W271 - Assignment3

Chandra Shekar Bikkanur

11/22/2019

```r
# Load required libraries
library(car)
library(dplyr)
library(Hmisc)
library(ggplot2)
library(ggfortify)
library(plotly)
library(astsa)
library(forecast)
library(fpp2)
library(GGally)
library("ggpubr")
library(gridExtra)
library(grid)
library(xts)
library(tidyverse)
library(xts)
library(vars)
library(zoo)
library(tseries)
```

Question 1 (1 point):

Backshift Operator Expression

a. Write down the $ARIMA(2,0,2)(1,0,1)_4$ in terms of (1) backshift operators and (2) the fully-expressed form as y_t as a function of lags of y_t and the shock ω_t.

a.1.

$$\Theta_1(B^4)\theta_2(B)y_t = \Phi_1(B^4)\phi_2(B)\omega_t$$

a.2.

$$y_t = \theta_1 y_{t-1} + \theta_2 y_{t-2} + \Theta_1 y_{t-4} + \Theta_2 y_{t-5} + \omega_t - \Phi_1\omega_{t-4} - \phi_1\omega_{t-1} - \phi_2\omega_{t-2}$$

b. Write down the $ARIMA(2,1,2)(1,0,1)_4$ in terms of (1) backshift operators and (2) the fully-expressed form.

b.1.

$$\Theta_1(B^4)\theta_2(B)(1-B)y_t = \Phi_1(B^4)\phi_2(B)\omega_t$$

b.2.

$$y_t = y_{t-1} + \theta_1 y_{t-1} + \theta_2 y_{t-2} + \Theta_1 y_{t-4} + \Theta_2 y_{t-5} + \omega_t - \Phi_1 \omega_{t-4} - \phi_1 \omega_{t-1} - \phi_2 \omega_{t-2}$$

c. Write down the $ARIMA(2,1,2)(1,1,1)_4$ in terms of (1) backshift operators and (2) the fully-expressed form.

c.1.

$$\Theta_1(B^4)\theta_2(B)(1-B^4)(1-B)y_t = \Phi_1(B^4)\phi_2(B)\omega_t$$

c.2.

$$y_t = y_{t-1} + y_{t-4} + \theta_1 y_{t-1} + \theta_2 y_{t-2} + \Theta_1 y_{t-4} + \Theta_2 y_{t-5} + \omega_t - \Phi_1 \omega_{t-4} - \phi_1 \omega_{t-1} - \phi_2 \omega_{t-2}$$

Parameter Redundancy, Stationarity, and Invertibility

In each of the following cases, (1) check for parameter redundancy and ensure that the $ARMA(p,q)$ notation is expressed in the simplest form, and (2) determine whether they are stationary and/or invertible.

a. $y_t = y_{t-1} - \frac{1}{4}y_{t-2} + \omega_t + \frac{1}{2}\omega_{t-1}$

$$y_t - y_{t-1} + \frac{1}{4}y_{t-2} = \omega_t + \frac{1}{2}\omega_{t-1}$$

$$(1 - B + \frac{1}{4}B^2)y_t = (1 + \frac{1}{2}B)\omega_t$$

$$\frac{(4 - 4B + B^2)}{4}y_t = \frac{(2+B)}{2}\omega_t$$
$$(B-2)^2 = 0||||(2+B) = 0$$
$$(B=2)||||B = -2$$

Here since the roots of θ and ϕ are above unity in absolute value, the process is both stationary and invertible

b. $y_t = 2y_{t-1} - y_{t-2} + \omega_t$

$$y_t - 2y_{t-1} + y_{t-2} = \omega_t$$

$$(1 - 2B + B^2)y_t = \omega_t$$

$$(B-1)^2 y_t = \omega_t$$
$$(B-1)^2 = 0 \implies B = 1$$

Here since the root of θ is 1, the process is non-stationary

c. $y_t = \frac{7}{10}y_{t-1} - \frac{1}{10}y_{t-2} + \omega_t + \frac{3}{2}\omega_{t-1}$

$$y_t - \frac{7}{10}y_{t-1} + \frac{1}{10}y_{t-2} = \omega_t + \frac{3}{2}\omega_{t-1}$$

$$(1 - \frac{7}{10}B + \frac{1}{10}B^2)y_t = (1 + \frac{3}{2}B)\omega_t$$
$$\frac{(B^2 - 7B + 10)}{10}y_t = (\frac{2 + 3B}{2})\omega_t$$
$$(B^2 - 7B + 10) = 0||||B = \frac{-2}{3}$$

$$B = (1, 5)||||B = -0.666$$

Here since the roots of θ has a value 1 and ϕ has the value below unity in absolute value, the process is neither stationary nor invertible

d. $y_t = \frac{7}{10}y_{t-1} - \frac{1}{10}y_{t-2} + \omega_t + \frac{1}{3}\omega_{t-1}$

$$y_t - \frac{7}{10}y_{t-1} + \frac{1}{10}y_{t-2} = \omega_t + \frac{1}{3}\omega_{t-1}$$

$$(1 - \frac{7}{10}B + \frac{1}{10}B^2)y_t = (1 + \frac{1}{3}B)\omega_t$$
$$\frac{(B^2 - 7B + 10)}{10}y_t = (\frac{3 + B}{3})\omega_t$$
$$(B^2 - 7B + 10) = 0||||3 + B = 0$$

$$B = (1, 5)||||B = -3$$

Here since the roots of θ has a value 1 and ϕ has the value above unity in absolute value, the process is not stationary but invertible

Question 2: Time series merging and interpolation (2 points)

The file `AMZN.csv` contains Amazon share price data obtained from Yahoo! Finance. The file `UMCSENT.csv` contains the University of Michigan Consumer Sentiment index obtained from the Federal Reserve Economic Database (FRED).

Read `AMZN.csv` and `UMCSENT.csv` into R as dataframes and convert them to time series objects. You may find it advantageous to work with `xts` rather than `ts` objects for the following questions (refer to `xts.Rmd` in the github repo's `live_session_7` folder).

a. Merge the two set of series together, preserving all of the observations in both set of series

b. Fill all of the missing values of the UMCSENT series with -9999

c. Create the following new series from the original UMCSENT series:

   - UMCSENT02, replacing all of the -9999 with NAs
   - UMCSENT03, replacing the NAs with the last observation
   - UMCSENT04, replacing the NAs using linear interpolation

   Print a few observations to ensure that your merge as well as the missing value imputation are done correctly. Choose a reasonable number of observations (do not print out the entire dataset).

d. Calculate the daily return of the Amazon closing price, where daily return is defined as $(x_t - x_{t-1})/x_{t-1}$. Plot the daily return series.

e. Create a 20-day and a 50-day rolling mean series from the AMZN close series.

EDA on AMAZON time-series data

```r
df_amzn <- read.csv("AMZN.csv", header = TRUE, stringsAsFactors = FALSE)
idx_amzn <- as.Date(df_amzn$Date, format = "%d/%m/%y")
xts_amzn <- xts(df_amzn[, 2:7], order.by = idx_amzn)
str(xts_amzn)
```

```
## An 'xts' object on 1997-05-15/2019-10-31 containing:
##   Data: num [1:5654, 1:6] 2.44 1.97 1.76 1.73 1.64 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:6] "Open" "High" "Low" "Close" ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
```

```r
head(xts_amzn, 4)
```

```
##                Open     High      Low    Close Adj.Close   Volume
## 1997-05-15 2.437500 2.500000 1.927083 1.958333  1.958333 72156000
## 1997-05-16 1.968750 1.979167 1.708333 1.729167  1.729167 14700000
## 1997-05-19 1.760417 1.770833 1.625000 1.708333  1.708333  6106800
## 1997-05-20 1.729167 1.750000 1.635417 1.635417  1.635417  5467200
```

```r
tail(xts_amzn, 4)
```

```
##               Open    High     Low   Close Adj.Close    Volume
## 2019-10-28 1748.06 1778.70 1742.50 1777.08   1777.08   3708900
## 2019-10-29 1774.81 1777.00 1755.81 1762.71   1762.71   2273700
## 2019-10-30 1760.24 1782.38 1759.12 1779.99   1779.99   2442400
## 2019-10-31 1775.99 1792.00 1771.48 1776.66   1776.66 277910000
```

EDA on UMCSENT time-series data

```r
df_umcsent <- read.csv("UMCSENT.csv", header = TRUE, stringsAsFactors =FALSE)
df_umcsent$UMCSENT <- as.numeric(df_umcsent$UMCSENT)
idx_umcsent <- as.Date(df_umcsent$DATE, format = "%Y-%m-%d")
xts_umcsent <- xts(df_umcsent[, 2], order.by = idx_umcsent)
str(xts_umcsent)
```

```
## An 'xts' object on 1952-11-01/2019-09-01 containing:
##   Data: num [1:803, 1] 86.2 NA NA 90.7 NA NA NA NA NA 80.8 ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
```

```r
head(xts_umcsent, 4)
```

```
##            [,1]
## 1952-11-01 86.2
## 1952-12-01   NA
## 1953-01-01   NA
## 1953-02-01 90.7
```

```r
tail(xts_umcsent, 4)
```

```
##            [,1]
## 2019-06-01 98.2
## 2019-07-01 98.4
## 2019-08-01 89.8
## 2019-09-01 93.2
```

2.a. Merge the two set of series together, preserving all of the observations in both set of series

We will be using the “outer” join method of xts package for merging Amazon data and UMCSENT data

```r
amzn_umcsent_xts <- merge(xts_amzn, xts_umcsent, join = "outer")
head(amzn_umcsent_xts)
```

```
##            Open High Low Close Adj.Close Volume xts_umcsent
## 1952-11-01   NA   NA  NA    NA        NA     NA        86.2
## 1952-12-01   NA   NA  NA    NA        NA     NA          NA
## 1953-01-01   NA   NA  NA    NA        NA     NA          NA
## 1953-02-01   NA   NA  NA    NA        NA     NA        90.7
## 1953-03-01   NA   NA  NA    NA        NA     NA          NA
## 1953-04-01   NA   NA  NA    NA        NA     NA          NA
```

```r
tail(amzn_umcsent_xts)
```

```
##               Open    High     Low   Close Adj.Close    Volume xts_umcsent
## 2019-10-24 1771.09 1788.34 1760.27 1780.78   1780.78   4446100          NA
## 2019-10-25 1697.55 1764.21 1695.00 1761.33   1761.33   9594600          NA
## 2019-10-28 1748.06 1778.70 1742.50 1777.08   1777.08   3708900          NA
## 2019-10-29 1774.81 1777.00 1755.81 1762.71   1762.71   2273700          NA
## 2019-10-30 1760.24 1782.38 1759.12 1779.99   1779.99   2442400          NA
## 2019-10-31 1775.99 1792.00 1771.48 1776.66   1776.66 277910000          NA
```

2.b. Fill all of the missing values of the UMCSENT series with -9999

```r
xts_umcsent[is.na(xts_umcsent)] <- -9999
head(xts_umcsent, 10)
```

```
##               [,1]
## 1952-11-01    86.2
## 1952-12-01 -9999.0
## 1953-01-01 -9999.0
## 1953-02-01    90.7
## 1953-03-01 -9999.0
## 1953-04-01 -9999.0
## 1953-05-01 -9999.0
## 1953-06-01 -9999.0
## 1953-07-01 -9999.0
## 1953-08-01    80.8
```

2.c. Create the following new series from the original UMCSENT series:

2.c.i. UMCSENT02, replacing all of the -9999 with NAs

```r
UMCSENT02 <- xts_umcsent
UMCSENT02[UMCSENT02 <= -9998] <- NA
cbind(head(xts_umcsent, 4), head(UMCSENT02,4))
```

```
##            head.xts_umcsent..4. head.UMCSENT02..4.
## 1952-11-01                 86.2               86.2
## 1952-12-01              -9999.0                 NA
## 1953-01-01              -9999.0                 NA
## 1953-02-01                 90.7               90.7
```

2.c. ii. UMCSENT03, replacing the NAs with the last observation

```
UMCSENT03 <- na.locf(UMCSENT02, na.rm = TRUE, fromLast = FALSE)
cbind(head(xts_umcsent, 6), head(UMCSENT02,6), head(UMCSENT03,6))
```

```
##            head.xts_umcsent..6. head.UMCSENT02..6. head.UMCSENT03..6.
## 1952-11-01                 86.2               86.2               86.2
## 1952-12-01              -9999.0                 NA               86.2
## 1953-01-01              -9999.0                 NA               86.2
## 1953-02-01                 90.7               90.7               90.7
## 1953-03-01              -9999.0                 NA               90.7
## 1953-04-01              -9999.0                 NA               90.7
```

2.c. iii. UMCSENT04, replacing the NAs using linear interpolation

```
UMCSENT04 <- UMCSENT02
UMCSENT04 <- na.approx(UMCSENT04, maxgap = 31)
cbind(head(xts_umcsent, 6), head(UMCSENT02,6),  head(UMCSENT03,6), head(UMCSENT04,6))
```

```
##            head.xts_umcsent..6. head.UMCSENT02..6. head.UMCSENT03..6.
## 1952-11-01                 86.2               86.2               86.2
## 1952-12-01              -9999.0                 NA               86.2
## 1953-01-01              -9999.0                 NA               86.2
## 1953-02-01                 90.7               90.7               90.7
## 1953-03-01              -9999.0                 NA               90.7
## 1953-04-01              -9999.0                 NA               90.7
##            head.UMCSENT04..6.
## 1952-11-01           86.20000
## 1952-12-01           87.66739
## 1953-01-01           89.18370
## 1953-02-01           90.70000
## 1953-03-01           89.16851
## 1953-04-01           87.47293
```

2.d. Calculate the daily return of the Amazon closing price, where daily return is defined as $(x_t - x_{t-1})/x_{t-1}$. Plot the daily return series.

```
df_amzn1 <- df_amzn[1:5653, 'Close']
df_amzn2 <- df_amzn[2:5654, 'Close']
df_amzn3 <- (df_amzn2 - df_amzn1)/df_amzn1
amzn3_ts <- as.ts(df_amzn3)
cbind(head(df_amzn1, 6),head(df_amzn2, 6),head(df_amzn3, 6))
```

```
##          [,1]     [,2]        [,3]
## [1,] 1.958333 1.729167 -0.11702096
## [2,] 1.729167 1.708333 -0.01204858
## [3,] 1.708333 1.635417 -0.04268254
## [4,] 1.635417 1.427083 -0.12738892
## [5,] 1.427083 1.395833 -0.02189782
## [6,] 1.395833 1.500000  0.07462712
```

```r
cbind(tail(df_amzn1, 6),tail(df_amzn2, 6),tail(df_amzn3, 6))
```

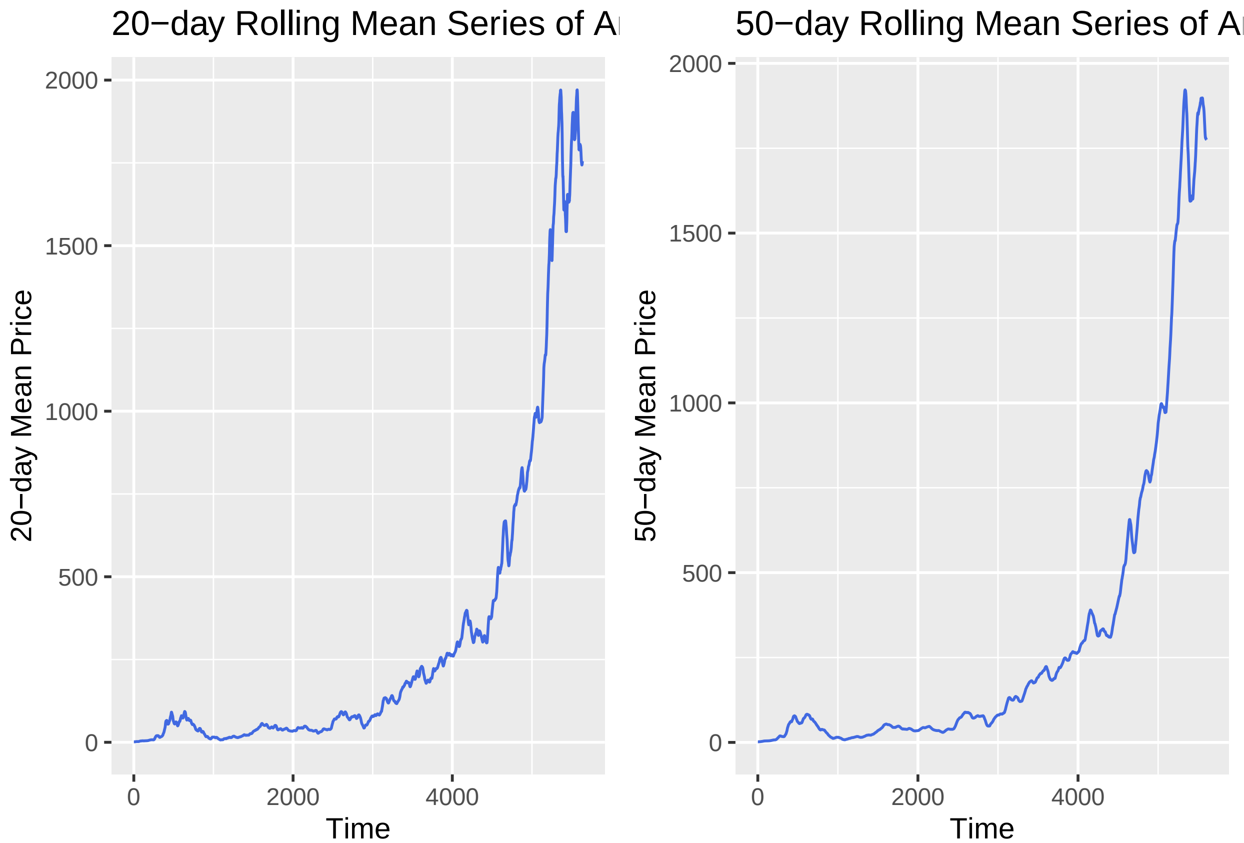
```
##         [,1]    [,2]         [,3]
## [1,] 1762.17 1780.78  0.010560834
## [2,] 1780.78 1761.33 -0.010922221
## [3,] 1761.33 1777.08  0.008942106
## [4,] 1777.08 1762.71 -0.008086296
## [5,] 1762.71 1779.99  0.009803104
## [6,] 1779.99 1776.66 -0.001870772
```

```r
ggplot(amzn3_ts, aes(x = time(amzn3_ts), y = amzn3_ts)) +
  geom_line(colour = "royalblue") +
  ggtitle("Daily Retun on Closing Price") +
  xlab("Time") + ylab("Relative change in Closing Price")
```

2.e. Create a 20-day and a 50-day rolling mean series from the AMZN close series.

```r
amzn_20day <- list()
for (i in 1:(nrow(df_amzn)-20)){
  amzn_20day_avg <- df_amzn[i:(i+20), 'Close']
  amzn_20day_avg <- mean(amzn_20day_avg)
  amzn_20day <- c(amzn_20day, amzn_20day_avg)
```

```
}

amzn_20day.df = data.frame(Reduce(rbind, amzn_20day),row.names = NULL)
amzn_20day_ts <- as.ts(amzn_20day.df)
amzn_20day_plot <- ggplot(amzn_20day_ts, aes(x = time(amzn_20day_ts), y = amzn_20day_ts)) +
  geom_line(colour = "royalblue") +
  ggtitle("20-day Rolling Mean Series of Amazon Closing Price") +
  xlab("Time") + ylab("20-day Mean Price")


amzn_50day <- list()
for (i in 1:(nrow(df_amzn)-50)){
  amzn_50day_avg <- df_amzn[i:(i+50), 'Close']
  amzn_50day_avg <- mean(amzn_50day_avg)
  amzn_50day <- c(amzn_50day, amzn_50day_avg)
}

amzn_50day.df = data.frame(Reduce(rbind, amzn_50day),row.names = NULL)
amzn_50day_ts <- as.ts(amzn_50day.df)
amzn_50day_plot <- ggplot(amzn_50day_ts, aes(x = time(amzn_50day_ts), y = amzn_50day_ts)) +
  geom_line(colour = "royalblue") +
  ggtitle("50-day Rolling Mean Series of Amazon Closing Price") +
  xlab("Time") + ylab("50-day Mean Price")

theme_set(theme_gray())
ggarrange(amzn_20day_plot, amzn_50day_plot, ncol = 2, nrow = 1)
```


20−day Rolling Mean Series of A
20−day Mean Price
2000
1500
1000
500
0
0
2000
4000
Time
50−day Rolling Mean Series of A
50−day Mean Price
2000
1500
1000
500
0
0
2000
4000
Time

Question 3: Atmospheric CO2 Concentration (4 points)

The file `mauna_loa.csv` contains weekly observations of atmospheric carbon dioxide concentration measured at the Mauna Loa observatory in Hawaii, obtained from the National Oceanic and Atmospheric Administration (NOAA), dating from 1974 to 2019.

a. Conduct a thorough EDA of the time series and develop a model that captures both trend and seasonality in the series, following all appropriate steps and conducting suitable diagnostics. Use the model to generate a 2-year-ahead forecast and plot this. In what year does your model predict that atmospheric CO2 will first reach 420 parts per million?

EDA on MAUNA_LOA time-series data

```r
df_noaa <- read.csv("mauna_loa.csv", header = TRUE, stringsAsFactors =FALSE)
df_noaa$Date <-  paste(df_noaa$day, df_noaa$mon, df_noaa$ï..yr, sep="/")
idx_noaa <- as.Date(df_noaa$Date, format = "%d/%m/%Y")
xts_noaa <- xts(df_noaa$CO2.ppm, order.by = idx_noaa)
summary(df_noaa)
```

```
##      ï..yr           mon              day           CO2.ppm
##  Min.   :1974   Min.   : 1.000   Min.   : 1.00   Min.   :-1000.0
##  1st Qu.:1985   1st Qu.: 4.000   1st Qu.: 8.00   1st Qu.:  346.2
##  Median :1997   Median : 7.000   Median :16.00   Median :  363.6
##  Mean   :1997   Mean   : 6.528   Mean   :15.72   Mean   :  354.9
##  3rd Qu.:2008   3rd Qu.: 9.000   3rd Qu.:23.00   3rd Qu.:  385.3
##  Max.   :2019   Max.   :12.000   Max.   :31.00   Max.   :  415.4
##      Date
##  Length:2367
##  Class :character
##  Mode  :character
##
##
##
```

```r
str(xts_noaa)
```

```
## An 'xts' object on 1974-05-19/2019-09-22 containing:
##   Data: num [1:2367, 1] 333 333 332 332 332 ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
```

```r
head(xts_noaa)
```

```
##               [,1]
## 1974-05-19 333.34
## 1974-05-26 332.95
## 1974-06-02 332.32
## 1974-06-09 332.18
## 1974-06-16 332.37
## 1974-06-23 331.59
```

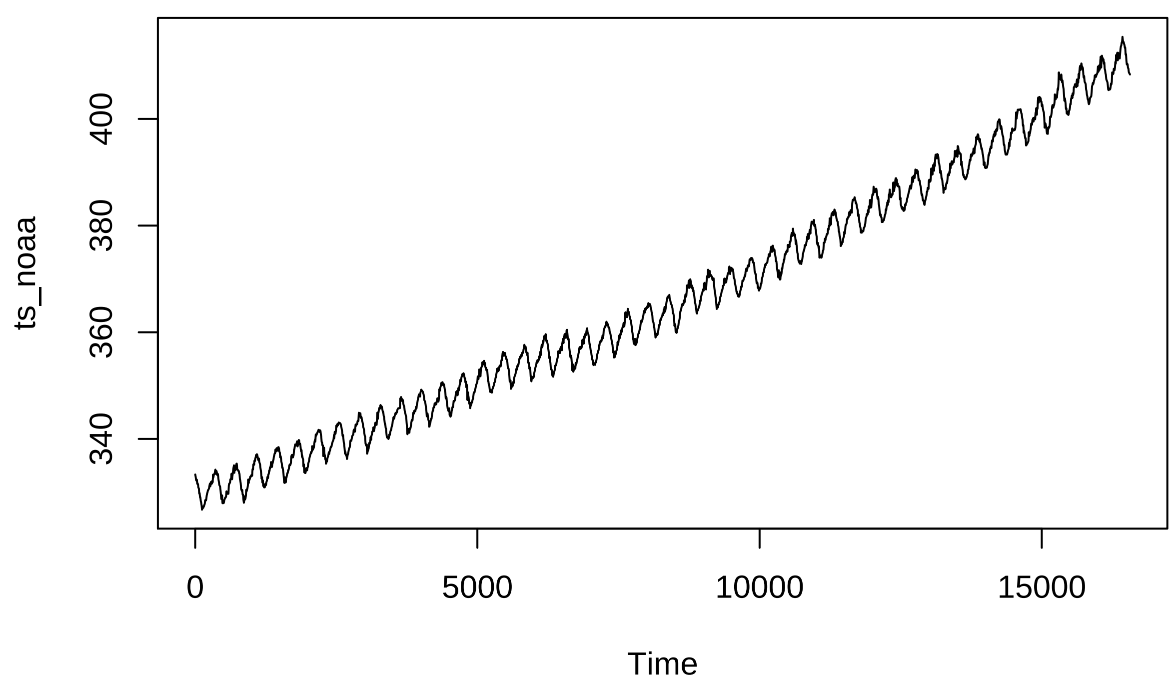
```r
tail(xts_noaa)
```

```
##              [,1]
## 2019-08-18 409.57
## 2019-08-25 409.45
## 2019-09-01 408.80
## 2019-09-08 408.59
## 2019-09-15 408.50
## 2019-09-22 408.32
```

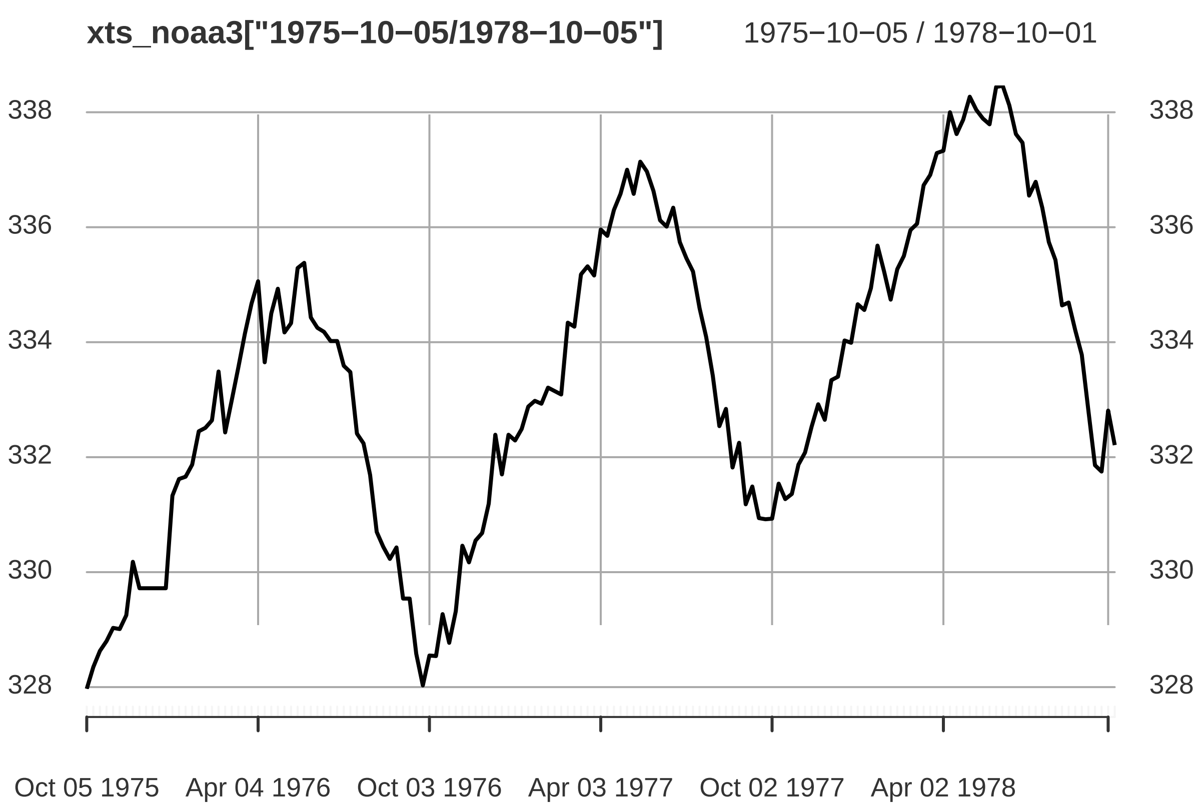
```r
xts_noaa2 <- xts_noaa
xts_noaa2[xts_noaa2 <= -998] <- NA # replace -999.99 with NAs
xts_noaa3 <- na.locf(xts_noaa2, na.rm = TRUE, fromLast = FALSE)# replace NAs with last oberved value
cbind(xts_noaa["1975-10-05"], xts_noaa2["1975-10-05"], xts_noaa3["1975-10-05"])
```

```
##            xts_noaa..1975.10.05.. xts_noaa2..1975.10.05..
## 1975-10-05                -999.99                      NA
##            xts_noaa3..1975.10.05..
## 1975-10-05                  327.97
```

```r
ts_noaa <- as.ts(xts_noaa3)
plot(ts_noaa)
```

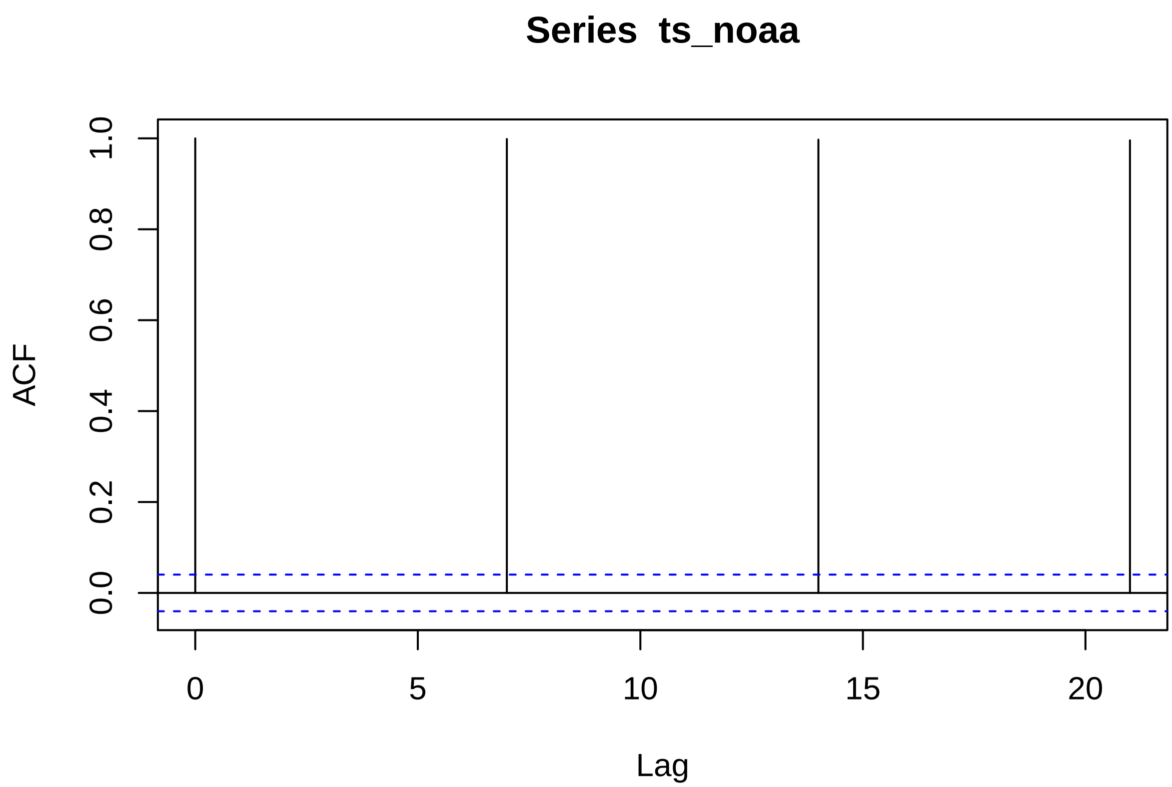


```r
plot(xts_noaa3["1975-10-05/1978-10-05"])
```

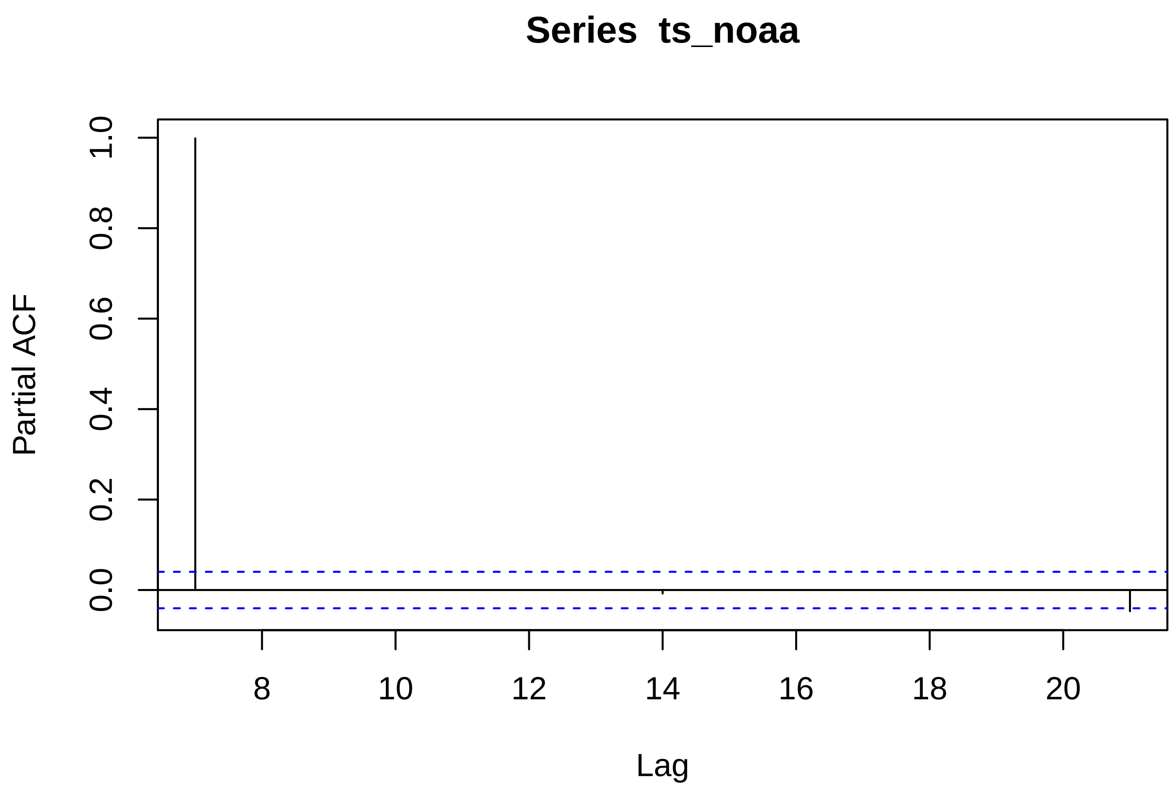


Let us find the best parameters for ARIMA model by applying grid search logic on the cleaned data

```r
acf(ts_noaa, 3)
```




```r
pacf(ts_noaa, 3)
```

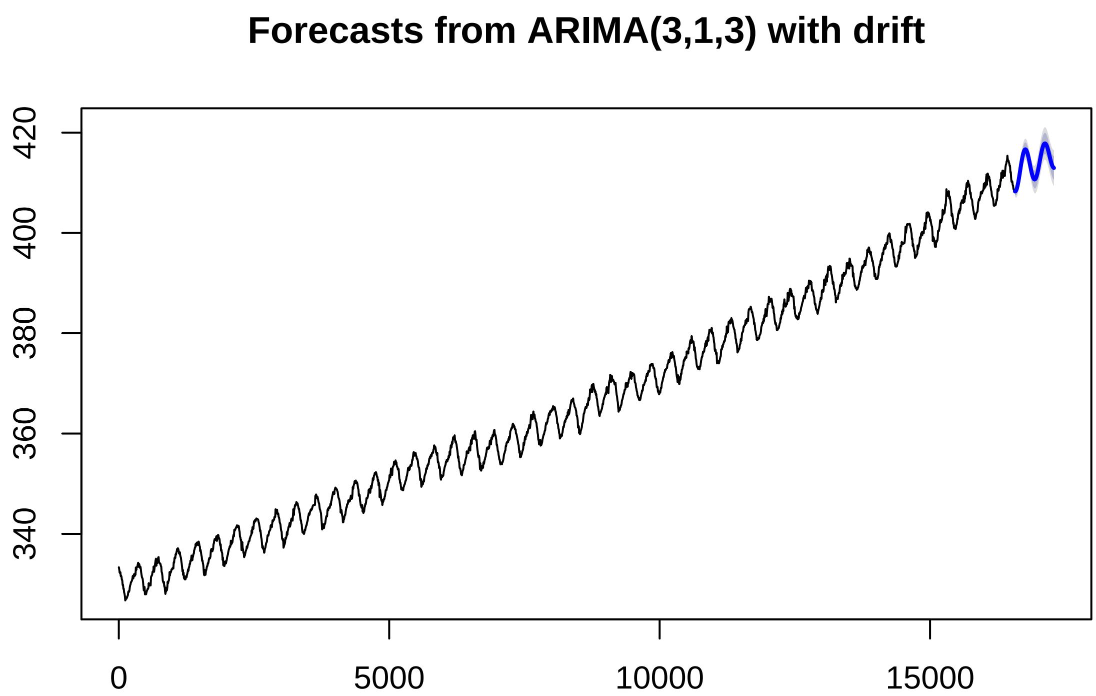



```r
best_bic = Inf
for (d in 0:2){
  for (q in 0:4){
    for(p in 0:4){

        ts_noaa.mod <- Arima(ts_noaa, order = c(p,d,q),
                      seasonal = list(order = c(0,0,0),52),
                      method = "ML",include.drift= TRUE)
        if (ts_noaa.mod$bic < best_bic){
          best_bic = ts_noaa.mod$bic
          best_p = p
          best_q = q
          best_d = d
          }
      }
    }
  }
cat("\np:", best_p, "\nd:", best_d, "\nq:", best_q, "\nBIC:", best_bic)
```

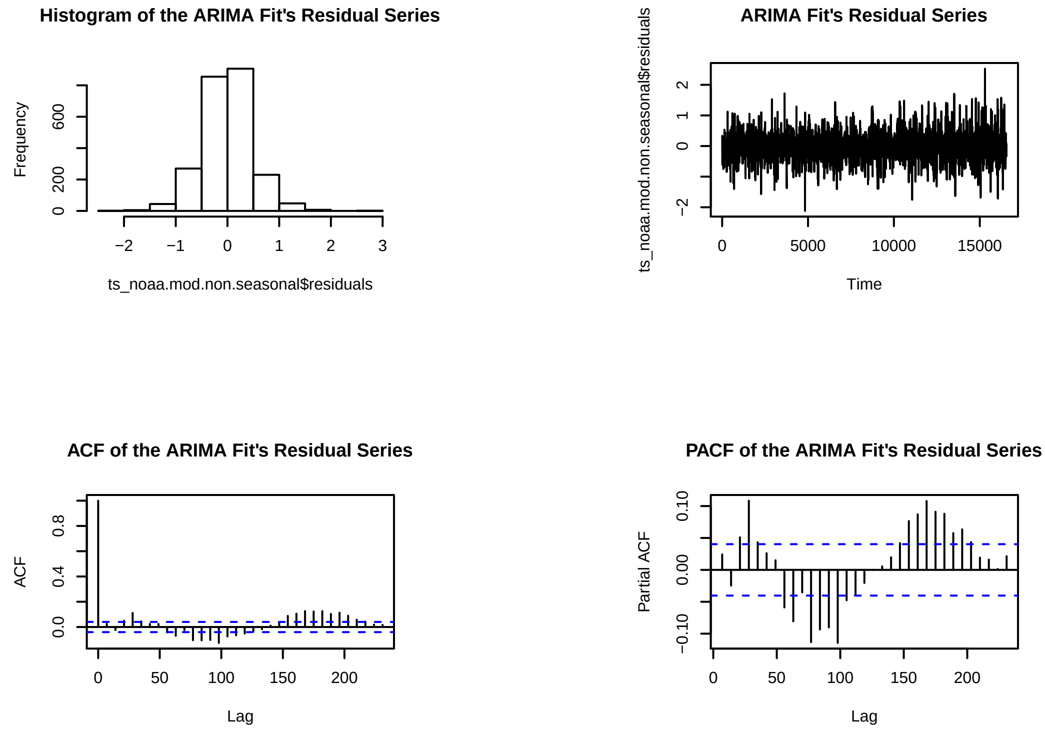
```
##
## p: 3
## d: 1
## q: 3
## BIC: 3252.52
```

Let us fit a model and see the diagnostic plots based on the parameters found from the grid search

```
ts_noaa.mod.non.seasonal <- Arima(ts_noaa, order = c(3,1,3),
                      seasonal = list(order = c(0,0,0),52),
                      method = "ML",include.drift= TRUE)
plot(forecast(ts_noaa.mod.non.seasonal, 104))
```

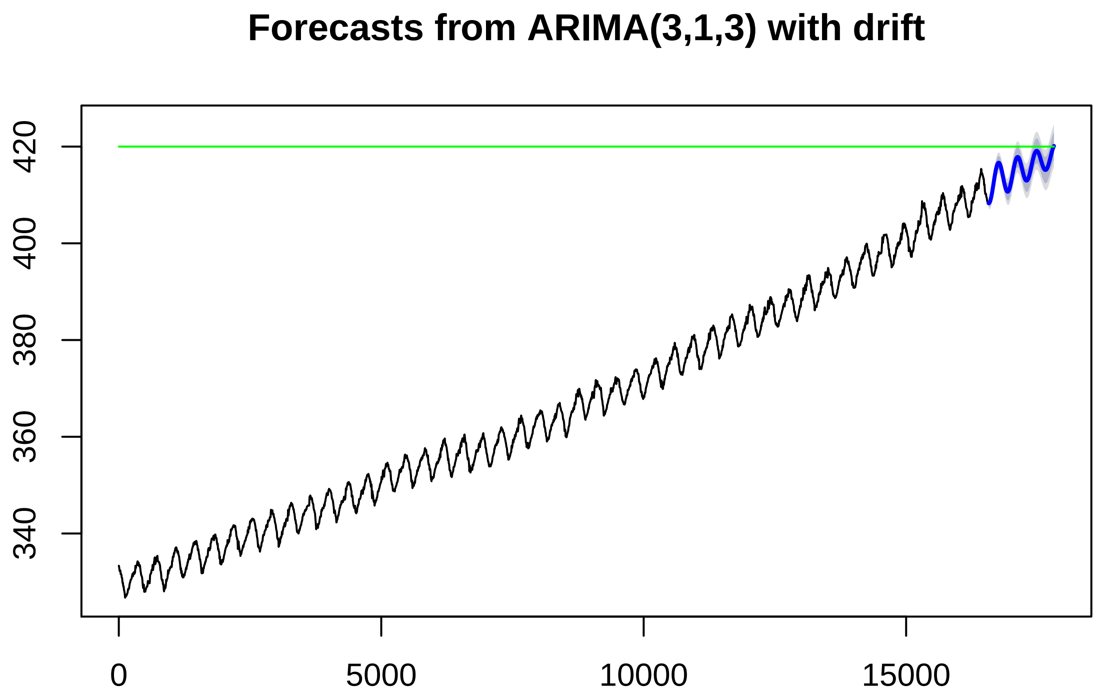



```
par(cex=0.5, mai=c(0.5,0.5,0.5,0.5))
par(fig=c(0.1,0.5,0.1,0.5))
acf(ts_noaa.mod.non.seasonal$residuals, main="ACF of the ARIMA Fit's Residual Series")
par(fig=c(0.6, 1,0.1,0.5), new=TRUE)
pacf(ts_noaa.mod.non.seasonal$residuals, main="PACF of the ARIMA Fit's Residual Series")
par(fig=c(0.1,0.5,0.6,1), new=TRUE)
hist(ts_noaa.mod.non.seasonal$residuals, main="Histogram of the ARIMA Fit's Residual Series")
par(fig=c(0.6,1,0.6,1), new=TRUE)
plot(ts_noaa.mod.non.seasonal$residuals, main="ARIMA Fit's Residual Series")
```



CO2 levels will reach 420 ppm on 1975-10-05

```
plot(forecast(ts_noaa.mod.non.seasonal, 179))
lines(x = 1:17816,y=rep(420, 17816),col="green")
```

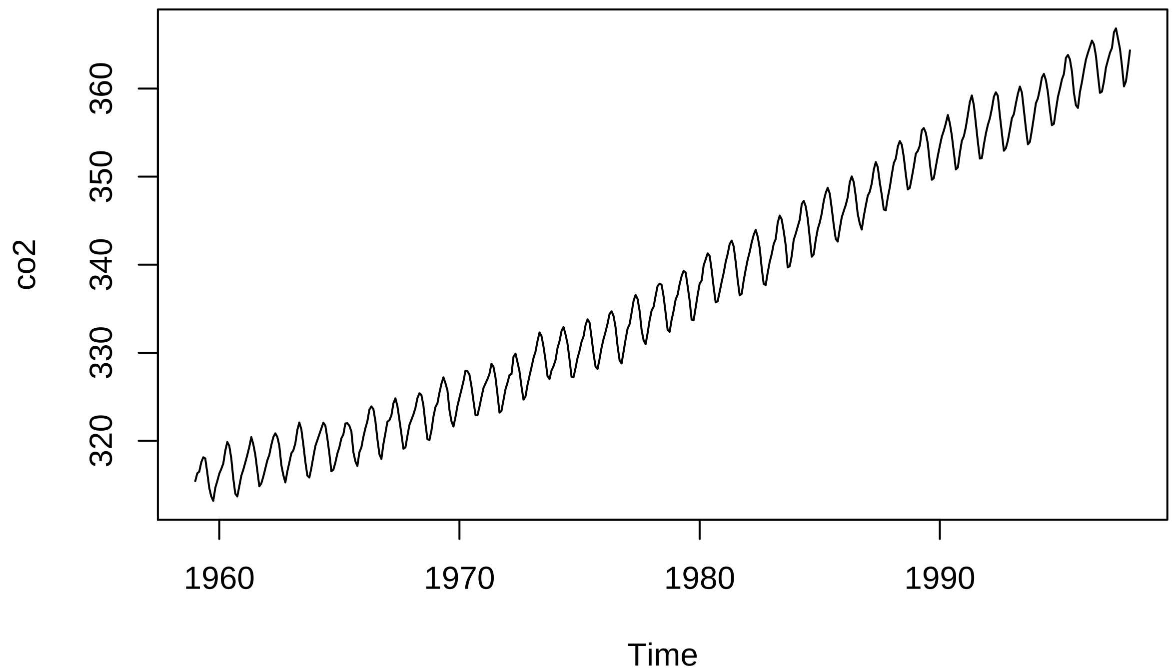



```r
paste("CO2 levels will reach 420 ppm on ", as.Date("1975-10-05") + 17816, sep = "")
```

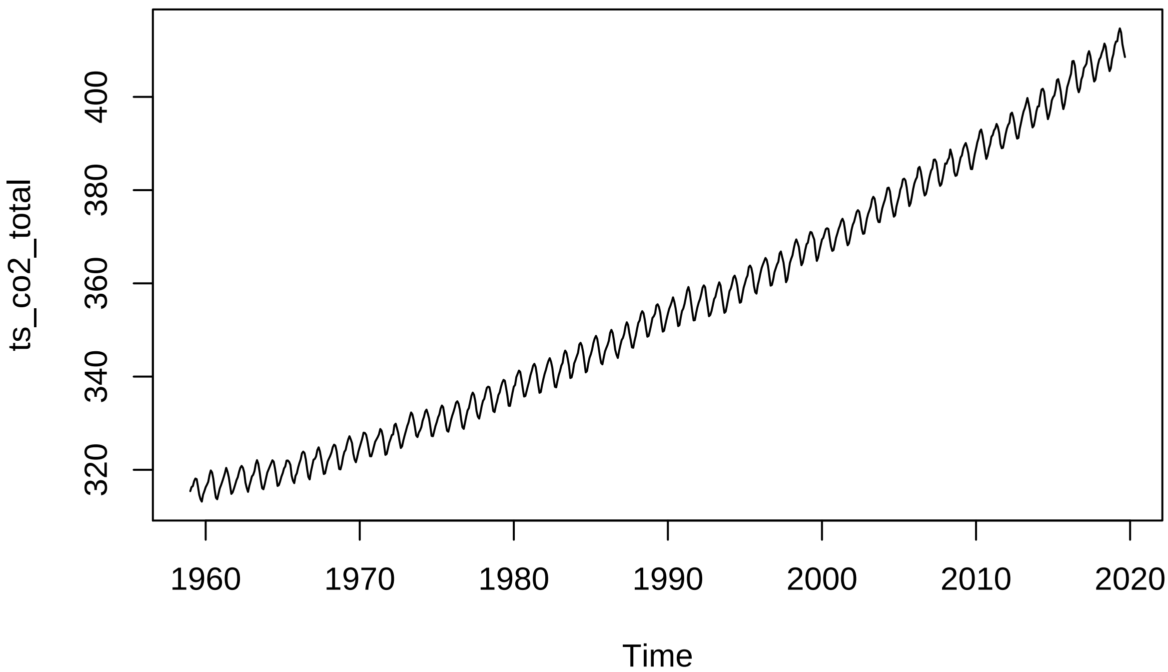
```
## [1] "CO2 levels will reach 420 ppm on 2024-07-15"
```

#3.b. Load the `co2` dataset from the R's `datasets` package. This is a time series of monthly observations from 1959 to 1997. Use averages from the NOAA data to extend the monthly series to 2019. Repeat the previous analysis and forecast for this monthly 1959-2019 series. Assess your model's pseudo-out-of-sample forecasting performance using a rolling test set window, and plot the distribution of RMSEs over this range of test sets.

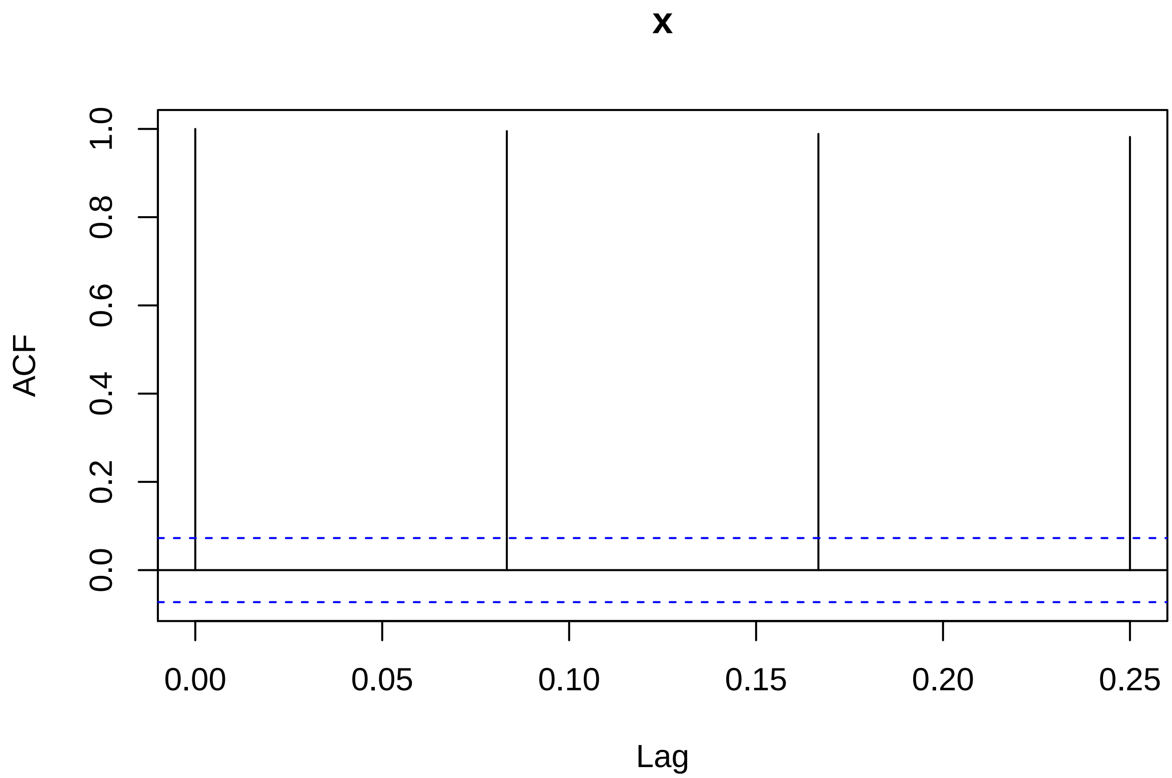
```r
data(co2)
ts.plot(co2)
```



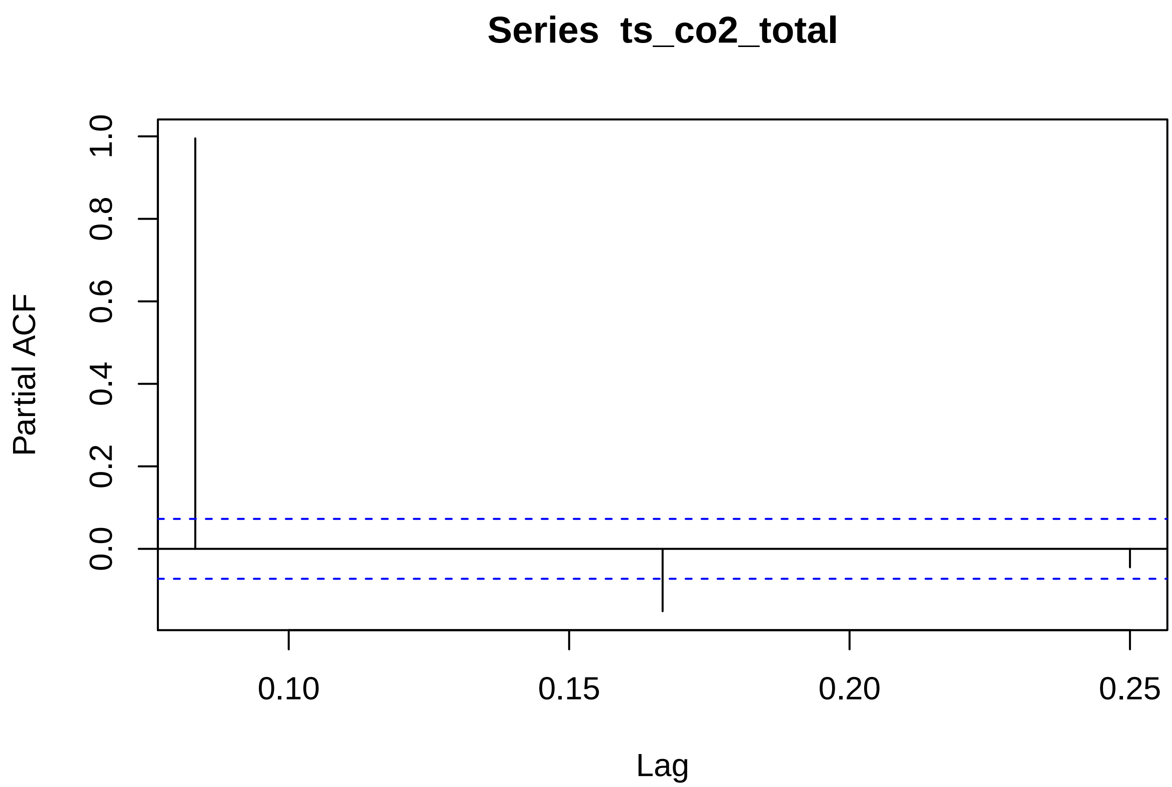

```
xts_noaa_ep <- endpoints(xts_noaa3, on = "months")
xts_noaa_monthlyMean <- round(period.apply(xts_noaa3, INDEX = xts_noaa_ep, FUN = mean),2)
xts_noaa_post_1998 <- xts_noaa_monthlyMean["1998-01-25/2019-09-22"]
ts_noaa_post_1998_df <- as.ts(xts_noaa_post_1998)
noaa_post_1998_df <- as.data.frame(as.numeric(ts_noaa_post_1998_df))

co2_df <- as.data.frame(as.numeric(co2))
colnames(co2_df) <- "x"
colnames(noaa_post_1998_df) <- "x"

df_co2_total <- rbind(co2_df, noaa_post_1998_df)
ts_co2_total <- ts(df_co2_total, start = c(1959, 1), end = c(2019, 9), frequency = 12)
ts.plot(ts_co2_total)
```




```r
acf(ts_co2_total, 3)
```



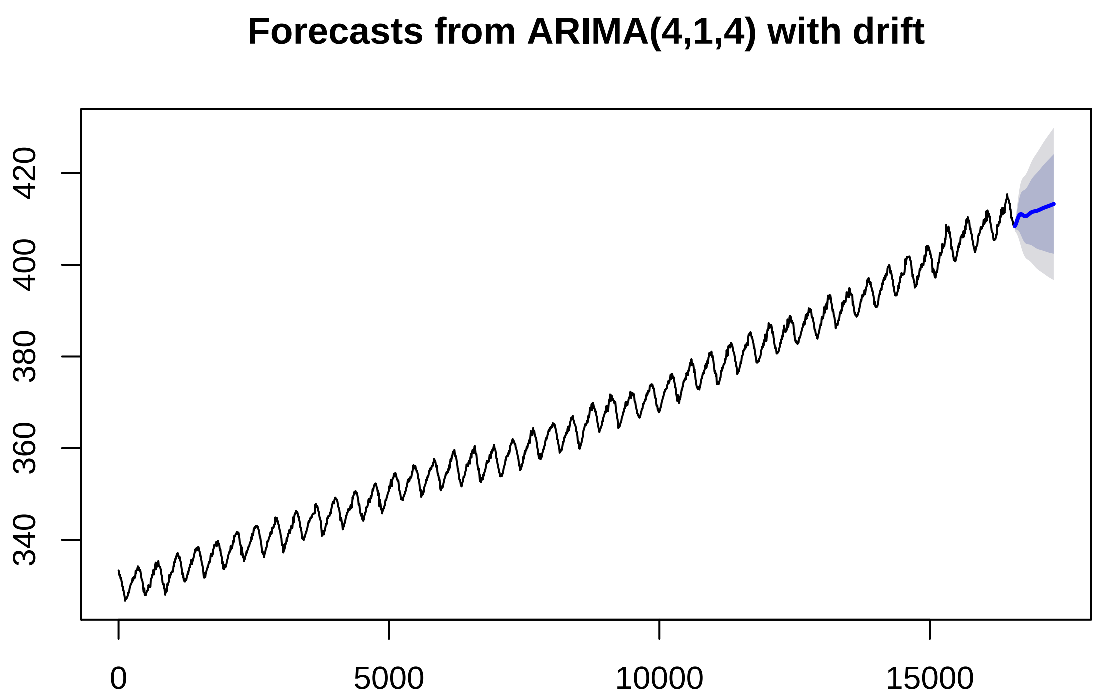

```r
pacf(ts_co2_total, 3)
```



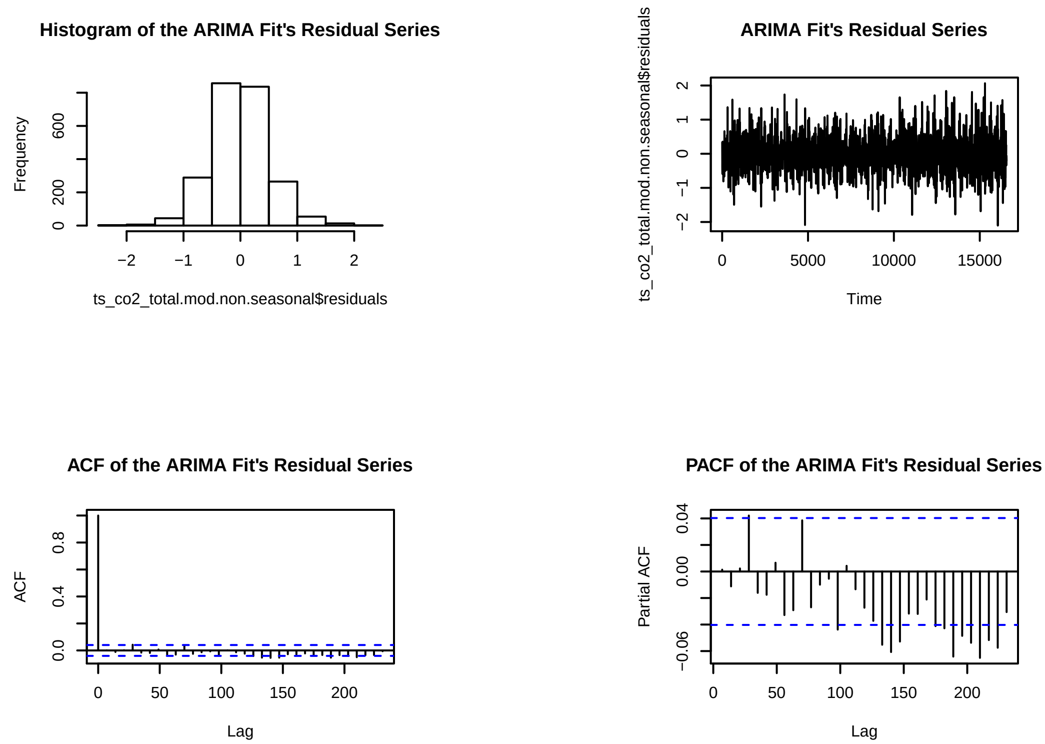

```
best_bic = Inf
for (d in 0:2){
  for (q in 0:4){
    for(p in 0:4){

        ts_co2_total.mod <- Arima(ts_co2_total, order = c(p,d,q),
                              seasonal = list(order = c(0,0,0),52),
                              method = "ML",include.drift= TRUE)
        if (ts_co2_total.mod$bic < best_bic){
          best_bic = ts_co2_total.mod$bic
          best_p = p
          best_q = q
          best_d = d
          }
      }
    }
  }
cat("\np:", best_p, "\nd:", best_d, "\nq:", best_q, "\nBIC:", best_bic)
```

```
##
## p: 4
## d: 1
## q: 4
## BIC: 657.6039
```

```r
ts_co2_total.mod.non.seasonal <- Arima(ts_noaa, order = c(4,1,4),
                       seasonal = list(order = c(0,0,0),52),
                       method = "ML",include.drift= TRUE)
plot(forecast(ts_co2_total.mod.non.seasonal, 104))
```




```r
par(cex=0.5, mai=c(0.5,0.5,0.5,0.5))
par(fig=c(0.1,0.5,0.1,0.5))
acf(ts_co2_total.mod.non.seasonal$residuals, main="ACF of the ARIMA Fit's Residual Series")
par(fig=c(0.6, 1,0.1,0.5), new=TRUE)
pacf(ts_co2_total.mod.non.seasonal$residuals, main="PACF of the ARIMA Fit's Residual Series")
par(fig=c(0.1,0.5,0.6,1), new=TRUE)
hist(ts_co2_total.mod.non.seasonal$residuals, main="Histogram of the ARIMA Fit's Residual Series")
par(fig=c(0.6,1,0.6,1), new=TRUE)
plot(ts_co2_total.mod.non.seasonal$residuals, main="ARIMA Fit's Residual Series")
```



Question 4: Vector Autoregression (3 points)

Use the series contained in `Q4.txt` to conduct a multivariate time series analysis and build a model to forecast the series. In model estimation, do not use the observations in 1993. All relevant time-series model building steps are applicable. Measure and discuss your model's performance, using both in-sample and out-of-sample model performance. When training your model, exclude all the observations in 1993. For the out-of-sample forecast, measure your model's performance in forecasting 1993. Discuss the model performance and forecast a 12-month forecast beyond the last observed month in the given series.

Let us load the "Q4.txt" data into a data-frame and convert it into a time-series (ts) object "df_q4_ts".

```
df_q4 <- read.table("Q4.txt",header = TRUE, stringsAsFactors = FALSE)
df_q4_ts <- ts(df_q4[, 3:6], start=c(1947,1),end = c(1993,12), frequency= 12)
str(df_q4_ts)
```

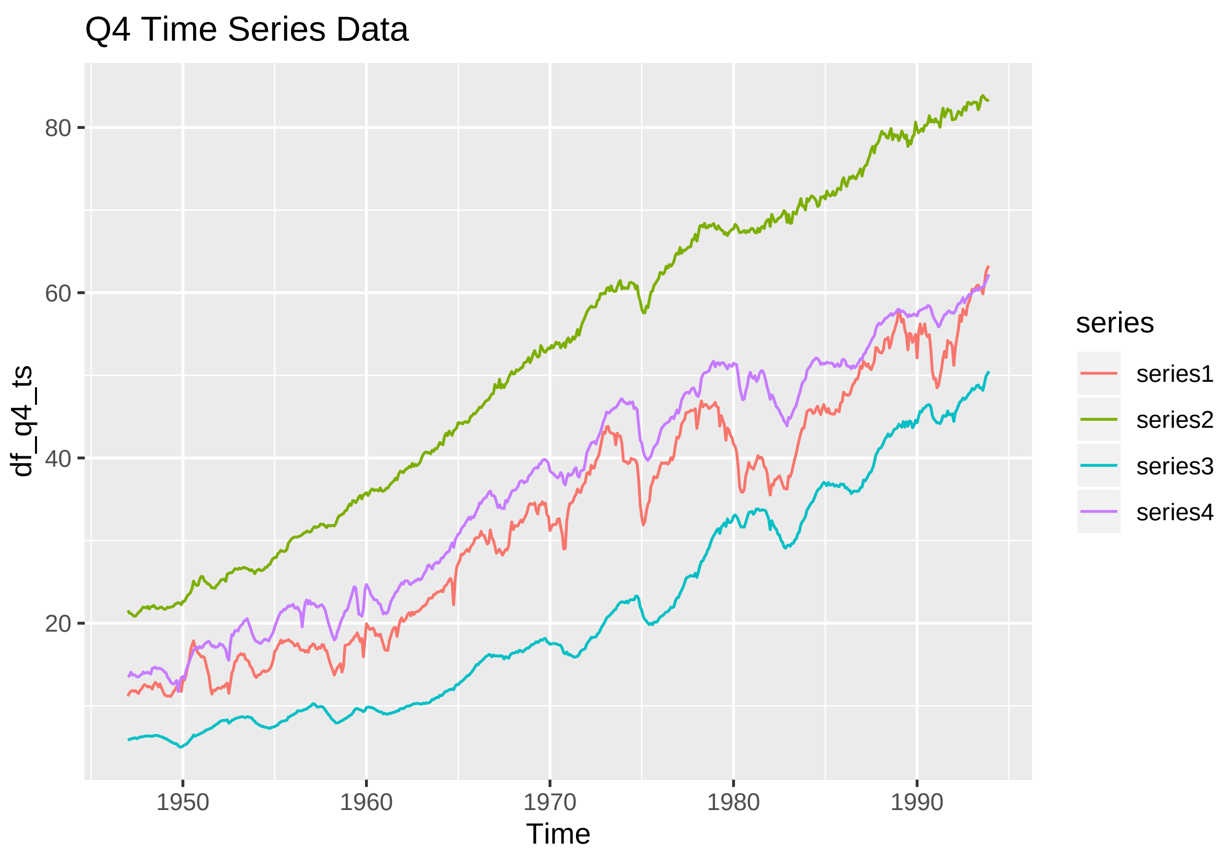
```
##  Time-Series [1:564, 1:4] from 1947 to 1994: 11.2 11.5 11.8 11.9 11.7 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:4] "series1" "series2" "series3" "series4"
```

Let us now segregate the time-series "df_q4_ts" into training and testing data.

```r
df_q4_test <- window(df_q4_ts, start= c(1993, 1))
df_q4_train <- window(df_q4_ts, end= c(1992, 12))
```

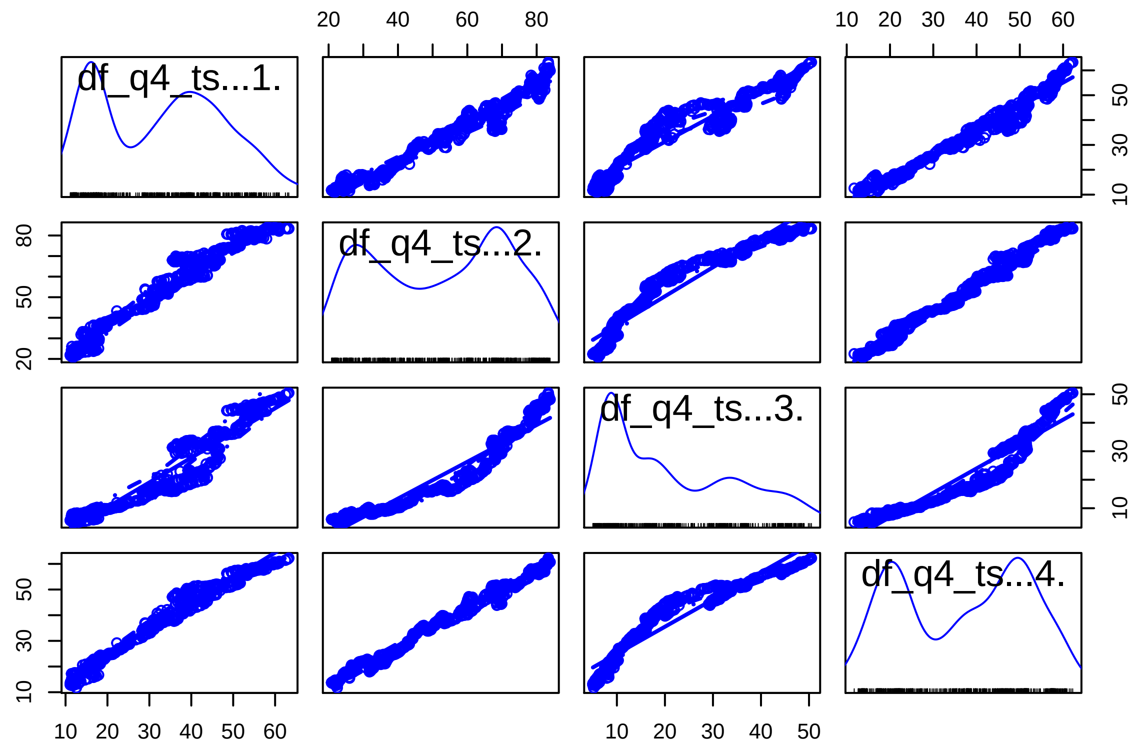
Let us now plot the time-series to see the distribution of 4 series

```r
autoplot(df_q4_ts, main = "Q4 Time Series Data")
```



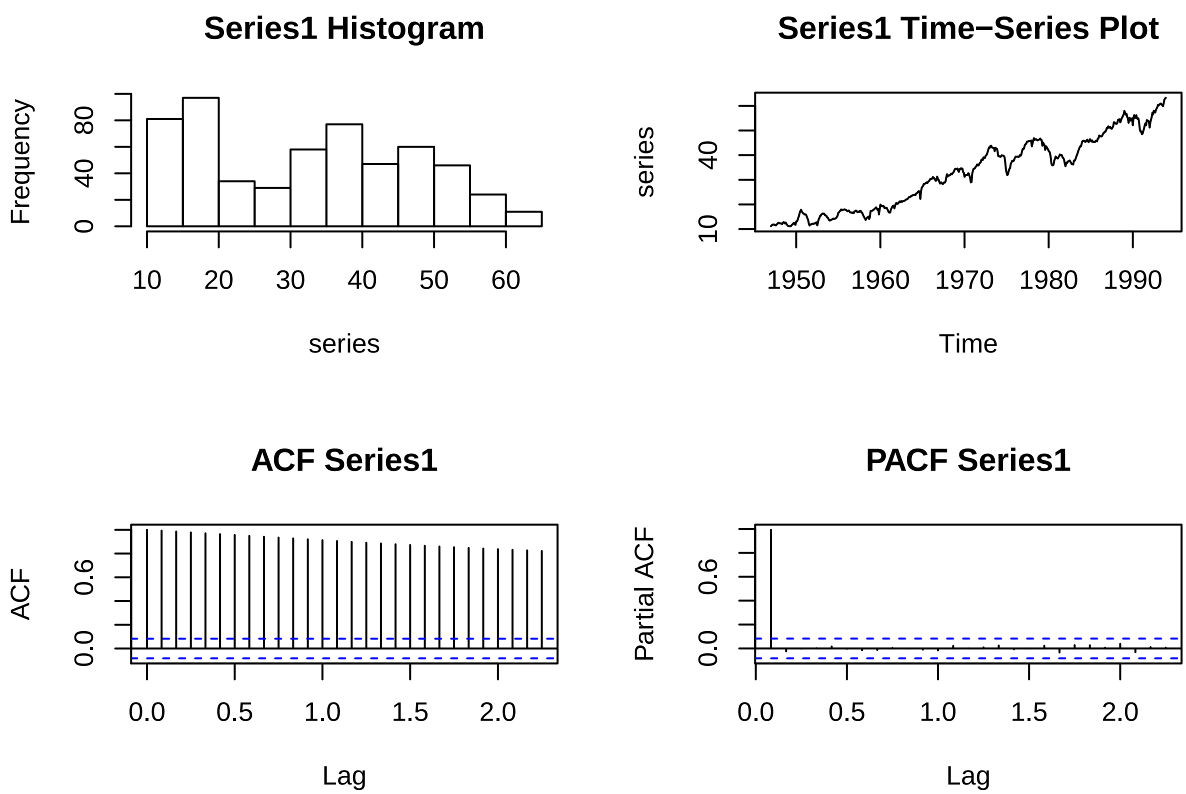
Let us now see the correlation among of all 4 series present in the "df_q4_ts"

```r
scatterplotMatrix(~df_q4_ts[, 1] + df_q4_ts[, 2] + df_q4_ts[, 3] +
df_q4_ts[, 4])
```

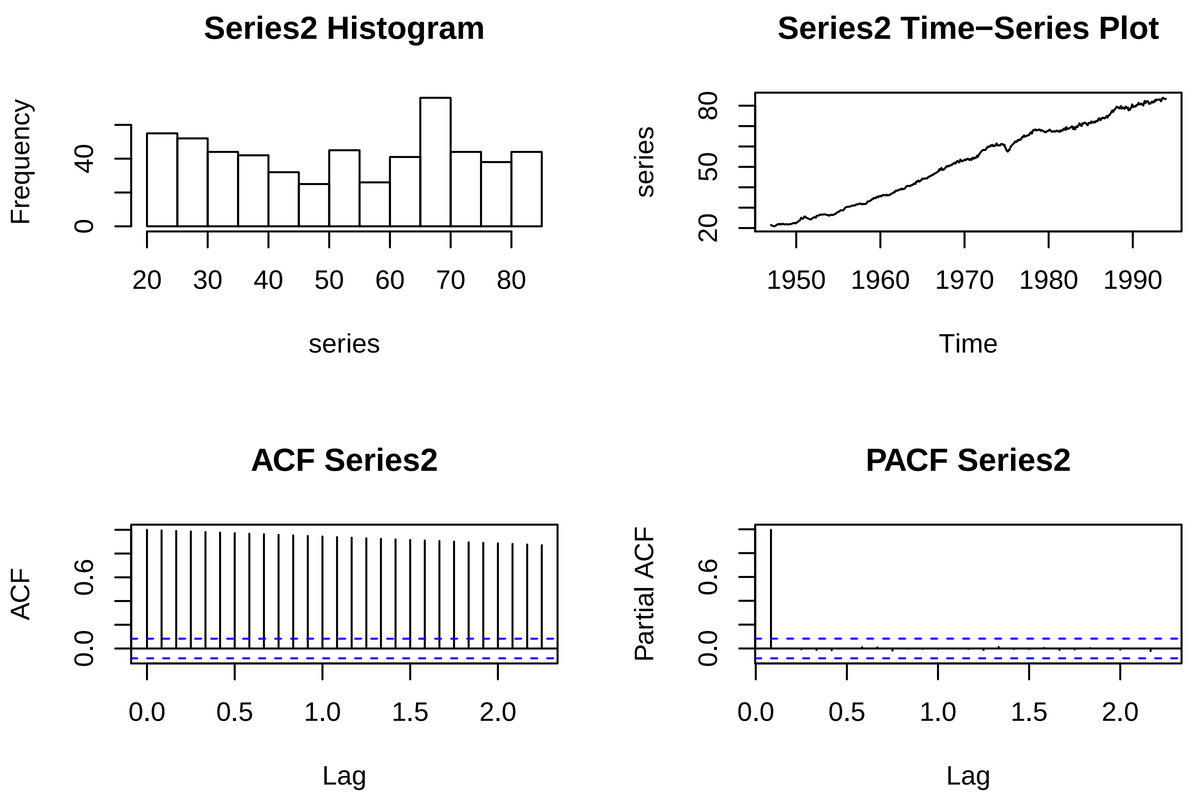


Let us now plot 1. Histogram ; 2. time-series plot, 3. ACF and 4. PACF of all 4 series.

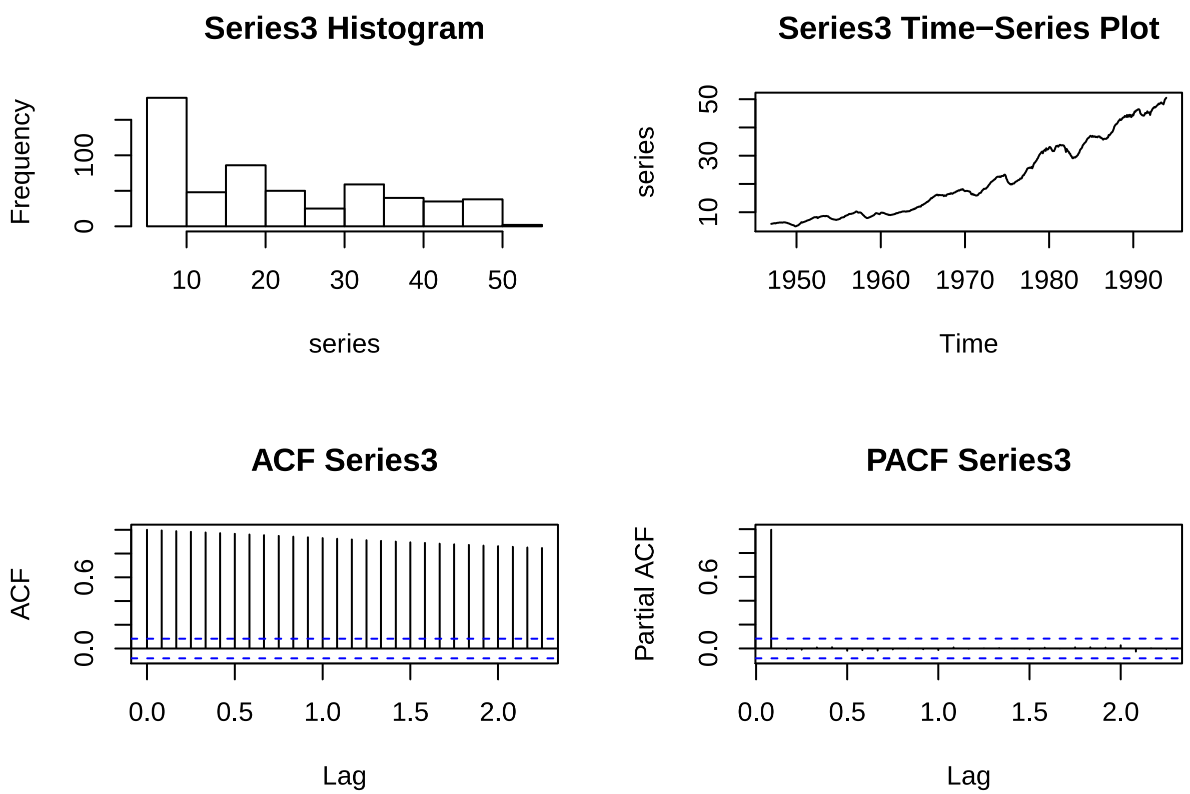
```
tsplot <- function(series, title) {
par(mfrow = c(2, 2))
hist(series, main = "")
title(paste(title, "Histogram"))
plot.ts(series, main = "")
title(paste(title, "Time-Series Plot"))
acf(series, main = "")
title(paste("ACF", title))
pacf(series, main = "")
title(paste("PACF", title))
}
tsplot(df_q4_ts[, 1], "Series1")
```



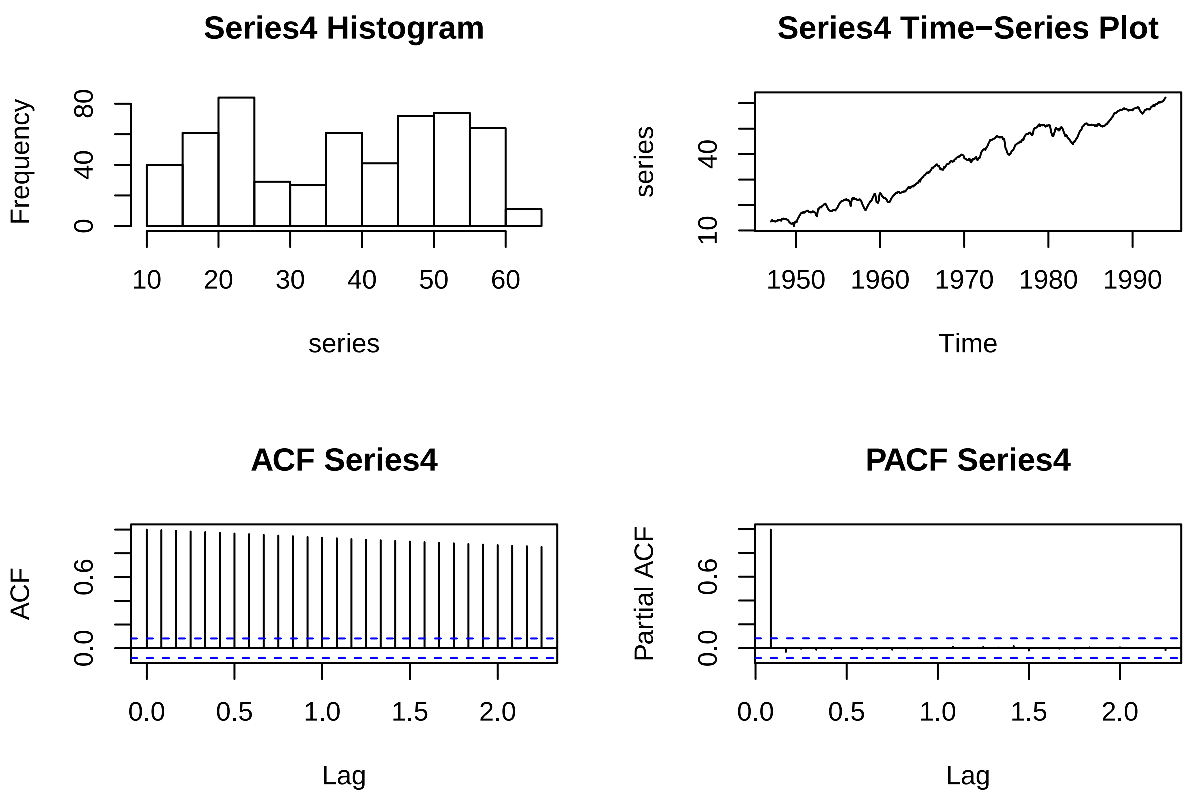

```
tsplot(df_q4_ts[, 2], "Series2")
```




```r
tsplot(df_q4_ts[, 3], "Series3")
```



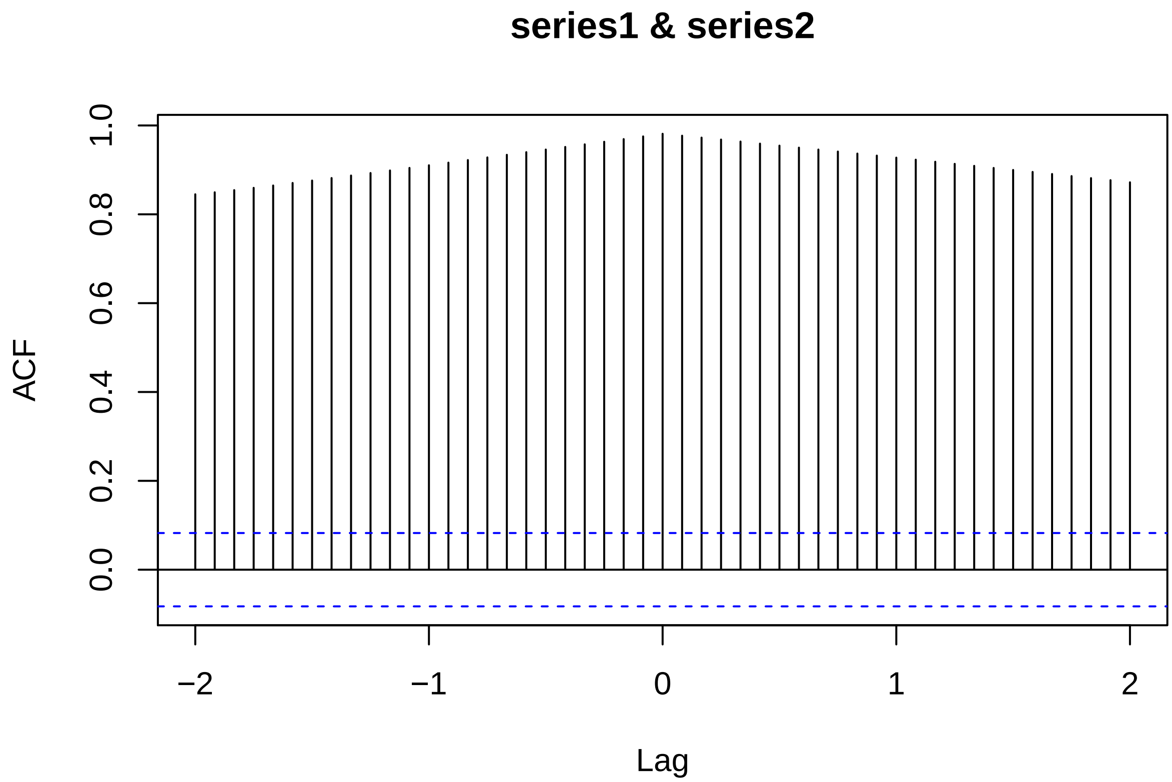

```
tsplot(df_q4_ts[, 4], "Series4")
```



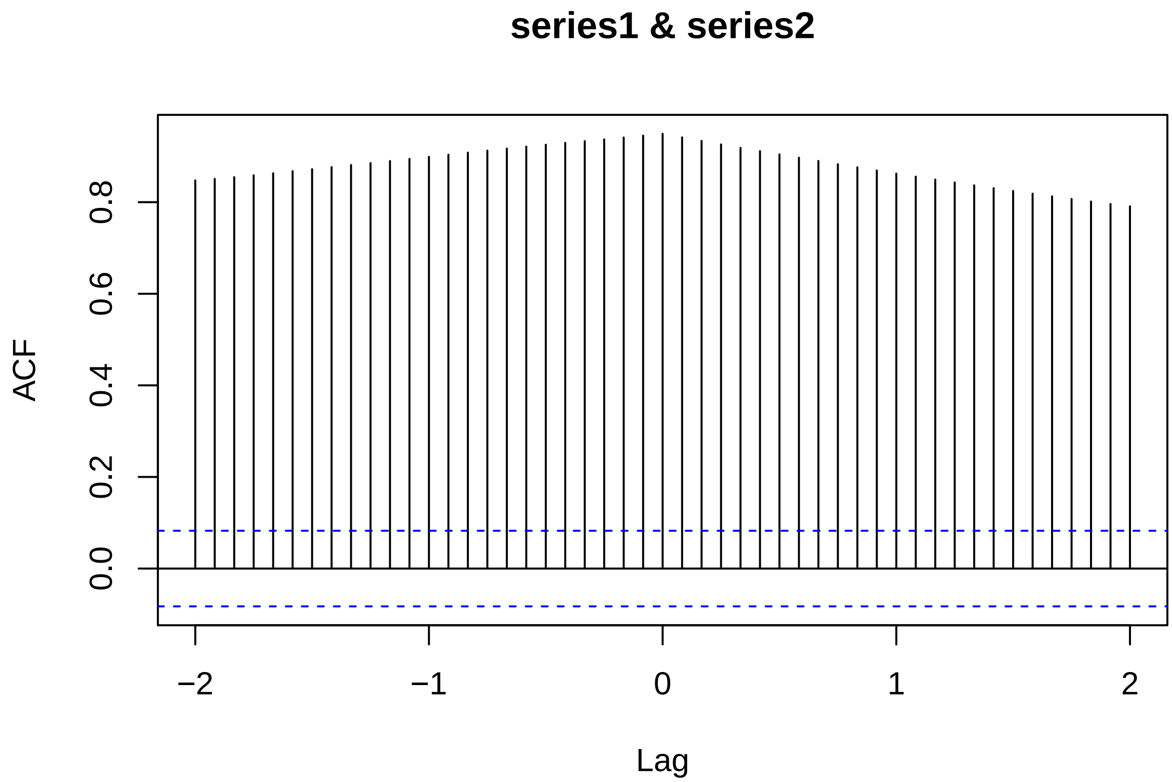
Let us now see how much correlation exists among all 4 series.

```
par(mfrow = c(1, 1))
corrfunc <- function(series1, series2) {
cat("Correlation Matrix: ", cor(series1, series2))
ccf(series1, series2)
}

for (i in 1:4) {
     for (j in 1:4) {
       if (i != j & j > i) {
           corrfunc(df_q4_ts[, i], df_q4_ts[, j])
           }
       }
}
```

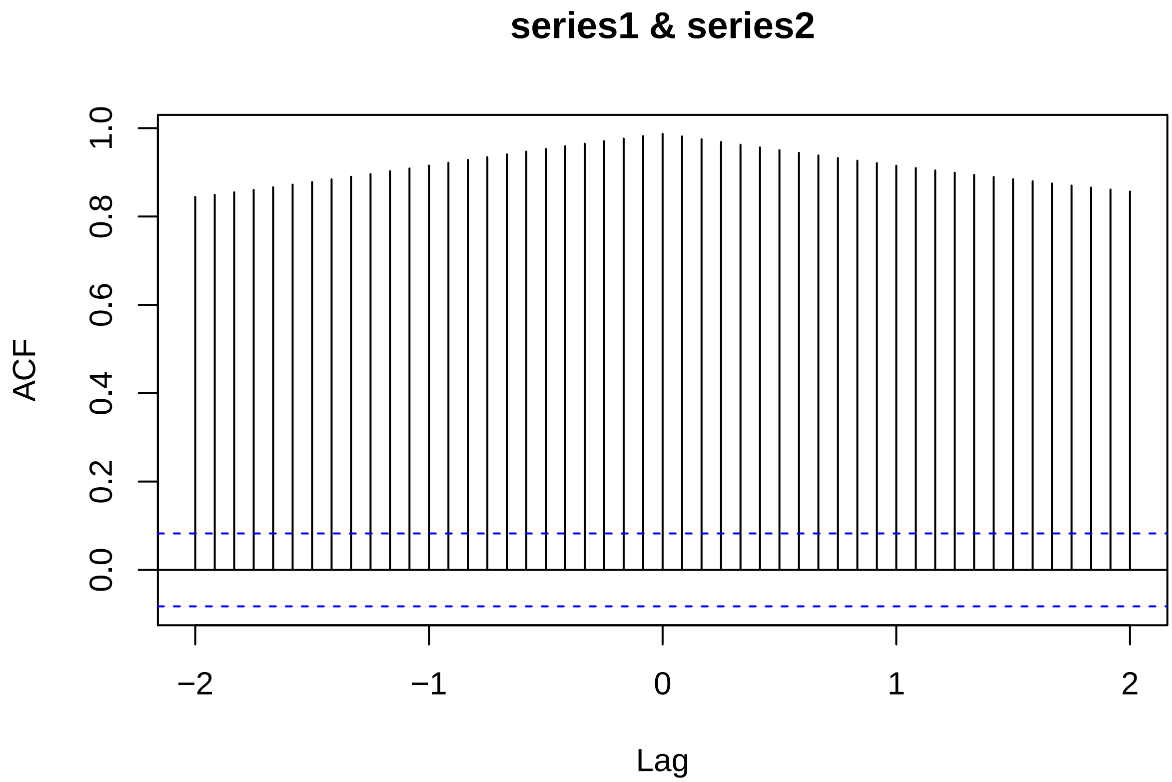
```
## Correlation Matrix:  0.9813096
```



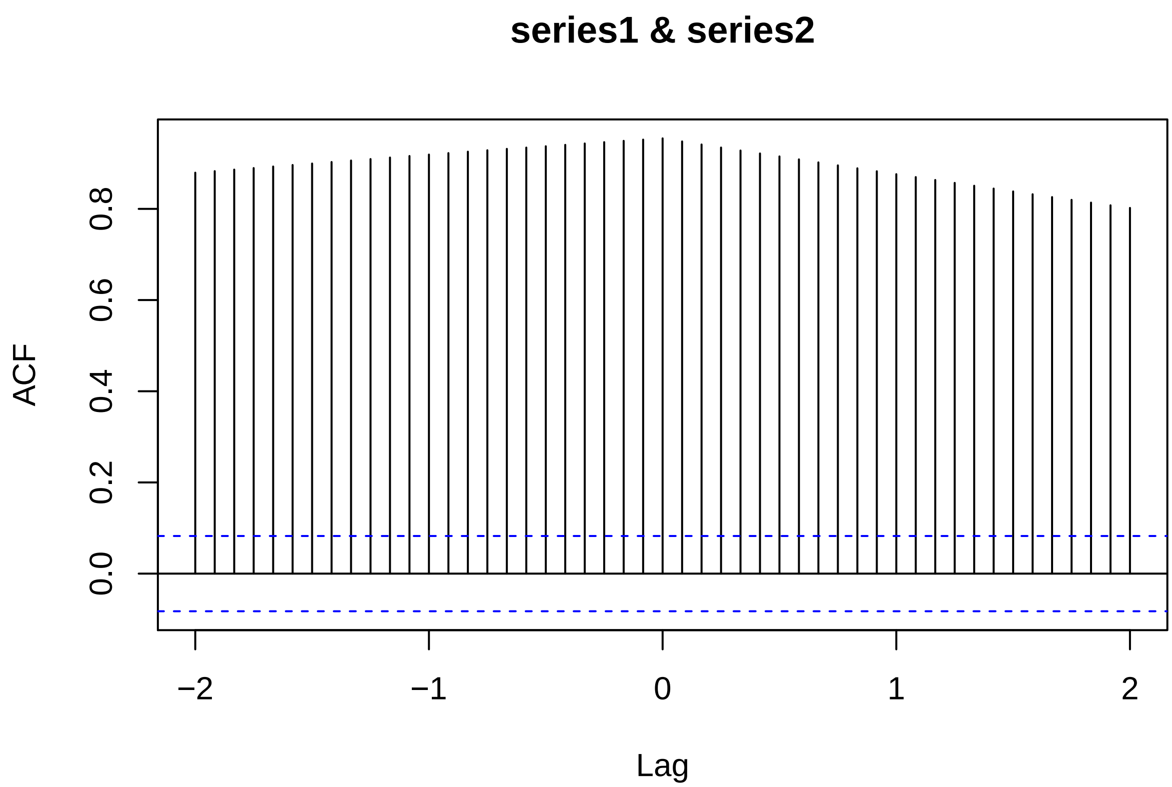

```
## Correlation Matrix:  0.9493428
```



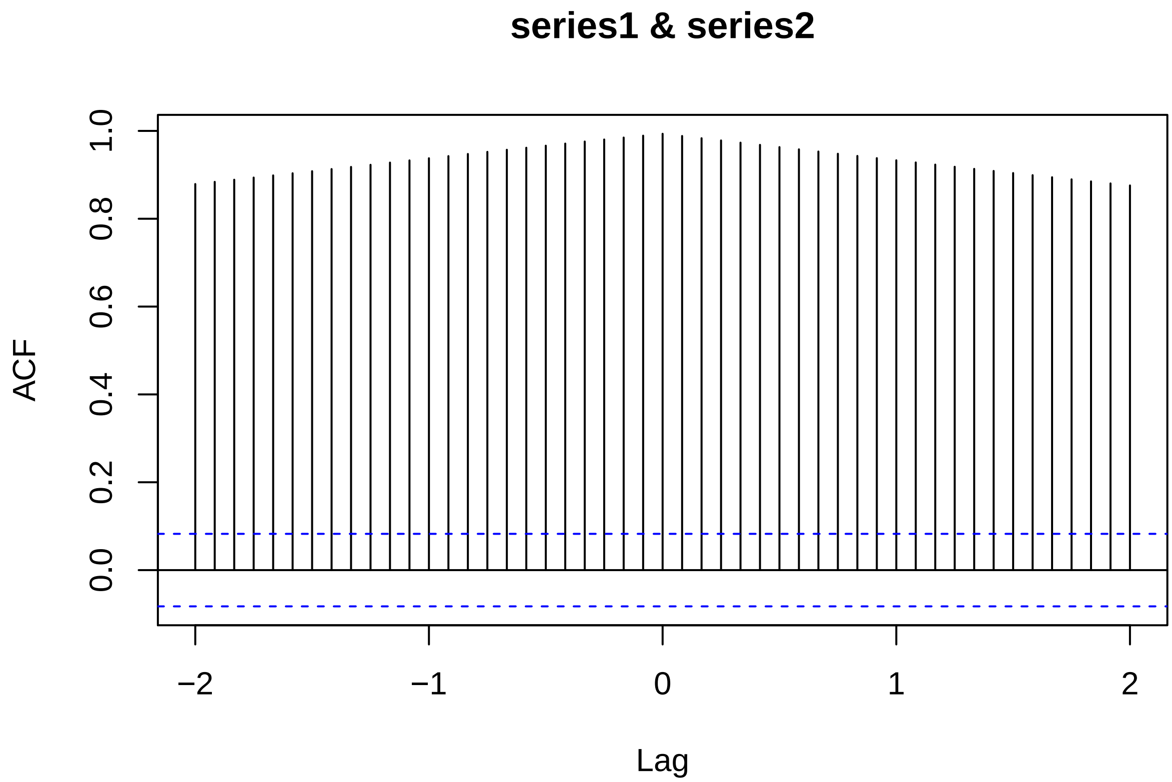

```
## Correlation Matrix:  0.987286
```



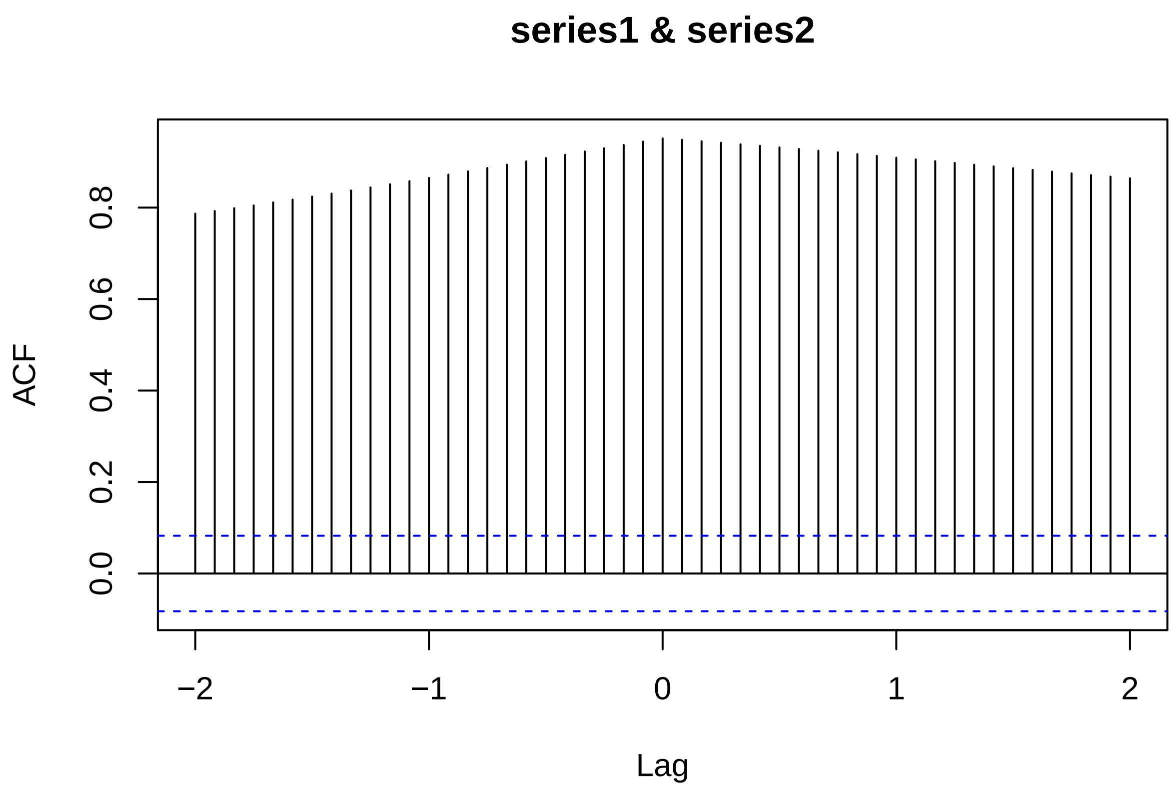

```
## Correlation Matrix:  0.9546637
```




```
## Correlation Matrix:  0.9934603
```




```
## Correlation Matrix:  0.9514153
```



We need to fit a Vector Autoregressive Model; so, let us find out what would be the regressive paramter (p) would be based on the least BIC score aka SC score.

```
VARselect(df_q4_train, lag.max = 8, type = "both")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      6      3      2      6
##
## $criteria
##                    1             2             3             4
## AIC(n) -7.4737990980 -7.6694465440 -7.7353310614 -7.7501785300
## HQ(n)  -7.3996476715 -7.5458608332 -7.5623110663 -7.5277242506
## SC(n)  -7.2841395724 -7.3533473347 -7.2927921683 -7.1811999532
## FPE(n)  0.0005677692  0.0004668839  0.0004371275  0.0004307073
##                    5             6             7             8
## AIC(n) -7.7415681431 -7.7530019249 -7.7214916648 -7.7000554890
## HQ(n)  -7.4696795794 -7.4316790769 -7.3507345325 -7.2798640724
## SC(n)  -7.0461498826 -6.9311439807 -6.7731940368 -6.6253181773
## FPE(n)  0.0004344666  0.0004295766  0.0004433976  0.0004530976
```

We see that "SC" score for parameter 2 is lowest (-7.3533473347) in top 8 paramters. So, we will model a VAR based on p = 2.

```r
var.fit1 <- VAR(df_q4_train, p = 2, type = "both")
summary(var.fit1)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: series1, series2, series3, series4
## Deterministic variables: both
## Sample size: 550
## Log Likelihood: -963.314
## Roots of the characteristic polynomial:
## 0.9962 0.9616 0.9616 0.912 0.3864 0.2438 0.1275 0.1167
## Call:
## VAR(y = df_q4_train, p = 2, type = "both")
##
##
## Estimation results for equation series1:
## ========================================
## series1 = series1.l1 + series2.l1 + series3.l1 + series4.l1 + series1.l2 + series2.l2 + series3.l2 +
##
##              Estimate Std. Error t value Pr(>|t|)
## series1.l1  1.0129768  0.0558601  18.134  < 2e-16 ***
## series2.l1  0.0256731  0.0787253   0.326    0.744
## series3.l1 -0.2318384  0.1545240  -1.500    0.134
## series4.l1  0.4297430  0.0845006   5.086 5.06e-07 ***
## series1.l2 -0.0227241  0.0550313  -0.413    0.680
## series2.l2  0.0415978  0.0787037   0.529    0.597
## series3.l2  0.2212607  0.1546375   1.431    0.153
## series4.l2 -0.5084159  0.0818174  -6.214 1.03e-09 ***
## const      -0.2285989  0.4161092  -0.549    0.583
## trend       0.0007667  0.0029284   0.262    0.794
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.7251 on 540 degrees of freedom
## Multiple R-Squared: 0.9974,  Adjusted R-squared: 0.9974
## F-statistic: 2.309e+04 on 9 and 540 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation series2:
## ========================================
## series2 = series1.l1 + series2.l1 + series3.l1 + series4.l1 + series1.l2 + series2.l2 + series3.l2 +
##
##             Estimate Std. Error t value Pr(>|t|)
## series1.l1  0.049840   0.031172   1.599   0.1104
## series2.l1  0.709703   0.043932  16.155  < 2e-16 ***
## series3.l1 -0.060668   0.086230  -0.704   0.4820
## series4.l1  0.073675   0.047155   1.562   0.1188
## series1.l2 -0.025371   0.030710  -0.826   0.4091
## series2.l2  0.234139   0.043920   5.331 1.44e-07 ***
## series3.l2  0.038038   0.086294   0.441   0.6595
## series4.l2 -0.078061   0.045657  -1.710   0.0879 .
```

```
## const       0.965874   0.232205   4.160 3.71e-05 ***
## trend       0.006883   0.001634   4.212 2.96e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.4047 on 540 degrees of freedom
## Multiple R-Squared: 0.9996,  Adjusted R-squared: 0.9996
## F-statistic: 1.397e+05 on 9 and 540 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation series3:
## ========================================
## series3 = series1.l1 + series2.l1 + series3.l1 + series4.l1 + series1.l2 + series2.l2 + series3.l2 +
##
##               Estimate Std. Error t value Pr(>|t|)
## series1.l1 -0.0148627  0.0197925  -0.751   0.4530
## series2.l1 -0.0328822  0.0278942  -1.179   0.2390
## series3.l1  1.0159275  0.0547515  18.555  < 2e-16 ***
## series4.l1  0.1849612  0.0299405   6.178 1.28e-09 ***
## series1.l2  0.0345086  0.0194989   1.770   0.0773 .
## series2.l2  0.0378522  0.0278865   1.357   0.1752
## series3.l2 -0.0205818  0.0547917  -0.376   0.7073
## series4.l2 -0.1978799  0.0289898  -6.826 2.36e-11 ***
## const      -0.1073883  0.1474372  -0.728   0.4667
## trend      -0.0005254  0.0010376  -0.506   0.6128
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.2569 on 540 degrees of freedom
## Multiple R-Squared: 0.9996,  Adjusted R-squared: 0.9996
## F-statistic: 1.525e+05 on 9 and 540 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation series4:
## ========================================
## series4 = series1.l1 + series2.l1 + series3.l1 + series4.l1 + series1.l2 + series2.l2 + series3.l2 +
##
##              Estimate Std. Error t value Pr(>|t|)
## series1.l1 -0.009157   0.031268  -0.293   0.7697
## series2.l1  0.011799   0.044067   0.268   0.7890
## series3.l1  0.202514   0.086495   2.341   0.0196 *
## series4.l1  1.243939   0.047299  26.299  < 2e-16 ***
## series1.l2  0.039681   0.030804   1.288   0.1982
## series2.l2  0.049710   0.044055   1.128   0.2597
## series3.l2 -0.211644   0.086559  -2.445   0.0148 *
## series4.l2 -0.334855   0.045797  -7.312 9.57e-13 ***
## const      -0.263090   0.232918  -1.130   0.2592
## trend      -0.001238   0.001639  -0.755   0.4504
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
```

```
## Residual standard error: 0.4059 on 540 degrees of freedom
## Multiple R-Squared: 0.9992,  Adjusted R-squared: 0.9992
## F-statistic: 7.831e+04 on 9 and 540 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##         series1 series2 series3 series4
## series1 0.52582 0.07924 0.11247 0.12687
## series2 0.07924 0.16374 0.02165 0.03891
## series3 0.11247 0.02165 0.06601 0.04318
## series4 0.12687 0.03891 0.04318 0.16475
##
## Correlation matrix of residuals:
##         series1 series2 series3 series4
## series1  1.0000  0.2701  0.6037  0.4310
## series2  0.2701  1.0000  0.2082  0.2369
## series3  0.6037  0.2082  1.0000  0.4141
## series4  0.4310  0.2369  0.4141  1.0000
```

Let us check the roots of the model to verify if the model/process is stable. If all the roots are less than one, the process is said to be stable.

```r
roots(var.fit1)
```

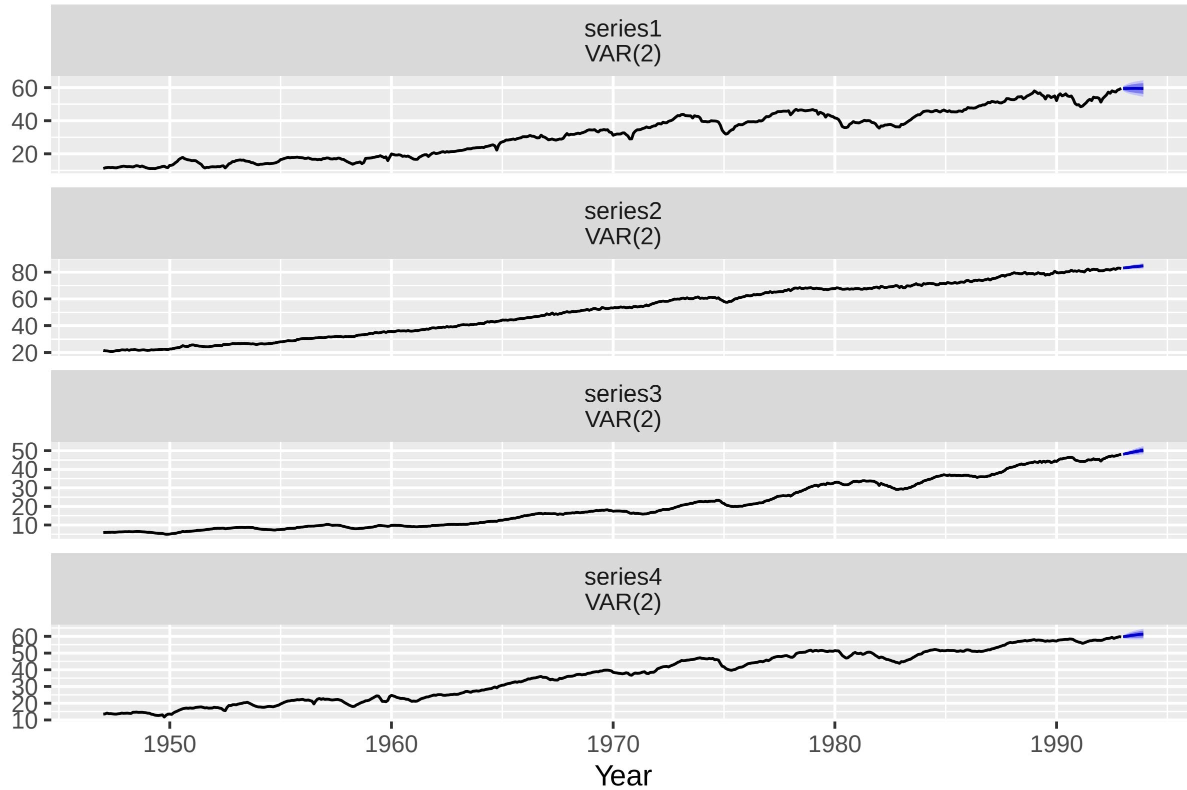
```
## [1] 0.9962454 0.9615507 0.9615507 0.9119872 0.3864429 0.2438240 0.1275290
## [8] 0.1167030
```

```r
var.fit1.ptasy <- serial.test(var.fit1, lags.pt = 12, type = "PT.asymptotic")
var.fit1.ptasy
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.fit1
## Chi-squared = 279.6, df = 160, p-value = 1.522e-08
```

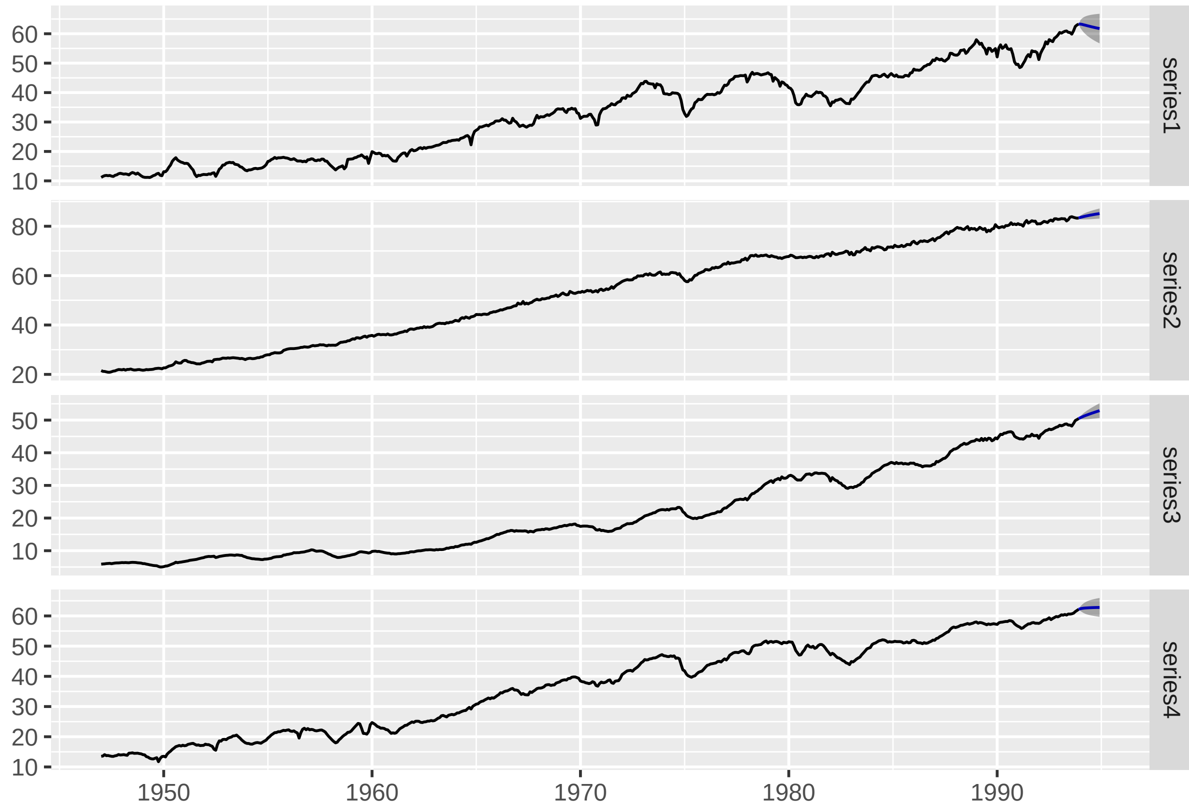
Let us now forecast the fitted model until the psudo-out-of-sample data.

```r
forecast(var.fit1, 12) %>% autoplot() + xlab("Year")
```



Let us now refit the model with p= 2 on the entire data and forecast 1 more year (= 12 months)

```
var.fit.total <- VAR( df_q4_ts, p=2, type = "both" )
var.fit.total %>% predict(n.ahead = 12, ci = 0.95) %>% autoplot()
```


60
50
40
30
20
10
series1
80
60
40
20
series2
50
40
30
20
10
series3
60
50
40
30
20
10
series4
1950
1960
1970
1980
1990