Hw 3

Xiang Yang Ng

May 5, 2018

Set the current directory, clear the environment and download the libraries

```
setwd("C:/Users/Xiang/OneDrive/Desktop/Econ-144/Homework/Hw 3")

rm(list=ls(all=TRUE))

library("quantmod")
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

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

```
## Loading required package: TTR
```

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
library(lattice)
library(foreign)
library(MASS)
library(car)
```

```
## Loading required package: carData
```

```
require(stats)
require(stats4)
```

```
## Loading required package: stats4
```

```
library(KernSmooth)
```

```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```
library(fastICA)
library(cluster)
library(leaps)
library(mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-23. For overview type 'help("mgcv-package")'.
```

```
library(rpart)
library(pan)
library(mgcv)
library(DAAG)
```

```
##
## Attaching package: 'DAAG'
```

```
## The following object is masked from 'package:car':
##
##     vif
```

```
## The following object is masked from 'package:MASS':
##
##     hills
```

```
library("TTR")
library(tis)
```

```
##
## Attaching package: 'tis'
```

```
## The following object is masked from 'package:mgcv':
##
##     ti
```

```
## The following object is masked from 'package:quantmod':
##
##     Lag
```

```
## The following object is masked from 'package:TTR':
##
##     lags
```

```
require("datasets")
require(graphics)
library("forecast")
```

```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:tis':
##
##     easter
```

```
## The following object is masked from 'package:nlme':
##
##     getResponse
```

```
#install.packages("astsa")
#require(astsa)
library(xtable)
# New libraries added:
library(stats)
library(TSA)
```

```
## Loading required package: locfit
```

```
## locfit 1.5-9.1    2013-03-22
```

```
## Loading required package: tseries
```

```
##
## Attaching package: 'TSA'
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

```
## The following object is masked from 'package:utils':
##
##     tar
```

```
library(timeSeries)
```

```
## Loading required package: timeDate
```

```
##
## Attaching package: 'timeDate'
```

```
## The following objects are masked from 'package:TSA':
##
##     kurtosis, skewness
```

```
## The following object is masked from 'package:xtable':
##
##     align
```

```
## The following objects are masked from 'package:tis':
##
##     dayOfWeek, dayOfYear, isHoliday
```

```
##
## Attaching package: 'timeSeries'
```

```
## The following objects are masked from 'package:tis':
##
##     description, interpNA
```

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

```
library(fUnitRoots)
```

```
## Loading required package: fBasics
```

```
##
## Attaching package: 'fBasics'
```

```
## The following object is masked from 'package:car':
##
##     densityPlot
```

```
## The following object is masked from 'package:TTR':
##
##     volatility
```

```
library(fBasics)
library(tseries)
library(timsac)
library(TTR)
library(fpp)
```

```
## Loading required package: fma
```

```
##
## Attaching package: 'fma'
```

```
## The following objects are masked from 'package:DAAG':
##
##     milk, ozone
```

```
## The following objects are masked from 'package:MASS':
##
##     cement, housing, petrol
```

```
## Loading required package: expsmooth
```

```
## Loading required package: lmtest
```

Problem 1

Clear the environment

```
rm(list = ls(all=TRUE))
```

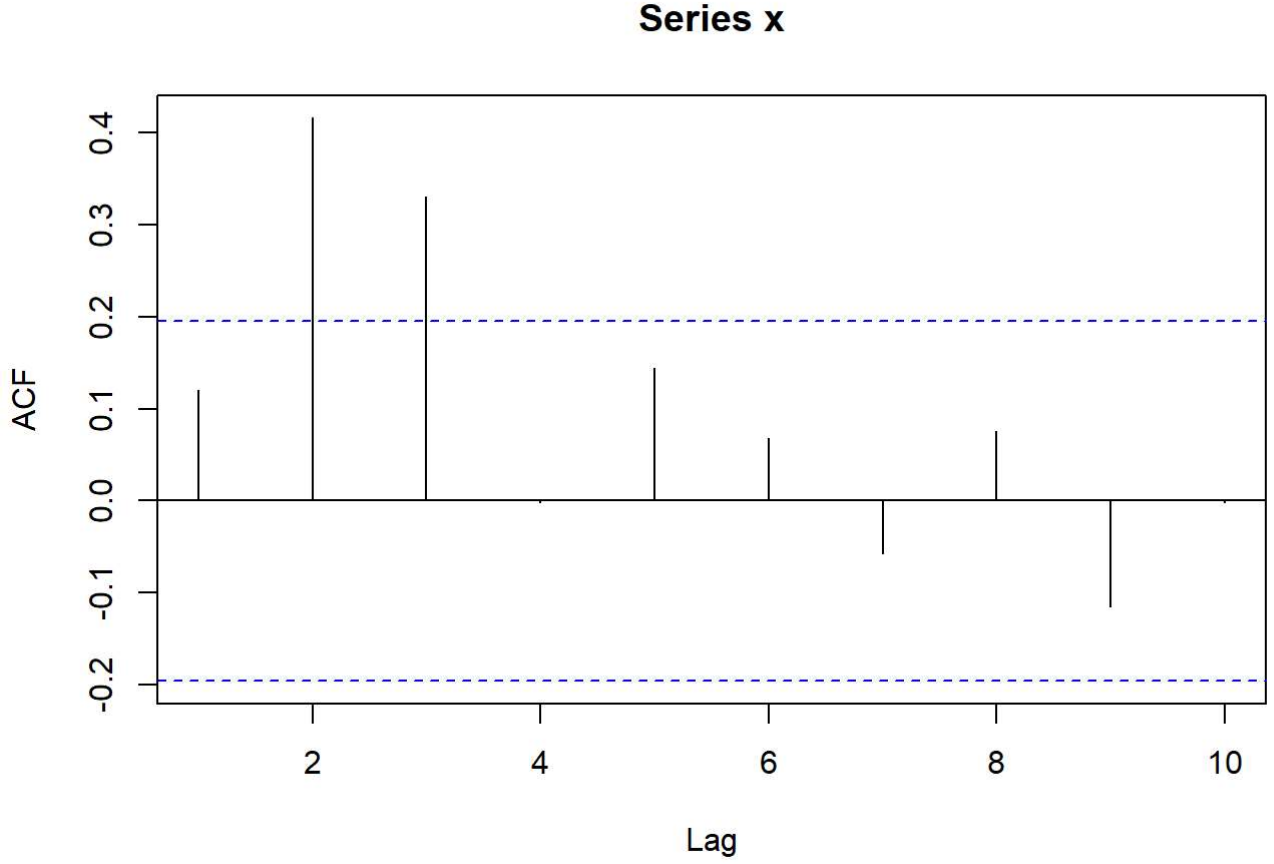
a. Obtaining the theoretical ACF

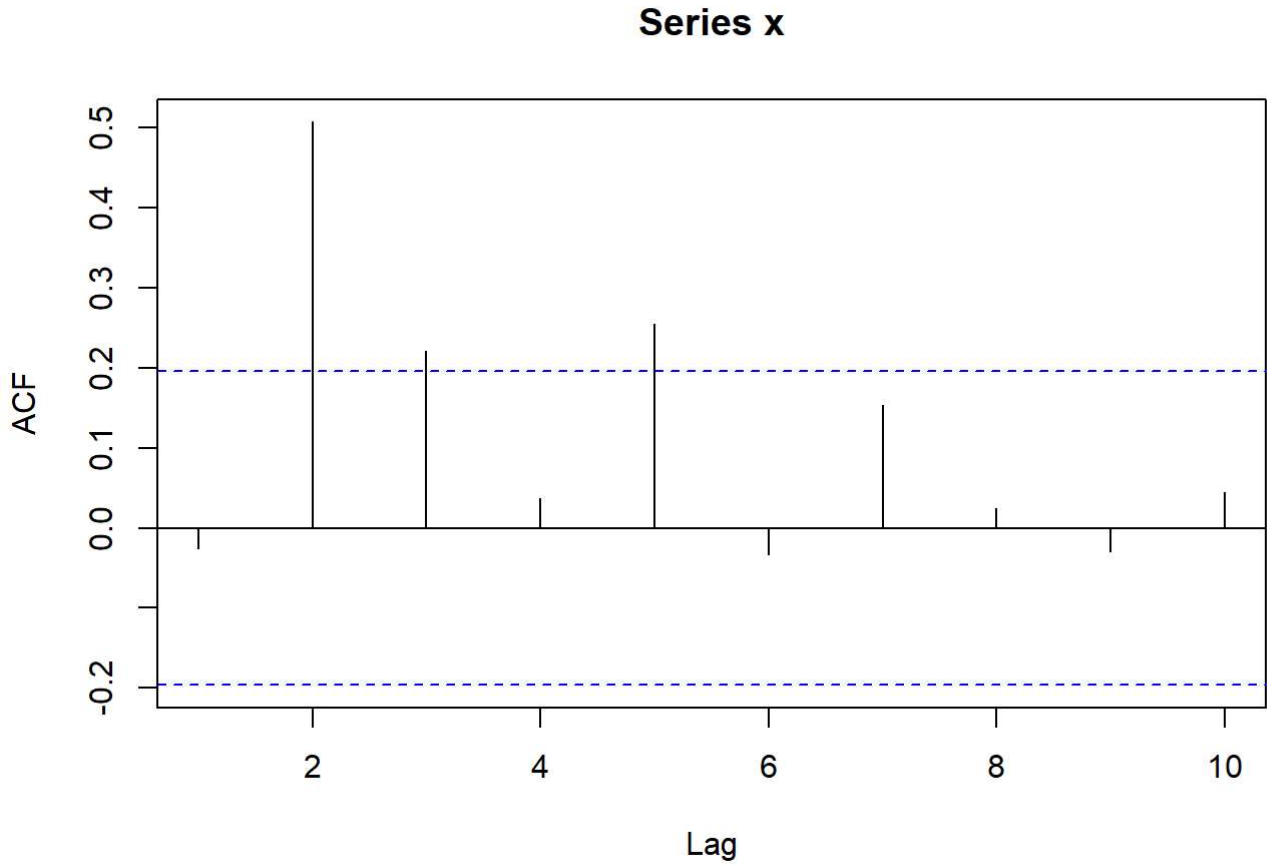
```
ARMAacf(ma = c(-2, 1.35), lag.max = 10)
```

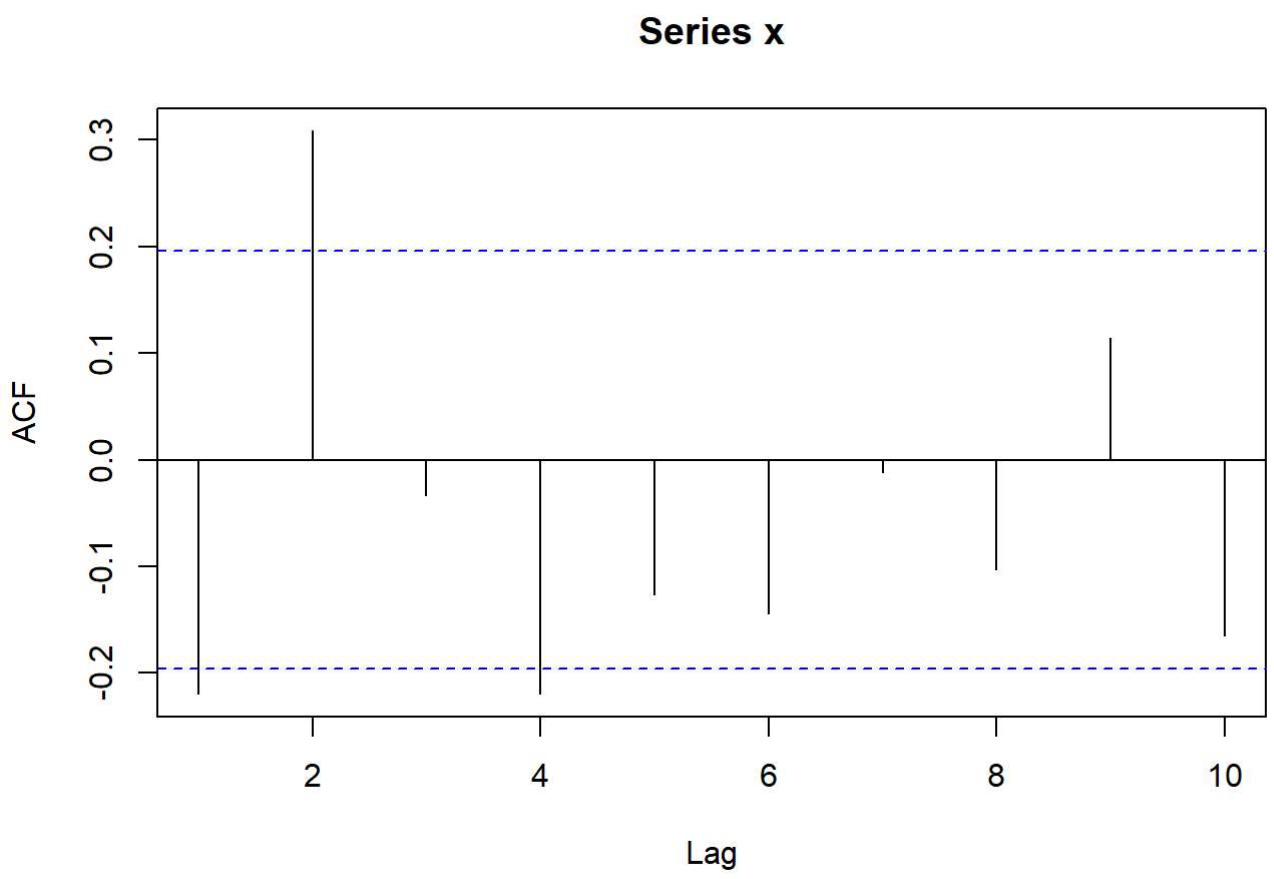
```
##          0          1          2          3          4          5
##  1.0000000 -0.6888970  0.1978747  0.0000000  0.0000000  0.0000000
##          6          7          8          9         10
##  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
```

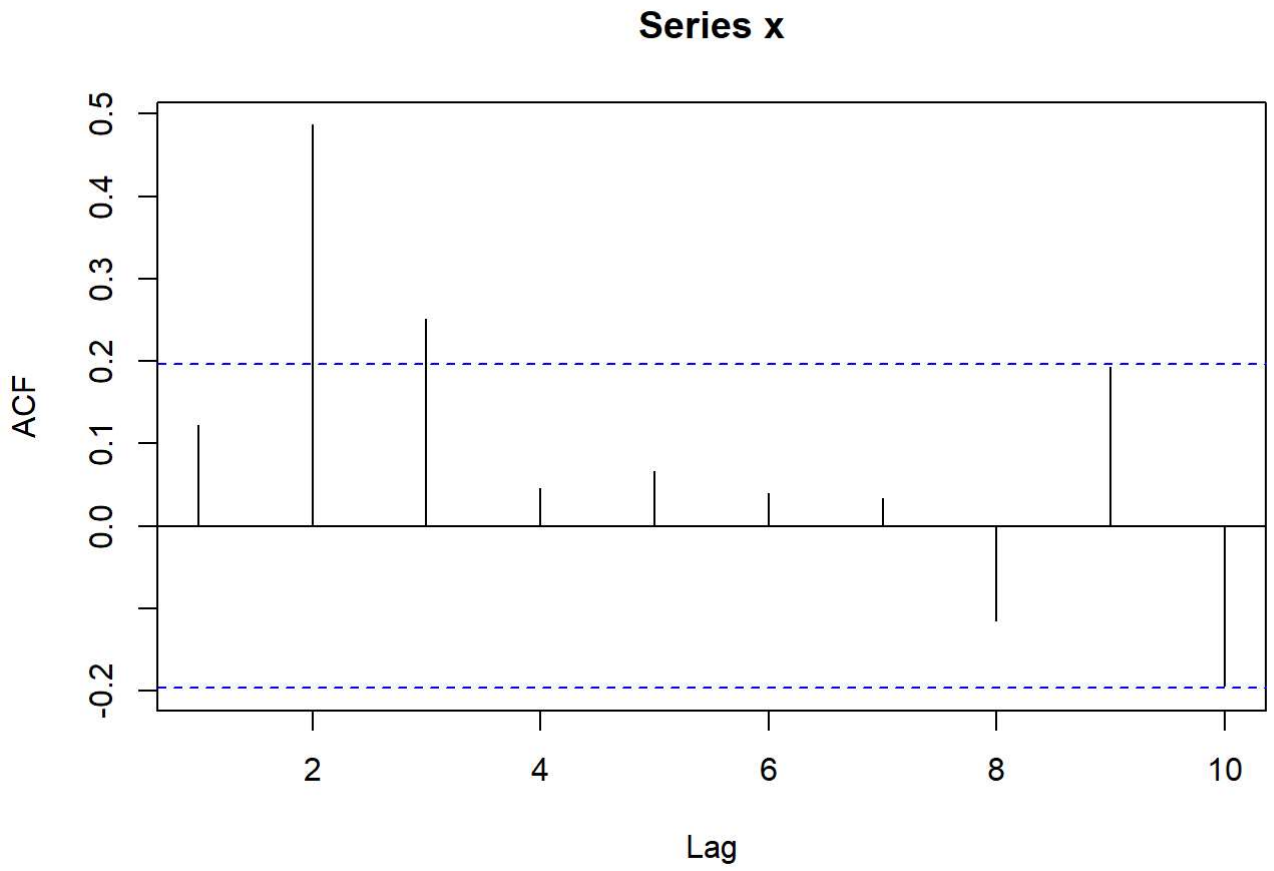
b. Obtaining the simulated MA(2) model with 100 points and it's ACF. Running this simulation 10 times

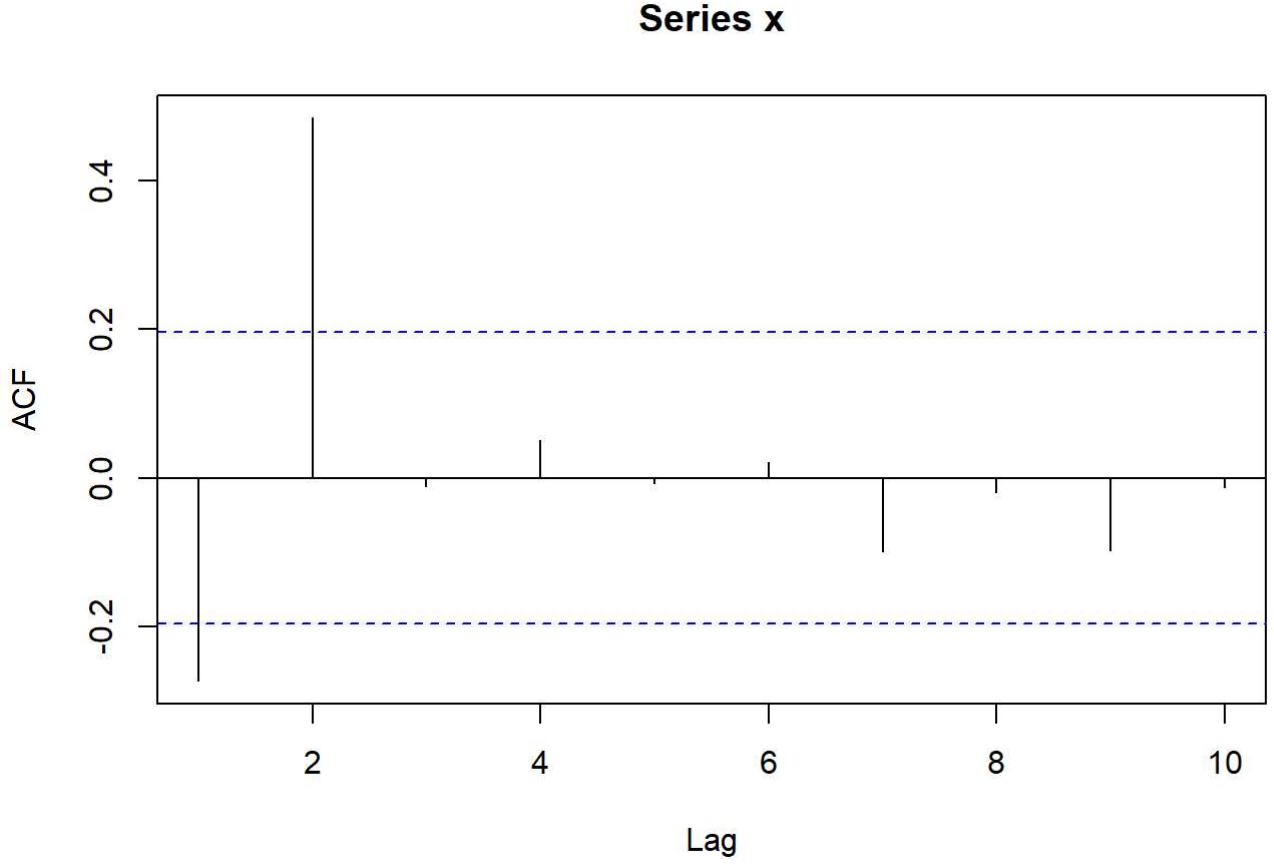
```
for (i in 1:10){
  x = e = rnorm(100)
  x[2] = 0.7 -0.2*e[1] + e[2]
  for (t in 3:100) x[t] = 0.7-0.2*e[t-1]+1.35*e[t-2] + e[t]
  acf(x, lag.max = 10)
}
```

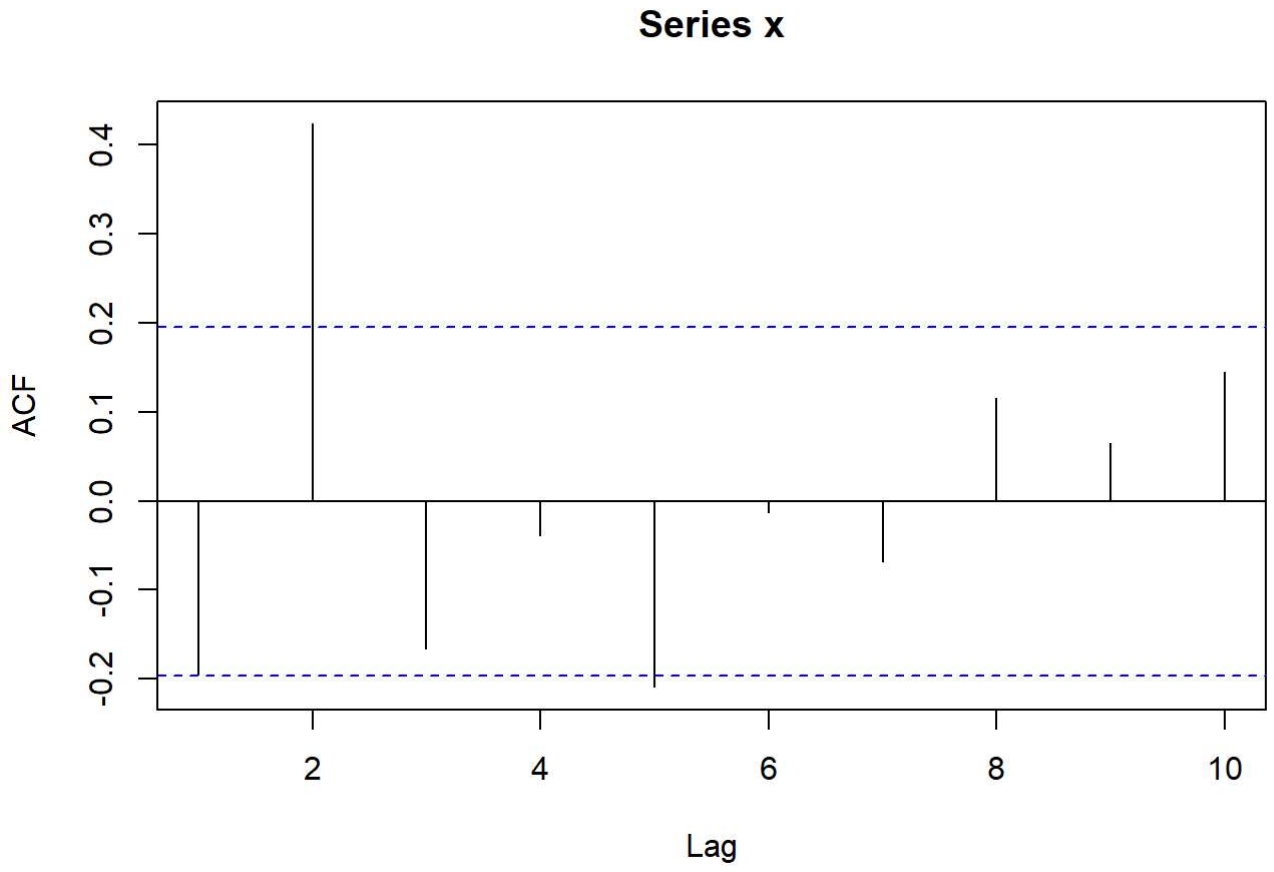

Series x
ACF
0.4
0.3
0.2
0.1
0.0
-0.1
-0.2
2
4
6
8
10
Lag

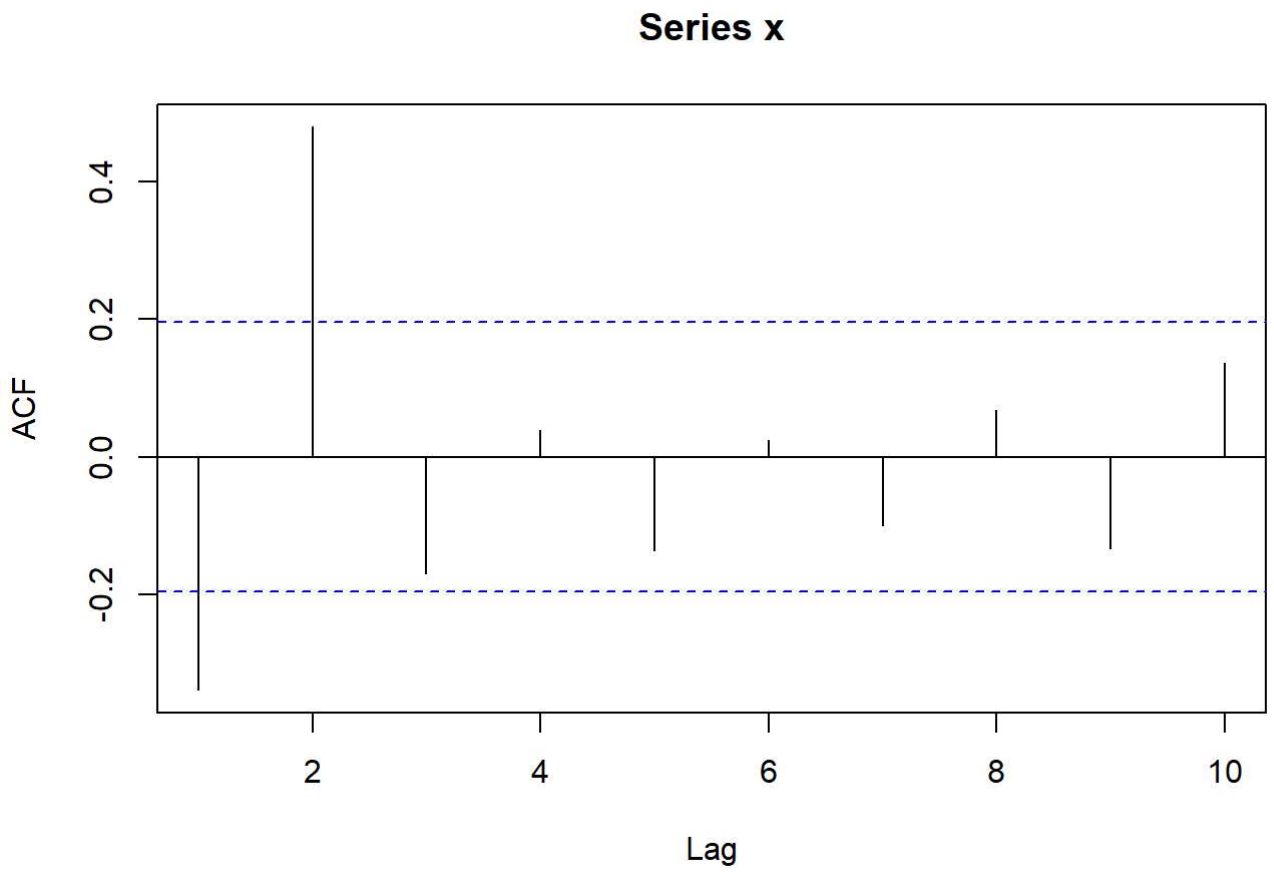

Series x
ACF
0.5
0.4
0.3
0.2
0.1
0.0
-0.2
2
4
6
8
10
Lag

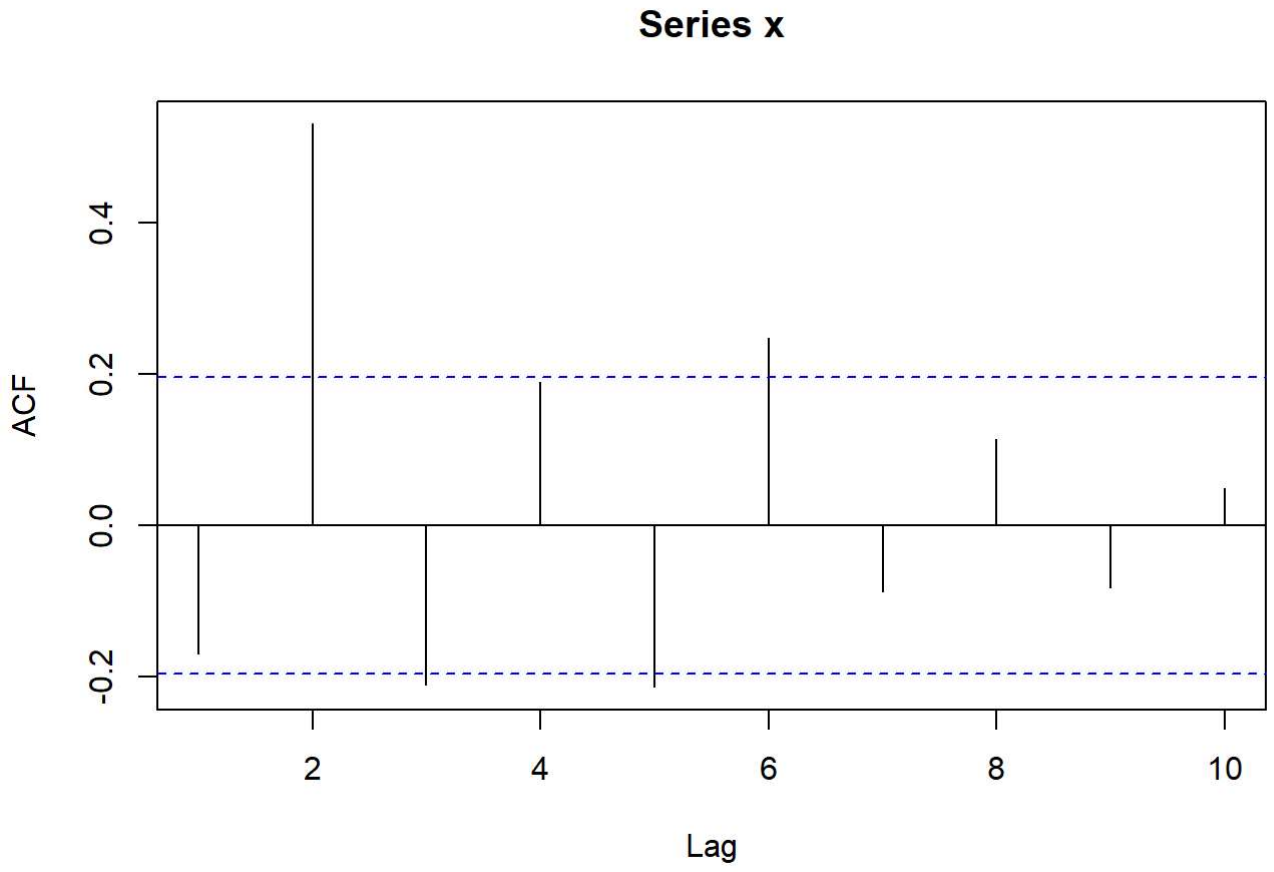

Series x
ACF
0.3
0.2
0.1
0.0
-0.1
-0.2
2
4
6
8
10
Lag

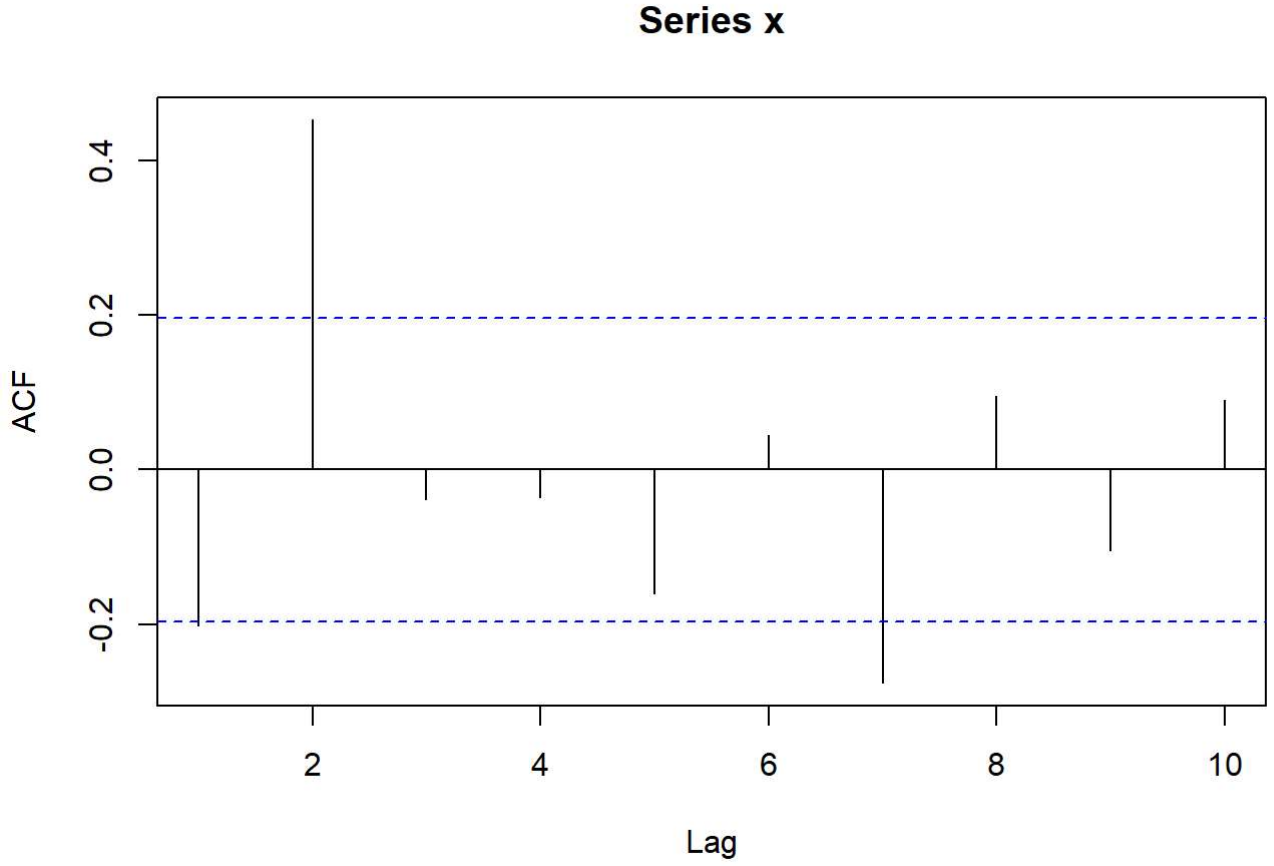

Series x
ACF
0.5
0.4
0.3
0.2
0.1
0.0
-0.2
2
4
6
8
10
Lag

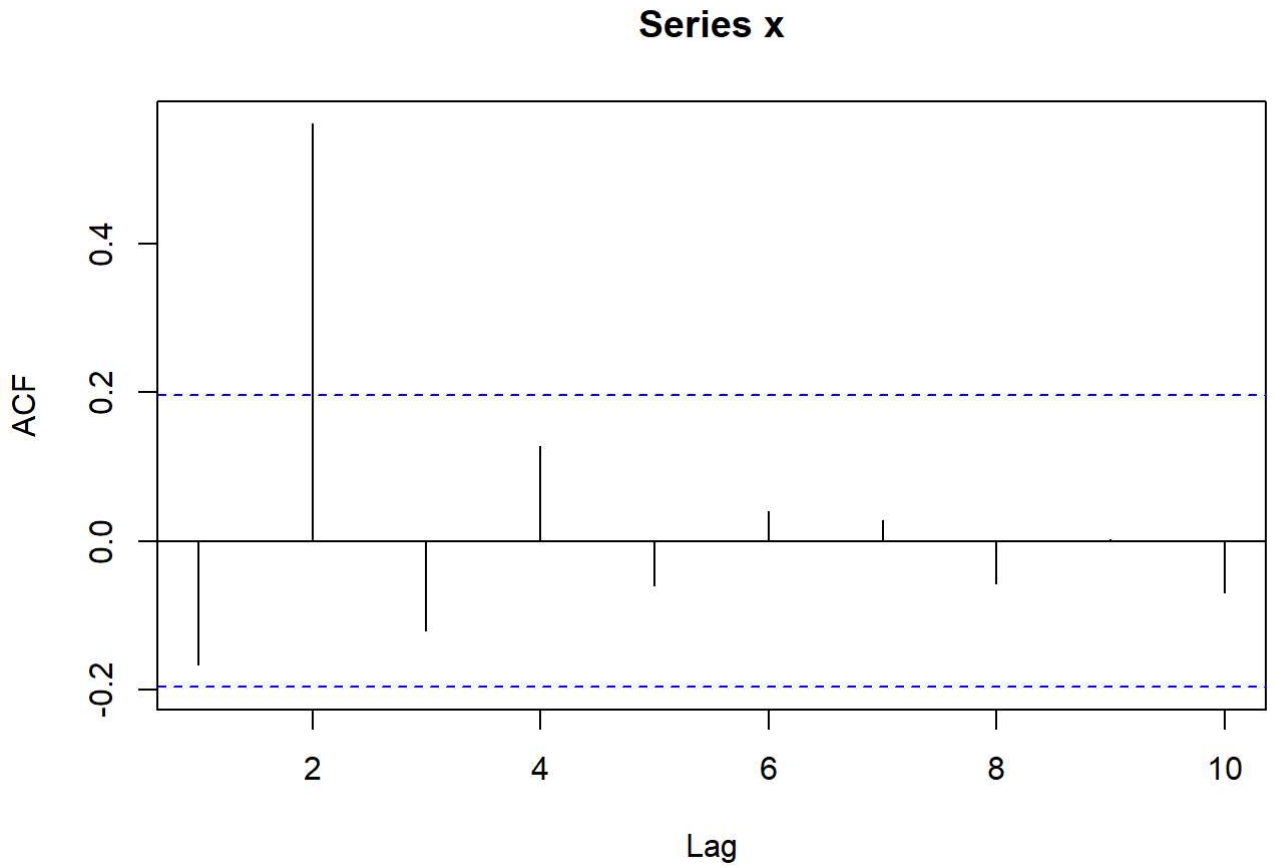

Series x
ACF
0.4
0.2
0.0
-0.2
2
4
6
8
10
Lag


Series x
ACF
0.4
0.3
0.2
0.1
0.0
-0.1
-0.2
2
4
6
8
10
Lag


Series x
ACF
0.4
0.2
0.0
-0.2
2
4
6
8
10
Lag


Series x
ACF
0.4
0.2
0.0
-0.2
2
4
6
8
10
Lag


Series x
ACF
0.4
0.2
0.0
-0.2
2
4
6
8
10
Lag


Series x
ACF
0.4
0.2
0.0
-0.2
2
4
6
8
10
Lag

The theoretical ACF gives a very different picture compared to the ACF of simulated MA(2). The theoretical ACF suggest that there will be 2 spikes at lag values 1 and 2 and no spikes after lag 2, while the simulated MA(2) produces an ACF that suggests at least an MA(1) process with possible seasonal AR. This means that the small sample is unable to approximate the the population estimate.

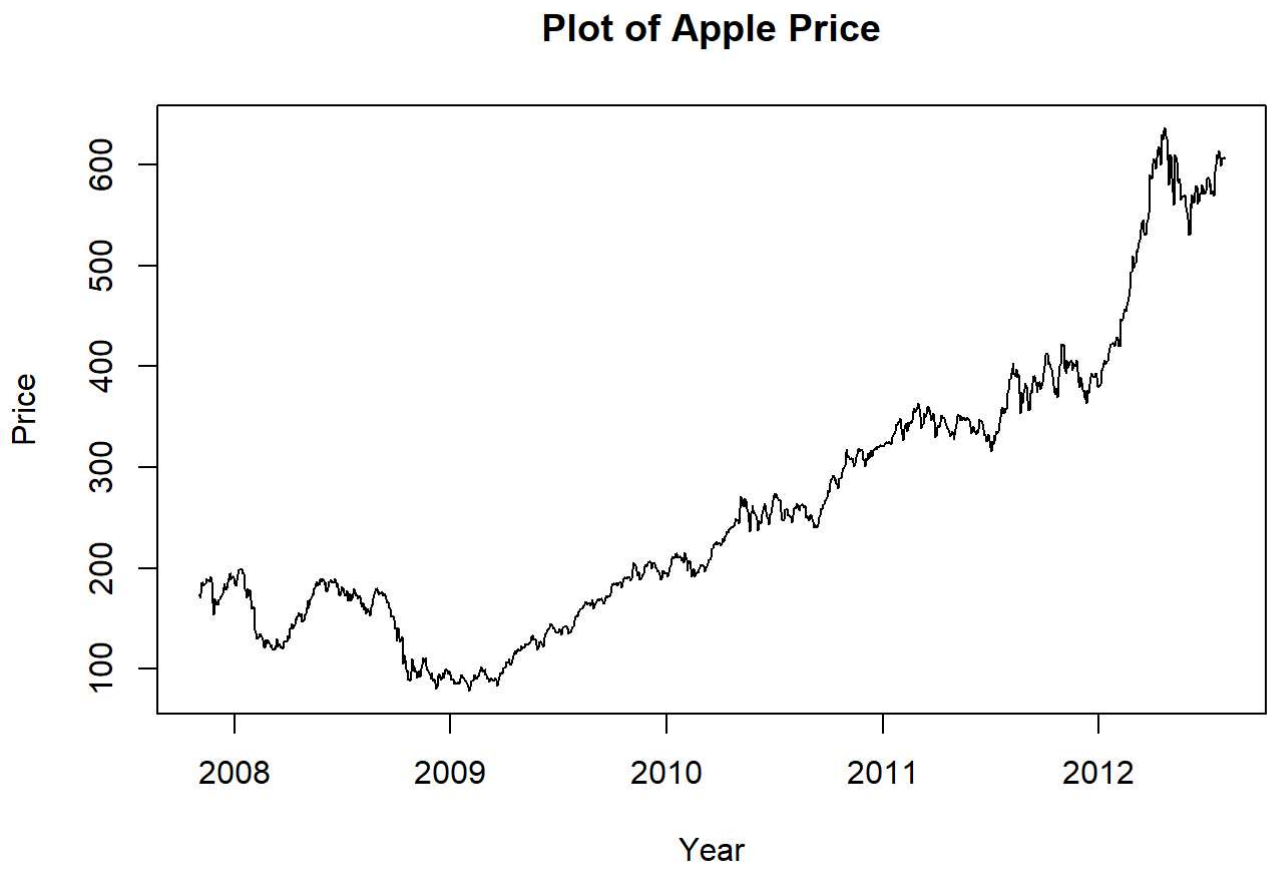
Problem 2

Clear the previous data and obtain the data

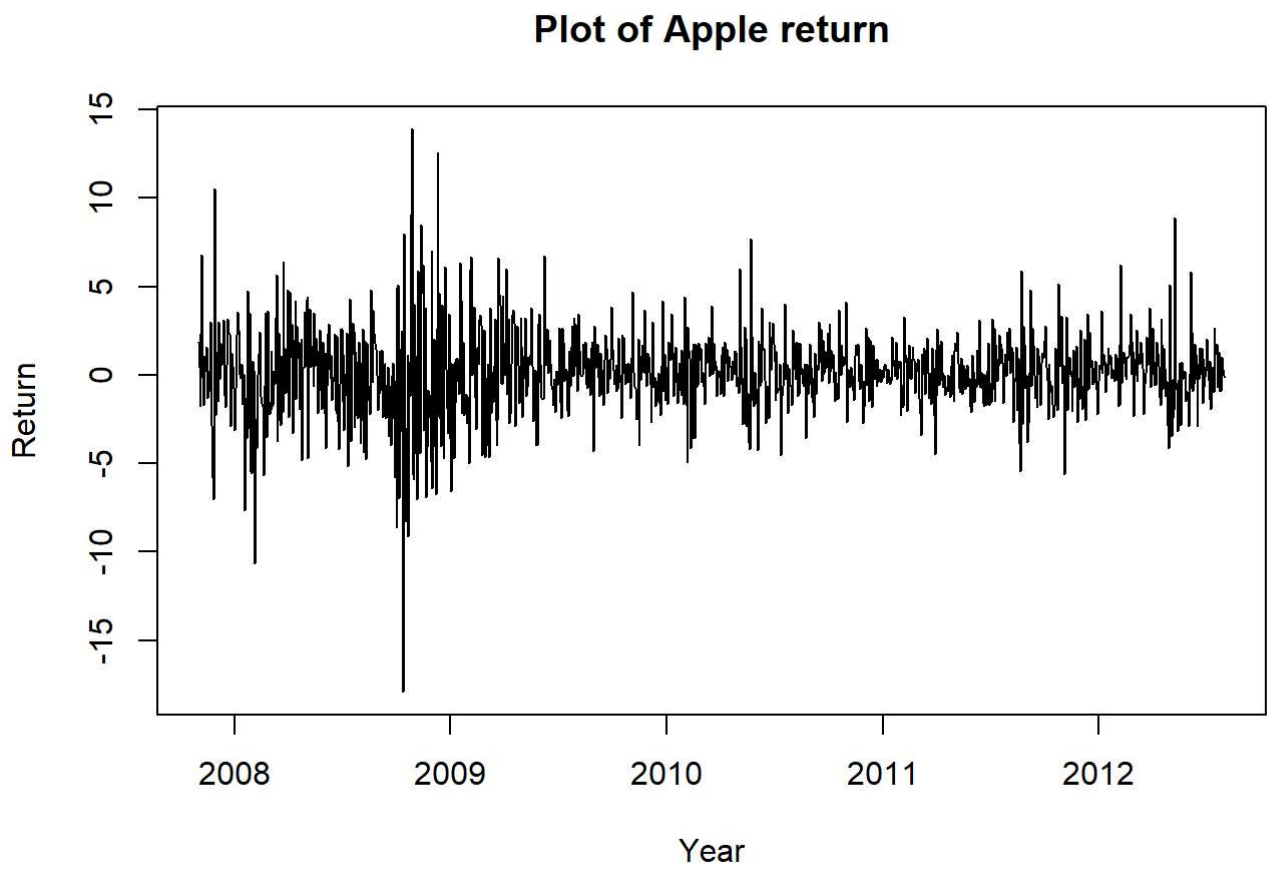
```
rm(list = ls(all = TRUE))
data = read.csv("6.10.csv")
names(data) = c("date", "ApplPrice", "return")
attach(data)
```

First look at the plots for both the price and the returns, as well as their ACF and PACF

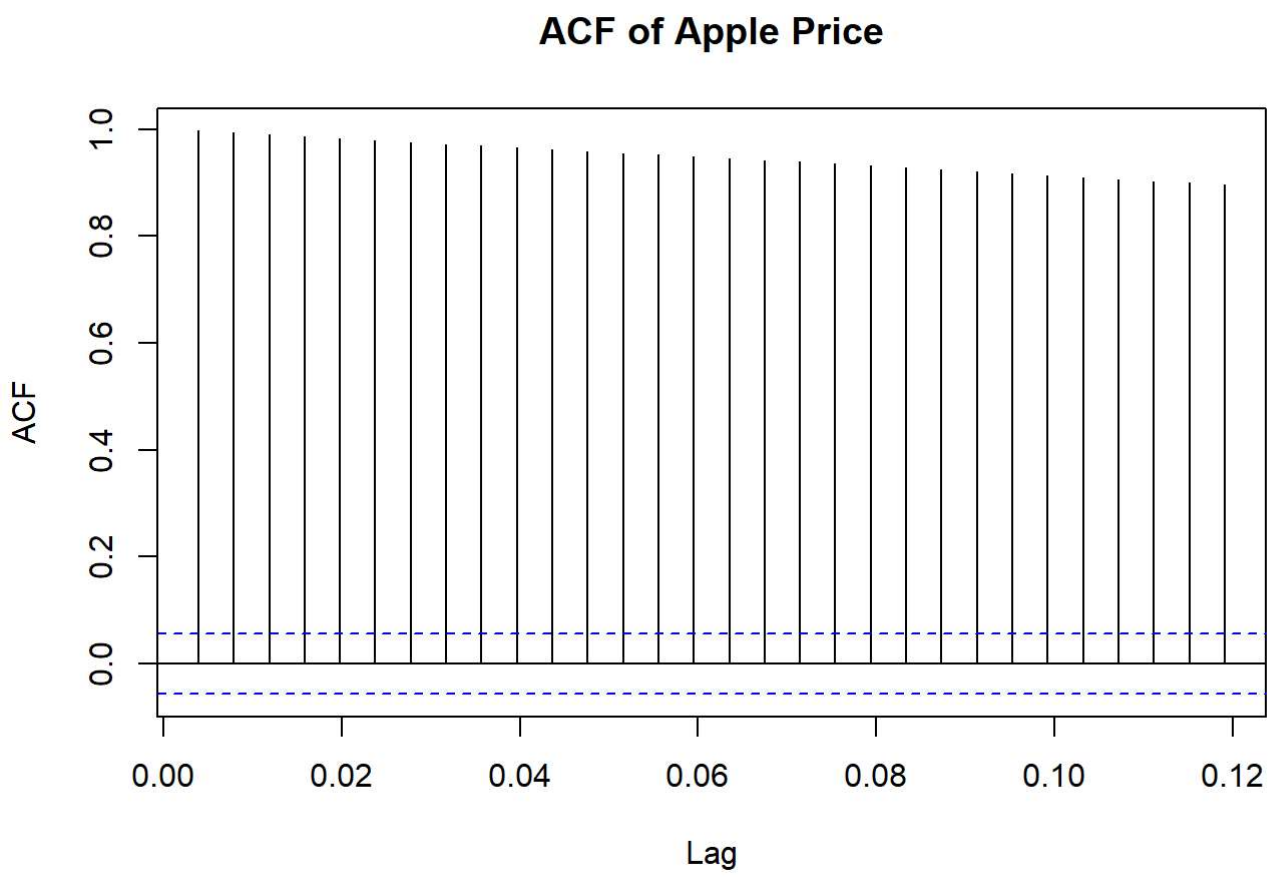
```
ApplPrice_ts = ts(ApplPrice, start = 2007+(10/12), end = 2012 + (7/12), freq = 252)
return_ts = ts(return, start = 2007+(10/12), end = 2012 + (7/12), freq = 252)
plot(ApplPrice_ts, type = "l", main = "Plot of Apple Price", xlab = "Year", ylab = "Price")
```



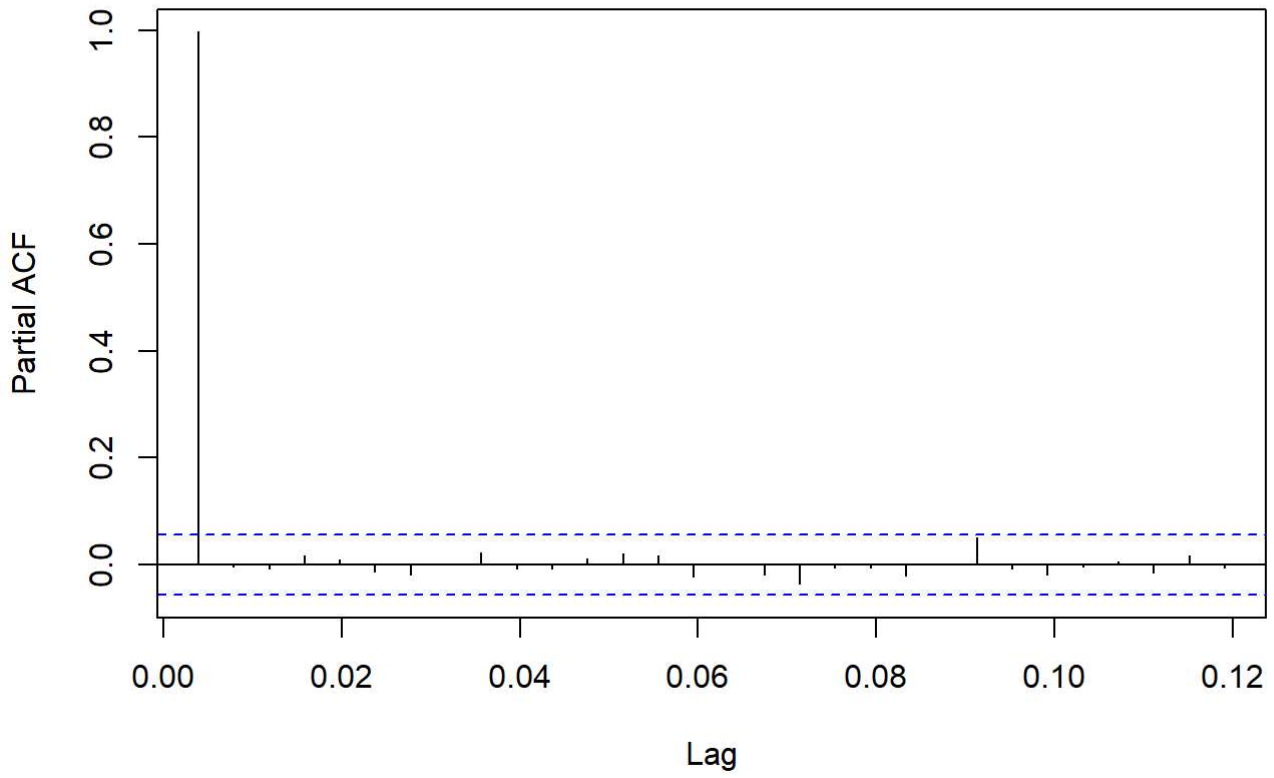

```
plot(return_ts, type = "l", main = "Plot of Apple return", xlab = "Year", ylab = "Return")
```



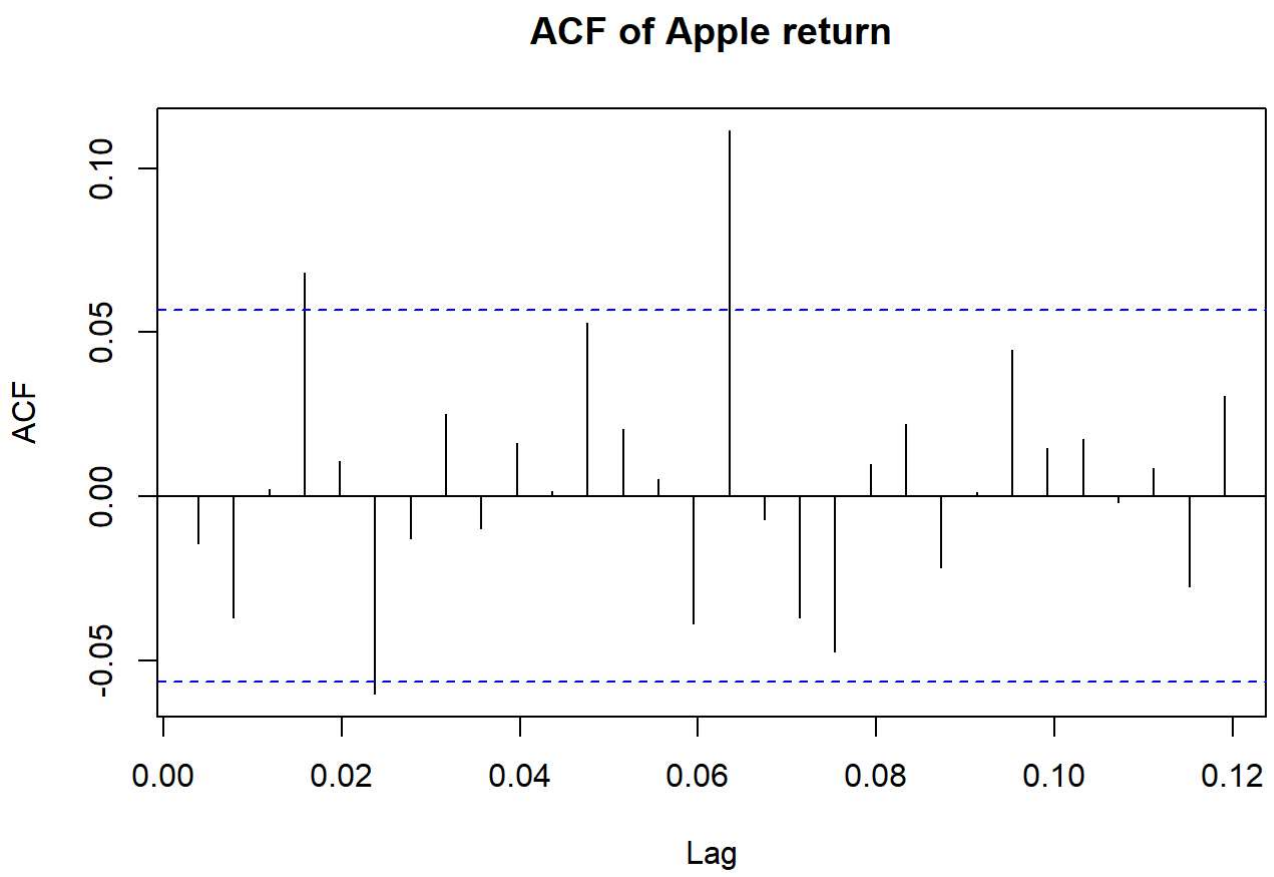

```
acf(ApplPrice_ts, main = "ACF of Apple Price", type = "correlation")
```



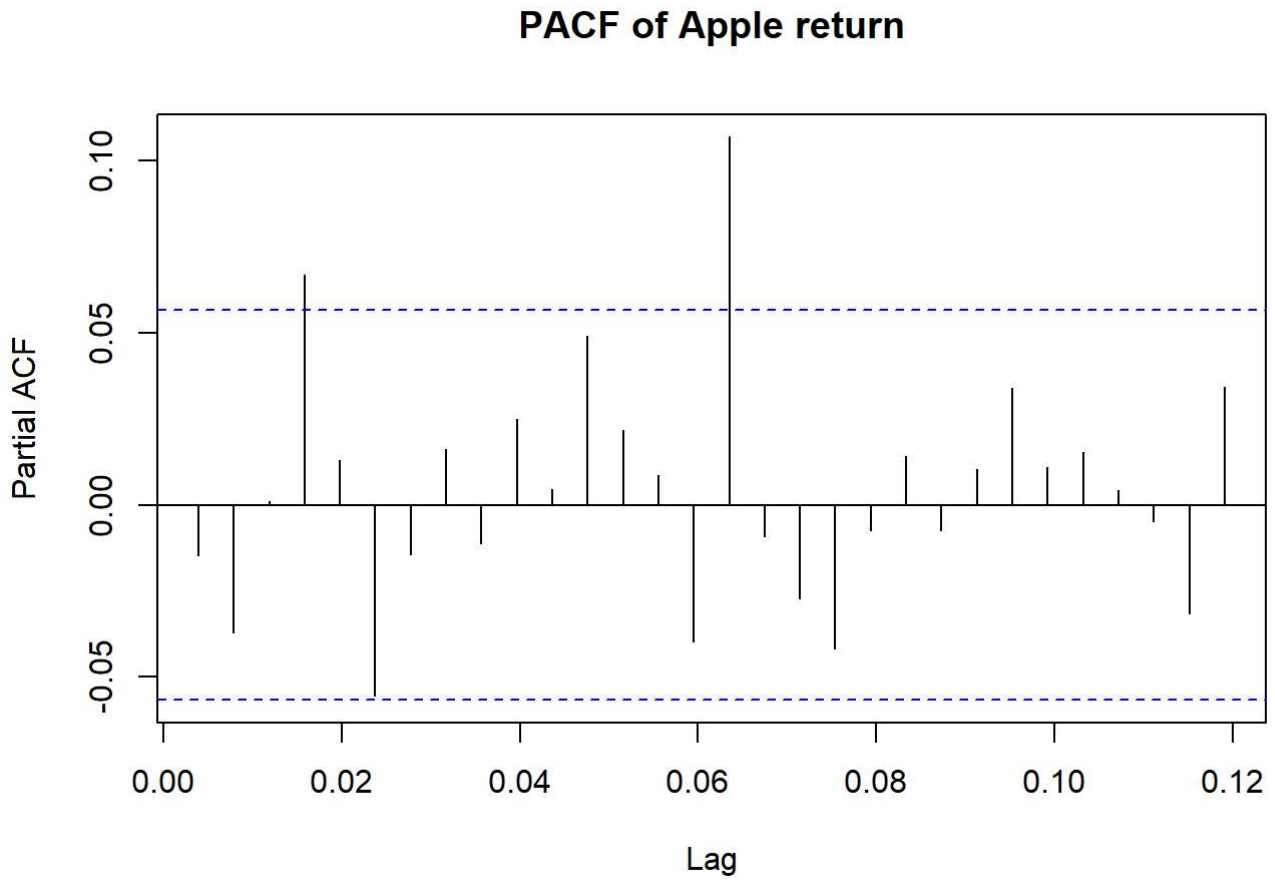

```
pacf(ApplPrice_ts, main = "pACF of Apple Price")
```




```
acf(return_ts, main = "ACF of Apple return")
```

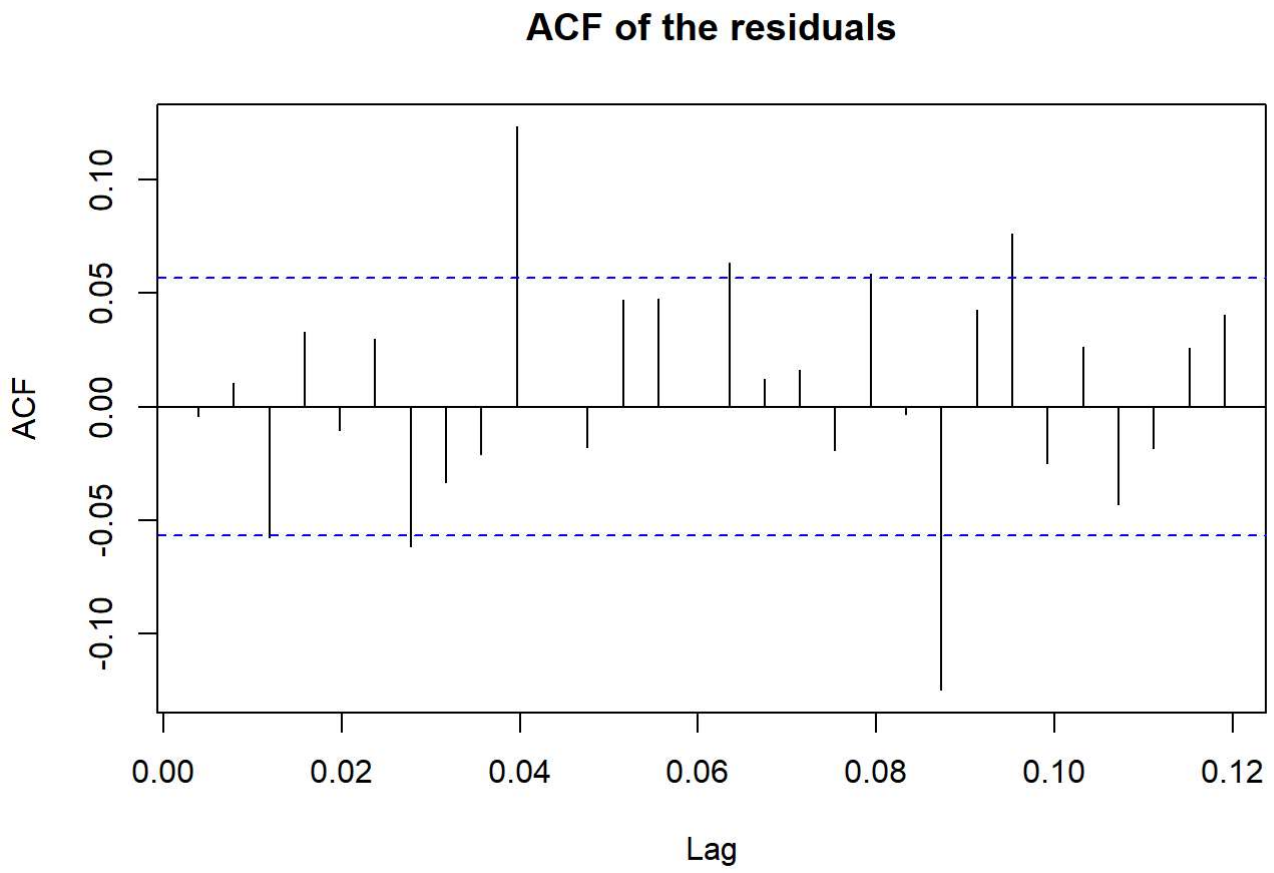



```
pacf(return_ts, main = "PACF of Apple return")
```

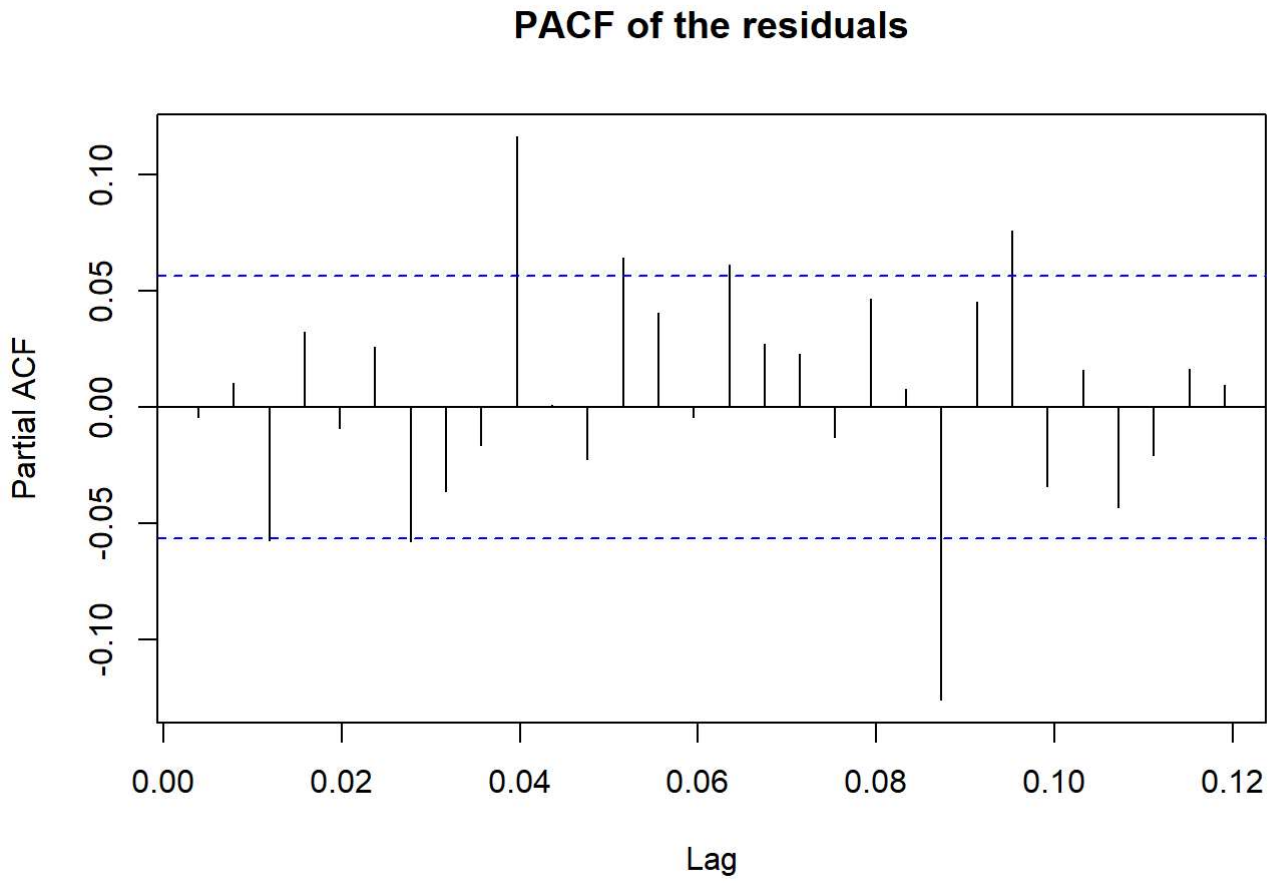


Looking at the plot of the price, there is an upward trend, while the ACF shows a slow decay and the PACF shows one spike. This means we should take the first difference of the price and then fit an AR(1) model to the data. Then we take the ACF and PACF of the residuals of the AR(1) model and also do the Box-Pierce test on it to test for time dependence between values of different lags.

```
ar1 = Arima(ApplPrice_ts, order = c(1,1,0))
acf(ar1$residuals, main = "ACF of the residuals")
```




```
pacf(ar1$residuals, main = "PACF of the residuals")
```




```
Box.test(ar1$residuals)
```

```
##
##  Box-Pierce test
##
## data:  ar1$residuals
## X-squared = 0.0229, df = 1, p-value = 0.8797
```

Since the Box-Pierce test shows that there is not time dependence between the lags, we can move on with the forecasting.

```
ar1.pred = forecast(ar1, h = 10)
print("The point forecast for the 10-step ahead is as follows:")
```

```
## [1] "The point forecast for the 10-step ahead is as follows:"
```

```
print(ar1.pred$mean)
```

```
## Time Series:
## Start = c(2012, 149)
## End = c(2012, 158)
## Frequency = 252
##  [1] 606.2544 606.2544 606.2544 606.2544 606.2544 606.2544 606.2544
##  [8] 606.2544 606.2544 606.2544
```

```
print(paste("While the value at the last observation is", toString(ar1$fitted[length(ar1$fitte
d)])))
```

```
## [1] "While the value at the last observation is 606.940246817187"
```

The values of the 10 step ahead point forecast are the same, which actually coincides with theory, since for mean is 0 and the persistence is 1, the point forecast for k-steps ahead at time t will just approximate to the realized value at time t.

Problem 3

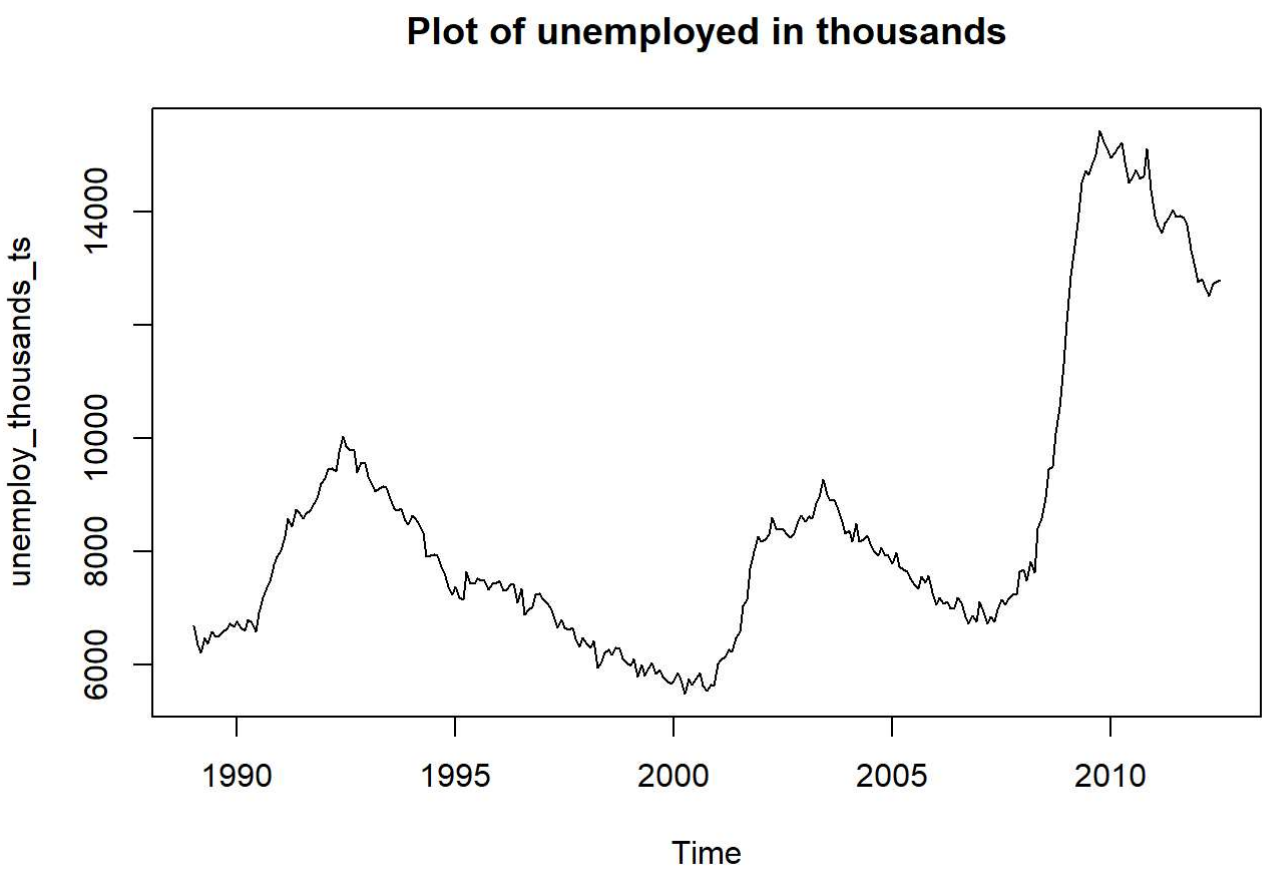
Clear the previous data and obtain the new data

```
rm(list = ls(all=TRUE))
data = read.csv("7.2.csv")
names(data) = c('date', 'unemploy_thousands')
attach(data)
```

```
## The following object is masked from data (pos = 3):
##
##     date
```

To explain the difference in acf or pacf, we need to look at the plot of the data

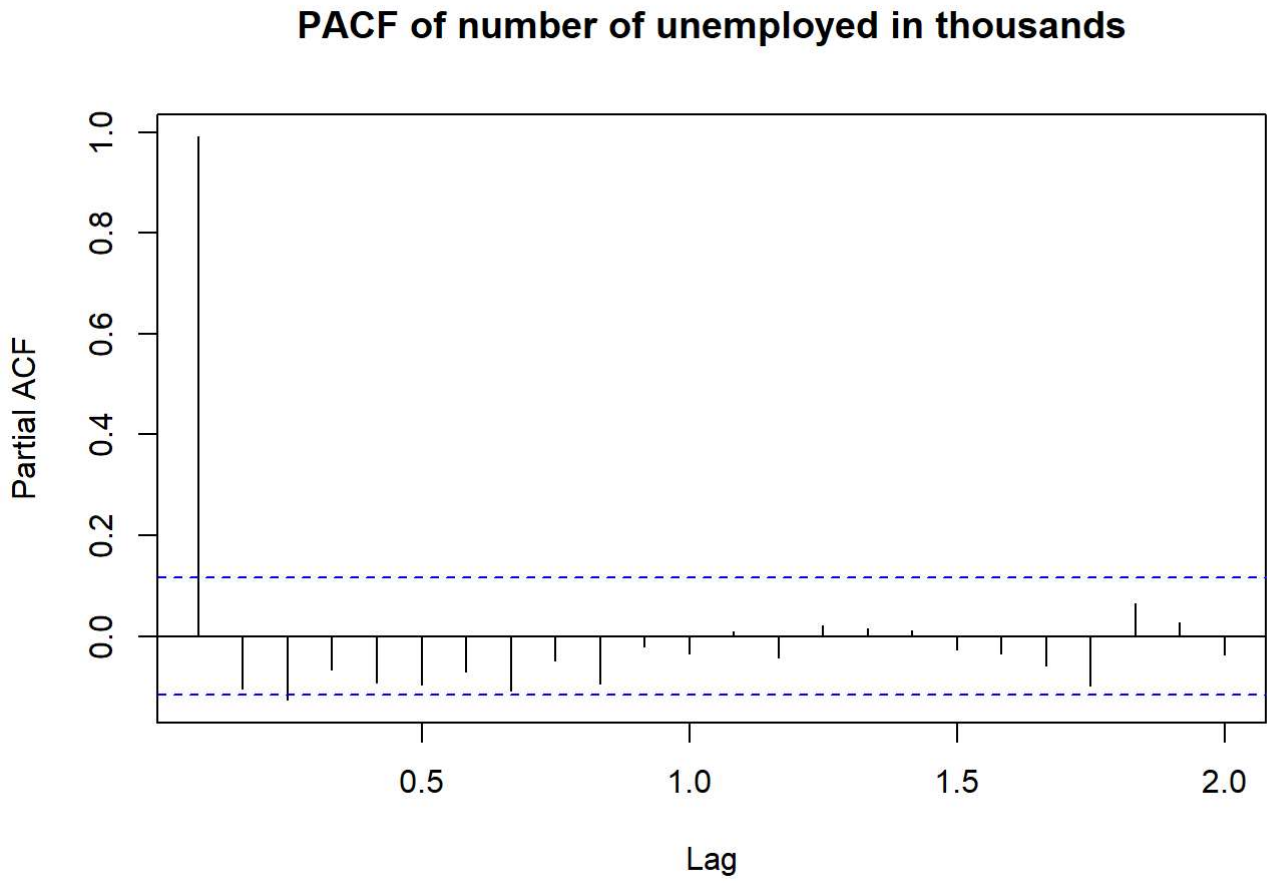
```
unemploy_thousands_ts = ts(unemploy_thousands, start = 1989, freq = 12)
plot(unemploy_thousands_ts, main = "Plot of unemployed in thousands")
```



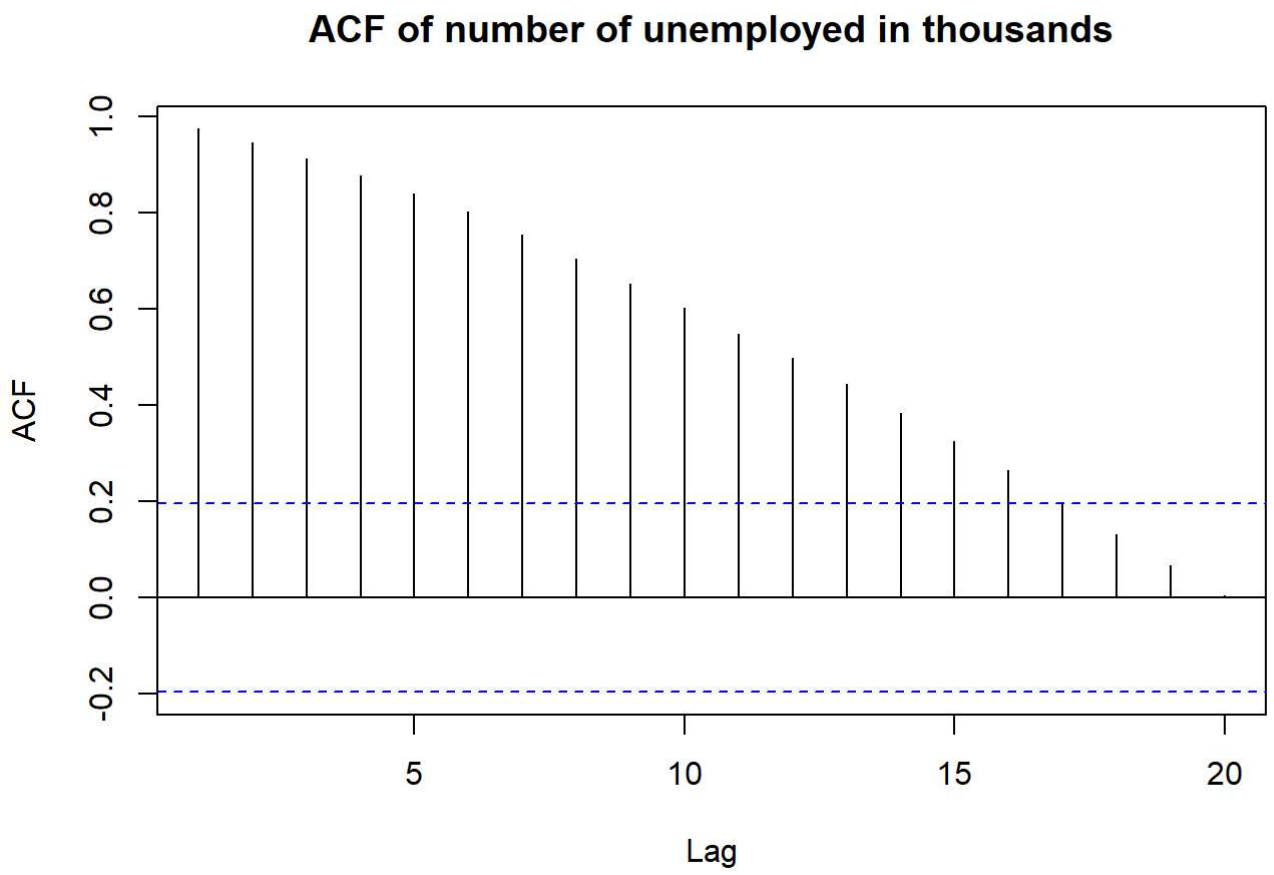
Create a time series object for the number of unemployed in thousands and calculate the acf and pacf in batches of 100. First pair of acf and pacf takes in all data points, second pair takes in the 1st hundred, the third pair takes in the 2nd hundred, while the last pair takes in 183th data point to the last data point.

```
unemploy_thousands_ts = ts(unemploy_thousands, start = 1989, frequency = 12)

acf(unemploy_thousands_ts, main = "ACF of number of unemployed in thousands")
```

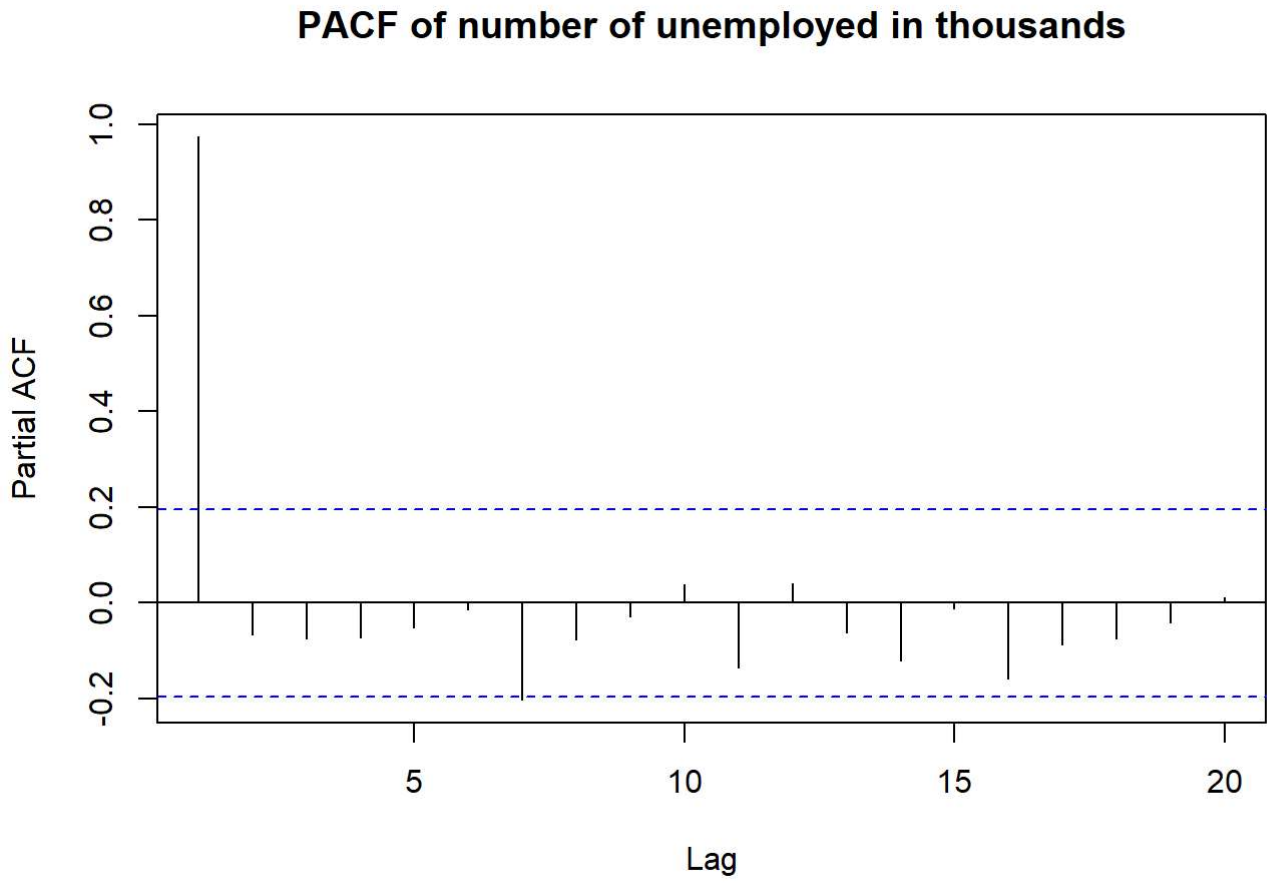

```
pacf(unemploy_thousands_ts, main = "PACF of number of unemployed in thousands")
```



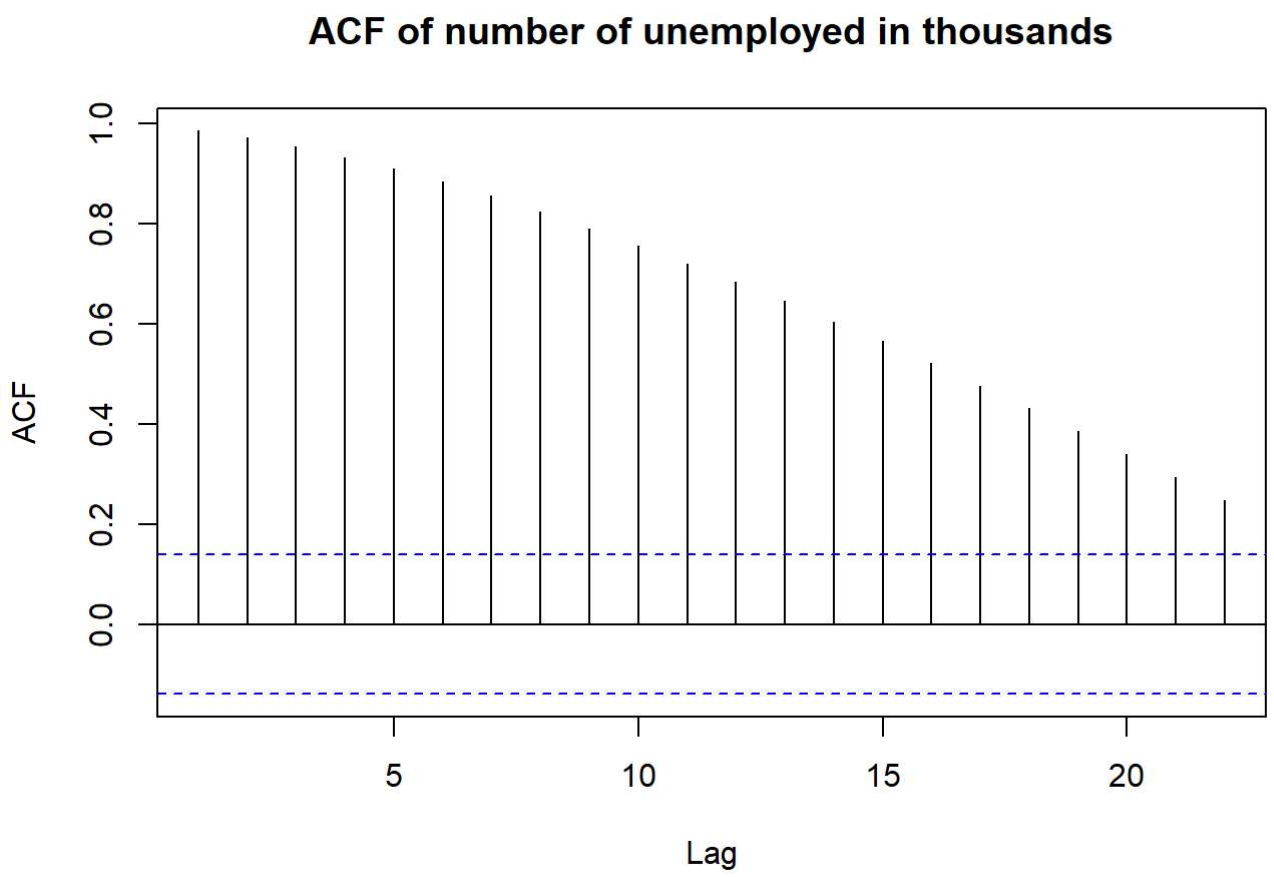

```
acf(unemploy_thousands_ts[1:100], main = "ACF of number of unemployed in thousands")
```



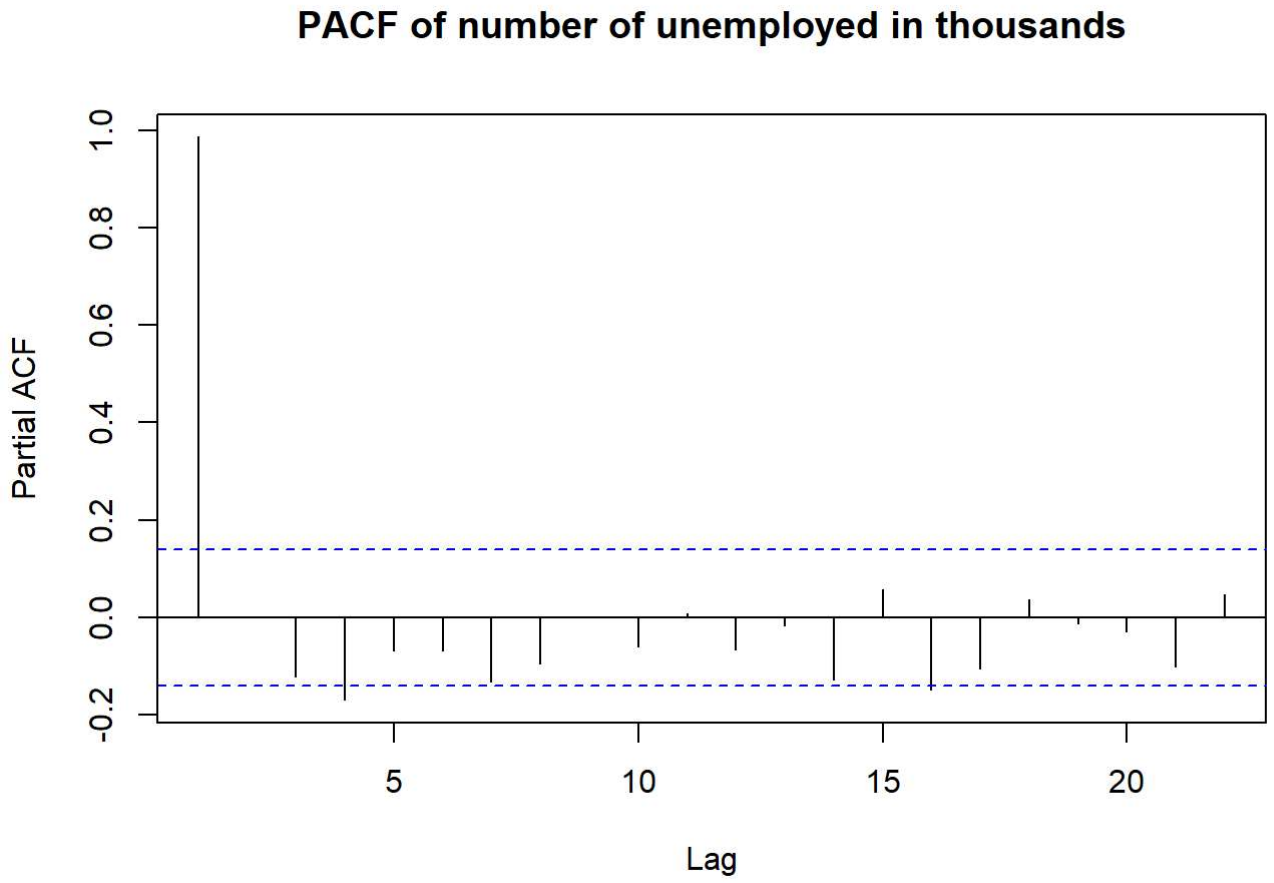

```
pacf(unemploy_thousands_ts[1:100], main = "PACF of number of unemployed in thousands")
```



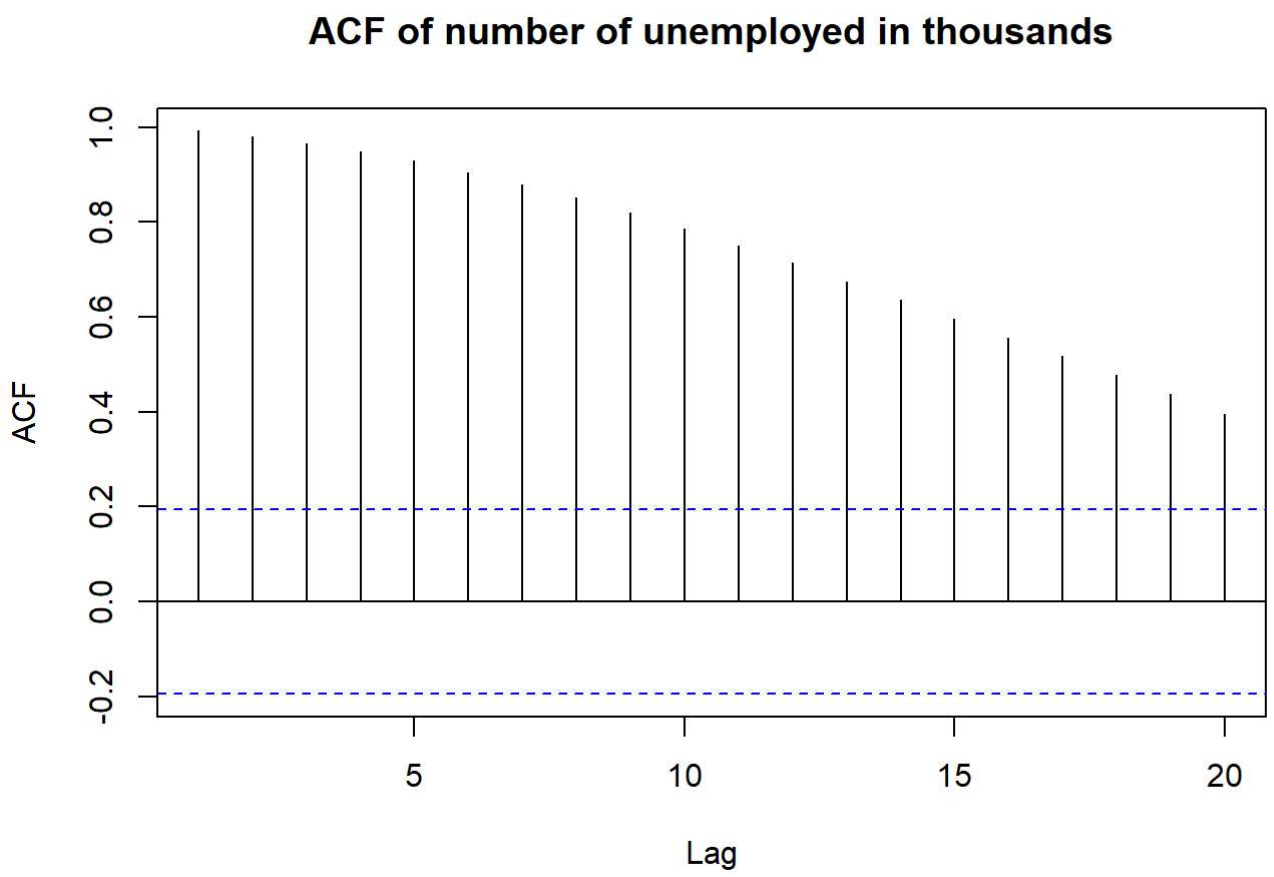

```
acf(unemploy_thousands_ts[2:200], main = "ACF of number of unemployed in thousands")
```



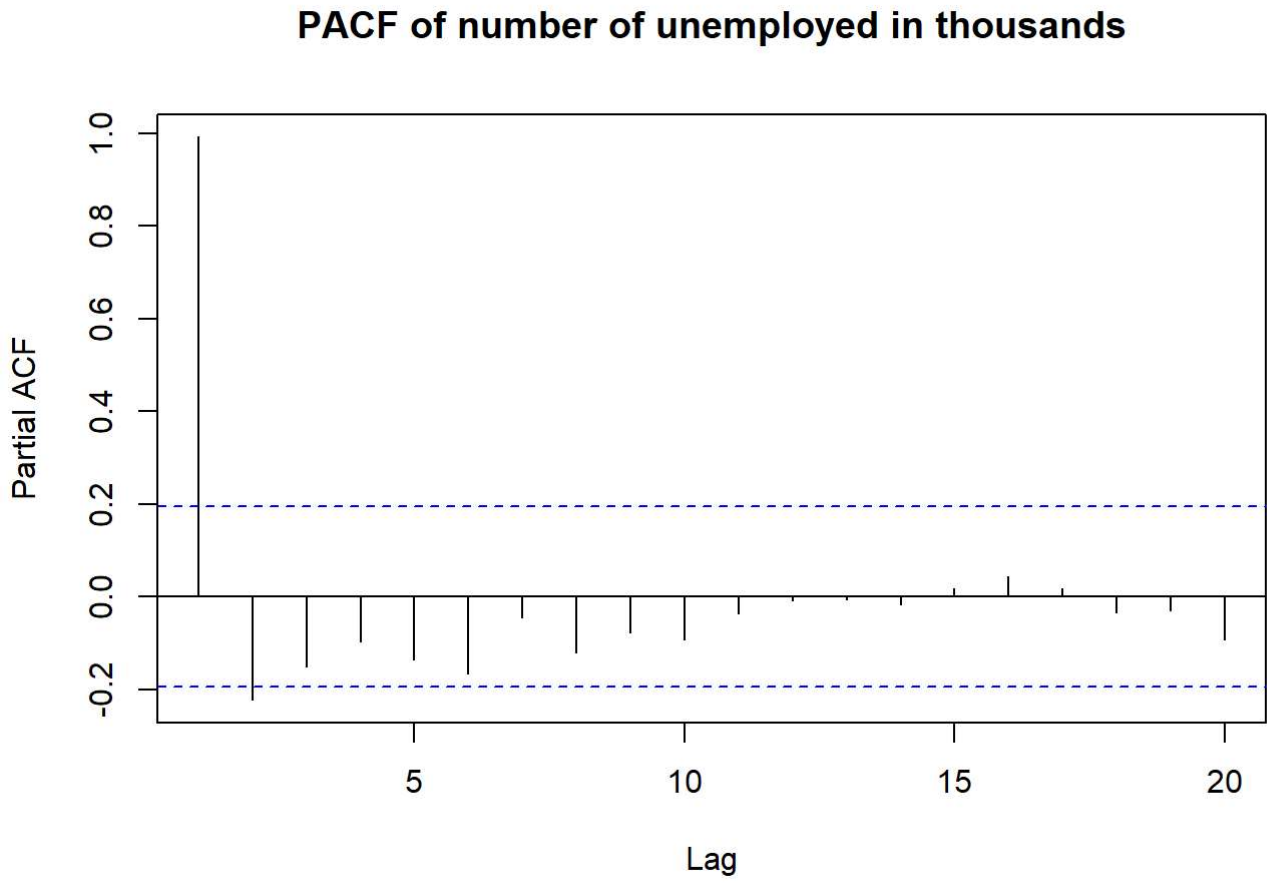

```
pacf(unemploy_thousands_ts[2:200], main = "PACF of number of unemployed in thousands")
```




```
acf(unemploy_thousands_ts[183:283], main = "ACF of number of unemployed in thousands")
```




```
pacf(unemploy_thousands_ts[183:283], main = "PACF of number of unemployed in thousands")
```



From the ACFs, there is a slow decay as the lag value increases and for each of the PACF, we can see there is a signifcant time dependence with lag 1 and that there seems to be a significant time dependence with other lags which is much smaller in magnitude compared to the first lag in various other lags. This means that there seems that for the time spans that I choose, there seems to be not much variation in the significance of the time dependence for for all lags.

AR model is also good to fit the data since there is a few spikes in the PACF and a decay of the ACF as lag values increases.

Problem 4

Clear all the previous environment and add in the new data

```
rm(list = ls(all=TRUE))
data = read.csv("7.6.csv")
names(data)[c(3,5)] = c("housing.inflation", "transportation.inflation")
attach(data)
```

```
## The following object is masked from data (pos = 3):
##
##     date
```

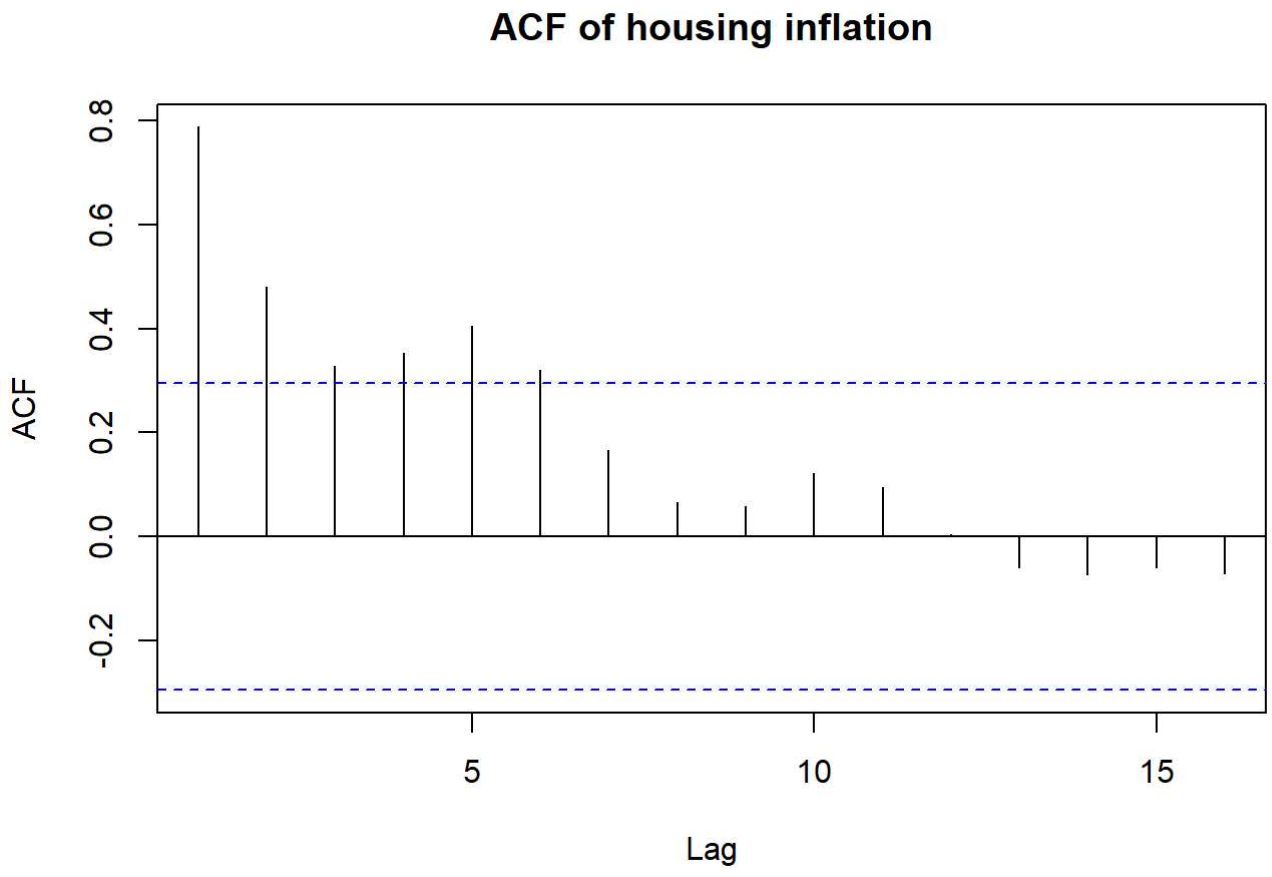
```
## The following object is masked from data (pos = 4):
##
##     date
```

To obtain the inflation rate, first create a time series object for all 4 and take the the diffence between the current values and its lag(can't do difference of log since there are negative values)

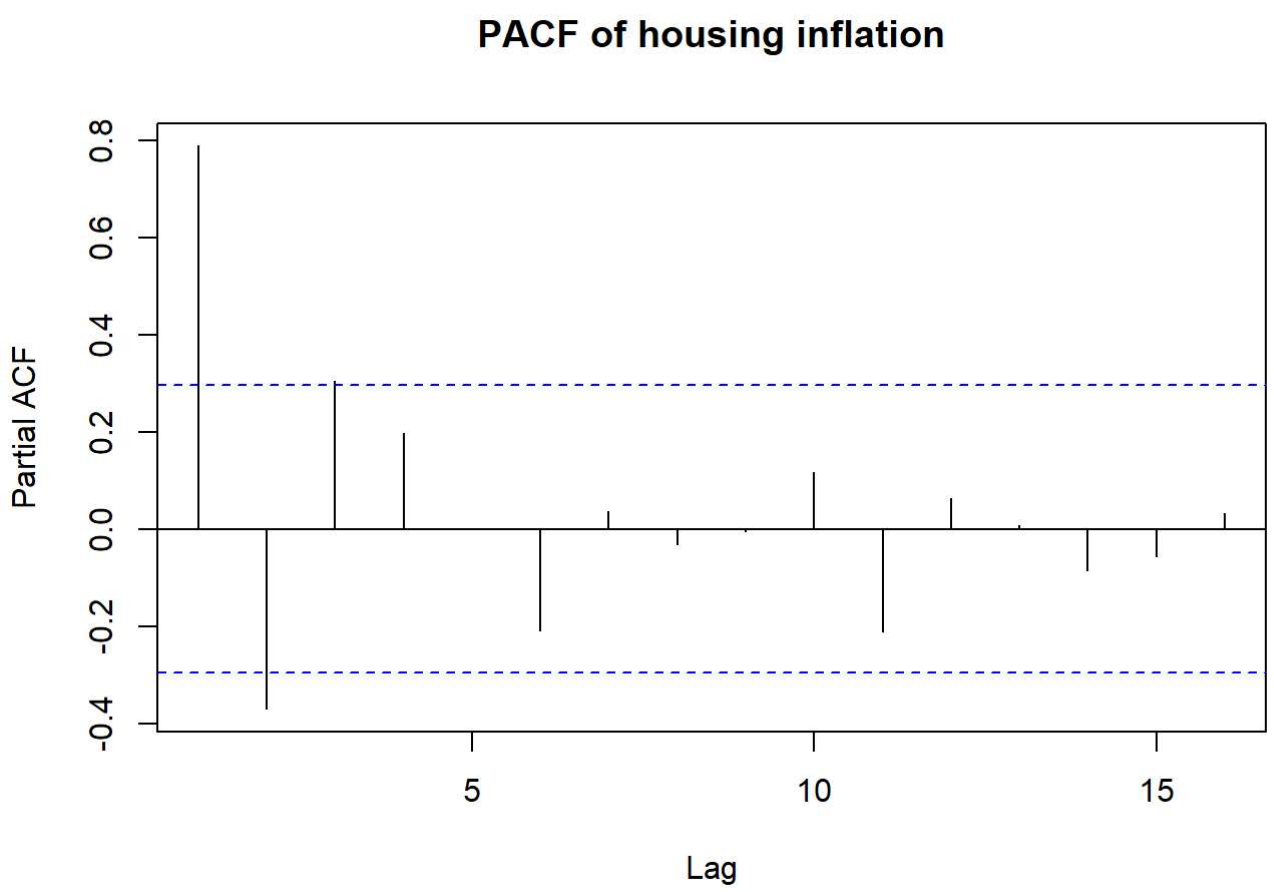
```
hinf_ts = ts(housing.inflation, start = 1976, freq = 1)
tinf_ts = ts(transportation.inflation, start = 1976, freq = 1)
```

To understand whether the data can be modeled as an AR(2) process, we have to look at the ACF and PACF of all the percentage change time series objects

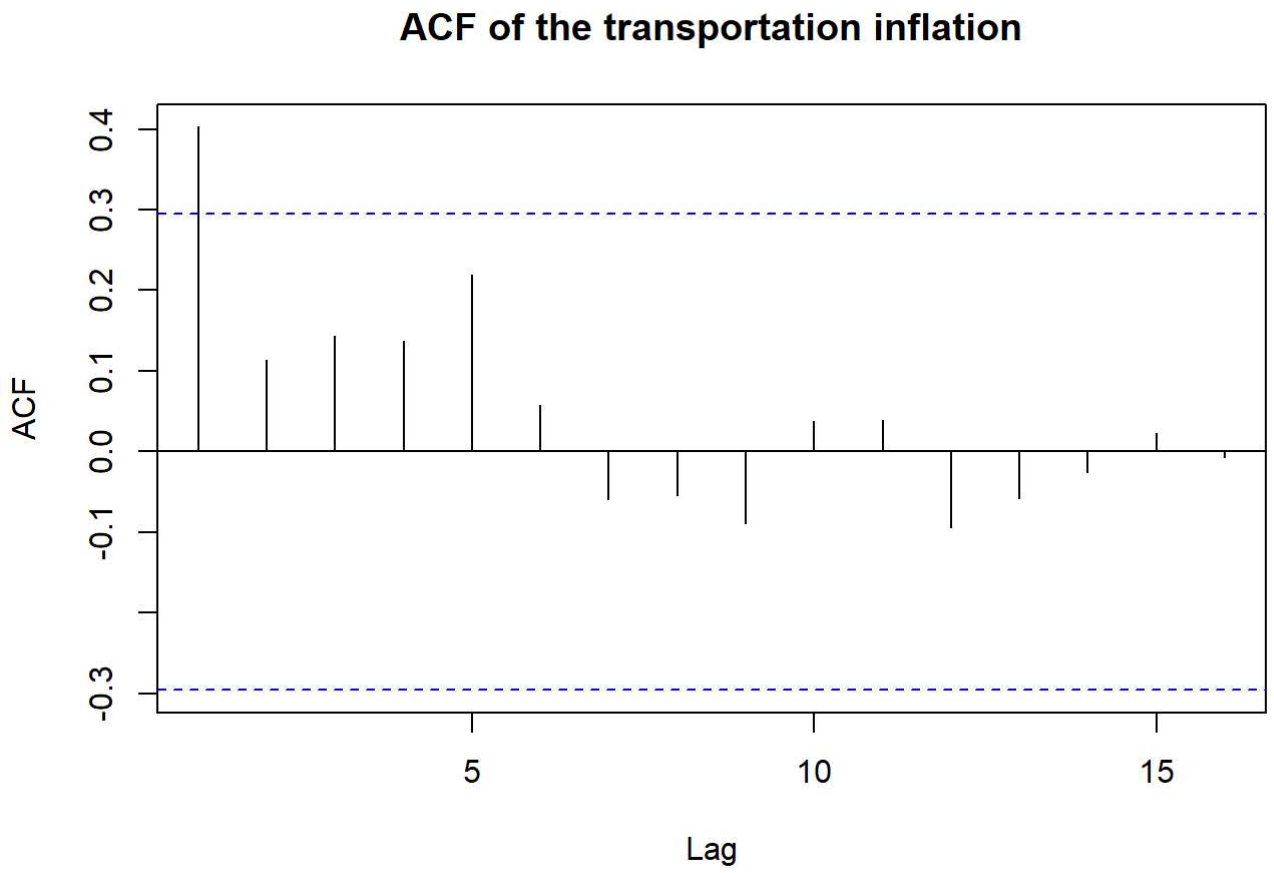
```
acf(hinf_ts[complete.cases(hinf_ts)], main = "ACF of housing inflation")
```



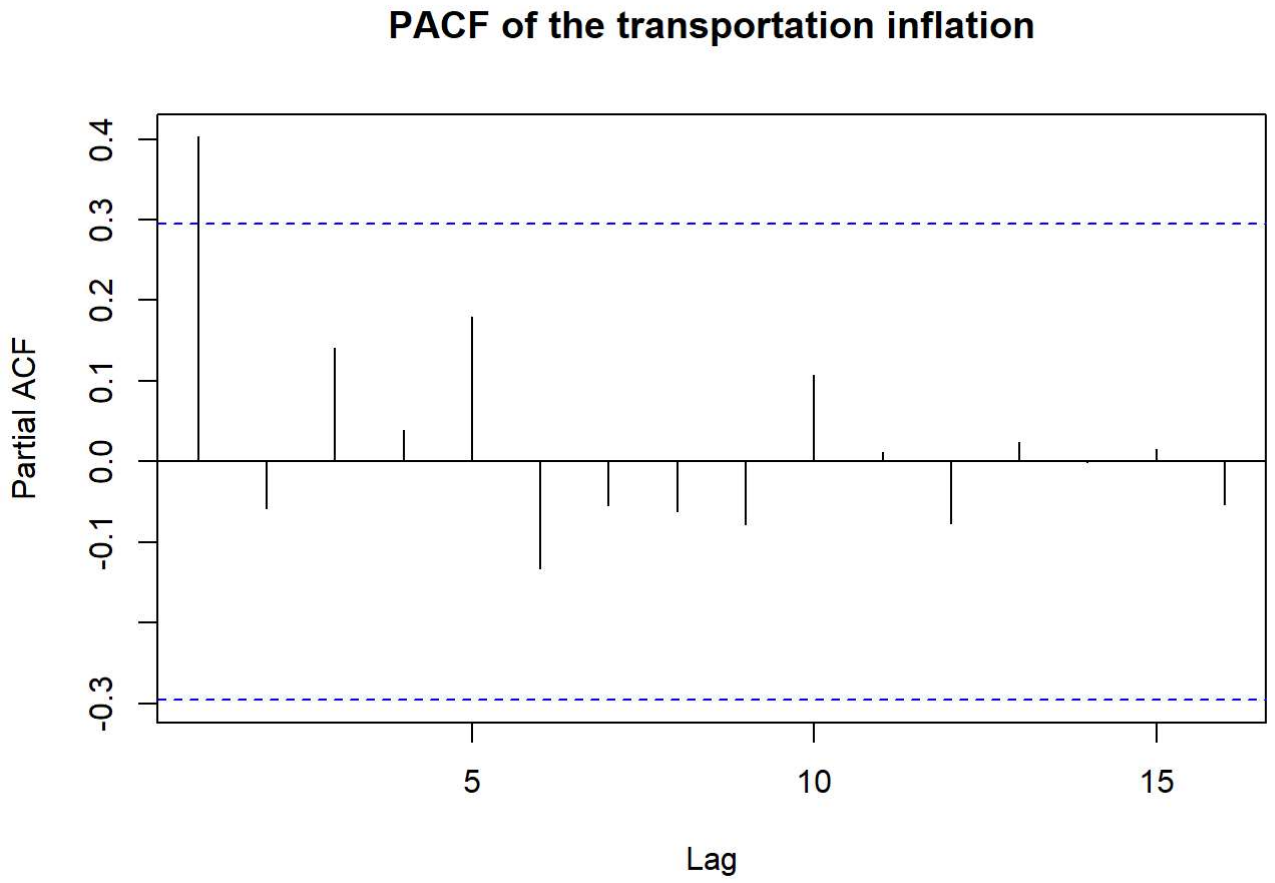

```
pacf(hinf_ts[complete.cases(hinf_ts)],  main = "PACF of housing inflation")
```




```
acf(tinf_ts[complete.cases(tinf_ts)], main = "ACF of the transportation inflation")
```




```
pacf(tinf_ts[complete.cases(tinf_ts)],  main = "PACF of the transportation inflation")
```



Based on the ACF and PACF, since there are 3 spikes for housing inflation, which means that we might be able to fit the AR(2), while since there is only 1 spike for the transporation inflation, which means we might not be able to fit it.

Problem 5

Clear all the previous environment and add in the new data

```
rm(list = ls(all=TRUE))
data = read.csv("7.7.csv")
names(data)[c(3,5)] = c("food.inflation", "Gas.inflation")
attach(data)
```

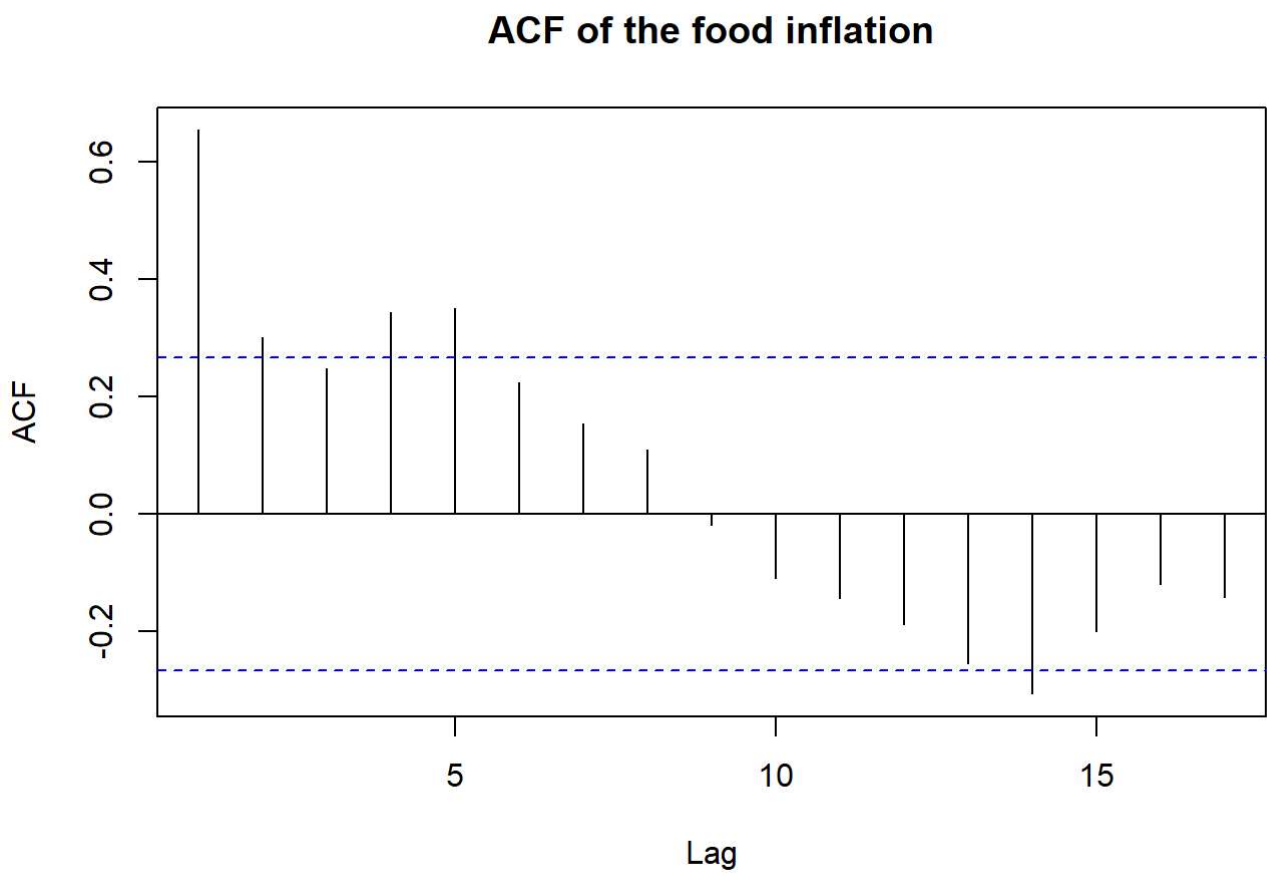
```
## The following object is masked from data (pos = 3):
##
##     date
```

```
## The following object is masked from data (pos = 4):
##
##     date
```

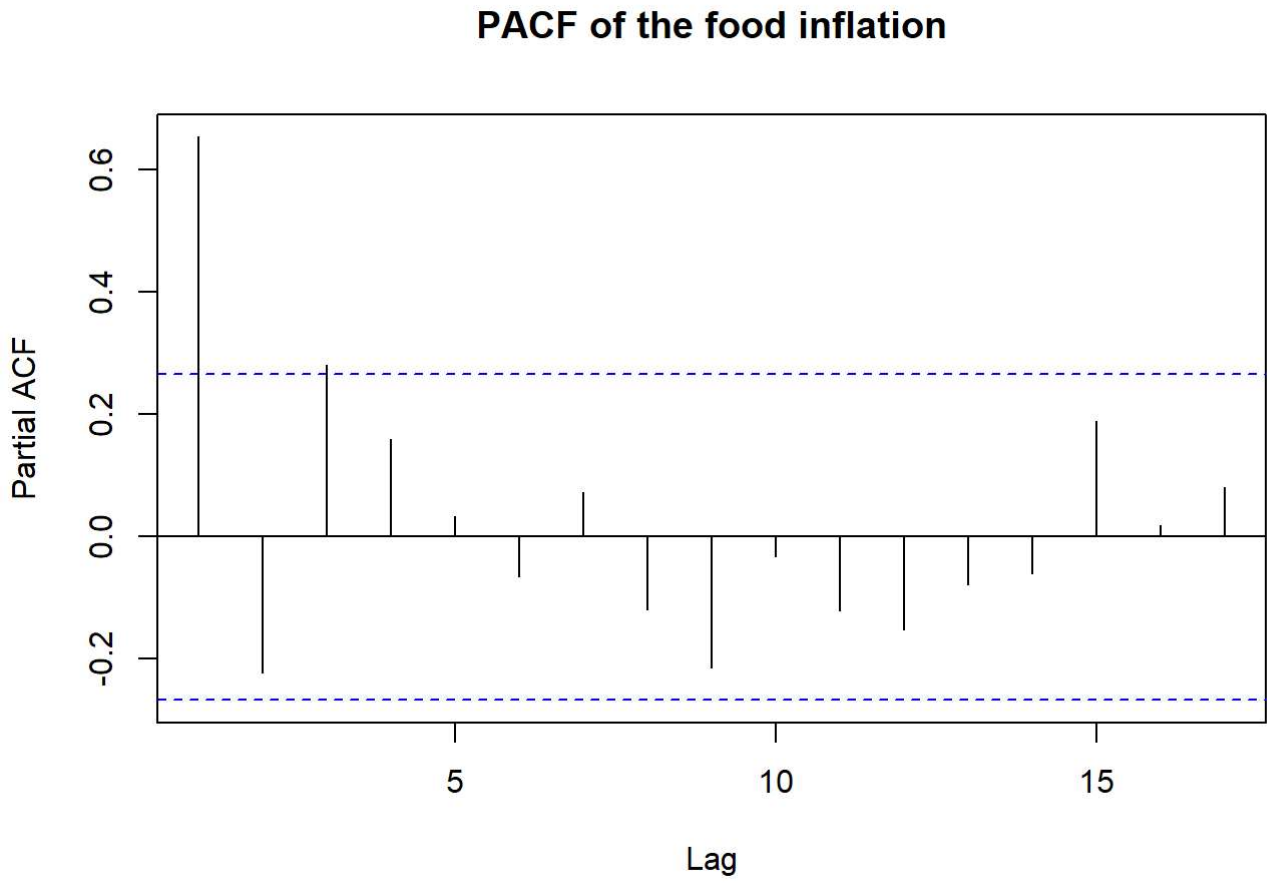
```
## The following object is masked from data (pos = 5):
##
##     date
```

Create a time series objects out of all the variables and find the ACF and PACF for all of them

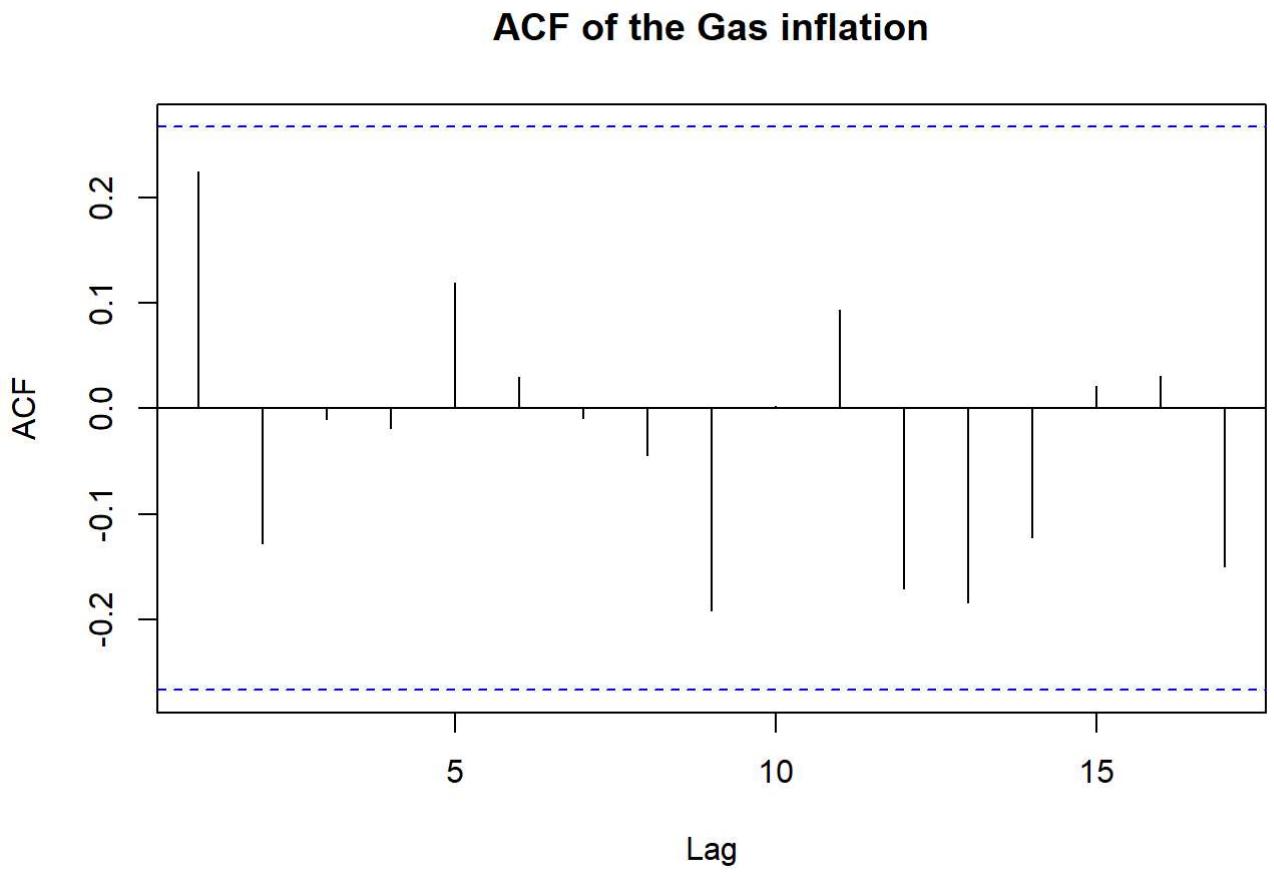
```
finf_ts = ts(food.inflation, start = 1957, freq = 1)
finf_ts = finf_ts [complete.cases(finf_ts )]

ginf_ts = ts(Gas.inflation, start = 1957, freq = 1)
ginf_ts = ginf_ts [complete.cases(ginf_ts)]

acf(finf_ts, main = "ACF of the food inflation")
```



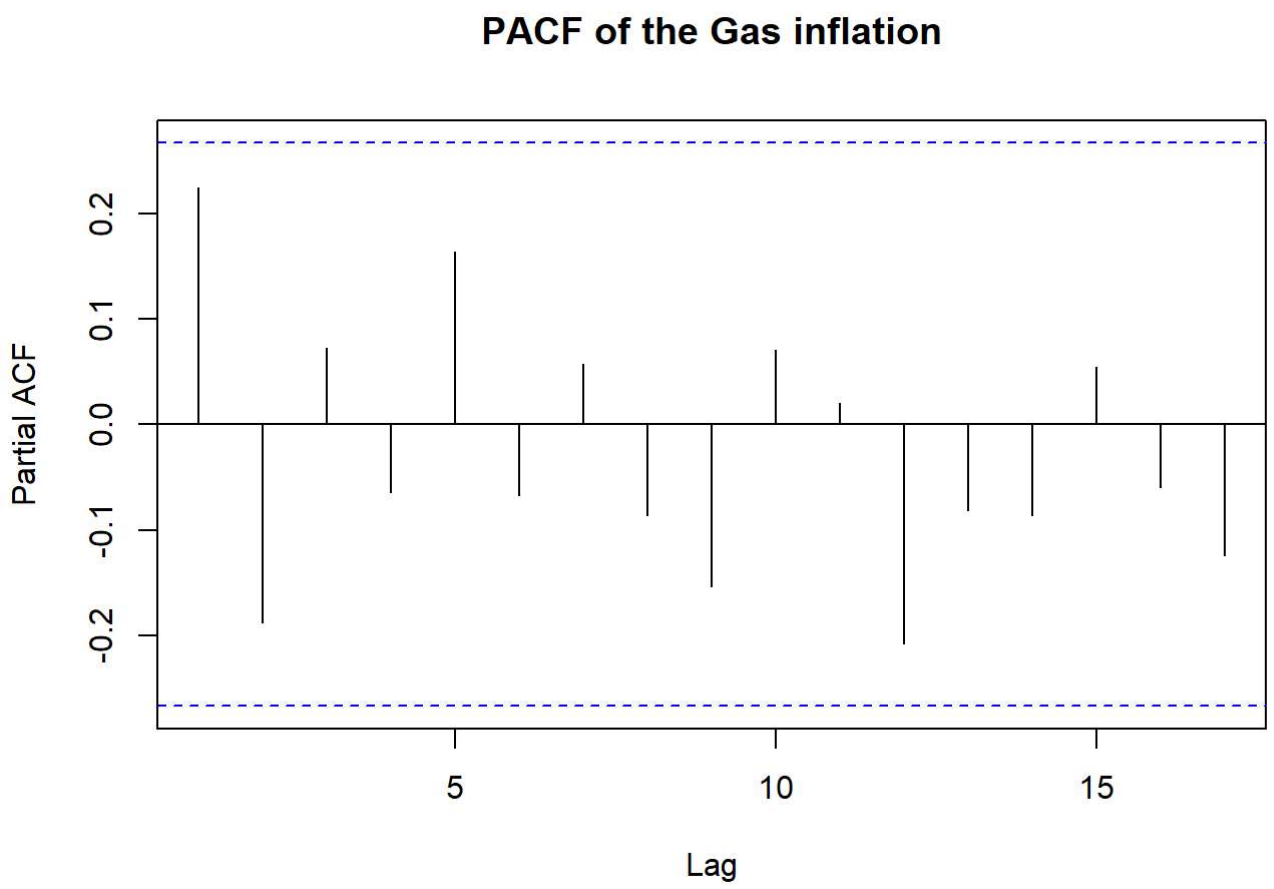

```
pacf(finf_ts, main = "PACF of the food inflation")
```




```
acf(ginf_ts, main = "ACF of the Gas inflation")
```

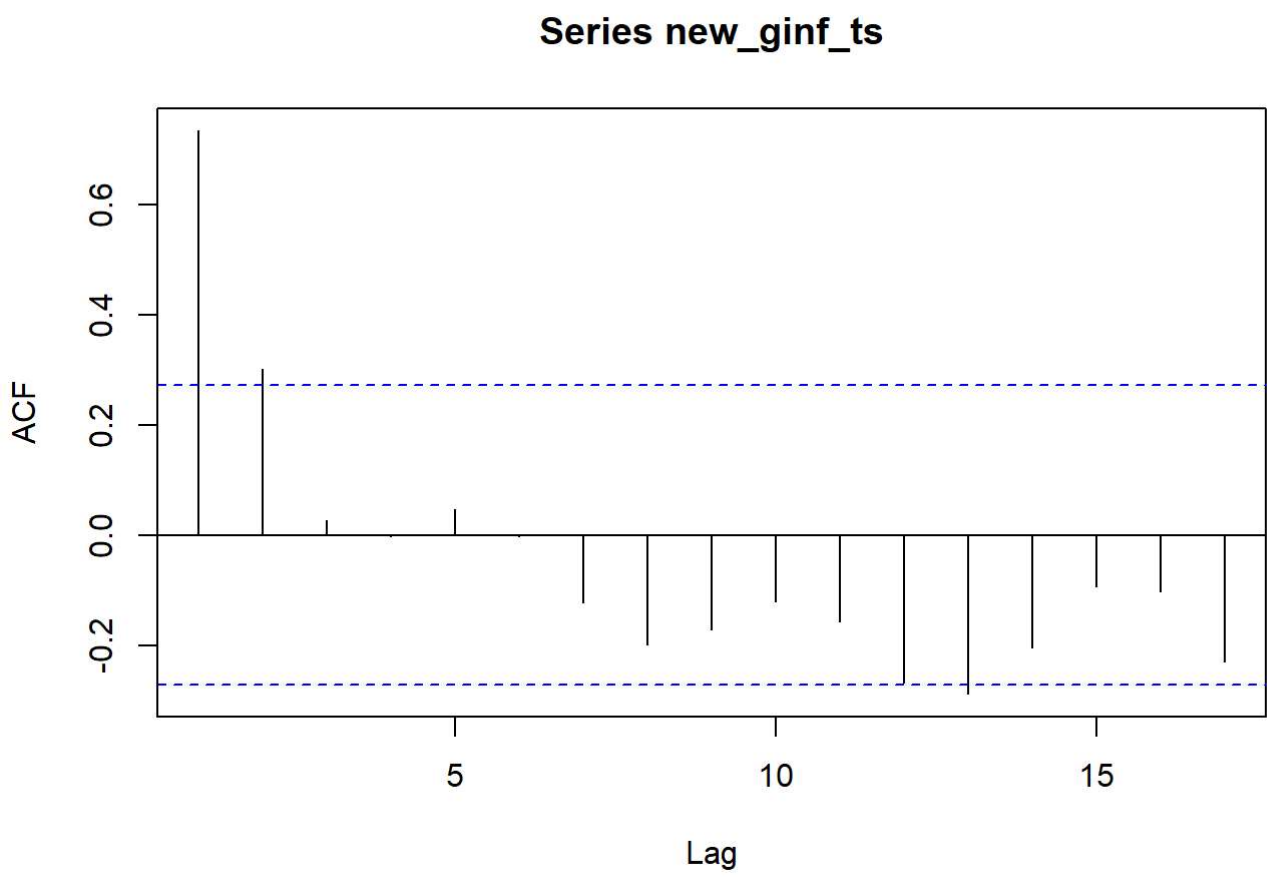



```
pacf(ginf_ts, main = "PACF of the Gas inflation")
```

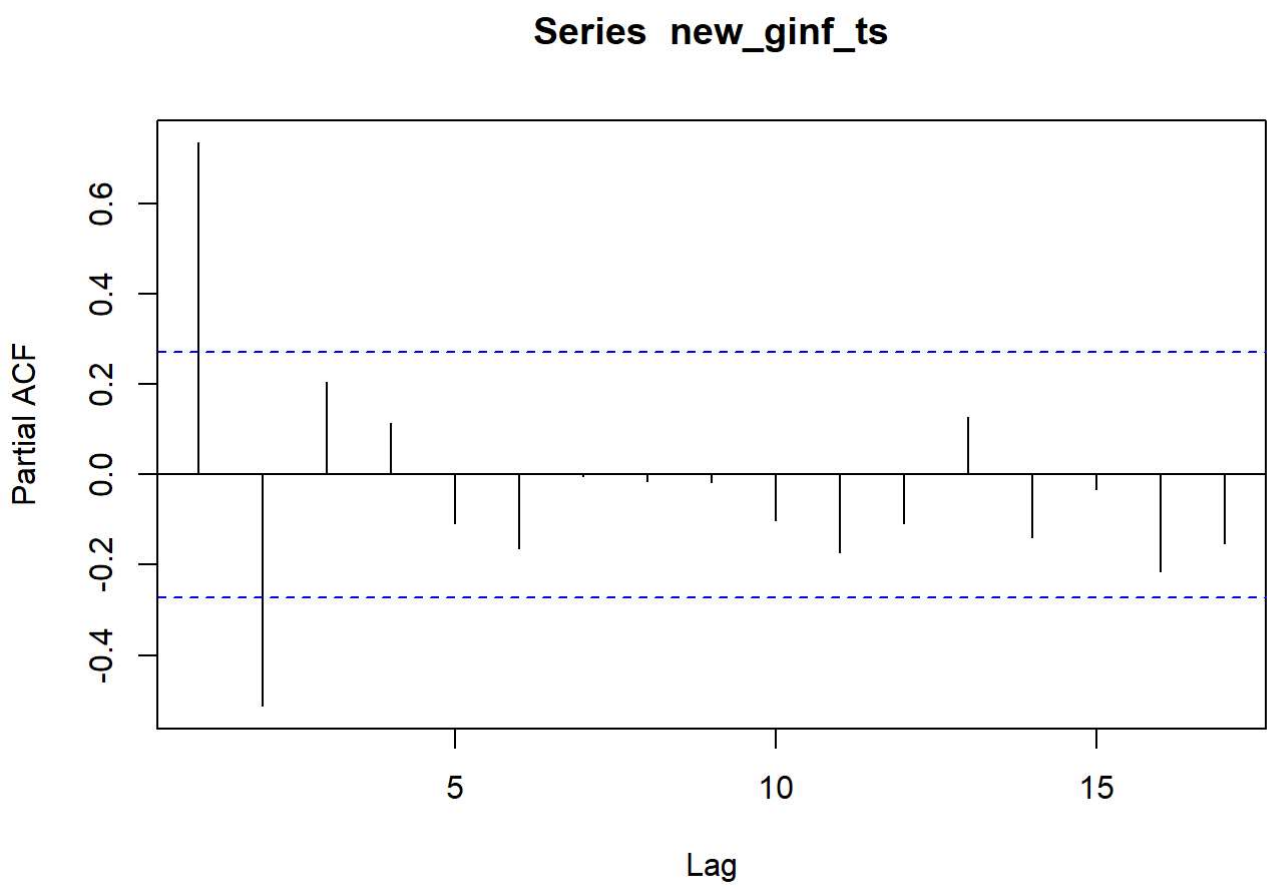


Looking at the PACF of all the time series objects, for food inflation, we might fit an AR(3) model; for the gas inflation however, since the ACF and PACF resembles those of a white noise, we need to do a 3-day moving average and look at the ACF and PACF again. After that, we'll check the ACF and the PACF of the residuals to see if there are any other structure in the data that we have not taken care of.

```
new_ginf_ts = rollmean(ginf_ts, 3)
acf(new_ginf_ts)
```

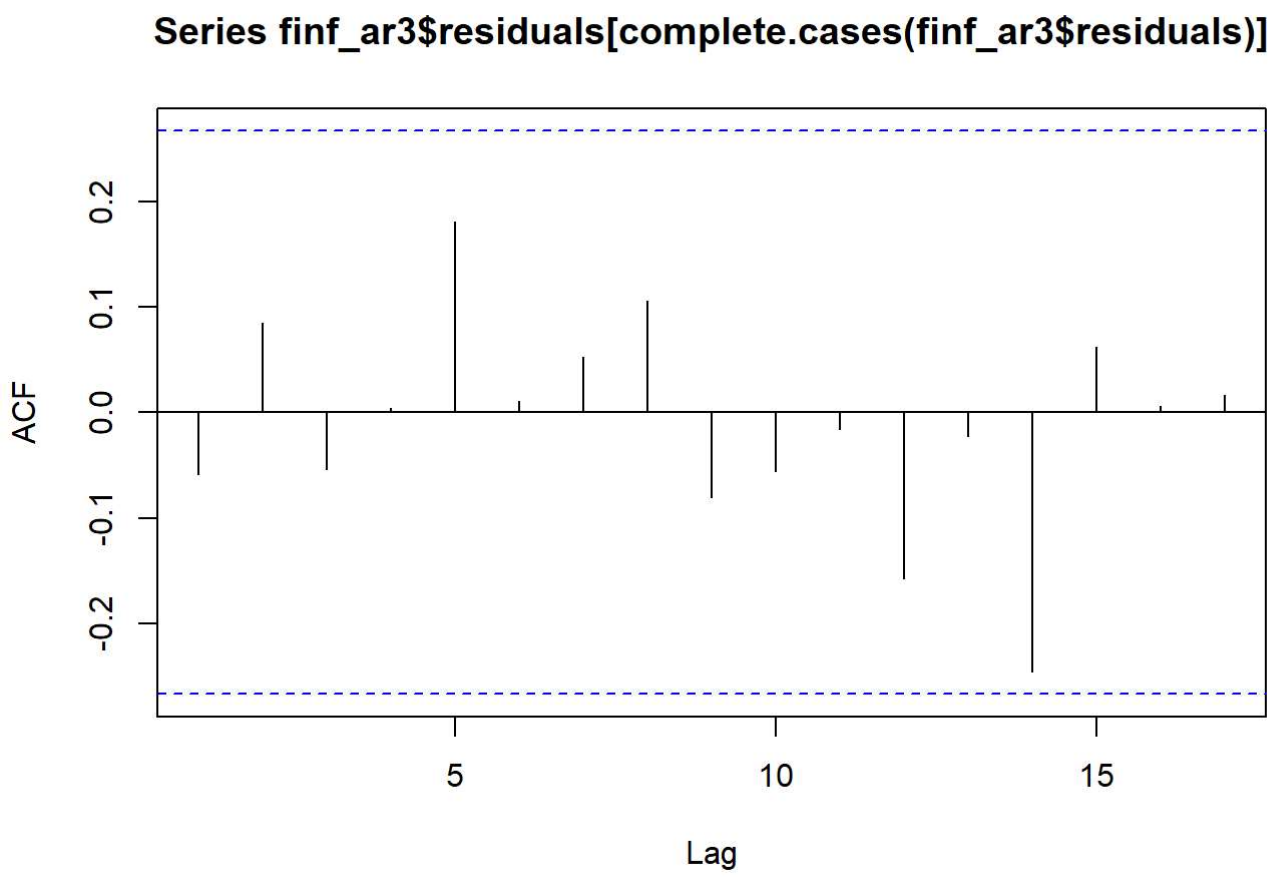



```
pacf(new_ginf_ts)
```

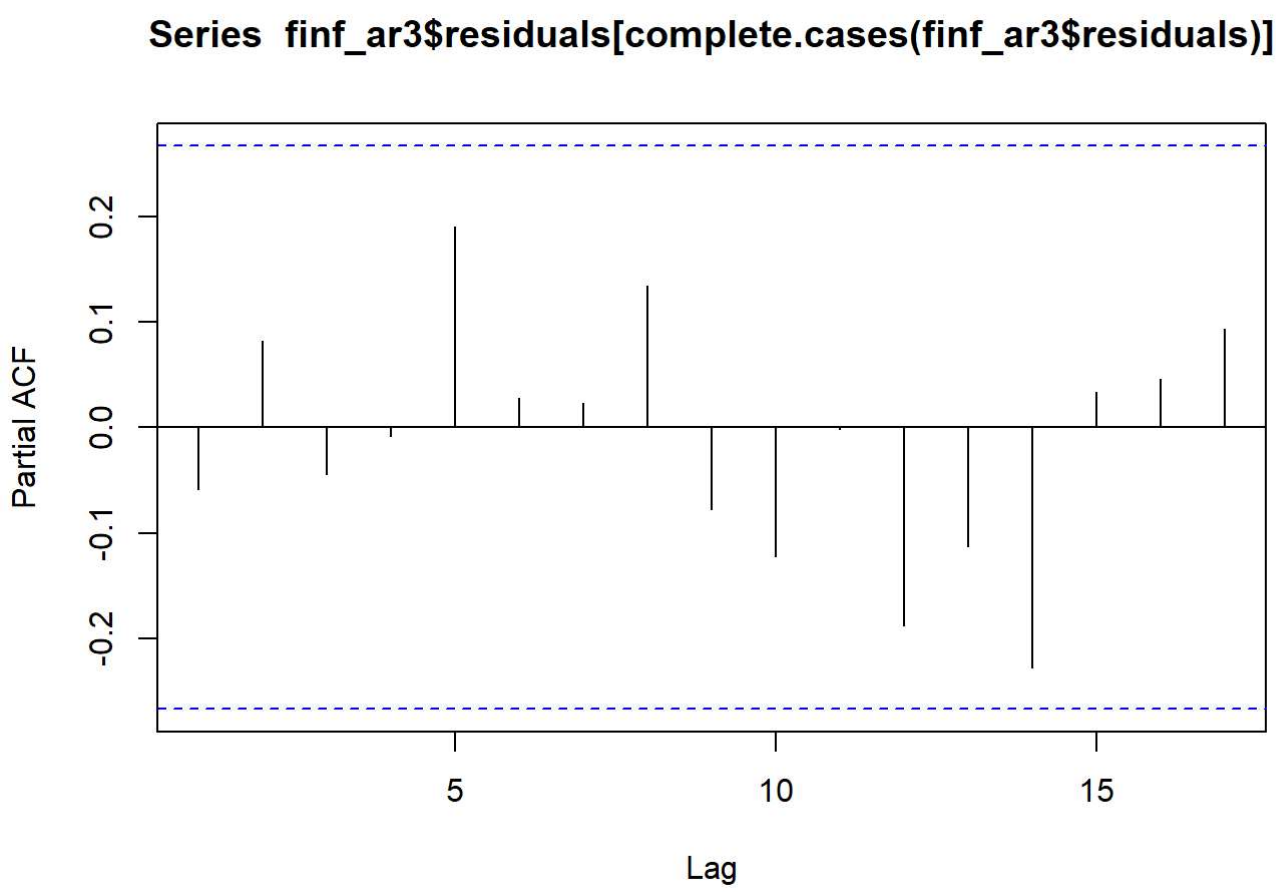


Since there's no more dynamics to look for, we do the fitting of model and check again the ACF and PACF of the residuals.

```
finf_ar3 = Arima(finf_ts, order = c(3,0,0), method = "ML")

ginf_ar2 = Arima(new_ginf_ts, order = c(2,0,0), method = "ML")


acf(finf_ar3$residuals[complete.cases(finf_ar3$residuals)])
```




```
pacf(finf_ar3$residuals[complete.cases(finf_ar3$residuals)])
```




```
acf(ginf_ar2 $residuals[complete.cases(ginf_ar2$residuals)])
```


```
pacf(ginf_ar2 $residuals[complete.cases(ginf_ar2$residuals)])
```

Now we see that the dynamics has been taken care of. So now we'll conduct a 1-step, 2-step and 3-step forecasts for all the models and its corresponding forecast error and uncertainty.

```
#Last element index
ind = length(food.CPI)

#Last element for the CPIs and the inflations
lastelfInf = food.inflation[ind]
lastelgInf = Gas.inflation[ind]

#for food inflation, create a list of realized future values
e = rnorm(3)
finf = vector()
finf[1] = food.inflation[length(food.inflation)-2]
finf[2] = food.inflation[length(food.inflation)-1]
finf[3] = food.inflation[length(food.inflation)]
finf[4] = finf_ar3$coef[4] + finf_ar3$coef[1]*finf[3] + finf_ar3$coef[2]*finf[2] +finf_ar3$coef[
3]*finf[1] + e[1]
finf[5] = finf_ar3$coef[4] + finf_ar3$coef[1]*finf[4] + finf_ar3$coef[2]*finf[3] +finf_ar3$coef[
3]*finf[2] + e[2]
finf[6] = finf_ar3$coef[4] + finf_ar3$coef[1]*finf[5] + finf_ar3$coef[2]*finf[4] +finf_ar3$coef[
3]*finf[3] + e[3]

finf_ar3.pred = forecast(finf_ar3,h=3)
print("For food inflation AR(3,0),")
```

```
## [1] "For food inflation AR(3,0),"
```

```
for (i in 1:3){
  print(paste("Point forecast for", toString(i), "step(s) is", toString(finf_ar3.pred$mean[i])))
  print(paste("Forecast error for", toString(i), "step(s) is", toString(finf[3+i] -finf_ar3.pred
$mean[i])))

  #Since the upper bound of 95% confidence interval = point_forecast + 1.96(forecast_variance)
  forecast_variance = (finf_ar3.pred$upper[i,2] - finf_ar3.pred$mean[i])/1.96
  print(paste("Forecast uncertainty for", toString(i), "step(s) is", toString(forecast_varianc
e)))
  cat("\n")
}
```

```
## [1] "Point forecast for 1 step(s) is 4.78608872619595"
## [1] "Forecast error for 1 step(s) is 2.07220123450175"
## [1] "Forecast uncertainty for 1 step(s) is 2.25500133527757"
##
## [1] "Point forecast for 2 step(s) is 3.75842087256037"
## [1] "Forecast error for 2 step(s) is 5.28904718800229"
## [1] "Forecast uncertainty for 2 step(s) is 3.04082654924731"
##
## [1] "Point forecast for 3 step(s) is 3.25428232526141"
## [1] "Forecast error for 3 step(s) is 5.57415395593766"
## [1] "Forecast uncertainty for 3 step(s) is 3.10184388210994"
```

```
#do similarly as the above
ginf = vector()
ginf[1] = Gas.inflation[length(Gas.inflation)-1]
ginf[2] = Gas.inflation[length(Gas.inflation)]
ginf[3] = ginf_ar2$coef[3] + ginf_ar2$coef[1]*ginf[2] + ginf_ar2$coef[2]*ginf[1] + e[1]
ginf[4] = ginf_ar2$coef[3] + ginf_ar2$coef[1]*ginf[3] + ginf_ar2$coef[2]*ginf[2] + e[2]
ginf[5] = ginf_ar2$coef[3] + ginf_ar2$coef[1]*ginf[4] + ginf_ar2$coef[2]*ginf[3]+ e[3]
ginf_ar2.pred = forecast(ginf_ar2,h=3)
print("For gas inflation AR(1,0),")
```

```
## [1] "For gas inflation AR(1,0),"
```

```
for (i in 1:3){
  print(paste("Point forecast for", toString(i), "step(s) is", toString(ginf_ar2.pred$mean[i])))
  print(paste("Forecast error for", toString(i), "step(s) is", toString(ginf[2+i] -ginf_ar2.pred
$mean[i])))

  #Since the upper bound of 95% confidence interval = point_forecast + 1.96(forecast_variance)
  forecast_variance = (ginf_ar2.pred$upper[i,2] - ginf_ar2.pred$mean[i])/1.96
  print(paste("Forecast uncertainty for", toString(i), "step(s) is", toString(forecast_varianc
e)))
  cat("\n")
}
```

```
## [1] "Point forecast for 1 step(s) is 7.28102102288834"
## [1] "Forecast error for 1 step(s) is 17.2289373812233"
## [1] "Forecast uncertainty for 1 step(s) is 4.61130565598962"
##
## [1] "Point forecast for 2 step(s) is 7.23441839148651"
## [1] "Forecast error for 2 step(s) is 12.4310654930953"
## [1] "Forecast uncertainty for 2 step(s) is 6.85343091702771"
##
## [1] "Point forecast for 3 step(s) is 6.44977152649723"
## [1] "Forecast error for 3 step(s) is 7.3712713461769"
## [1] "Forecast uncertainty for 3 step(s) is 7.58754652042357"
```

We see that gas inflation has the higher uncertainty which means that gas inflation is harder to forecast.