Assignment #5 Daily Returns

Scott Morgan

PART A

Solve textbook exercises 12.16: 1 and 2 on pages 356 (Ruppert). (20 pts.)

12.16 Exercises

1. This problem and the next use CRSP daily returns. First, get the data and plot the ACF in two ways:

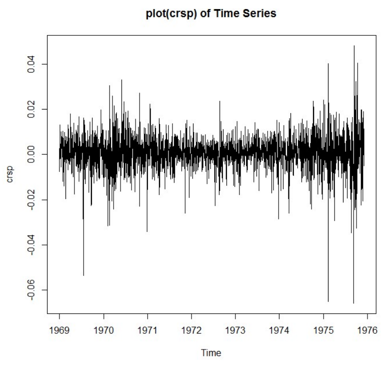
```
library(stats)
library(Ecdat)
library(tseries)
library(forecast)
library(PerformanceAnalytics)
library(dplyr)

data(CRSPday)
crsp=CRSPday[,7]
```

(a)Explain what "lag" means in the two ACF plots. Why does lag differ between the plots?

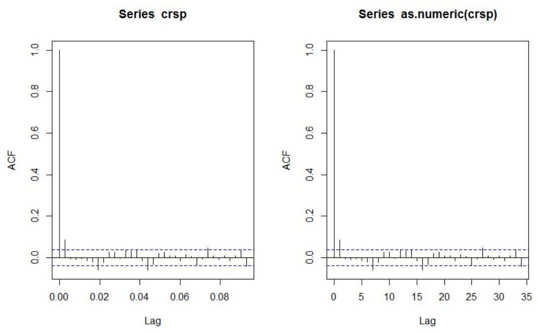
The term lag refers to the time distance between two time series that are being correlated and are useful because of autocorrelation which is the tendency for values in a time series to be correlated with previous copies of itself (Matt Dancho, 2017).

By executing the following a partial summary is given:

```
crsp

# Time Series:
# Start = c(1969, 1)
# End = c(1975, 338)
# Frequency = 365
#<truncated#

plot(crsp)
```



```
# acf graphs
par(mfrow=c(1,2))
acf(crsp)
acf(as.numeric(crsp))
par(mfrow=c(1,1))
```

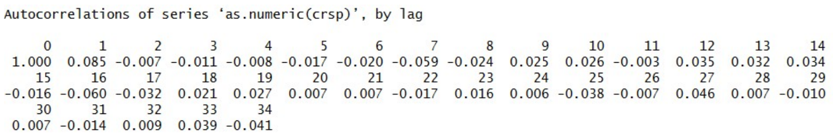


In the left graph plotting *acf(crsp),* the data series is a time series (ts). When plotting the time series with *acf()*, the lags are displayed relative to the time dimension associated with the data. Viewing the partial summary of the data using the topmost portion of the *crsp <enter>* code, we can see that the frequency is equal to 365, this a single lag which equates to 1/365 = 0.002739726 year time units. These figures are the values for the x-axis in the left chart. In the right chart that plots *acf(as.numeric(crsp)),* the column is coerced into a numeric array where row indexes are returned.

For example, note on the right chart that there is a spike at lag 1. This equates to 1/365 = 0.0027397 units of time in the left hand plot.

(b) At what values of lag are there significant autocorrelations in the CRSP returns? For which of these values do you think the statistical significance might be due to chance?
By using the above acf plots, there appears to be three significant autocorrelations with small lags at 1, 7, and 16. I anticipate that the autocorrelation at lag 1 to be statistically significant and the remaining at larger lags to be mostly due to chance. Autocorrelations at higher lags that appears significant (2 sd error bars) are likely due to chance. We can also see the *acf(as.numeric(crsp)* values excluding the plot by executing the following code:

```
acf(as.numeric(crsp),plot = FALSE)
```



```
Autocorrelations of series 'as.numeric(crsp)', by lag

     0      1      2      3      4      5      6      7      8      9     10     11     12     13     14 
 1.000  0.085 -0.007 -0.011 -0.008 -0.017 -0.020 -0.059 -0.024  0.025  0.026 -0.003  0.035  0.032  0.034 
    15     16     17     18     19     20     21     22     23     24     25     26     27     28     29 
-0.016 -0.060 -0.032  0.021  0.027  0.007  0.007 -0.017  0.016  0.006 -0.038 -0.007  0.046  0.007 -0.010 
    30     31     32     33     34 
 0.007 -0.014  0.009  0.039 -0.041 
```

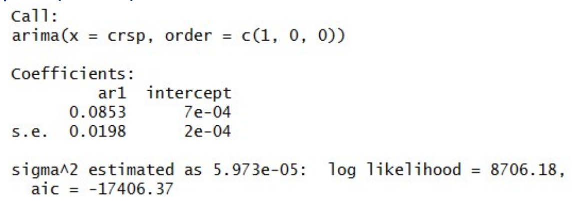
2. Next, fit AR(1) and AR(2) models to the CRSP returns:

```
fitAR1 = arima(crsp,order=c(1,0,0))
fitAR2 =  arima(crsp,order=c(2,0,0))
```

(a) Would you prefer an AR(1) or an AR(2) model for this time series? Explain your answer.
AR(1) model is a linear model that predicts the present value of a time series using the immediately prior value in time where AR(2) uses the prior two values in time. The series from the following question actually looks to be a better fit for MA(1) but for purposes of this assignment we fit two AR models. For final selection, we choose the model that has the lower value of the Akaike information criterion (AIC) or the Bayesian information criterion (BIC). The AR(1) model is preferred for either of these criteria.

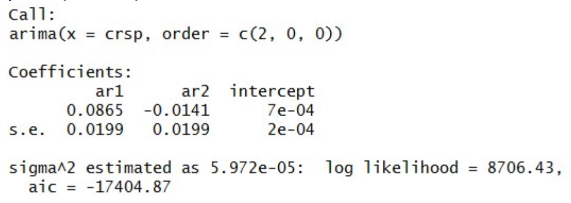
Executing the following code provides a summary of the fitting models and lists the AIC and BIC.

print(fitAR1)



```
Call:
arima(x = crsp, order = c(1, 0, 0))

Coefficients:
         ar1  intercept
      0.0853      7e-04
s.e.  0.0198      2e-04

sigma^2 estimated as 5.973e-05:  log likelihood = 8706.18,
  aic = -17406.37
```

print(fitAR2)



```
Call:
arima(x = crsp, order = c(2, 0, 0))

Coefficients:
         ar1      ar2  intercept
      0.0865  -0.0141      7e-04
s.e.  0.0199   0.0199      2e-04

sigma^2 estimated as 5.972e-05:  log likelihood = 8706.43,
  aic = -17404.87
```

options("scipen"=100, "digits"=4)

AIC(fitAR1)
#[1] -17406.37
AIC(fitAR2)
#1] -17404.87

BIC(fitAR1)
#[1] -17389
BIC(fitAR2)
#[1] -17382

(b) Find a 95% confidence interval for ϕ for the AR(1) model.

The confidence interval for ϕ for the AR(1) model is given by the following function which leveraged the fitted summary from the previous question:

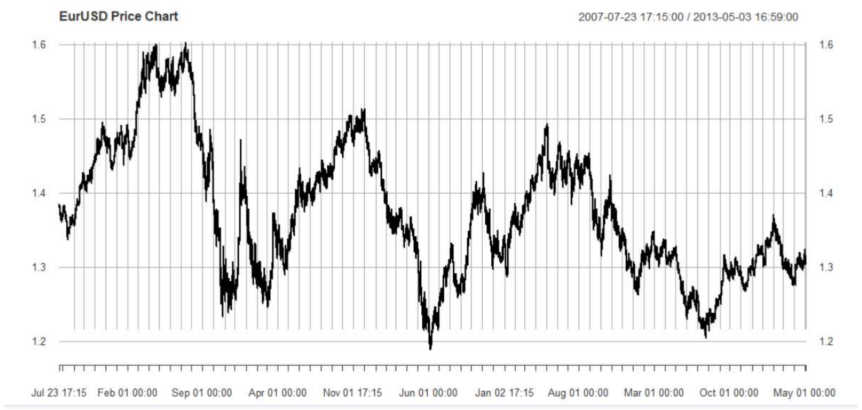
alpha = 0.05
0.0853 + 0.0198 * qnorm(1 - 0.5 * alpha) * c(-1,+1)
#[1] 0.04649271 0.12410729

PART B

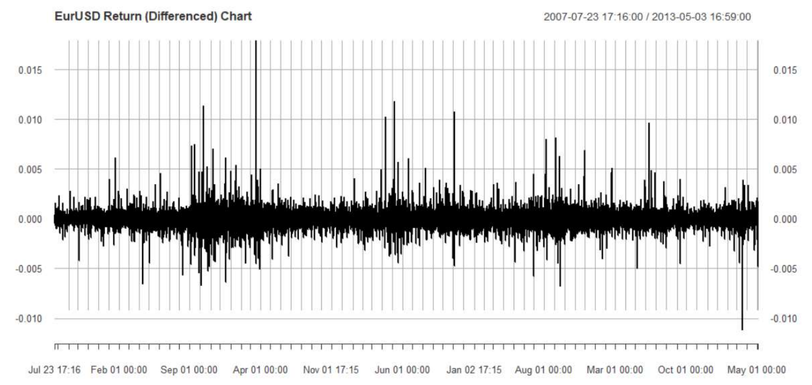
1.Selecting only the last column of the file, which represents the closing prices of 1-minute bars, calculate the 1-minute returns. What are their means and standard deviations? (5 pts)

The mean and standard deviation of the 1-minute price series is 1.373845 and 0.08908675, respectively. The mean and standard deviation of the calculated 1-minute return series is -0.000000004573 and 0.0001979, respectively. This is for the total data series, not the training data.

```
library(stats)
library(Ecdat)
library(tseries)
library(forecast)
library(PerformanceAnalytics)
library(dplyr)
library(lubridate)
library(xts)

raw<-read.csv('C:/Users/Scott/Desktop/451/Module5/EURUSD mid.csv', header = FALSE, sep=',')

mydata<-raw
colnames(mydata) <- c("date",
                "time",
                "Notneed1",
                "Notneed2",
                "Notneed3",
                "EurUSD")

mydata <- transform(mydata, date = as.Date(as.character(date), "%Y%m%d"))
mydata$time<-gsub('^([0-9]{1,2})([0-9]{2})$', '\\1:\\2', sprintf('%04d',mydata$time))
mydata$date<-as.POSIXct(paste(mydata$date, mydata$time), format="%Y-%m-%d %H:%M")
mydata<-as.data.frame(mydata)
mydata.t<-mydata
keep = c("date","EurUSD")
mydata.t<-mydata.t[keep]
mydata.t<-xts(mydata.t$EurUSD, as.POSIXct(mydata.t$date, format="%Y%m%d %H:%M"))
is.xts(mydata.t)
colnames(mydata.t) <- ("EurUSD")

par(mfrow=c(1,1))
plot(mydata.t,main= "EurUSD Price Chart")
```



```
returns = Return.calculate(mydata.t, method="simple")
returns = as.xts(returns)
returns=na.omit(returns)
plot(returns, main = "EurUSD Return (Differenced) Chart")
```

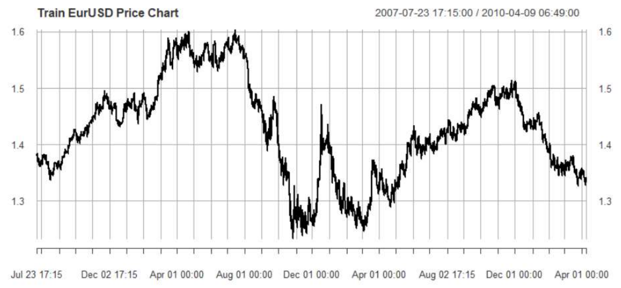


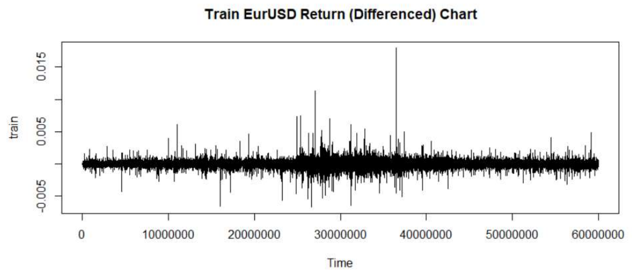
```
#Mean and Standard Deviation of PRICE
avg=mean(mydata.t)
avg

stddev=sd(mydata.t)
stddev

#Mean and Standard Deviation of RETURNS
m = mean(returns)
m

s=sd(returns)
s
```

2.)Find the best AR(p) model m0 for the returns series using the BIC criteria, using only the first one million bars (the training set): please tell us what the optimal p is. (5 pts)

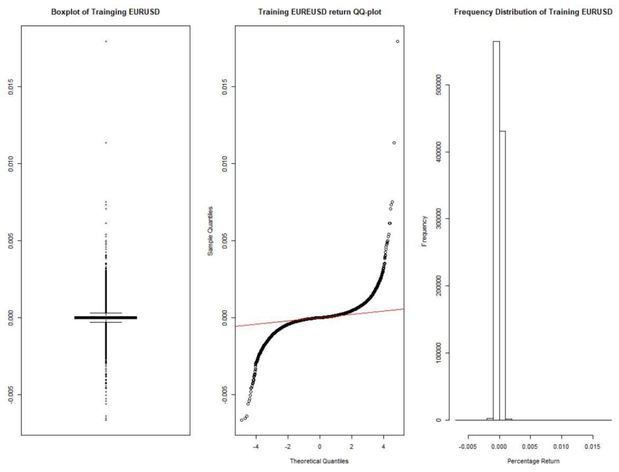
To summarize, the best AR(p) model m0 for the return series using the BIC criteria on the training set is AR(4). Here is how I reached this conclusion:

The training price time series appears to have strong persistence, neighboring observations are similar and short upward and downward trends appear at random. This is indicative of a random walk series. The training price time series possibly has a negative drift coefficient given the slight downward trend. The difference transformation of the training price data is a white noise series and appears non-periodic and oscillates around the common mean of 0, suggesting stationarity.

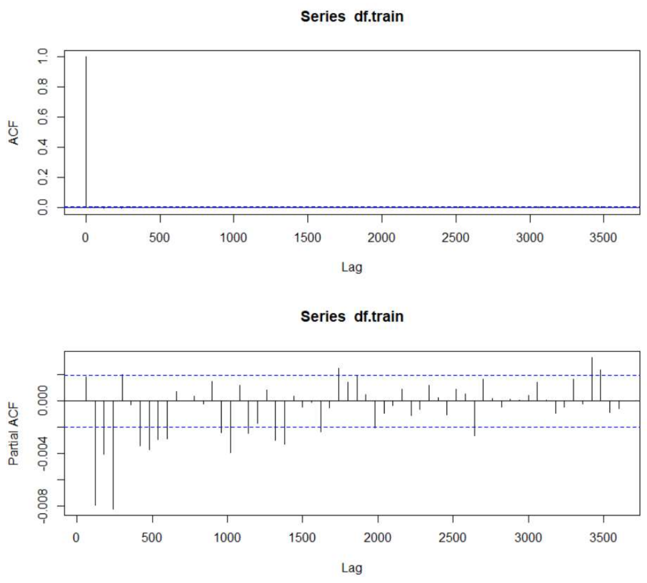




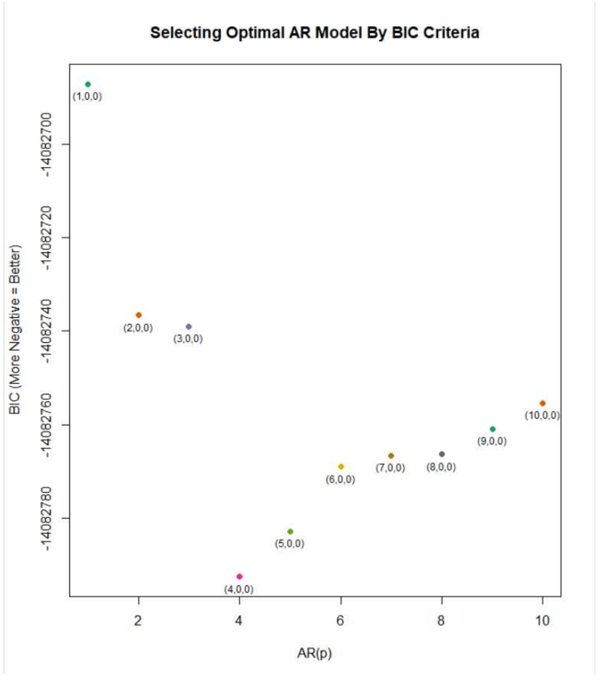
Looking at plots of the EurUSD training distribution, note that the majority of returns are near zero, but some daily returns are greater than 0.0% percentage points in magnitude. Similarly, note the clear departure from normality, especially in the tails of the distributions, as evident in the normal quantile plots.



Plotting the estimated ACF and PACF of the training data shows large negative correlations for several lags and then is zero at most other lags. It appears this series may not be linearly related to it’s past. The PACF chart suggests AR(4) might be the optimal model. This is tested in the next graph which displays the BIC criteria metric for lags 1 through 10.



The below visual of the first 10 lags confirms that AR(4) is the optimal model by BIC.



```
#Partition data sets
returns<-as.ts(returns)
train = subset(returns, end = 1000000)
test = subset(returns,start=1000001, end = 1001000)
```

```r
#Data exploration
par(mfrow=c(2,1))
#Draw line chart for training set
plot(mydata.t[1:1000000], main = "Train EurUSD Price Chart")
#Draw line chart for training set
plot(train, main = "Train EurUSD Return (Differenced) Chart")
par(mfrow=c(1,1))

par(mfrow=c(1,3))
# Draw box and whisker plot
boxplot(coredata(train), main = "Boxplot of Trainging EURUSD")
# Draw q-q plot and add a red line for normality
qqnorm(train,
      main = "Training EUREUSD return QQ-plot")
qqline(train,
      col = "red")
#Histogram
lapply(train, MARGIN = 2, FUN = hist, xlab = "Percentage Return", main = "Frequency Distribution of Training
EURUSD")
par(mfrow=c(1,1))

# Draw autocorrelogram
par(mfrow=c(2,1))
acf(df.train)
pacf(df.train)
par(mfrow=c(1,1))


#bic criterion
m100<-Arima(train, order=c(1,0,0))
m200<-Arima(train, order=c(2,0,0))
m300<-Arima(train, order=c(3,0,0))
m400<-Arima(train, order=c(4,0,0))
m500<-Arima(train, order=c(5,0,0))
m600<-Arima(train, order=c(6,0,0))
m700<-Arima(train, order=c(7,0,0))
m800<-Arima(train, order=c(8,0,0))
m900<-Arima(train, order=c(9,0,0))
m1000<-Arima(train, order=c(10,0,0))

a=m100$bic
b=m200$bic
c=m300$bic
d=m400$bic
e=m500$bic
f=m600$bic
g=m700$bic
h=m800$bic
i=m900$bic
j=m1000$bic

library(RColorBrewer)
colors <- brewer.pal(n = 8, name = "Dark2")
```

```
total<-rbind(a,b,c,d,e,f,g ,h ,i ,j)
rownames(total)<-c("(1,0,0)","(2,0,0)","(3,0,0)","(4,0,0)","(5,0,0)","(6,0,0)","(7,0,0)","(8,0,0)","(9,0,0)","(10,0,0)")
colnames(total)<-"bic"
total

plot(total,xlab = 'AR(p)', ylab = 'BIC (More Negative = Better)'
    , col = colors, pch = 16
    , main = "Selecting Optimal AR Model By BIC Criteria"
    , cex.lab=1.0)
text(total, row.names(total), cex=0.75, pos=1, col="black")
```

3) Using the AR model chosen in 2), make a 1-step forecast of the EURUSD return in the next minute from bar 1,000,001 to 1,001,000. What percentage of times this forecast correctly predicts the sign of the return of the next minute (from the 1,000,001th to the 1,001,000th bar)? (The sign has values -1, 0, +1). (Hint: There is a way to "fix" a model so that you don't have to re-fit a model for every new bar and instead use a fixed set of parameters determined in the first million bars for predicting the values in the next one thousand bars. Use the Arima, not arima, function for this problem, and specify model=m0, then you won't need to use a loop.) (20 pts)

The accuracy for bars 1,000,001 to 1,001,000 using m0 (i.e. AR(4)) is 44.1%.

```
#Fit Model to training data
m0<-Arima(train, order=c(4,0,0))

#Apply fit model to later data
m1 <- Arima(test,model=m0, include.constant = TRUE)#Include.contact = TRUE For differenced timeseries
m1forecast<-forecast(m1,h=1000)

#Tally up Correct signs
yhat<-m1forecast$fitted
yhat.sign<-sign(m1forecast$fitted)
y<-test
y.sign<-sign(test)

mytable<-cbind(yhat,yhat.sign,y,y.sign, yhat.sign - y.sign)
mytable=as.data.frame(mytable)
# View(mytable)

num=sum(mytable[5] == '0')
num
den=nrow(mytable[5])
den

directional_accuracy=num/den
print(directional_accuracy)
```

4) (Optional): You can attempt to backtest a trading strategy based on this AR model and compute the cumulative return of such a strategy. Sometimes, we don't need a high percentage of correct guesses of the sign to be profitable.

The back tested model has an accuracy of 44.28% and a cumulative return of -0.01623 versus -0.05029 for the actual time series. So my model is better! Losing less over time does help preserve capital, as this model points out.

```
#Fit Model to All Data
m2 <- Arima(returns,model=m0, include.constant = TRUE)
m2forecast<-forecast(m2,h=2136591)

yhat2<-m2forecast$fitted
yhat2.sign<-sign(m2forecast$fitted)
y2<-returns
y2.sign<-sign(returns)

mytable2<-cbind(yhat2,yhat2.sign,y2,y2.sign, yhat2.sign - y2.sign)
mytable2=as.data.frame(mytable2)

num2=sum(mytable2[5] == '0')
num2
den2=nrow(mytable2[5])
den2

directional_accuracy2=num2/den2
print(directional_accuracy2)

Return.cumulative.yhat = cumprod(1+yhat2) - 1
Return.cumulative.y = cumprod(1+y2) - 1

Return.cumulative.yhat[2136591]
Return.cumulative.y[2136591]
```

REFERENCES

Ruppert, D., & Matteson, D. S. (2015). Statistics and data analysis for financial engineering: With R examples. New York: Springer.

Dancho, Matt. “Tidy Time Series Analysis, Part 4: Lags and Autocorrelation.” Business Science, 30 Aug. 2017, www.business-science.io/timeseries-analysis/2017/08/30/tidy-timeseries-analysis-pt-4.html.