Homework 4

Xiang Yang Ng

May 17, 2018

Problem 1

Download all the libraries and set the directory

```
rm(list=ls(all=T))
setwd("/Users/Xiang/OneDrive/Desktop/Econ-144/Homework/Hw 4")
library(forecast)
library(stats)
library(timeSeries)
```

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

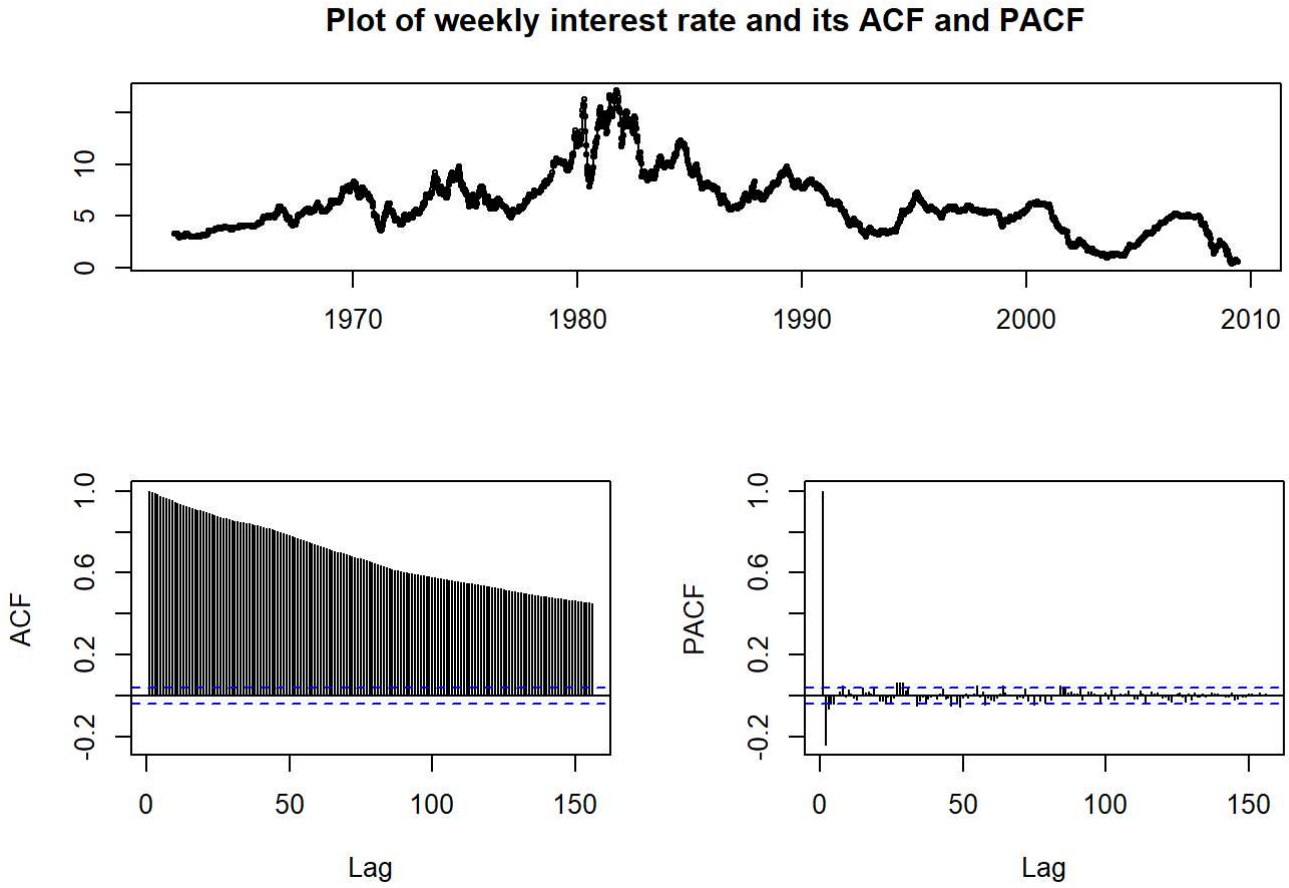
```
library(tseries)
```

Read the dataset

```
data = read.table ("w-gs1yr.txt",header = TRUE)
attach(data)
```

a.

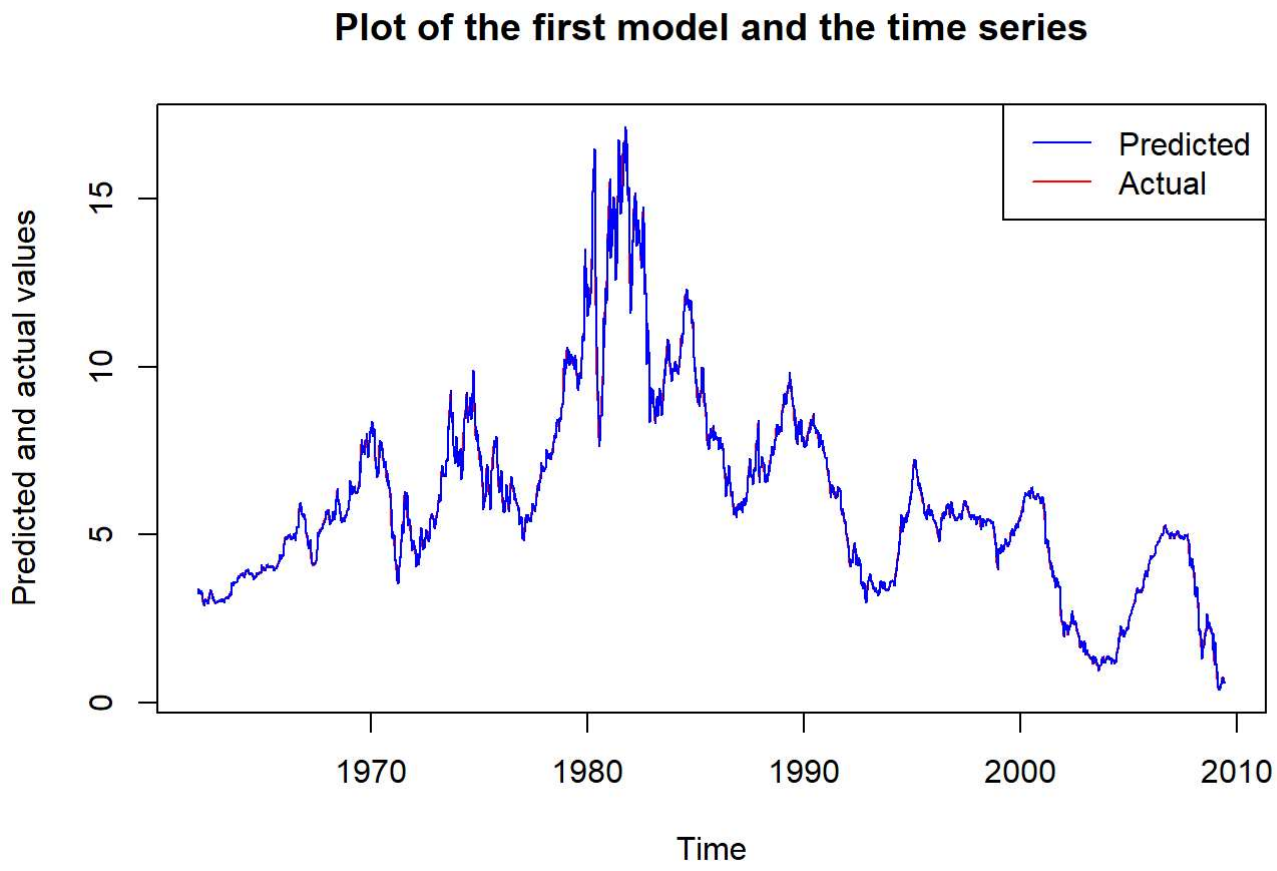
```
#create a time series object for the data
ir_ts = ts(rate, start = 1962+(5/365), freq = 52)

#display the plot, acf and pacf
tsdisplay(ir_ts, main = "Plot of weekly interest rate and its ACF and PACF")
```



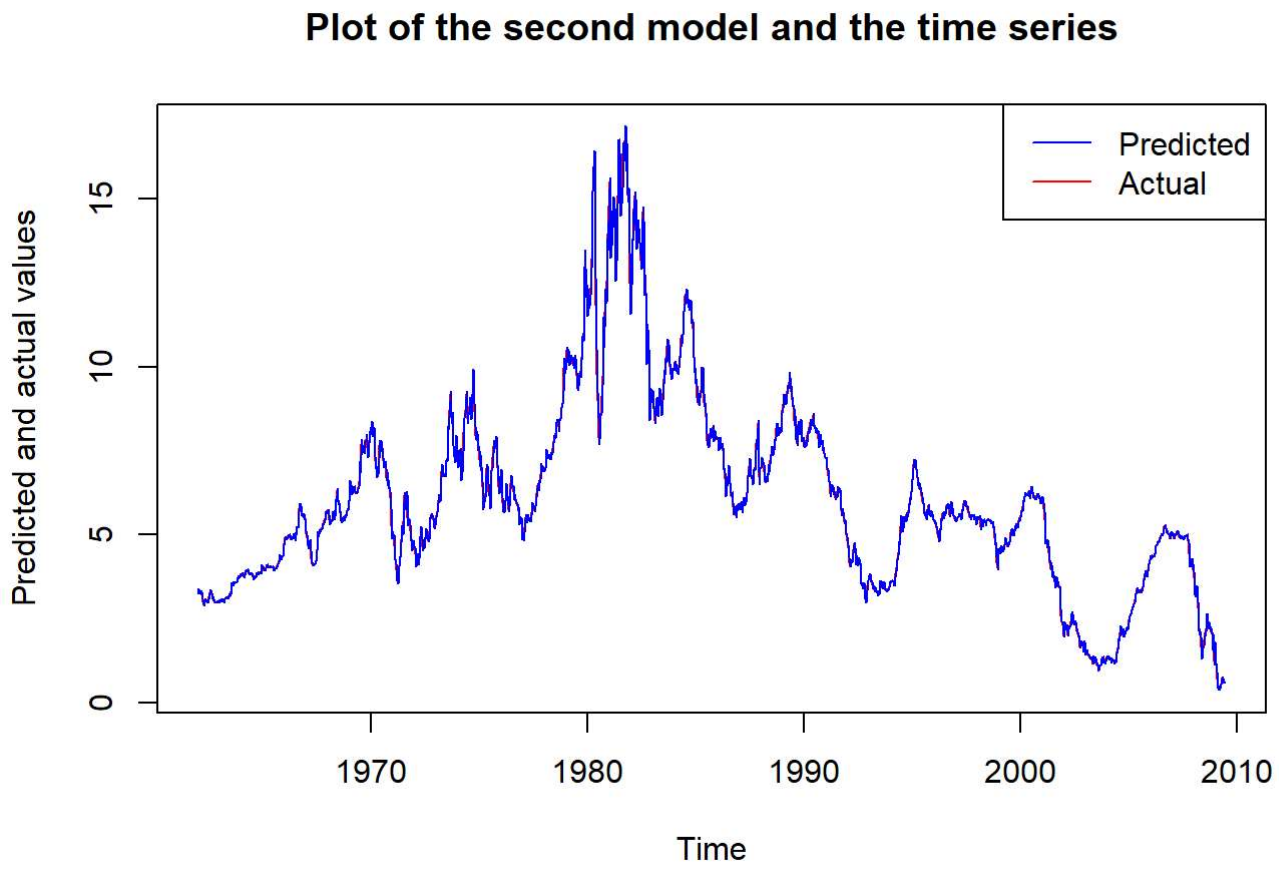
There is very strong persistence in the plot, which might mean that there is cycles in the data. There doesn't seem to have to have a strong upward or downward trend. Looking at the PACF, we can see that there is presence of seasonality, particularly S-MA(3) model might fit the data. Two strong spikes in the PACF might also suggest an AR(2) process.

b.

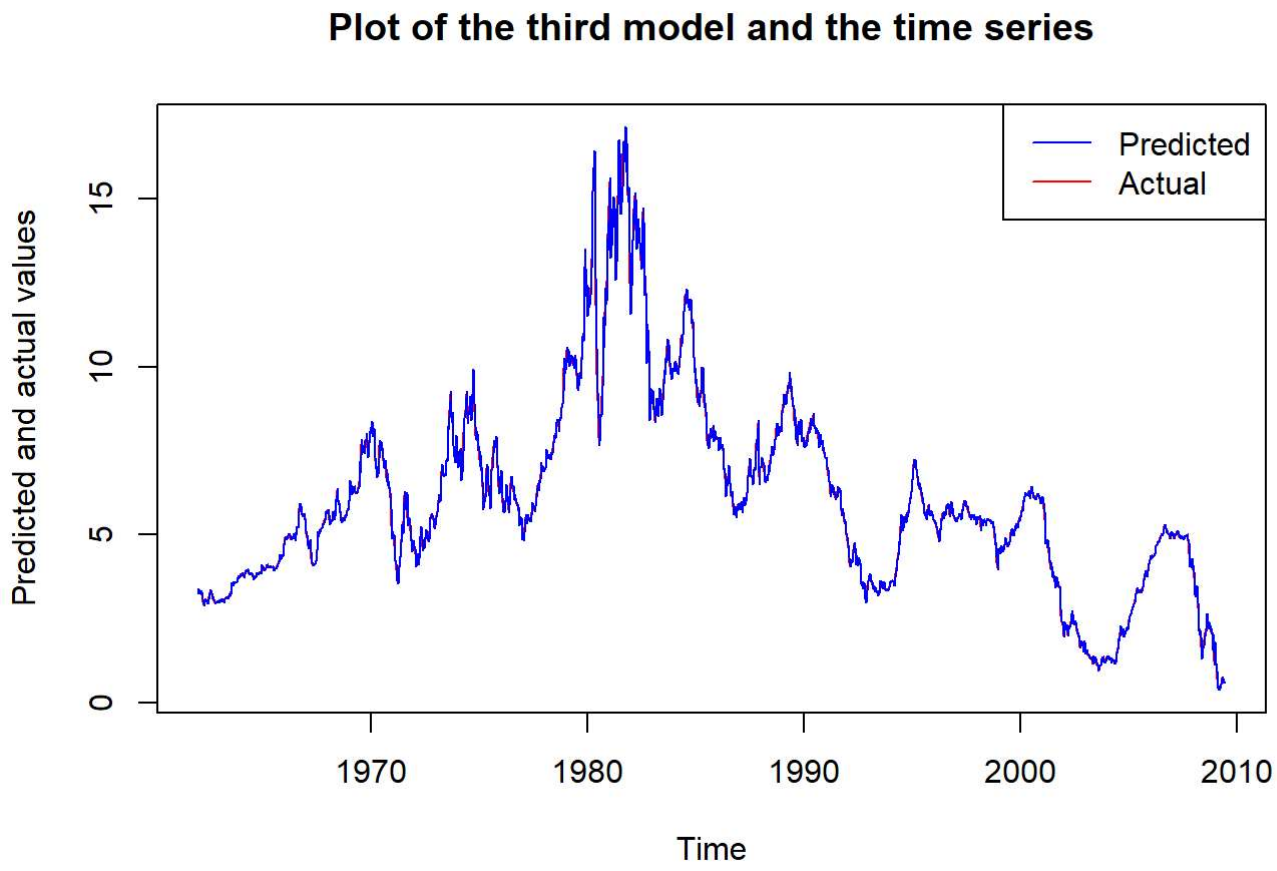
```
#Constructing the models
arma1 = Arima(ir_ts, order = c(2,0,0), seasonal = list(order = c(0,0,3), period = 6))
arma2 = Arima(ir_ts, order = c(3,0,0), seasonal = list(order = c(0,0,3), period = 5))
arma3 = Arima(ir_ts, order = c(3,0,0), seasonal = list(order = c(0,0,2), period = 6))

plot(arma1$x,col="red", main = "Plot of the first model and the time series", xlab = "Time", yla
b = "Predicted and actual values")
legend("topright", legend = c("Predicted", "Actual"), lty = 1, col = c("blue","red"))
lines(fitted(arma1),col="blue")
```




```
plot(arma2$x,col="red", main = "Plot of the second model and the time series", xlab = "Time", yl
ab = "Predicted and actual values")
legend("topright", legend = c("Predicted", "Actual"), lty = 1, col = c("blue","red"))
lines(fitted(arma2),col="blue")
```



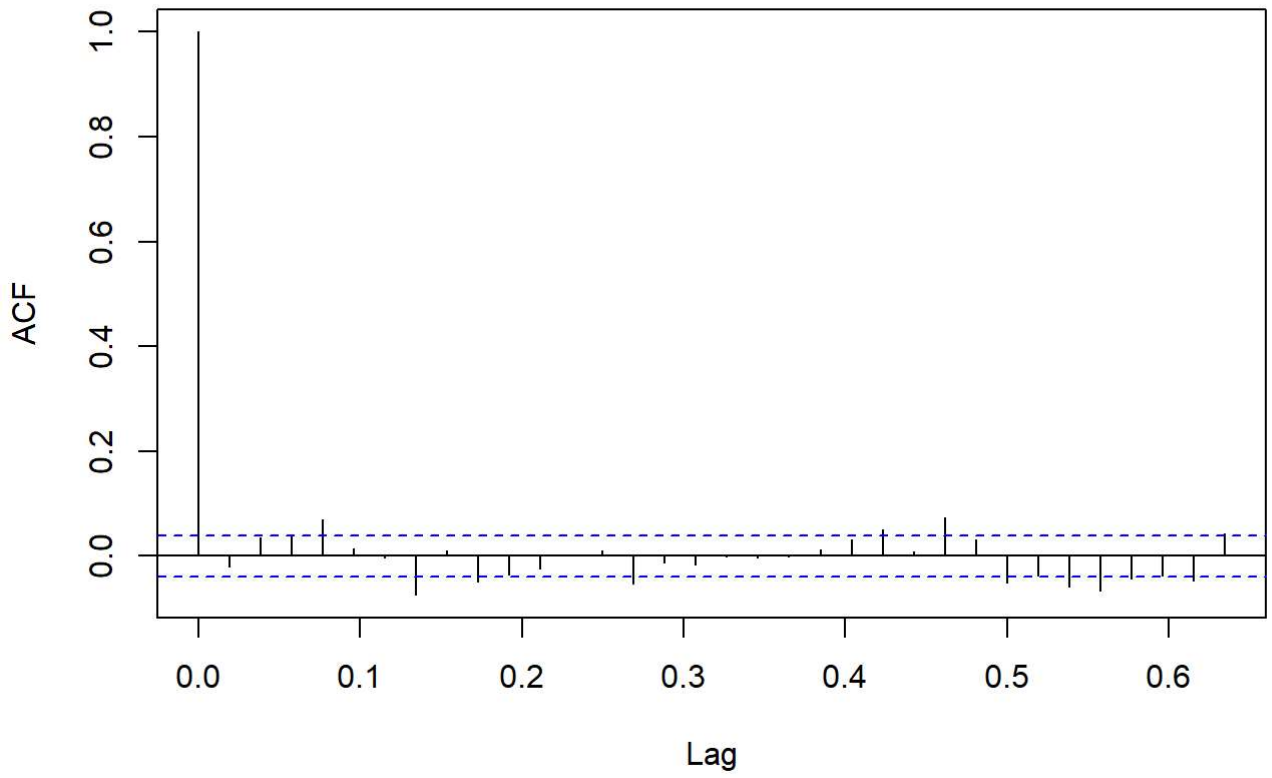

```
plot(arma3$x,col="red", main = "Plot of the third model and the time series", xlab = "Time", yla
b = "Predicted and actual values")
legend("topright", legend = c("Predicted", "Actual"), lty = 1, col = c("blue","red"))
lines(fitted(arma3),col="blue")
```



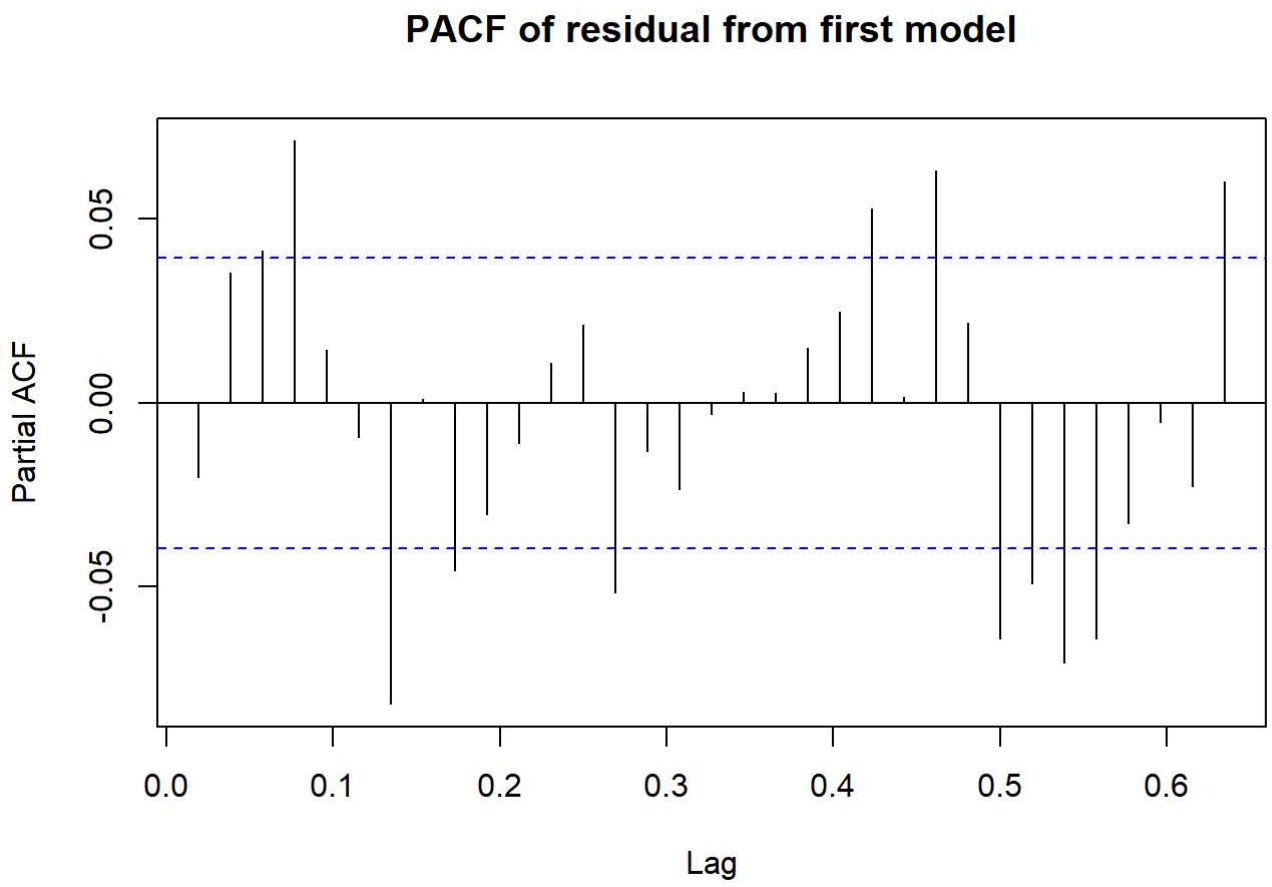
All 3 models seem to fit the data pretty well. We would need to look at the ACF and the PACF of all 3 models to see which one fits best.

c. Running 3 models: (1) ARMA(2,0) and S-ARMA(0,3) with frequency 6, (2) ARMA(3,0) and S-ARMA(0,3) with frequency 5, (3)ARMA(3,0) and S-ARMA(0,2) with frequency 6

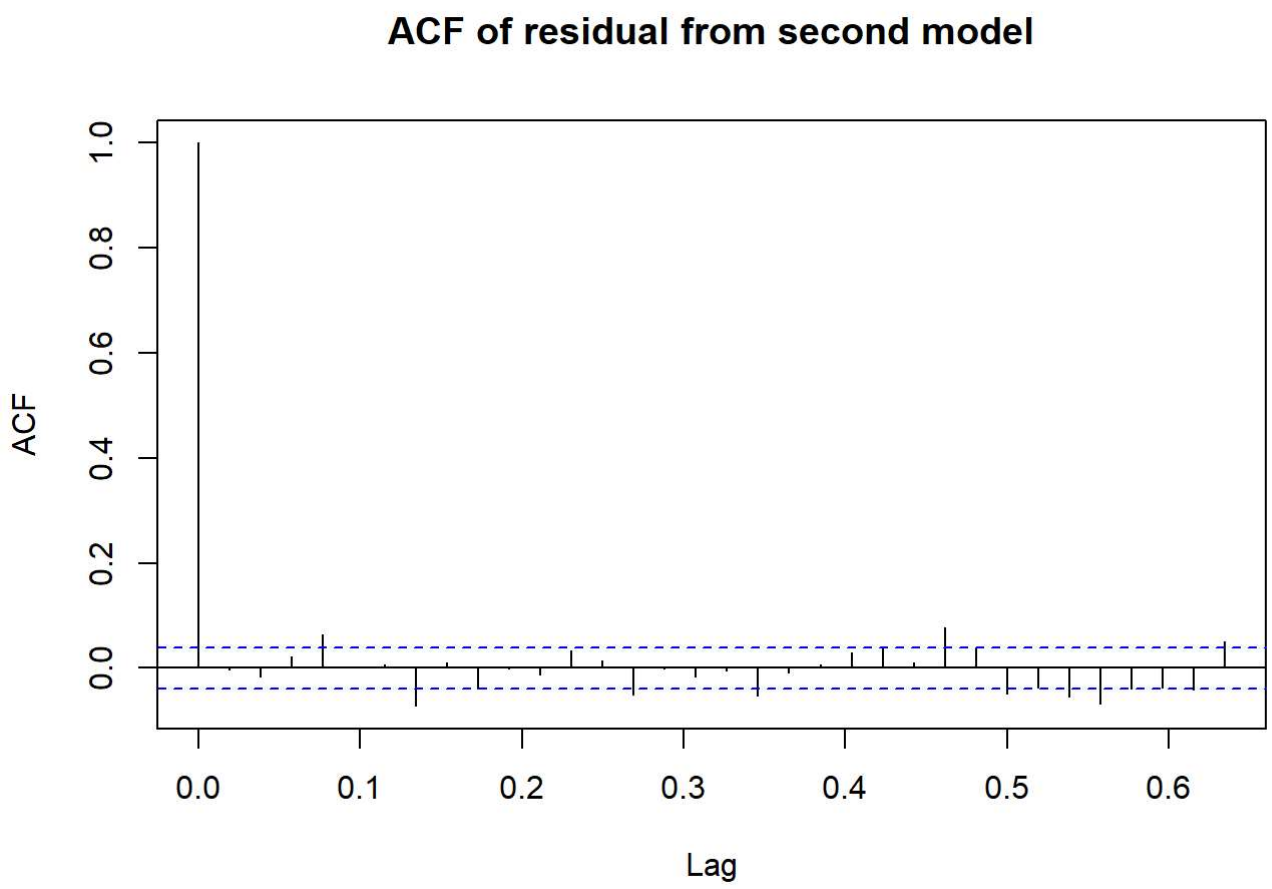
```
#Look at the ACF and the PACF of residuals from the models and run a box test
acf(arma1$residuals, main = "ACF of residual from first model")
```



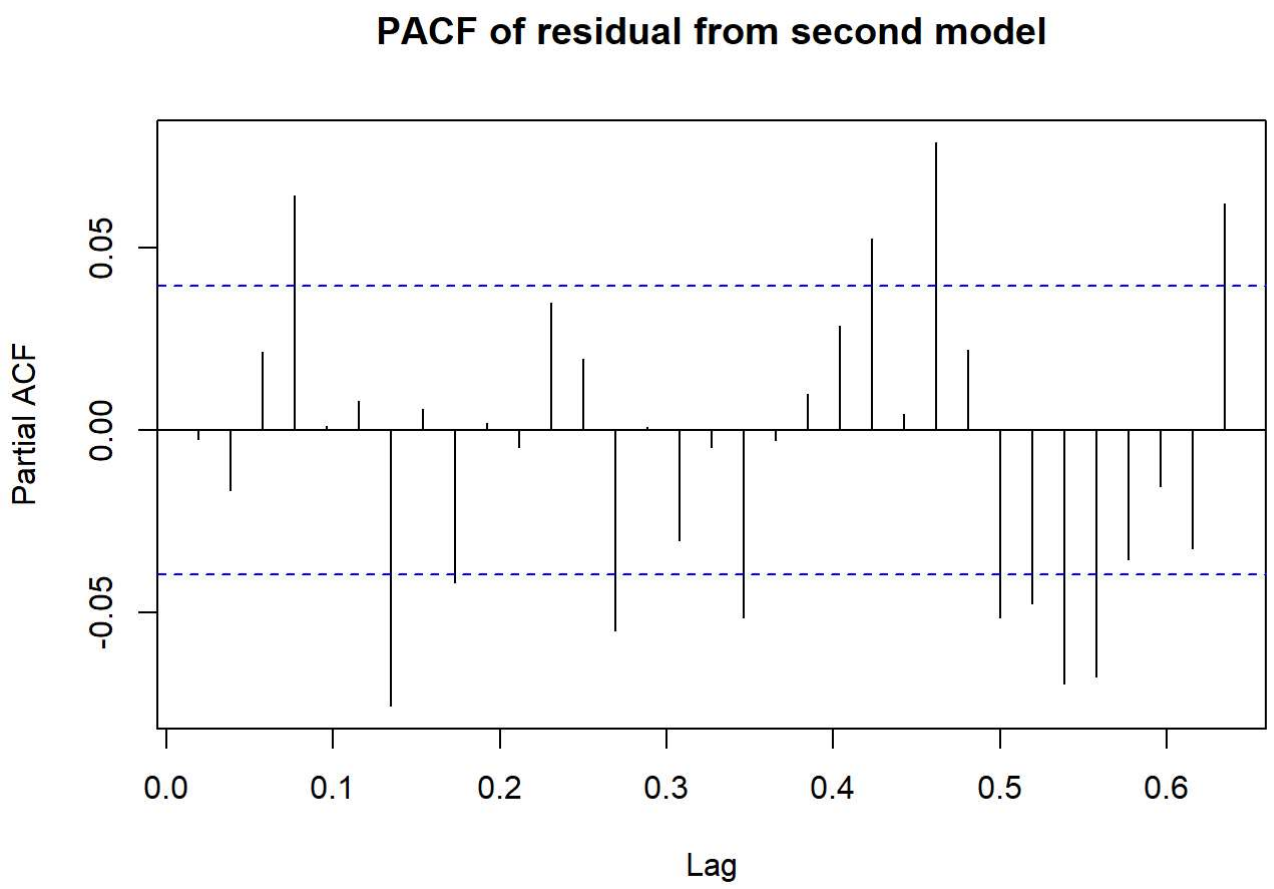

```
pacf(arma1$residuals, main = "PACF of residual from first model")
```



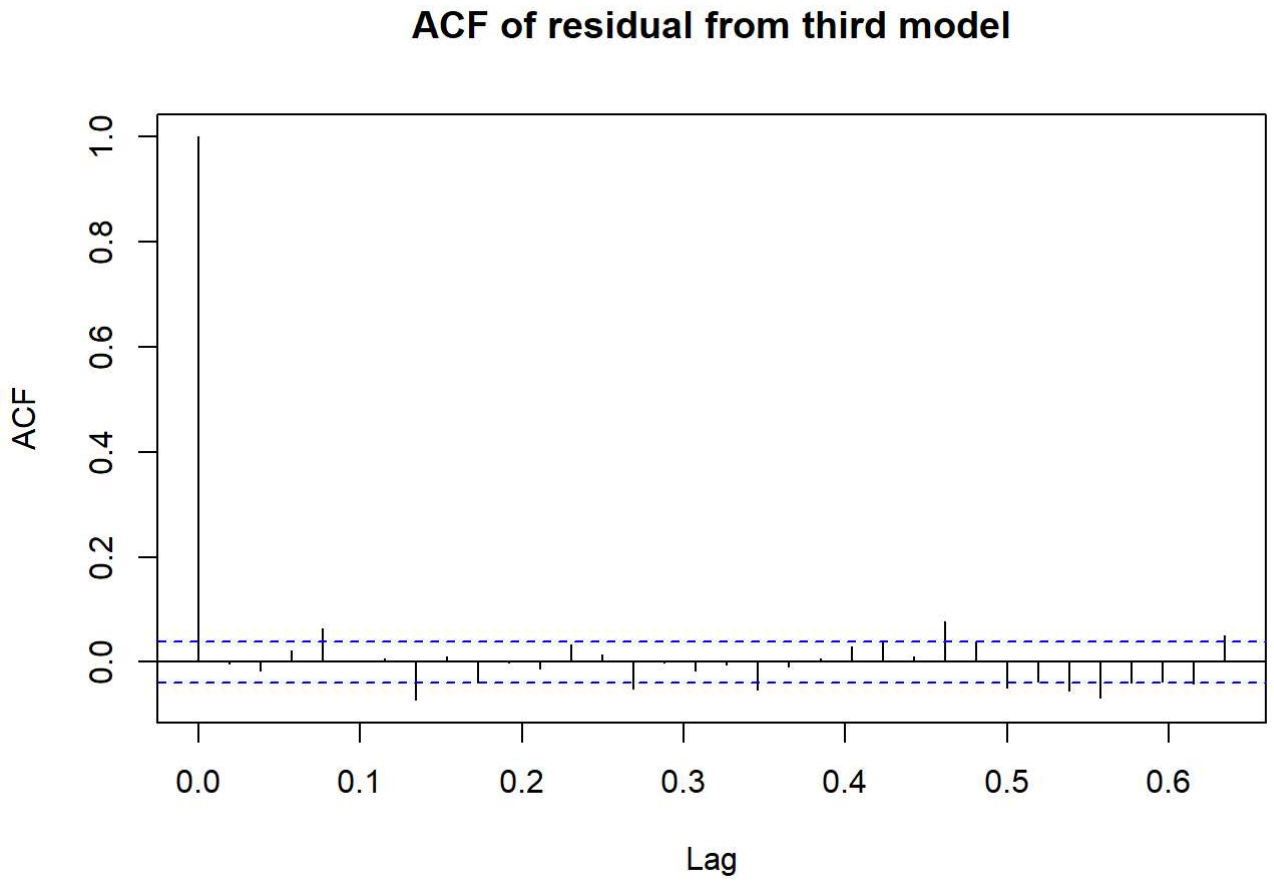

```
acf(arma2$residuals, main = "ACF of residual from second model")
```



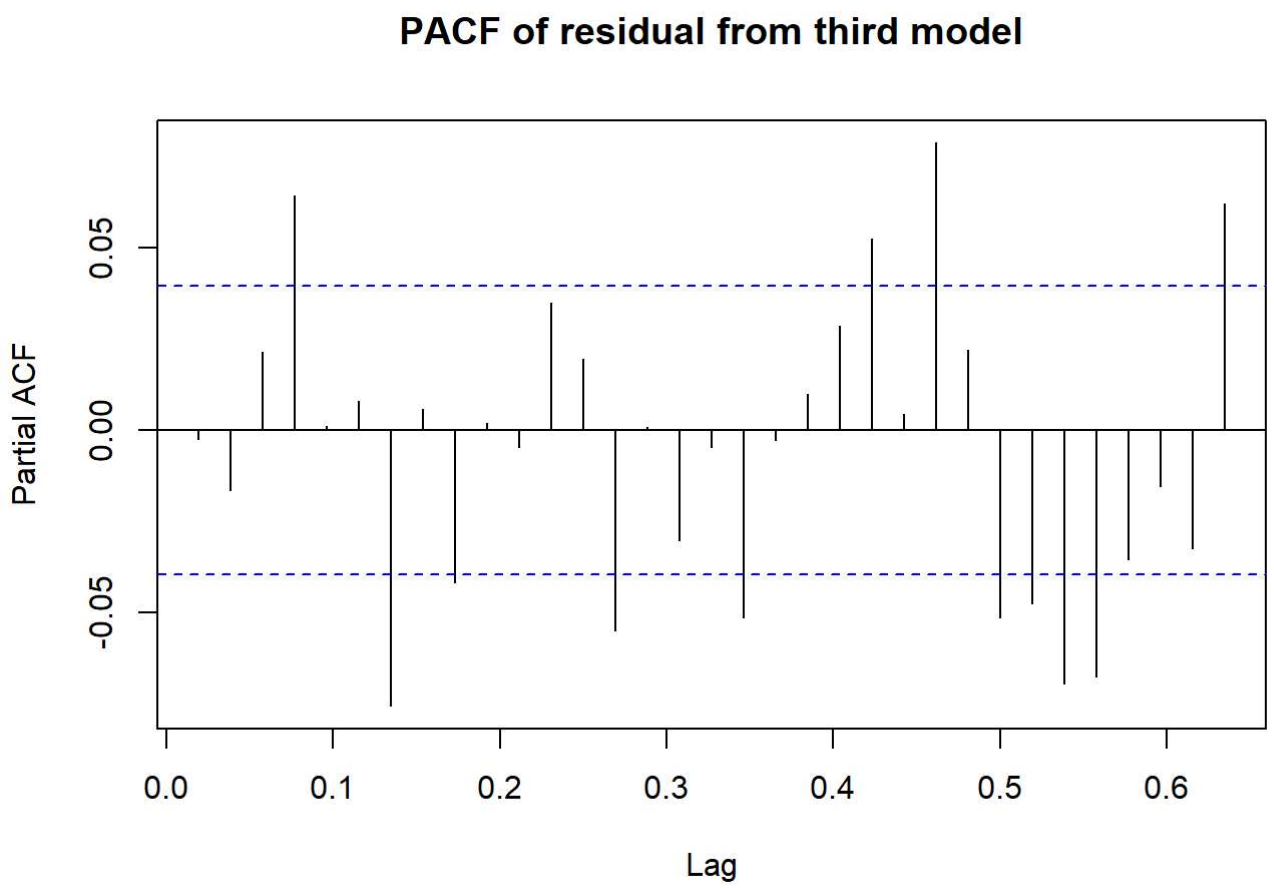

```
pacf(arma2$residuals, main = "PACF of residual from second model")
```




```
acf(arma2$residuals, main = "ACF of residual from third model")
```




```
pacf(arma2$residuals, main = "PACF of residual from third model")
```




```
Box.test(arma1$residuals, lag = 20)
```

```
##
##  Box-Pierce test
##
## data:  arma1$residuals
## X-squared = 53.484, df = 20, p-value = 6.897e-05
```

```
Box.test(arma2$residuals, lag = 20)
```

```
##
##  Box-Pierce test
##
## data:  arma2$residuals
## X-squared = 46.645, df = 20, p-value = 0.000657
```

```
Box.test(arma3$residuals, lag = 20)
```

```
##
##  Box-Pierce test
##
## data:  arma3$residuals
## X-squared = 49.224, df = 20, p-value = 0.0002857
```

All three models seem to fit the data well, with the residuals practically reduced to white noise looking at the ACF and the PACF as well as the Box-Pierce test. However, since by the Box-Pierce test, the first model has the lowest p-values, we'll take that model.

d.

```
library(strucchange)
```

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

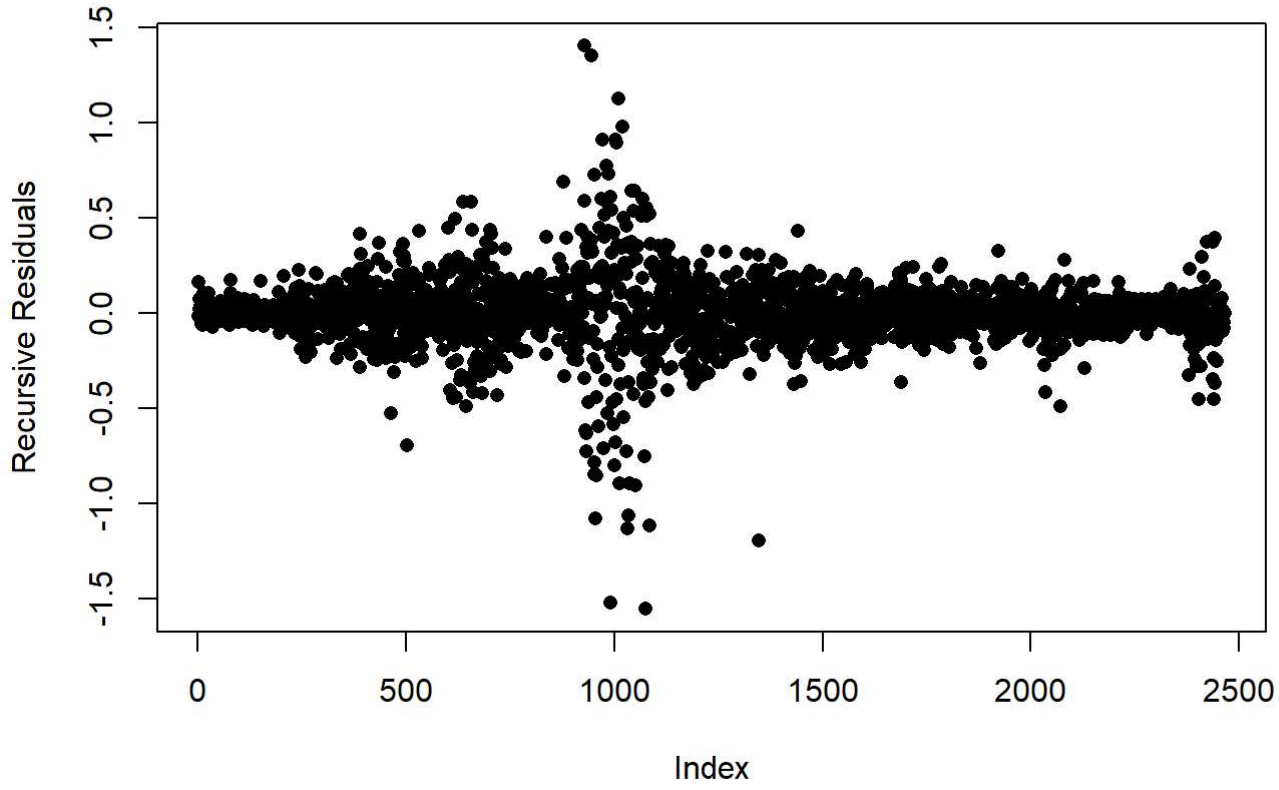
```
##
## Attaching package: 'zoo'
```

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

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

```
## Loading required package: sandwich
```

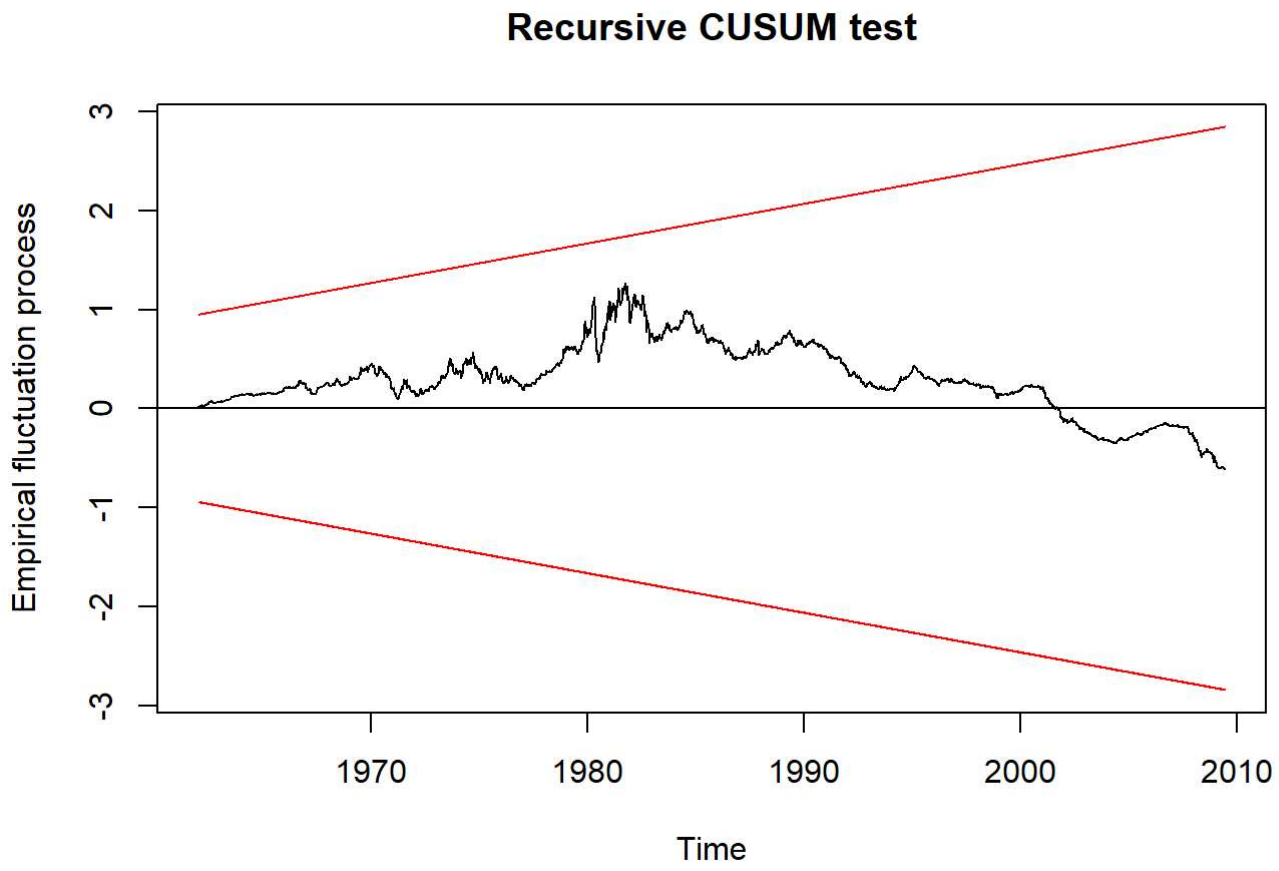
```
y=recresid(arma1$residuals~1)
plot(y, pch=16,ylab="Recursive Residuals")
```



The recursive residuals suggests that there might a structural break of my ARMA model predicting the data. This occurs at the middle of the years, maybe around the year 1982. But we need to be sure by looking at its cummulative sum.

e.

```
plot(efp(arma1$residuals~1, type = "Rec-CUSUM"))
```



The recursive sum of the recursive residuals do not suggest that there is a structural break, though there are irregularities in the middle. Nevertheless, the model is robust enough to predict the data.

d.

```
#to obtain the best fit model according to R, we use auto.arima function
armaR = auto.arima(ir_ts)
summary(armaR)
```

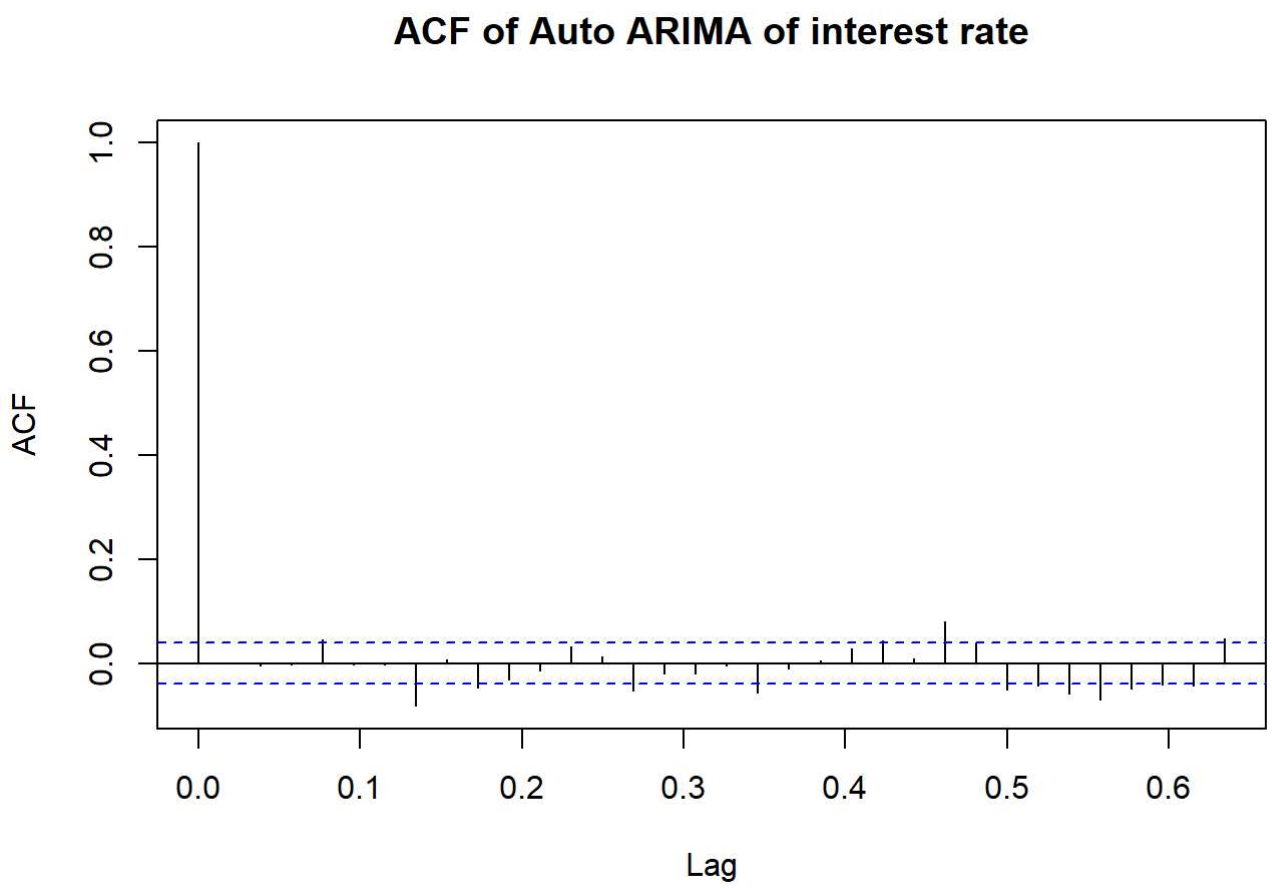
```
## Series: ir_ts
## ARIMA(1,1,2)
##
## Coefficients:
##          ar1      ma1      ma2
##       0.6284  -0.3065  -0.0527
## s.e.  0.0642   0.0675   0.0299
##
## sigma^2 estimated as 0.03143:  log likelihood=768.32
## AIC=-1528.65   AICc=-1528.63   BIC=-1505.41
##
## Training set error measures:
##                        ME      RMSE       MAE         MPE     MAPE
## Training set -0.0006210918 0.1771539 0.1046884 -0.05084758 1.820023
##                    MASE          ACF1
## Training set 0.07539674 -0.0002601686
```

```
summary(arma1)
```

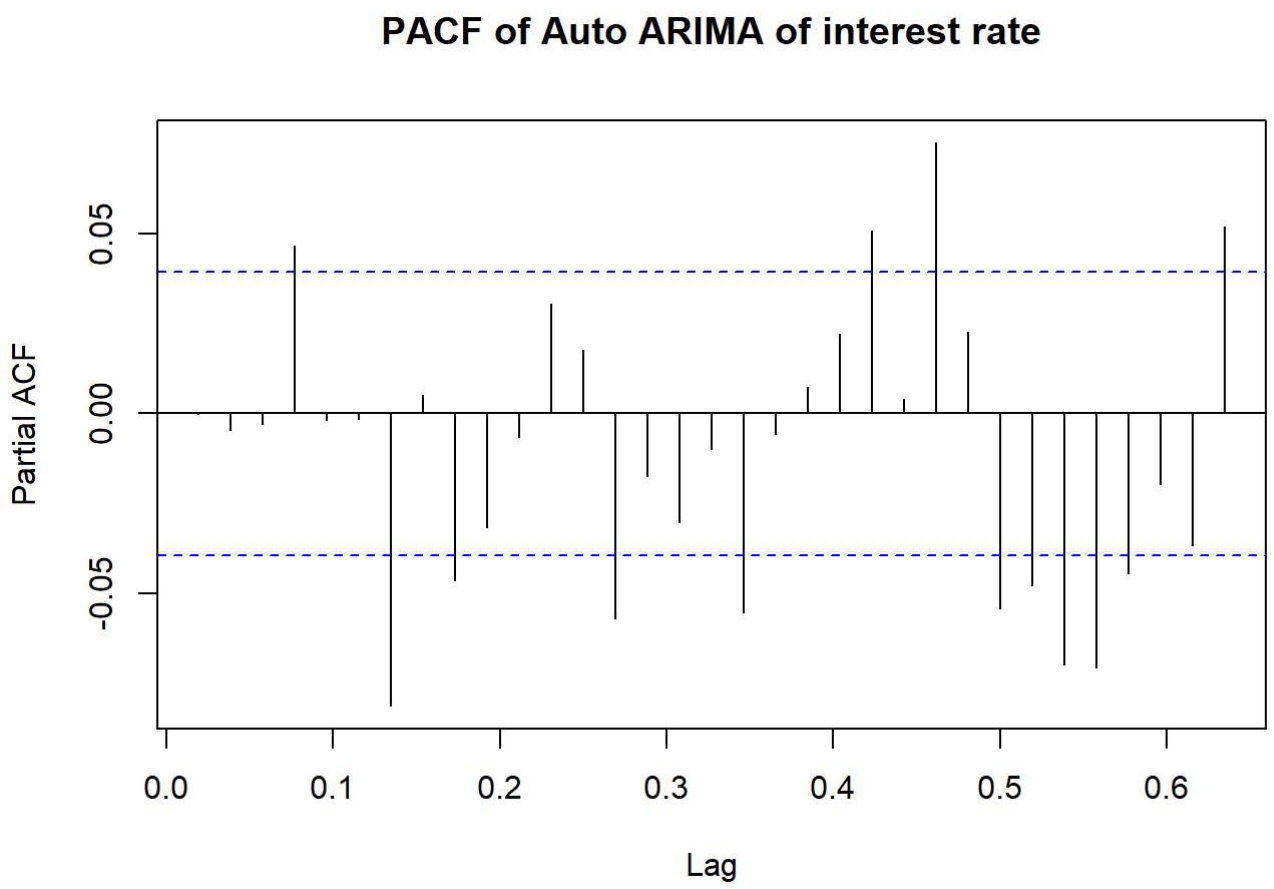
```
## Series: ir_ts
## ARIMA(2,0,0)(0,0,3)[6] with non-zero mean
##
## Coefficients:
##          ar1      ar2    sma1    sma2     sma3    mean
##       1.3448  -0.3472  0.0190  0.0201  -0.0485  6.0860
## s.e.  0.0189   0.0189  0.0204  0.0190   0.0194  1.3876
##
## sigma^2 estimated as 0.03148:  log likelihood=765.3
## AIC=-1516.6   AICc=-1516.55   BIC=-1475.92
##
## Training set error measures:
##                         ME      RMSE       MAE        MPE     MAPE
## Training set -0.0007431643 0.1772215 0.1049781 -0.1596363 1.833548
##                     MASE       ACF1
## Training set 0.07560533 -0.0202107
```

The auto ARIMA function from R provides us the ARIMA(1,1,2) model to fit the data. The AIC and the BIC seems to be similar for both the models. So we can't say much from these 2 measures about which one is a better model to fit the data. Thus, we want to look at the ACF and the PACF of the residuals as well as conduct a Box-Pierce test to look at time dependence.

```
acf(armaR$residuals, main = "ACF of Auto ARIMA of interest rate")
```




```
pacf(armaR$residuals, main = "PACF of Auto ARIMA of interest rate")
```




```
Box.test(armaR$residuals, lag = 20)
```

```
##
##  Box-Pierce test
##
## data:  armaR$residuals
## X-squared = 49.591, df = 20, p-value = 0.0002534
```

The ACF and PACF seems to have been reduced to that of a white noise, which means that the ARIMA(1,1,2) took care of all the dynamics in the data as well. The Box-Pierce seems to also conclude that result. However, from the ACF and PACF of my model seems to suggest an even less spikes in the plots, as well as an even lower P-value for the Box-Pierce. This sugggests that my model seems to be doing better than what 'R' suggested but the models seem to be comparable in fitting the data.

g.

```
#obtain a forecast object from each ARIMA model and then show its point forecast
arma1.forecast = forecast(arma1, h= 24)
arma1.forecast$mean
```

```
## Time Series:
## Start = 2009.45600632244
## End = 2009.89831401475
## Frequency = 52
##  [1] 0.6231658 0.6460586 0.6672275 0.6877722 0.7051344 0.7280438 0.7477210
##  [8] 0.7642885 0.7813322 0.7979600 0.8157029 0.8306220 0.8530309 0.8727072
## [15] 0.8959690 0.9188072 0.9393944 0.9589638 0.9780169 0.9968452 1.0155504
## [22] 1.0341681 1.0527110 1.0711835
```

```
armaR.forecast = forecast(armaR, h= 24)
armaR.forecast$mean
```

```
## Time Series:
## Start = 2009.45600632244
## End = 2009.89831401475
## Frequency = 52
##  [1] 0.6038364 0.6048191 0.6054366 0.6058246 0.6060685 0.6062217 0.6063180
##  [8] 0.6063785 0.6064165 0.6064404 0.6064555 0.6064649 0.6064708 0.6064746
## [15] 0.6064769 0.6064784 0.6064793 0.6064799 0.6064802 0.6064805 0.6064806
## [22] 0.6064807 0.6064808 0.6064808
```

We see that there seems to be a difference in the prediction. We take a difference between these 2 forecasted values to look at how different they are.

```
forecast.diff = arma1.forecast$mean - armaR.forecast$mean
forecast.diff
```

```
## Time Series:
## Start = 2009.45600632244
## End = 2009.89831401475
## Frequency = 52
##  [1] 0.01932942 0.04123949 0.06179092 0.08194758 0.09906596 0.12182211
##  [7] 0.14140294 0.15790998 0.17491566 0.19151959 0.20924748 0.22415715
## [13] 0.24656012 0.26623264 0.28949206 0.31232879 0.33291507 0.35248395
## [19] 0.37153667 0.39036471 0.40906976 0.42768741 0.44623022 0.46470271
```

At every period, my model forecasted higher values. Not only that, there also seems to be upward trend that my model is predicting, while the model given by 'R' seems to be fluctuating around 0.6.

Problem 2

Clear the previous environment and obtain the new data

```
require(openxlsx)
```

```
## Loading required package: openxlsx
```

```
rm(list = ls(all=T))
data = read.xlsx("Chapter8_Exercises_Data.xlsx", sheet = "Exercise 7")
names(data)[4:5] = c("SP500_returns", "FTSE_returns")
attach(data)
```

In order to see if FTSE returns can influence SP500 returns, we want to see if FTSE returns granger causes SP500 returns. But first we would need to clean the data since there are many NA's in between. In order to solve that problem, I would discard the data points for both returns if only one of the datapoints is NA

```
index = vector()
for (i in 1:length(SP500_returns)){
  if (is.na(SP500_returns[i]) || is.na(FTSE_returns[i]))
    index = append(index, i)
}

data = data[-index,]
attach(data)
```

```
## The following objects are masked from data (pos = 3):
##
##     FTSE_OPEN, FTSE_returns, OBS, SP500_OPEN, SP500_returns
```

After removing all the empty data points, we first find the order for the VAR model.

```
require("vars")
```

```
## Loading required package: vars
```

```
## Loading required package: MASS
```

```
## Loading required package: urca
```

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

```
require("VAR.etp")
```

```
## Loading required package: VAR.etp
```

```
SP500_ts = ts(SP500_returns, start = 1990, freq = 252)
FTSE_ts = ts(FTSE_returns, start = 1990, freq = 252)
y = cbind(SP500_ts, FTSE_ts)
y_tot=data.frame(y)
y_cri=VAR.select(y_tot, pmax = 10)
print(y_cri$p)
```

```
## [1] 8
```

From VAR.select function, we find that the optimal order is 8. So we select the order to be 8 and check Granger causality. But just in case, we check for Granger causality of FTSE returns on SP500 returns

```
for (i in 1:8){
print(grangertest(FTSE_returns ~ SP500_returns, order = i))
print(grangertest(SP500_returns~FTSE_returns , order = i))
}
```

```
## Granger causality test
##
## Model 1: FTSE_returns ~ Lags(FTSE_returns, 1:1) + Lags(SP500_returns, 1:1)
## Model 2: FTSE_returns ~ Lags(FTSE_returns, 1:1)
##   Res.Df Df      F    Pr(>F)
## 1   5348
## 2   5349 -1 516.64 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Granger causality test
##
## Model 1: SP500_returns ~ Lags(SP500_returns, 1:1) + Lags(FTSE_returns, 1:1)
## Model 2: SP500_returns ~ Lags(SP500_returns, 1:1)
##   Res.Df Df      F Pr(>F)
## 1   5348
## 2   5349 -1 1.0704 0.3009
## Granger causality test
##
## Model 1: FTSE_returns ~ Lags(FTSE_returns, 1:2) + Lags(SP500_returns, 1:2)
## Model 2: FTSE_returns ~ Lags(FTSE_returns, 1:2)
##   Res.Df Df      F    Pr(>F)
## 1   5345
## 2   5347 -2 265.68 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Granger causality test
##
## Model 1: SP500_returns ~ Lags(SP500_returns, 1:2) + Lags(FTSE_returns, 1:2)
## Model 2: SP500_returns ~ Lags(SP500_returns, 1:2)
##   Res.Df Df     F  Pr(>F)
## 1   5345
## 2   5347 -2 2.539 0.07904 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Granger causality test
##
## Model 1: FTSE_returns ~ Lags(FTSE_returns, 1:3) + Lags(SP500_returns, 1:3)
## Model 2: FTSE_returns ~ Lags(FTSE_returns, 1:3)
##   Res.Df Df      F    Pr(>F)
## 1   5342
## 2   5345 -3 179.81 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Granger causality test
##
## Model 1: SP500_returns ~ Lags(SP500_returns, 1:3) + Lags(FTSE_returns, 1:3)
## Model 2: SP500_returns ~ Lags(SP500_returns, 1:3)
##   Res.Df Df      F Pr(>F)
## 1   5342
## 2   5345 -3 1.6741 0.1703
## Granger causality test
##
## Model 1: FTSE_returns ~ Lags(FTSE_returns, 1:4) + Lags(SP500_returns, 1:4)
```

```
## Model 2: FTSE_returns ~ Lags(FTSE_returns, 1:4)
##   Res.Df Df      F    Pr(>F)
## 1   5339
## 2   5343 -4 133.74 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Granger causality test
##
## Model 1: SP500_returns ~ Lags(SP500_returns, 1:4) + Lags(FTSE_returns, 1:4)
## Model 2: SP500_returns ~ Lags(SP500_returns, 1:4)
##   Res.Df Df      F Pr(>F)
## 1   5339
## 2   5343 -4 1.4657 0.2098
## Granger causality test
##
## Model 1: FTSE_returns ~ Lags(FTSE_returns, 1:5) + Lags(SP500_returns, 1:5)
## Model 2: FTSE_returns ~ Lags(FTSE_returns, 1:5)
##   Res.Df Df      F    Pr(>F)
## 1   5336
## 2   5341 -5 106.71 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Granger causality test
##
## Model 1: SP500_returns ~ Lags(SP500_returns, 1:5) + Lags(FTSE_returns, 1:5)
## Model 2: SP500_returns ~ Lags(SP500_returns, 1:5)
##   Res.Df Df      F   Pr(>F)
## 1   5336
## 2   5341 -5 3.3913 0.004626 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Granger causality test
##
## Model 1: FTSE_returns ~ Lags(FTSE_returns, 1:6) + Lags(SP500_returns, 1:6)
## Model 2: FTSE_returns ~ Lags(FTSE_returns, 1:6)
##   Res.Df Df      F    Pr(>F)
## 1   5333
## 2   5339 -6 88.104 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Granger causality test
##
## Model 1: SP500_returns ~ Lags(SP500_returns, 1:6) + Lags(FTSE_returns, 1:6)
## Model 2: SP500_returns ~ Lags(SP500_returns, 1:6)
##   Res.Df Df      F   Pr(>F)
## 1   5333
## 2   5339 -6 3.0756 0.005245 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Granger causality test
##
## Model 1: FTSE_returns ~ Lags(FTSE_returns, 1:7) + Lags(SP500_returns, 1:7)
## Model 2: FTSE_returns ~ Lags(FTSE_returns, 1:7)
##   Res.Df Df      F    Pr(>F)
```

```
## 1   5330
## 2   5337 -7 75.635 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Granger causality test
##
## Model 1: SP500_returns ~ Lags(SP500_returns, 1:7) + Lags(FTSE_returns, 1:7)
## Model 2: SP500_returns ~ Lags(SP500_returns, 1:7)
##   Res.Df Df      F    Pr(>F)
## 1   5330
## 2   5337 -7 4.5631 4.309e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Granger causality test
##
## Model 1: FTSE_returns ~ Lags(FTSE_returns, 1:8) + Lags(SP500_returns, 1:8)
## Model 2: FTSE_returns ~ Lags(FTSE_returns, 1:8)
##   Res.Df Df      F    Pr(>F)
## 1   5327
## 2   5335 -8 65.965 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Granger causality test
##
## Model 1: SP500_returns ~ Lags(SP500_returns, 1:8) + Lags(FTSE_returns, 1:8)
## Model 2: SP500_returns ~ Lags(SP500_returns, 1:8)
##   Res.Df Df      F    Pr(>F)
## 1   5327
## 2   5335 -8 3.5615 0.0003987 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the Granger causality test, we could see for sure that SP500 returns does influence FTSE returns, while FTSE returns affect SP500 returns as the number of lags is more than 4. Maybe because as time goes by, the eventual shock on the FTSE market would eventually linger to the SP500. Thus, we can use SP500 returns to forecase FTSE returns, and also the other way around for lags more than 4.

Problem 3

Clear the previous environment and add in the new data

```
rm(list = ls(all=T))
data = read.xlsx("Chapter11_exercises_data.xlsx")
data = data[,-8]
attach(data)
```

Construct 4 VAR models, which are of GSF and GSJ(VAR_1), GSF and GAL(VAR_2), GSj and GAL(VAR_3), and GSF, GSJ and GAL together(VAR_4).(All these 3 symbols are quarterly house price growth rate based on original Freddie Mac House Price Indices of San Francisco-Oakland-Freemont, San Jose-Sunnyvale-Santa Clara, Albany-Schenectady-Oakland-Freemont respectively)

```
GSF_ts = ts(GSF[complete.cases(GSF)], start = 1957, freq = 4)
GSJ_ts = ts(GSJ[complete.cases(GSJ)], start = 1957, freq = 4)
GAL_ts = ts(GAL[complete.cases(GAL)], start = 1957, freq = 4)

y1 = cbind(GSF_ts, GSJ_ts)
y2 = cbind(GSF_ts, GAL_ts)
y3 = cbind(GSJ_ts, GAL_ts)
y4 = cbind(GSF_ts, GSJ_ts, GAL_ts)

y_tot1=data.frame(y1)
y_cri1=VAR.select(y_tot1, pmax = 20)
y_tot2=data.frame(y2)
y_cri2=VAR.select(y_tot2, pmax = 10)
y_tot3=data.frame(y3)
y_cri3=VAR.select(y_tot3, pmax = 10)
y_tot4=data.frame(y4)
y_cri4=VAR.select(y_tot4, pmax = 10)

print(y_cri1$p)
```

```
## [1] 3
```

```
print(y_cri2$p)
```

```
## [1] 7
```

```
print(y_cri3$p)
```

```
## [1] 7
```

```
print(y_cri4$p)
```

```
## [1] 3
```

This shows that the order to use for VAR_1, VAR_2, VAR_3 and VAR_4 are 3, 7, 7 and 3 respectively.

```
VAR_1 = VAR(y_tot1, p = 3)
VAR_2 = VAR(y_tot2, p = 7)
VAR_3 = VAR(y_tot3, p = 7)
VAR_4 = VAR(y_tot4, p = 3)

VAR_1.predict = predict(object=VAR_1, n.ahead=1)
VAR_2.predict = predict(object=VAR_2, n.ahead=1)
VAR_3.predict = predict(object=VAR_3, n.ahead=1)
VAR_4.predict = predict(object=VAR_4, n.ahead=1)

print(VAR_1.predict$fcst)
```

```
## $GSF_ts
##                   fcst     lower    upper       CI
## GSF_ts.fcst -2.102955 -6.075083 1.869172 3.972127
##
## $GSJ_ts
##                   fcst     lower    upper      CI
## GSJ_ts.fcst -2.105309 -5.317858 1.107241 3.21255
```

```
print(VAR_2.predict$fcst)
```

```
## $GSF_ts
##                   fcst    lower    upper       CI
## GSF_ts.fcst -0.989752 -4.98998 3.010476 4.000228
##
## $GAL_ts
##                    fcst     lower    upper       CI
## GAL_ts.fcst -0.7083885 -4.544204 3.127426 3.835815
```

```
print(VAR_3.predict$fcst)
```

```
## $GSJ_ts
##                    fcst     lower    upper       CI
## GSJ_ts.fcst -0.3327676 -3.481943 2.816408 3.149175
##
## $GAL_ts
##                    fcst     lower    upper      CI
## GAL_ts.fcst -0.2545135 -4.175454 3.666427 3.92094
```

```
print(VAR_4.predict$fcst)
```

```
## $GSF_ts
##                   fcst     lower    upper       CI
## GSF_ts.fcst -2.126521 -6.058267 1.805225 3.931746
##
## $GSJ_ts
##                   fcst     lower    upper       CI
## GSJ_ts.fcst -2.066979 -5.244068 1.110111 3.177089
##
## $GAL_ts
##                   fcst     lower    upper       CI
## GAL_ts.fcst -1.074373 -5.133941 2.985194 4.059568
```

We see that with or without adding the GAL, the forecasted values of GSF and GSJ are similar. That implies that since Albany-Schenectady-Oakland-Freemont is so far from the other 2, there is no significant effect on the forecast.