Final Exam

Nicholas Colonna

5/7/2018

## Practical Problems

library("forecast")

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

## Loading required package: timeDate

## This is forecast 7.3

library("fBasics")

## Loading required package: timeSeries

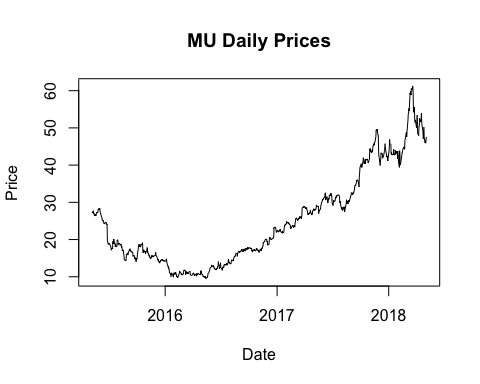
##   
## Attaching package: 'timeSeries'

## The following object is masked from 'package:zoo':  
##   
## time<-

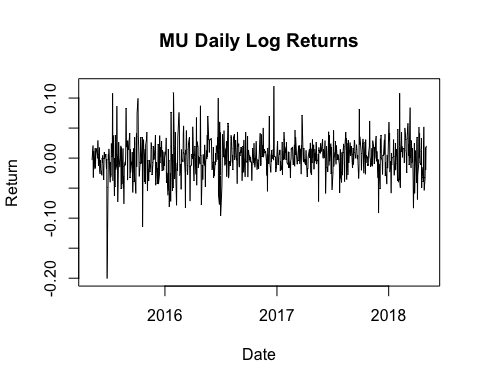
library("fUnitRoots")

**#Part A**  
MU <- read.csv("MU.csv", header=T)  
MU$Date <- as.Date(MU$Date, format="%Y-%m-%d")

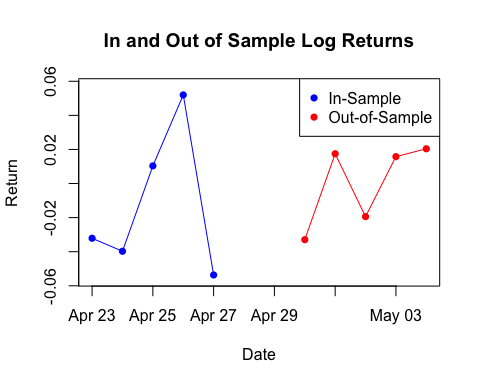
prices <- data.frame(MU$Date, MU$Adj.Close)  
colnames(prices) <- c("Date", "Price")  
plot(prices, main="MU Daily Prices", type="l")



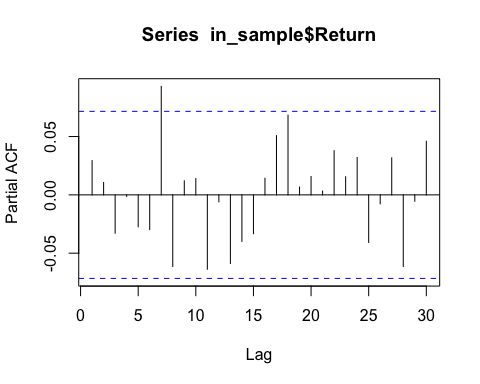
returns <- data.frame(MU$Date[2:755], diff(log(prices$Price), lag=1))  
colnames(returns) <- c("Date", "Return")  
plot(returns, main="MU Daily Log Returns", type="l")



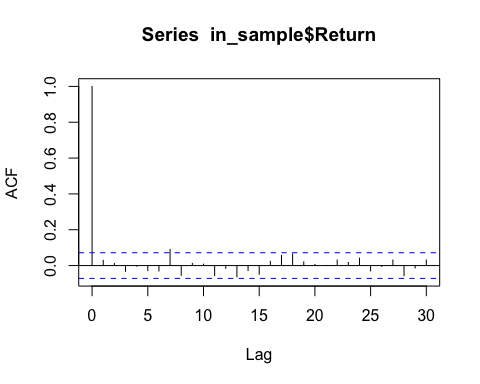
**#Part B**  
in\_sample <- data.frame(returns$Date[1:749], returns$Return[1:749])  
colnames(in\_sample) <- c("Date", "Return")  
  
out\_sample <- data.frame(returns$Date[750:754], returns$Return[750:754])  
colnames(out\_sample) <- c("Date", "Return")  
  
in\_sample\_last5 <- data.frame(in\_sample$Date[745:749], in\_sample$Return[745:749])  
colnames(in\_sample\_last5) <- c("Date", "Return")  
  
plot(in\_sample\_last5, main="In and Out of Sample Log Returns", col="blue", xlim=as.Date(c("2018-04-23","2018-05-04")), ylim=c(min(out\_sample$Return, in\_sample\_last5$Return)-.002, max(out\_sample$Return, in\_sample\_last5$Return)+.005), type="o", pch=16)  
lines(out\_sample, col="red", type="o", pch=16)  
legend("topright", c("In-Sample", "Out-of-Sample"), col=c("blue", "red"), pch=16)



**#Part C**  
pacf(in\_sample$Return, lag.max=30) #The PACF is used to give a recommended order for the AR model. In this case it is order 7.



acf(in\_sample$Return,lag.max=30) #The ACF is used to give a recommended order for the MA model. In this case it is order 7.



adfTest(in\_sample$Return, lags=7) #one test we learned in class was the Augmented Dickey-Fuller Test (ADF). If the p-value is small, it is likely that there are no unit roots. Since the p-value is significant here, we can conclude no unit-root.

## Warning in adfTest(in\_sample$Return, lags = 7): p-value smaller than  
## printed p-value

##   
## Title:  
## Augmented Dickey-Fuller Test  
##   
## Test Results:  
## PARAMETER:  
## Lag Order: 7  
## STATISTIC:  
## Dickey-Fuller: -9.7529  
## P VALUE:  
## 0.01   
##   
## Description:  
## Fri May 11 17:12:05 2018 by user:

Box.test(in\_sample$Return, type="Ljung") #When no lag, this gives us a pretty high p-value, so we accept H0 that all correlations are 0.

##   
## Box-Ljung test  
##   
## data: in\_sample$Return  
## X-squared = 0.64955, df = 1, p-value = 0.4203

Box.test(in\_sample$Return, lag=7, type="Ljung") #When we run the test again at lag 7, we still accept H0, since the p-value is greater than 0.05. However, we saw improvement from no lag. This could potentially mean this lag isn’t the best fit for our model.

##   
## Box-Ljung test  
##   
## data: in\_sample$Return  
## X-squared = 9.033, df = 7, p-value = 0.2503

max.p=7  
max.q=7  
model.aic <- matrix(NA,nrow=max.p+1,ncol=max.q+1)  
for (p in 0:max.p){  
 for (q in 0:max.q){  
 model.aic[p+1,q+1] = arima(in\_sample$Return, order=c(p,0,q), method="ML")$aic  
 }  
}

min(model.aic)

## [1] -3111.207

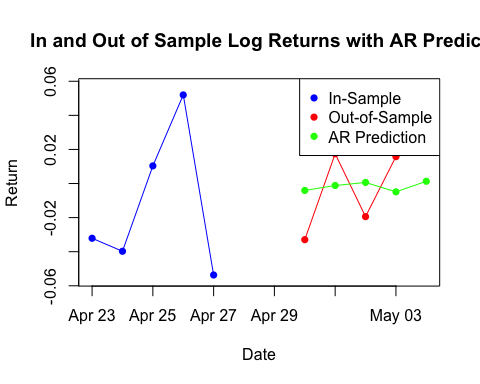
which(model.aic == min(model.aic), arr.ind = TRUE)

## row col  
## [1,] 3 3

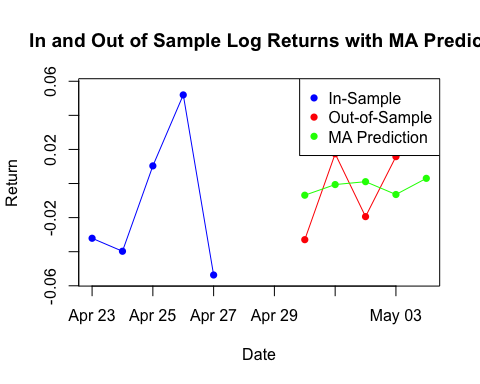
#This means that our recommended p=2 and recommonded q=2 for our ARMA model.  
  
  
#Now we build our models with this information  
AR <- arima(in\_sample$Return, order=c(7,0,0)) #We build our AR(p) model with p=7, since we found that lag to be significant with our PACF.  
MA <- arima(in\_sample$Return, order=c(0,0,7)) #We build our MA(q) model with q=7, since we found that lag to be significant with our ACF.  
ARMA <- arima(in\_sample$Return, order=c(2,0,2)) #Above, we tested various values for p and q for our ARMA(p,q) model to find the one with the lowest AIC value. After checking multiple values, I found that the best model to be p=2 and q=2.

#**Part D**  
predictAR <- data.frame(out\_sample$Date,predict(AR, 5))  
predictMA <- data.frame(out\_sample$Date,predict(MA, 5))  
predictARMA <- data.frame(out\_sample$Date,predict(ARMA, 5))  
colnames(predictAR) <- c("Date", "Return")  
colnames(predictMA) <- c("Date", "Return")  
colnames(predictARMA) <- c("Date", "Return")

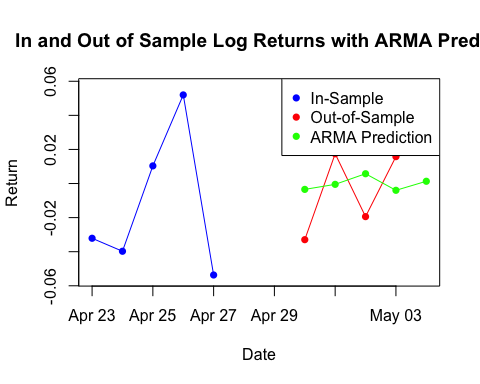
#plotting AR prediction over actual  
plot(in\_sample\_last5, main="In and Out of Sample Log Returns with AR Predict", col="blue", xlim=as.Date(c("2018-04-23","2018-05-04")), ylim=c(min(out\_sample$Return, in\_sample\_last5$Return, predictAR$Return)-.002, max(out\_sample$Return, in\_sample\_last5$Return, predictAR$Return)+.005), type="o", pch=16)  
lines(out\_sample, col="red", type="o", pch=16)  
lines(predictAR, col="green", type="o", pch=16)  
legend("topright", c("In-Sample", "Out-of-Sample","AR Prediction"), col=c("blue", "red", "green"), pch=16)



#plotting MA prediction over actual  
plot(in\_sample\_last5, main="In and Out of Sample Log Returns with MA Predict", col="blue", xlim=as.Date(c("2018-04-23","2018-05-04")), ylim=c(min(out\_sample$Return, in\_sample\_last5$Return, predictMA$Return)-.002, max(out\_sample$Return, in\_sample\_last5$Return, predictMA$Return)+.005), type="o", pch=16)  
lines(out\_sample, col="red", type="o", pch=16)  
lines(predictMA, col="green", type="o", pch=16)  
legend("topright", c("In-Sample", "Out-of-Sample","MA Prediction"), col=c("blue", "red", "green"), pch=16)



#plotting ARMA prediction over actual  
plot(in\_sample\_last5, main="In and Out of Sample Log Returns with ARMA Predict", col="blue", xlim=as.Date(c("2018-04-23","2018-05-04")), ylim=c(min(out\_sample$Return, in\_sample\_last5$Return, predictARMA$Return)-.002, max(out\_sample$Return, in\_sample\_last5$Return, predictARMA$Return)+.005), type="o", pch=16)  
lines(out\_sample, col="red", type="o", pch=16)  
lines(predictARMA, col="green", type="o", pch=16)  
legend("topright", c("In-Sample", "Out-of-Sample","ARMA Prediction"), col=c("blue", "red","green"), pch=16)



#**Part E**  
sseAR <- sum((out\_sample$Return-predictAR$Return)^2)  
sseAR

## [1] 0.002374043

sseMA <- sum((out\_sample$Return-predictMA$Return)^2)  
sseMA

## [1] 0.002222692

sseARMA <- sum((out\_sample$Return-predictARMA$Return)^2)  
sseARMA

## [1] 0.002580017

#Here we computed the sum of squared errors between each predicted model and the out of sample data. AR gave a value of 0.002374043, MA gave a value of 0.002222692, and ARMA gave a value of 0.002580017. You will notice that all of these models gave low values, however, the best choice of a model would be the one with the lowest sum of squared errors, which is the MA model in this case.

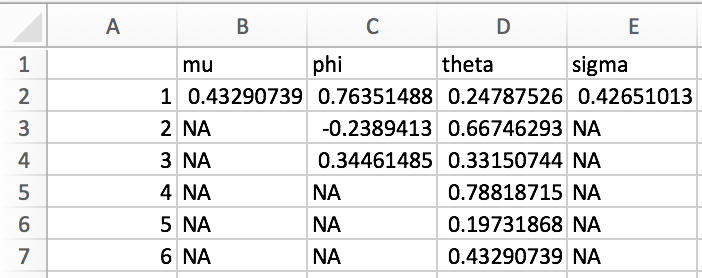
## 

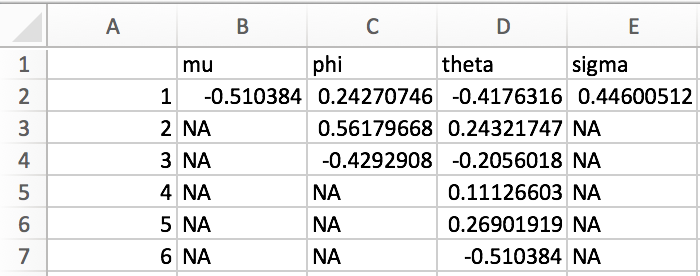
## Interview Problem

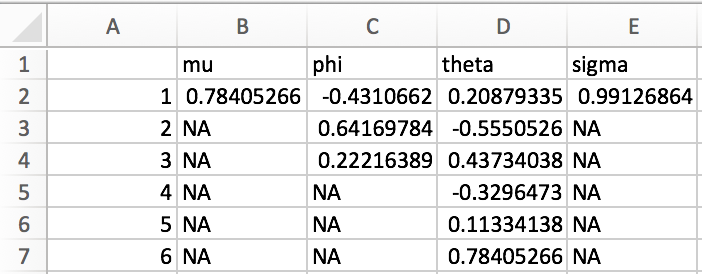
x <- read.csv("qf-202-final-data-xxx.csv", header=T)  
y <- read.csv("qf-202-final-data-yyy.csv", header=T)  
z <- read.csv("qf-202-final-data-zzz.csv", header=T)  
  
estimateParameters <- function(data, fileName){  
 model <- arima(data, order=c(3,2,6))  
 mu <- model$coef[9]  
 mu <- c(mu, NA,NA,NA,NA,NA)  
 phi <- model$model$phi  
 phi <- c(phi,NA,NA,NA)  
 theta <- model$model$theta  
 sigma <- sqrt(model$sigma2)  
 sigma <- c(sigma, NA,NA,NA,NA,NA)  
 result <- data.frame(mu, phi, theta, sigma)  
 write.csv(result, file=fileName)  
}  
  
estimateParameters(x$x, "xxx-result.csv")  
estimateParameters(y$x, "yyy-result.csv")  
estimateParameters(z$x, "zzz-result.csv")

#The approach I took to this problem is as follows. First, I read each csv file, saving it with its corresponding letter. I made sure to indicate header=T, since there are headers to our data. The next step I took was to create a function that would estimate the parameters for the data, no matter the data length. The function, estimateParameters, takes in 2 parameters, the first being the data and the second being the name of the file you want the result to output to. The function creates an ARIMA(3,2,6) model using the data it was called with. From there, I was able to access each component of the result. First, I found the mean (mu), which is the last coefficient for the model. Next, I found all the phi and theta parameters, referencing them from our created model. Lastly, I took the variance of the model and took the square root, so that I can get the standard deviation. Since there were a different number of parameters for each of the values we were seeking (mu, phi, theta, sigma), I padded my results with 'NA' to fill in empty spots, so that I could return a data frame that was nicely formatted.

**Below are screenshots from the outputted csv files from the Interview Type Problem**

****xxx-result.csv**

**yyy-result.csv**

**zzz-result.csv**