3 = as.numeric(lag(dly_rets,k=3))[-(1:10)]
x4 = as.numeric(lag(dly_rets,k=4))[-(1:10)]
x5 = as.numeric(lag(dly_rets,k=5))[-(1:10)]
x6 = as.numeric(lag(dly_rets,k=6))[-(1:10)]
x7 = as.numeric(lag(dly_rets,k=7))[-(1:10)]
x8 = as.numeric(lag(dly_rets,k=8))[-(1:10)]
x9 = as.numeric(lag(dly_rets,k=9))[-(1:10)]
x10 = as.numeric(lag(dly_rets,k=10))[-(1:10)]
y = as.numeric(dly_rets)[-(1:10)]

df = data.frame(y, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10)

nrows = nrow(df)
train_rows = sample(nrows, floor(.5*nrows), replace=FALSE)

train_df = df[train_rows, ]
test_df = df[-train_rows, ]

library(glmnet)
ar10 = lm(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10)
summary(ar10)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
##     x10)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.086493 -0.005919  0.000397  0.006301  0.084753
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0001293  0.0002476   0.522    0.601
## x1          -0.0036281  0.0200249  -0.181    0.856
## x2          -0.0021796  0.0200143  -0.109    0.913
## x3          -0.0106141  0.0200124  -0.530    0.596
## x4          -0.0246832  0.0200123  -1.233    0.218
## x5          -0.0271430  0.0200096  -1.356    0.175
## x6          -0.0238077  0.0200083  -1.190    0.234
## x7          -0.0061048  0.0200079  -0.305    0.760
## x8          -0.0133299  0.0200059  -0.666    0.505
## x9           0.0215940  0.0200038   1.079    0.280
```

```
## x10          0.0270854  0.0200083   1.354    0.176
##
## Residual standard error: 0.01238 on 2494 degrees of freedom
## Multiple R-squared:  0.003581,   Adjusted R-squared:  -0.0004147
## F-statistic: 0.8962 on 10 and 2494 DF,  p-value: 0.5359
```

```
y_test_actual = test_df["y"]
y_test_pred = predict(ar10, test_df[, 2:ncol(test_df)])

RSS_error = sum((y_test_pred - y_test_actual)^2)
RSS_error
```

```
## [1] 0.1826033
```

Question 1.4

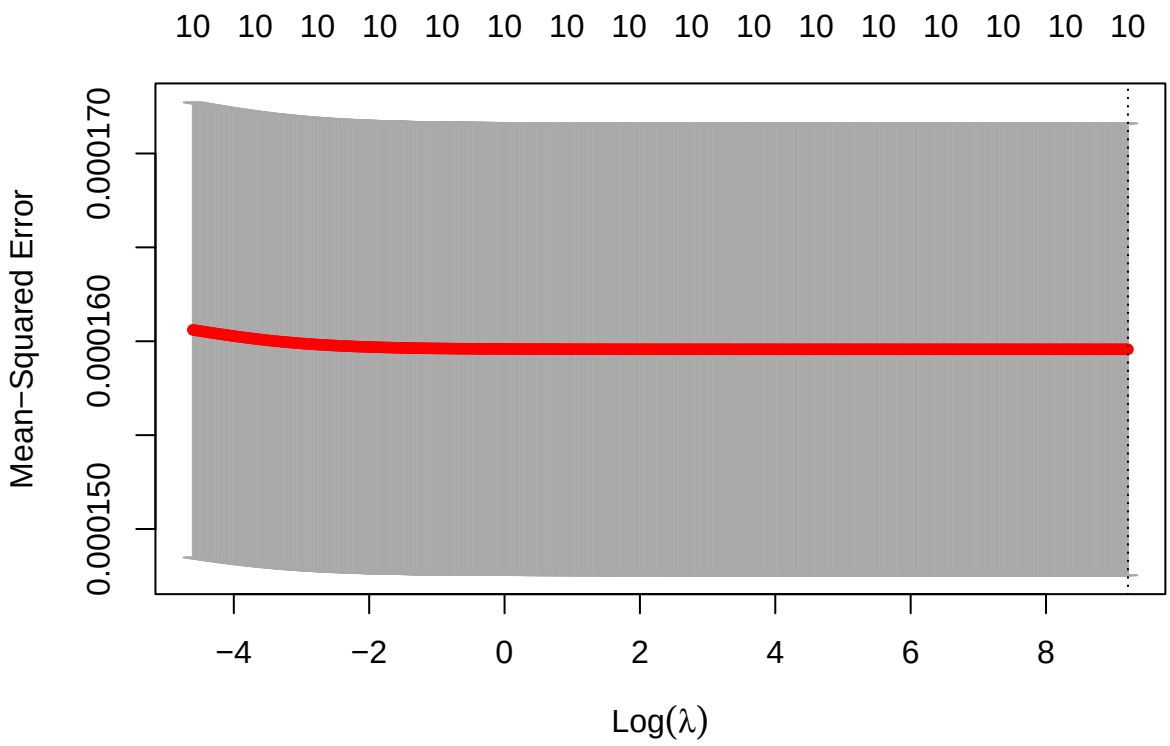
Using Ridge regression, produce an AR(10) linear regression. Use the same train/test split as in Question 1.3. Find the optimal tuning parameter for this regression. What is the test mean squared error?

Solution:

```
# X, Y train and test split
X = data.matrix(data.frame(df[2:ncol(df)]))
Y = data.matrix(data.frame(df["y"]))

X_train = data.matrix(data.frame(train_df[2:ncol(train_df)]))
Y_train = data.matrix(data.frame(train_df["y"]))

X_test = data.matrix(data.frame(test_df[2:ncol(test_df)]))
Y_test = data.matrix(data.frame(test_df["y"]))

# Baseline GLM model
library(glmnet)
base=glm(y~.,data=train_df) #Compare to base model
pred=predict(base,data.frame(X_test))
cat("Baseline GLM Model: ", mean((pred-test_df$y)^2), "\n") #Baseline MSE
```

```
## Baseline GLM Model:  0.000148846
```

```
# Ridge Regression
lambda=10^seq(-2,4,by=.01)
ridge.mod = glmnet(X_train, Y_train, lambda=lambda)
cv.out=cv.glmnet(X_train, Y_train, alpha=0,lambda=lambda)
plot(cv.out)
```




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

```
## Lambda (Ridge):  10000
```

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

```
## Ridge Model MSE:  0.00014727
```

```
ridgecoef=glmnet(X,Y,alpha=0,lambda=lambda)
predict(ridgecoef,type="coefficients",s=bestlam)
```

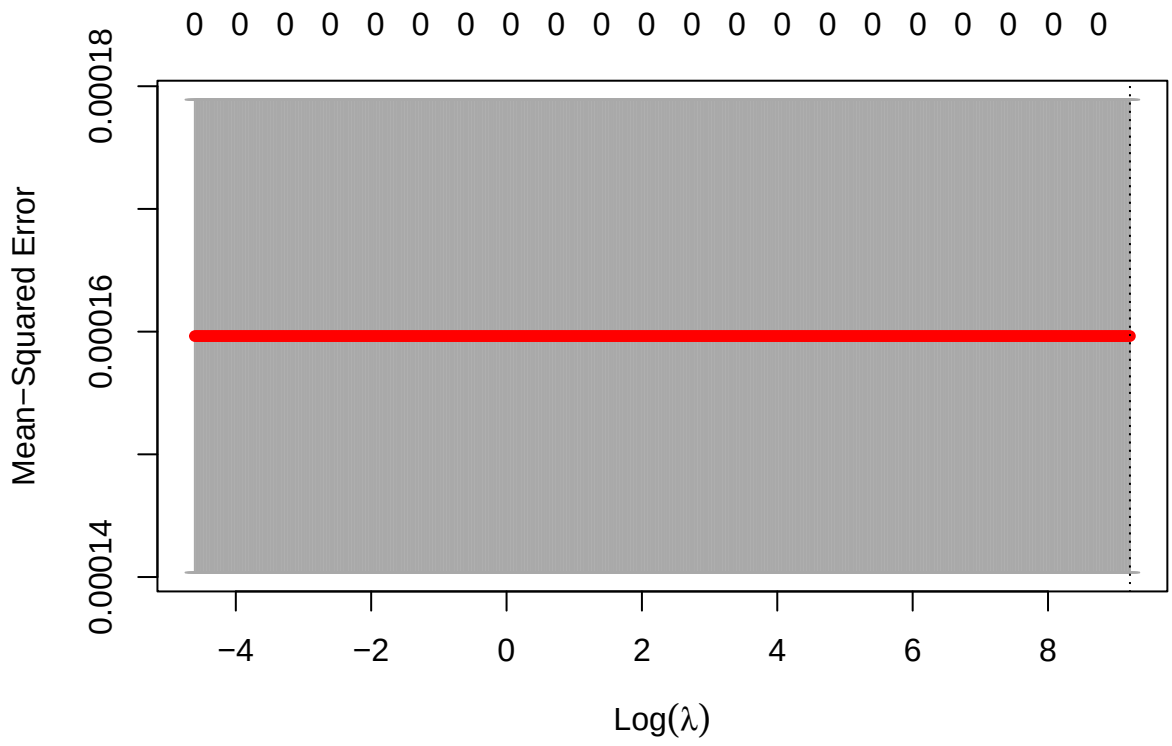
```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)  1.205919e-04
## x1          -1.725622e-09
## x2          -1.477228e-09
## x3          -1.293110e-08
## x4          -3.148455e-08
## x5          -3.474999e-08
## x6          -3.015497e-08
## x7          -6.983245e-09
## x8          -1.493838e-08
## x9           2.912109e-08
## x10          3.600027e-08
```

Question 1.4

Using LASSO regression, produce an AR(10) linear regression. Use the same train/test split as in Question 1.3. Find the optimal tuning parameter for this regression. What is the test mean squared error?

Solution:

```
# Lasso Regression
lasso.mod=glmnet(X_train,Y_train, alpha=1,lambda=lambda) #LASSO is with alpha=1
cv.out=cv.glmnet(X_train,Y_train, alpha=1,lambda=lambda)
plot(cv.out)
```




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

```
## Lambda (Lasso):  10000
```

```
bestlasso.pred=predict(lasso.mod,s=bestlam,newx=X_test)
cat("Lasso Model MSE: ", mean((bestlasso.pred-test_df$y)^2), "\n") #Lasso MSE
```

```
## Lasso Model MSE:  0.00014727
```

```
lassocoef=glmnet(X,Y,alpha=1,lambda=lambda)
predict(lassocoef,type="coefficients",s=bestlam)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) 0.0001205919
## x1          0.0000000000
## x2          .
## x3          .
## x4          .
## x5          .
## x6          .
## x7          .
## x8          .
## x9          .
## x10         .
```

Question 1.4

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

Solution:

I have already computed the coefficients for Ridge and Lasso above. The main differences I can spot is that Ridge still assigns a coefficient to every factor while Lasso completely eliminates the factors that are unimportant.

Lasso seems to provide feature selection in addition to regression.

After a bit of research, it seems that Lasso's weight constraint region |w1|+|w2|<=t is more likely to intersect with the least squares error function's contour line in such a way that a certain component can receive no weight.