QF301. Homework #2.

2022-10-07

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: Arjun Koshal

CWID: 10459064

Date: 10/07/2022

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 = 10459064 #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 both both stocks. Print the first and last 6 lines of your data frame.

Solution:

```
library(quantmod)
getSymbols("AMZN", from = "2019-01-01", to = "2021-12-31", src = "yahoo")
```

```
## [1] "AMZN"
```

```
getSymbols("TSLA", from = "2019-01-01", to = "2021-12-31", src = "yahoo")
```

```
## [1] "TSLA"
```

```
AMZN <- dailyReturn(AMZN$AMZN.Adjusted, type = "log")
TSLA <- dailyReturn(TSLA$TSLA.Adjusted, type = "log")
df <- data.frame(AMZN, TSLA)
colnames(df) = c("AmazonReturns", "TeslaReturns")
head(df, 6)
```

```
##            AmazonReturns TeslaReturns
## 2019-01-02   0.000000000  0.000000000
## 2019-01-03  -0.025565525 -0.031977580
## 2019-01-04   0.048851116  0.056094294
## 2019-01-07   0.033776507  0.052935088
## 2019-01-08   0.016475880  0.001163641
## 2019-01-09   0.001712895  0.009437908
```

```
tail(df, 6)
```

```
##            AmazonReturns TeslaReturns
## 2021-12-22  0.0036315092  0.072271410
## 2021-12-23  0.0001841186  0.056020023
## 2021-12-27 -0.0082116459  0.024934916
## 2021-12-28  0.0058267057 -0.005012817
## 2021-12-29 -0.0085917224 -0.002096908
## 2021-12-30 -0.0032944260 -0.014699782
```

Question 1.2

List the names of the variables in the data set.

Solution:

```
names(df)
```

```
## [1] "AmazonReturns" "TeslaReturns"
```

Question 1.3

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

Solution:

```
df2 <- data.frame(df, row.names = NULL)
head(df2)
```

```
##   AmazonReturns TeslaReturns
## 1   0.000000000  0.000000000
## 2  -0.025565525 -0.031977580
## 3   0.048851116  0.056094294
## 4   0.033776507  0.052935088
## 5   0.016475880  0.001163641
## 6   0.001712895  0.009437908
```

Question 1.4

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

Solution:

```
table <- matrix(c(mean(df2$AmazonReturns), sd(df2$AmazonReturns),
mean(df2$TeslaReturns), sd(df2$TeslaReturns)), ncol = 2, byrow = FALSE)

colnames(table) <- c("Amazon","Tesla")
rownames(table) <- c("Mean", "Standard Deviation")
table
```

```
##                        Amazon       Tesla
## Mean               0.001037768 0.003767474
## Standard Deviation 0.018480715 0.042144209
```

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:

```
model <- lm(df$AmazonReturns~df$TeslaReturns)
summary(model)
```

```
##
## Call:
## lm(formula = df$AmazonReturns ~ df$TeslaReturns)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.081601 -0.009470 -0.000611  0.008858  0.073590
```

```
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.0003671  0.0006171   0.595    0.552
## df$TeslaReturns 0.1780277  0.0145944  12.198   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0169 on 754 degrees of freedom
## Multiple R-squared:  0.1648, Adjusted R-squared:  0.1637
## F-statistic: 148.8 on 1 and 754 DF,  p-value: < 2.2e-16
```

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 3 days back.

Solution:

```
r1 <- as.numeric(lag (AMZN, k = 1))[-(1:3)]
r2 <- as.numeric(lag (AMZN, k = 2))[-(1:3)]
r3 <- as.numeric(lag (AMZN, k = 3))[-(1:3)]
r0 <- as.numeric(AMZN)[-(1:3)]
df_lag <- data.frame(r0, r1, r2, r3)
head(df_lag)
```

```
##             r0           r1           r2           r3
## 1  0.033776507  0.048851116 -0.025565525  0.000000000
## 2  0.016475880  0.033776507  0.048851116 -0.025565525
## 3  0.001712895  0.016475880  0.033776507  0.048851116
## 4 -0.001930295  0.001712895  0.016475880  0.033776507
## 5 -0.009500214 -0.001930295  0.001712895  0.016475880
## 6 -0.014335242 -0.009500214 -0.001930295  0.001712895
```

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:

```
x1 <- as.numeric(lag(AMZN, k=1))[-(1:1)]
y <- as.numeric(AMZN)[-(1:1)]
df_ar1 <- data.frame(y, x1)
length <- length(AMZN)
```

```r
train <- sample(length, length/2, replace = FALSE)
ar1 = lm(y[train]~x1[train])
summary(ar1)
```

```
##
## Call:
## lm(formula = y[train] ~ x1[train])
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.079106 -0.010855  0.000661  0.010352  0.070957
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0003297  0.0009713   0.339    0.734
## x1[train]   -0.0134270  0.0540824  -0.248    0.804
##
## Residual standard error: 0.01883 on 376 degrees of freedom
## Multiple R-squared:  0.0001639,  Adjusted R-squared:  -0.002495
## F-statistic: 0.06164 on 1 and 376 DF,  p-value: 0.8041
```

```r
Ar1_Test = lm(y[-train]~x1[-train])

MSE <- mean(Ar1_Test$residuals^2)
MSE
```

```
## [1] 0.0003244583
```

Question 2.3

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

Solution:

```r
summary(ar1)$coefficients
```

```
##                  Estimate   Std. Error    t value  Pr(>|t|)
## (Intercept)  0.0003296759 0.0009713414  0.3394028 0.7344958
## x1[train]   -0.0134269890 0.0540824350 -0.2482689 0.8040619
```

```r
b0 <- summary(ar1)$coefficients[1,1]
b1 <- summary(ar1)$coefficients[2,1]
b1.se <- summary(ar1)$coefficients[2,2]
abs(b1) < 1
```

```
## [1] TRUE
```

```r
t <- (b1 - 1)/b1.se
t
```

```
## [1] -18.73856
```

```r
pnorm(t)
```

```
## [1] 1.200166e-78
```

We can see from the fact the absolute value of b1 is less than 1 that model is weakly stationary.

(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
mu <- b0 / (1 - b1)
gamma0 <- MSE / (1 - b1^2)
gamma1 <- b1*gamma0
gamma2 <- b1*gamma1
gamma3 <- b1*gamma2
mean(AMZN) - mu
```

```
## [1] 0.00071246
```

```r
cov(AMZN, AMZN) - gamma0
```

```
##                daily.returns
## daily.returns  1.702001e-05
```

```r
cov(AMZN[-1], lag(AMZN, k = 1)[-1]) - gamma1
```

```
##                daily.returns
## daily.returns -1.855918e-05
```

```r
cov(AMZN[-(1:2)], lag(AMZN, k = 2)[-(1:2)]) - gamma2
```

```
##                daily.returns
## daily.returns  1.853037e-06
```

```r
cov(AMZN[-(1:3)], lag(AMZN, k = 3)[-(1:3)]) - gamma3
```

```
##                daily.returns
## daily.returns -1.834645e-05
```

We can see that since the difference between the empirical and theoretical data is extremely small and near 0, that the empirical data is similar to the theoretical data.

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

Solution:

```
N <- length(df_ar1$y)
sig <- sqrt(MSE)
Hat_R <- matrix(nrow = length(df_ar1$y)-5, ncol = 5)
e <- matrix(nrow = N - 5, ncol = 5)
Var_e <- rep(NA, 5)
Hat_R[,1] <- b0 + b1*df_ar1$y[1:(N-5)]
e[,1] <- df_ar1$y[2:(N-4)] - Hat_R[,1]
Var_e[1] <- sig^2
for (l in seq(2,5)) {
Hat_R[,l] <- b0 + b1*Hat_R[,l-1]
e[,l] <- df_ar1$y[(l+1):(N-5+l)] - Hat_R[,l]
Var_e[l] <- sig^2 + b1^2*Var_e[l-1]
}

Emp_e <- c(var(e[,1]),var(e[,2]),var(e[,3]),var(e[,4]),var(e[,5]))
sqrt(Emp_e)
```

```
## [1] 0.01850600 0.01844163 0.01840357 0.01839817 0.01839880
```

Question 2.3

Using the same train/test split as in Question 2.2. Create an AR(3) model for your stock returns. Provide the test mean squared error.

Solution:

```
train_r0 = sample(length(r0), length(r0)/2, replace = FALSE)
lin.mod = glm(r0~., data = df_lag, subset = train_r0)
summary(lin.mod)
```

```
##
## Call:
## glm(formula = r0 ~ ., data = df_lag, subset = train_r0)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.080972  -0.009135  -0.000080   0.009036   0.073521
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0012966  0.0009937   1.305    0.193
## r1          -0.0775517  0.0542491  -1.430    0.154
## r2          -0.0142695  0.0520467  -0.274    0.784
## r3          -0.0187208  0.0510887  -0.366    0.714
```

```
##
## (Dispersion parameter for gaussian family taken to be 0.0003655312)
##
##     Null deviance: 0.13682  on 375  degrees of freedom
## Residual deviance: 0.13598  on 372  degrees of freedom
## AIC: -1902.7
##
## Number of Fisher Scoring iterations: 2
```

```
pred = predict(lin.mod, df_lag[-train_r0,])
MSE_ar3 = mean((pred-df_lag$r0[-train_r0])^2)
MSE_ar3
```

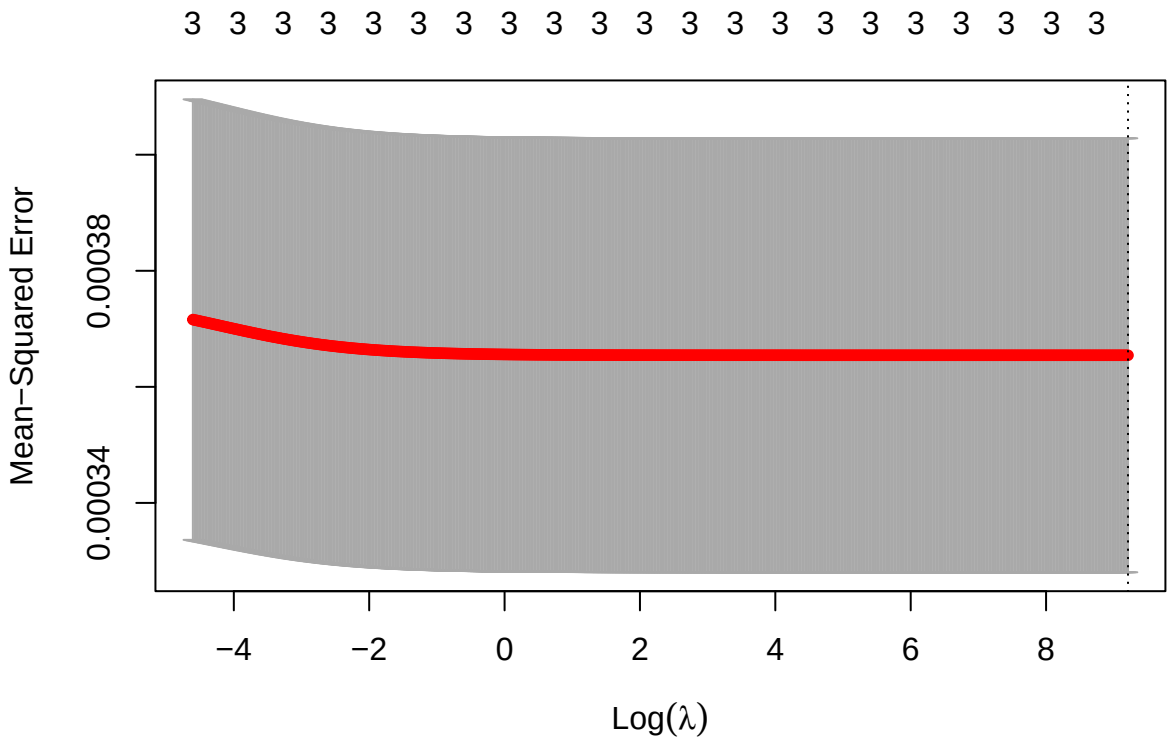
```
## [1] 0.0003119182
```

Question 2.4

Using the same train/test split as in Question 2.2. Using Ridge regression, produce an AR(3) linear regression. Find the optimal tuning parameter for this regression. What is the test mean squared error?

Solution:

```
library(glmnet)
x = model.matrix(r0~., df_lag)[,-1]
y = df_lag$r0
lambda = 10^seq(-2, 4, by = .01)
ridge.mod = glmnet(x[train_r0,], y[train_r0], alpha = 0, lambda = lambda)
cv.out = cv.glmnet(x[train_r0,], y[train_r0], alpha = 0, lambda = lambda)
plot(cv.out)
```




```
ridge.bestlam = cv.out$lambda.min
ridge.bestlam
```

```
## [1] 10000
```

```
bestridge.pred = predict(ridge.mod, s = ridge.bestlam, newx = x[-train_r0,])
mean((bestridge.pred-y[-train_r0])^2)
```

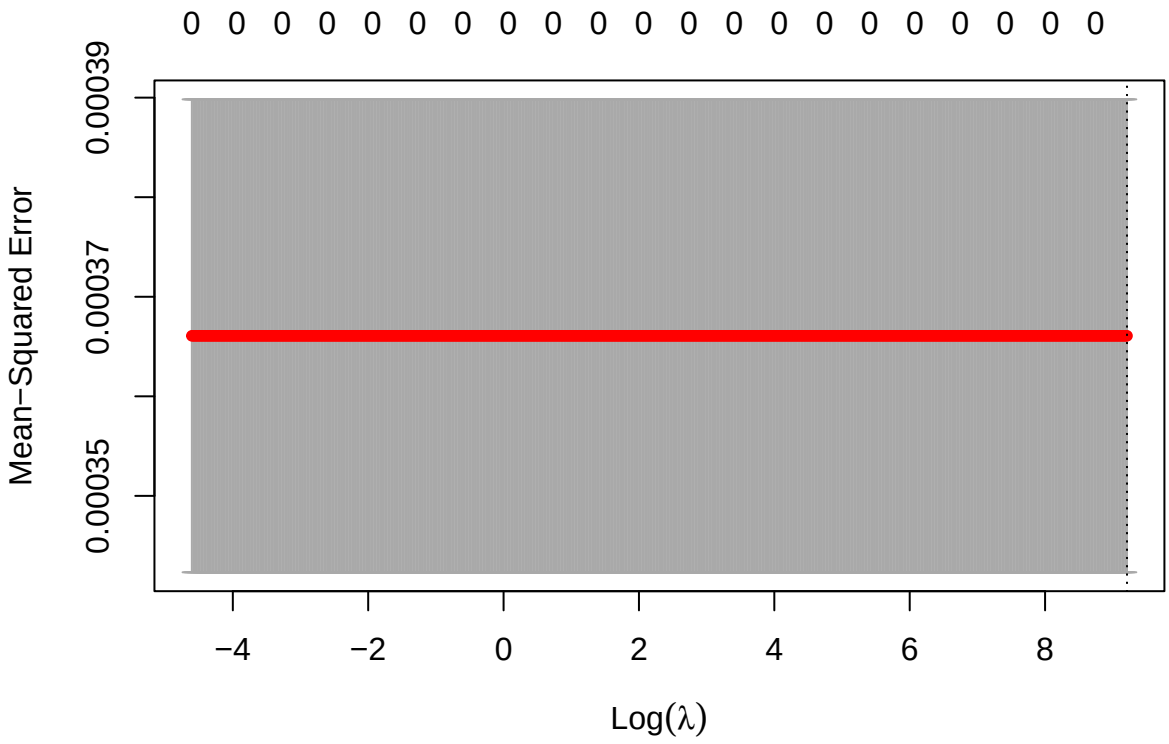
```
## [1] 0.0003131761
```

Question 2.5

Using the same train/test split as in Question 2.2. Using LASSO regression, produce an AR(3) linear regression. Find the optimal tuning parameter for this regression. What is the test mean squared error?

Solution:

```
library(glmnet)
lasso.mod = glmnet(x[train_r0,], y[train_r0], alpha = 1, lambda = lambda)
cv.out = cv.glmnet(x[train_r0,], y[train_r0], alpha = 1, lambda = lambda)
plot(cv.out)
```




```
lasso.bestlam=cv.out$lambda.min
lasso.bestlam
```

```
## [1] 10000
```

```
bestlasso.pred = predict(lasso.mod, s = lasso.bestlam, newx = x[-train_r0,])
mean((bestlasso.pred-y[-train_r0])^2)
```

```
## [1] 0.0003131761
```

Question 2.6

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

Solution:

```
summary(lin.mod)$coefficients
```

```
##                Estimate   Std. Error    t value  Pr(>|t|)
## (Intercept)  0.001296567 0.0009937227  1.3047570 0.1927822
## r1          -0.077551675 0.0542490849 -1.4295481 0.1536860
## r2          -0.014269539 0.0520467364 -0.2741678 0.7841079
## r3          -0.018720774 0.0510887268 -0.3664365 0.7142477
```

```
ridgecoef = glmnet(x[train_r0,], y[train_r0], alpha = 0, lambda = lambda)
predict(ridgecoef, type="coefficients", s = ridge.bestlam)
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)  1.201648e-03
## r1          -1.503172e-07
## r2          -3.247837e-08
## r3          -3.838424e-08
```

```
lassocoef = glmnet(x[train_r0,], y[train_r0], alpha = 1, lambda = lambda)
predict(lassocoef, type="coefficients", s = lasso.bestlam)
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) 0.001201647
## r1          0.000000000
## r2          .
## r3          .
```

We can see that the LASSO regression coefficients are nearly the same, as all of them are very close to 0. The same can be said about the Ridge regression coefficients, as all of them are nearly 0 as well, exlcuding the intercept. From this we can infer that there is a weak historical relationship.

Question 3 (15pt)

Question 3.1

Write a function that works in R (or python if submitting a Jupyter notebook) 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 can**not** use the `lm` command in this function or any of the other built in regression models.

Solution:

```
linear_regression = function(X, y) {
  beta = solve(t(X) %*% X) %*% t(X) %*% y
  sig2 = mean((y-X %*% beta)^2)
  se = sig^2*solve(t(X) %*% X)
  se = diag(se)
  output = list(beta, se)
  names(output) = c("beta","SE")
  return(output)
}
```

Question 3.2

Compare the output of your function to that of the `lm` command in R (or your favorite choice of built in linear regression function) on sample data. You may use the financial data from Questions 1 and 2 or examples from lecture notes.

Solution:

```
fit <- lm(AMZN ~ TSLA, df)
fit
```

```
##
## Call:
## lm(formula = AMZN ~ TSLA, data = df)
##
## Coefficients:
## (Intercept)         TSLA
##   0.0003671    0.1780277
```

```
linear_regression(TSLA, AMZN)
```

```
## $beta
##                daily.returns
## daily.returns     0.1788012
##
## $SE
## daily.returns
##  0.0002400351
```

The output for the function created in 3.1 and the lm function are nearly the same.