Arima Models

Shovan Biswas

2020/10/18

Libraries

```
library(tidyverse)
library(kableExtra)
library(corrplot)
library(reshape2)
library(caret)
library(Amelia)
library(dlookr)
library(fpp2)
library(plotly)
library(gridExtra)
library(readxl)
library(ggplot2)
library(urca)
library(tseries)
```

Exercise 8.1

Figure 8.31 shows the ACFs for 36 random numbers, 360 random numbers and 1,000 random numbers.

a)Explain the differences among these fgures. Do they all indicate that the data are white noise?

Answer: The number of numbers considered in each of the figures are 36, 360 and 1000. The diferences in pattern in each of the figures could be influenced by the numbers.

According to chapter 2, page 43 (printed book), "For a white noise series, we expect 95% of the spikes in ACF to lie within $\pm 2/\sqrt{(T)}$ where T is the length of the time serie".

So, in the first figure $T = 36$. Therefore, the bounds are $\pm 2/\sqrt{(36)} = \pm 0.33$. In left figure, the boundary line indicated by the blue dash-line appears to be ± 0.33 distance away from x-axis. There are 18 points here, and only 1 spike touches the lower boundary. So, $1/18 * 100 = 5.55\%$ touch the boundary. This is slightly more that 5%. Therefore, this should not qualify as White Noise. If at all, it's marginally White Noise.

In the second figure, $T = 360$. Therefore, the bounds are $\pm 2/\sqrt{(360)} = 2/18.97 = \pm 0.105$. In middle figure, the the blue dash-line appears to be ± 0.105 distance away from x-axis. There are 15 points here, out of which 3 spikes touch the boundaries. So, $3/15 * 100 = 20\%$ touch the boundary. This is way more than 5%. Therefore, this should not qualify as White Noise.

In the third figure, $T = 1000$. Therefore, the bounds are $\pm 2/\sqrt{(T)} = 2/31.62 = \pm 0.063$. In right figure, the the blue dash-line appears to be ± 0.06 distance away from x-axis. By plain observation, none of the spikes appear to touch the boundaries. So, this should comfortably qualify as White Noise.

b)Why are the critical values at different distances from the mean of zero? Why are the autocorrelations different in each fgure when they each refer to white noise?

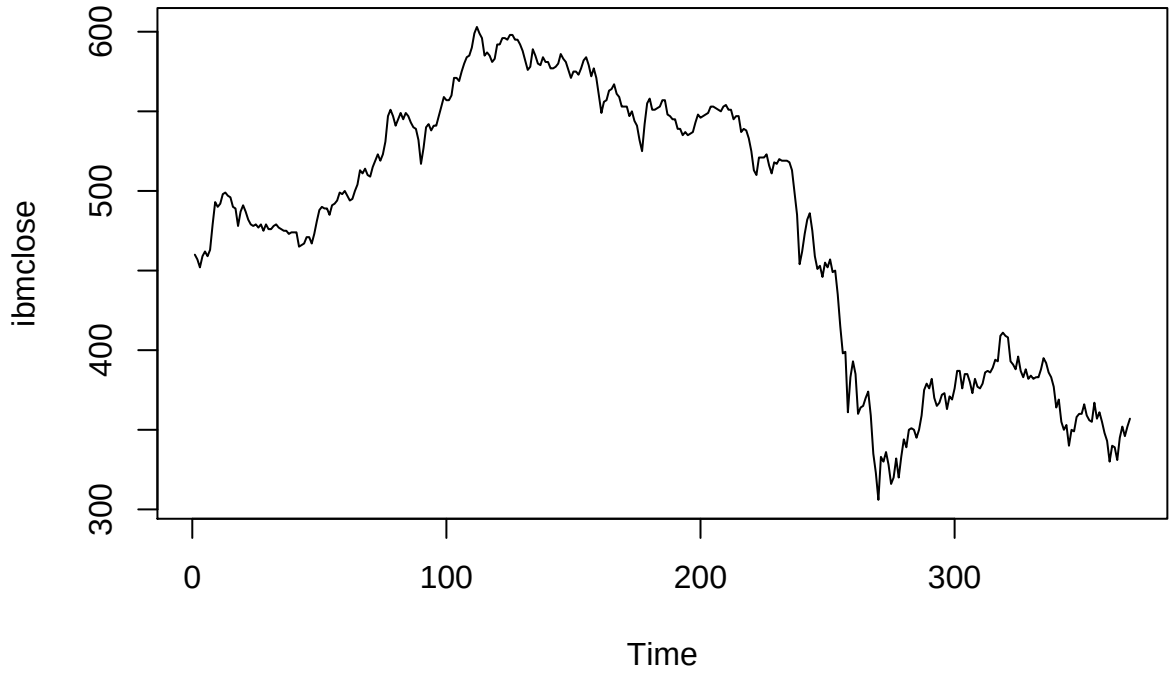
Answer: I think the critical values are at different distances from the mean, because the numbers are randomly generated. In the two first cases, the randomly generated numbers are such that the results are not White Noise. So, in the figures, the critical values touch the boundaries. In the third figure, the random numbers are such that none of the critical values touch the boundary line. This makes the figures different.

So, I think randomness to be the root cause for the difference in the figures.

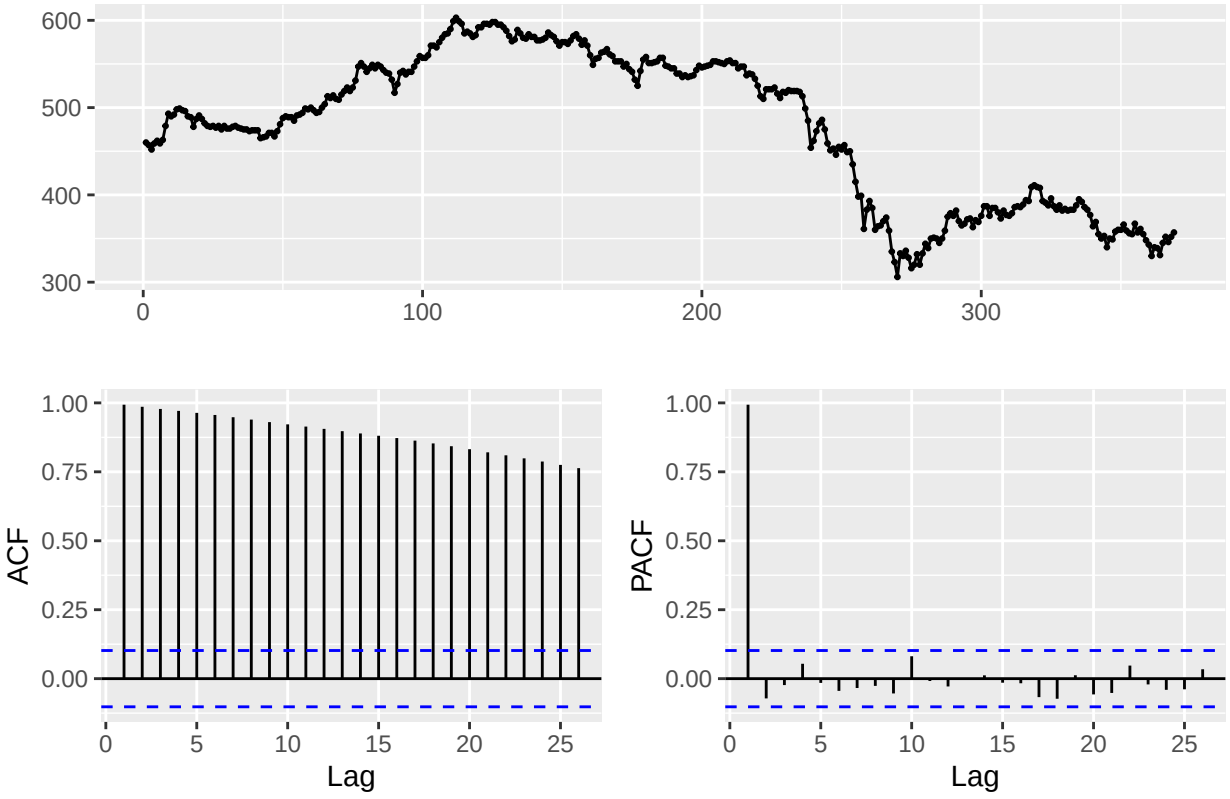
Exercise 8.2

A classic example of a non-stationary series is the daily closing IBM stock price series (data set ibmclose). Use R to plot the daily closing prices for IBM stock and the ACF and PACF. Explain how each plot shows that the series is non-stationary and should be differenced.

```
plot(ibmclose)
```




```
ggtsdisplay(ibmclose)
```



"A stationary time series is one whose properties do not depend on the time at which the series is observed. Thus, time series with trends, or with seasonality, are not stationary..." (page 223, printed copy).

Contrary to the above definition, the plot of *ibmclose* is seasonal, downward trending. Therefore, it's not sttionary. Furthermore, ACF plot shows that 95% of the spikes have crossed the boundary, which suggests that the daily change in IBM stock price is random and uncorrelated with the price of the previous day. The PACF shows that there is a strong correlation with 1 lagged values.

So, IBM prices are predicted by 1 lagged values and non-stationary.

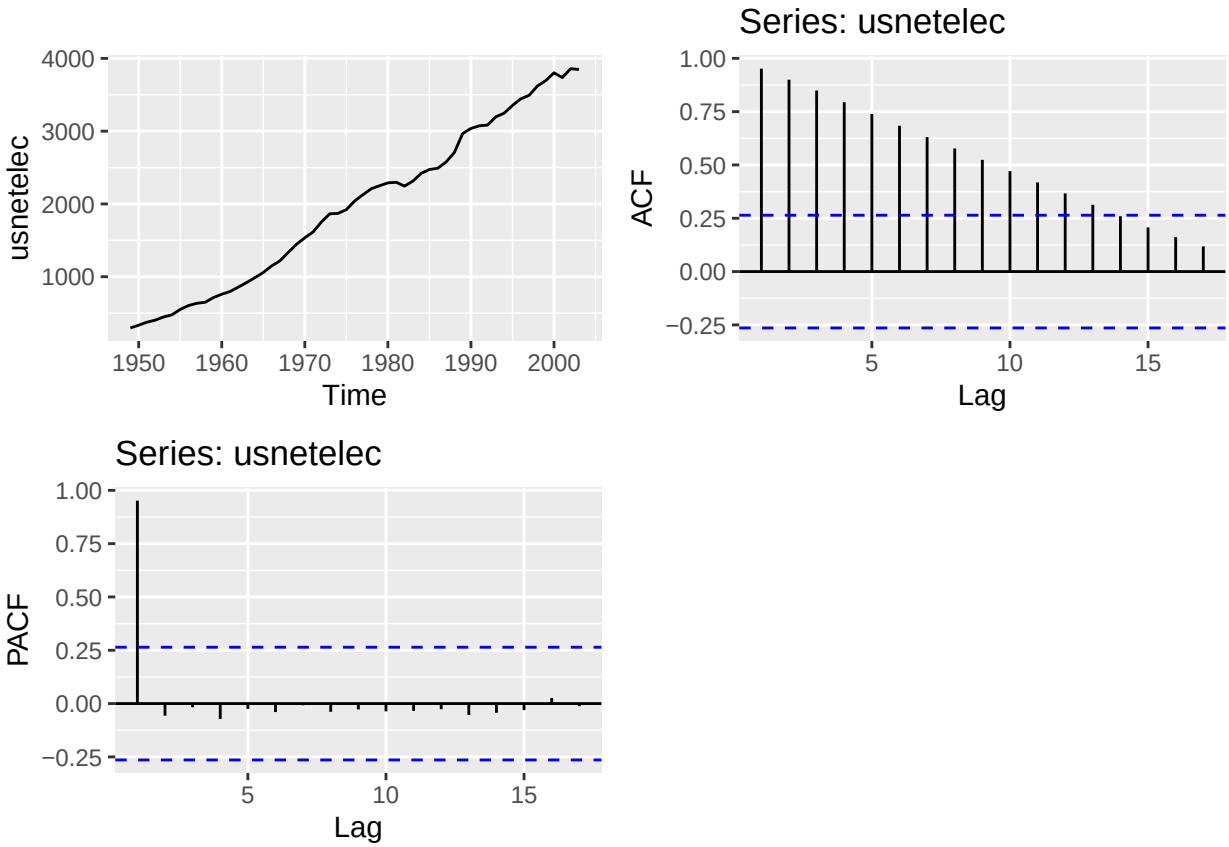
Exercise 8.3

For the following series, find an appropriate Box-Cox transformation and order of differencing in order to obtain stationary data.
a.usnetelec
b.usgdp
c.mcopper
d.enplanements
e.visitors

Answer a:

First plot of usnetelec, without differencing.

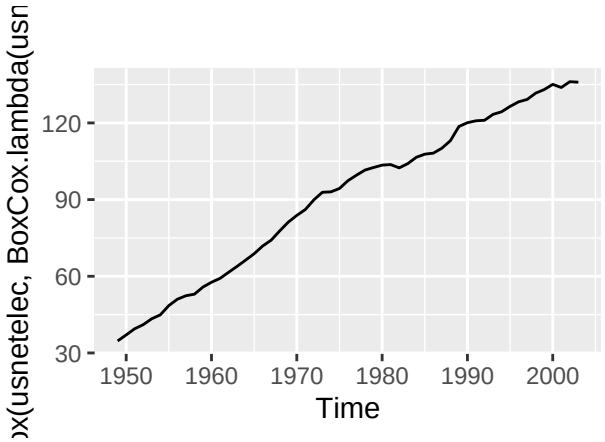
```
grid.arrange(autoplot(usnetelec), ggAcf(usnetelec), ggPacf(usnetelec), nrow = 2)
```

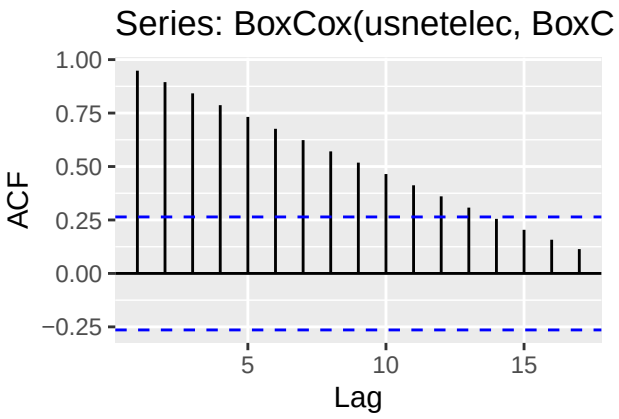


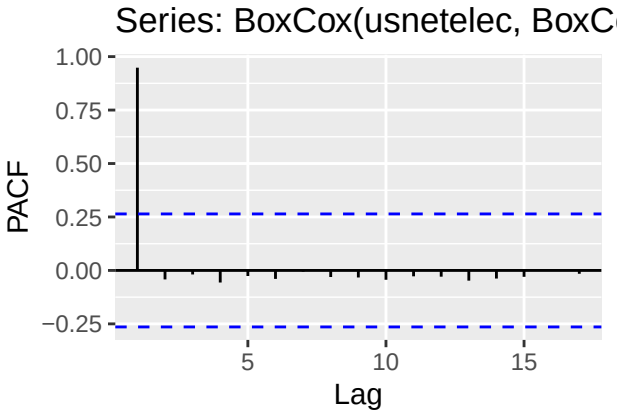
The graph is almost linearly increasing, with no seasonality. The ACF plot shows that majority of the spikes have crossed the critical values (blue dash-line) and are decreasing.

Now, let me observe the effect of doing BoxCox transformation.

```
grid.arrange(autoplot(BoxCox(usnetelec, BoxCox.lambda(usnetelec))), ggAcf(BoxCox(usnetelec, BoxCox.lambd
```







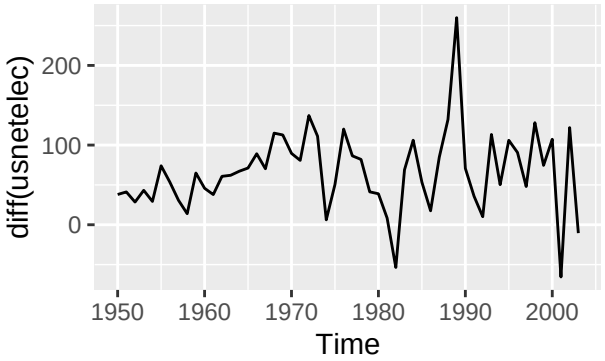
BoxCox transformation didn't produce a significant difference.

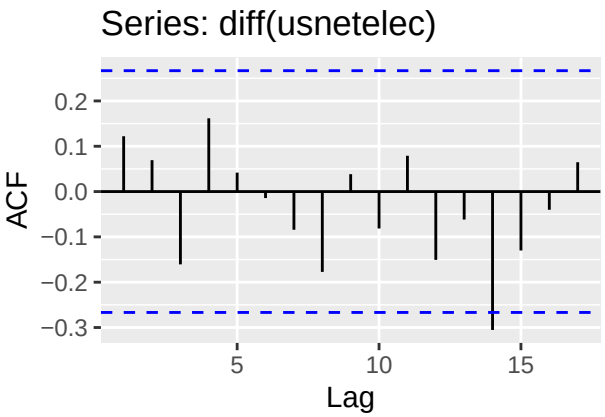
```
print(paste0('Number of differencing: ', ndiffs(usnetelec)))
```

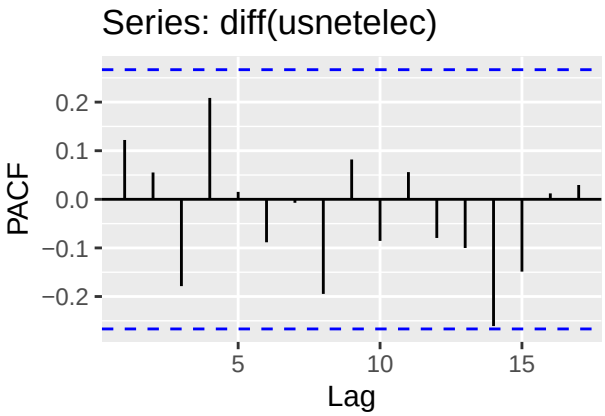
```
## [1] "Number of differencing: 1"
```

Now, I'll try to difference the time series and see if I can make it stationary.

```
grid.arrange(autoplot(diff(usnetelec)), ggAcf(diff(usnetelec)), ggPacf(diff(usnetelec)), nrow = 2)
```



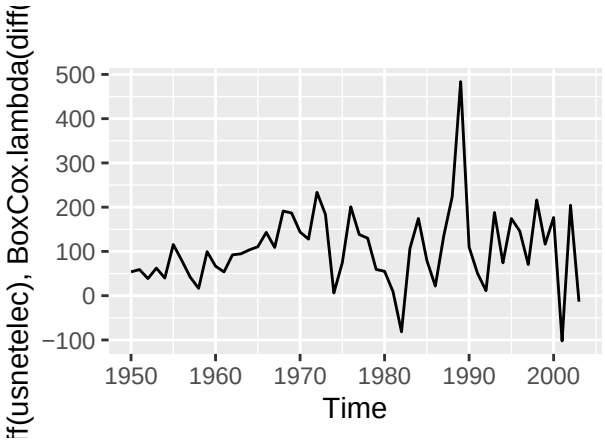


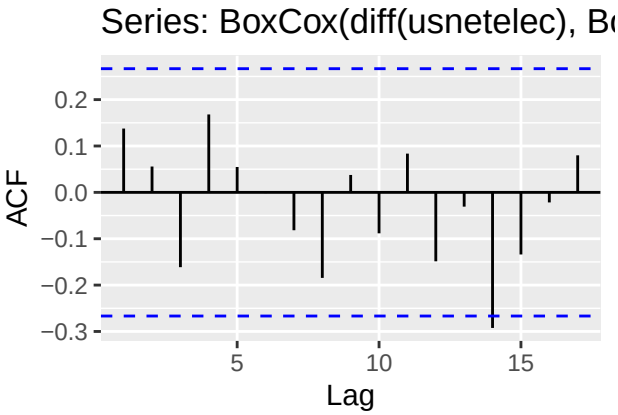


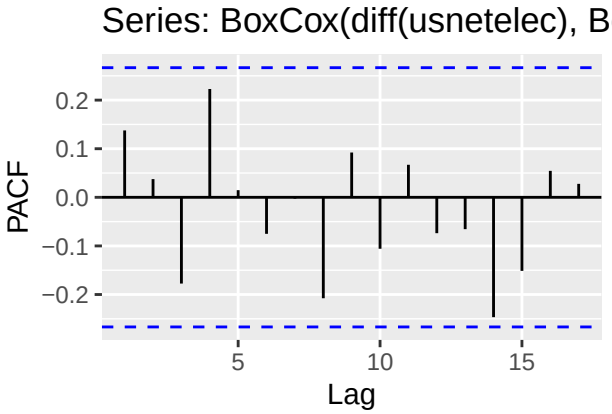
So, by ordinary differencing once, I am able to get stationary plot.

Now, I'll do a BoxCox again on the differenced time series.

```
grid.arrange(autoplot(BoxCox(diff(usnetelec), BoxCox.lambda(diff(usnetelec)))), ggAcf(BoxCox(diff(usnet
```







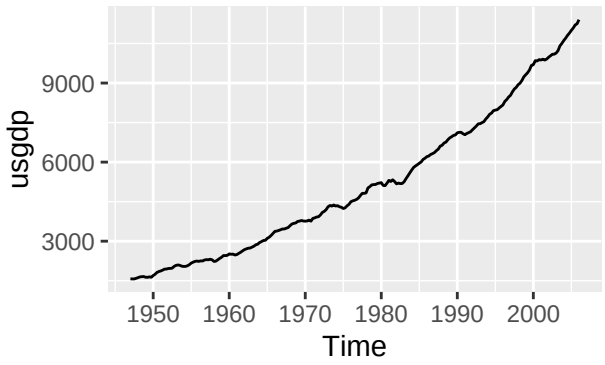
Didn't make any difference. So, it's enough to do one differencing, BoxCox transformation is not required.

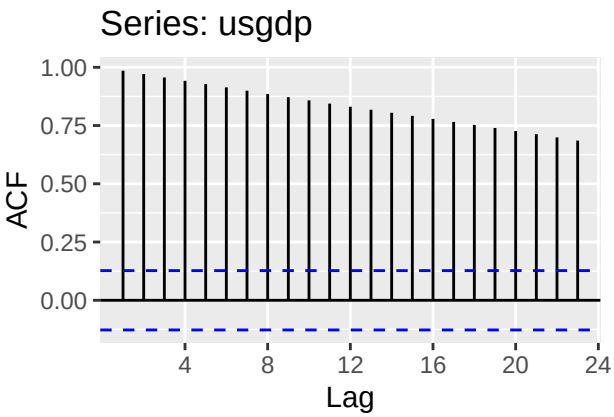
The order of differencing is 1.

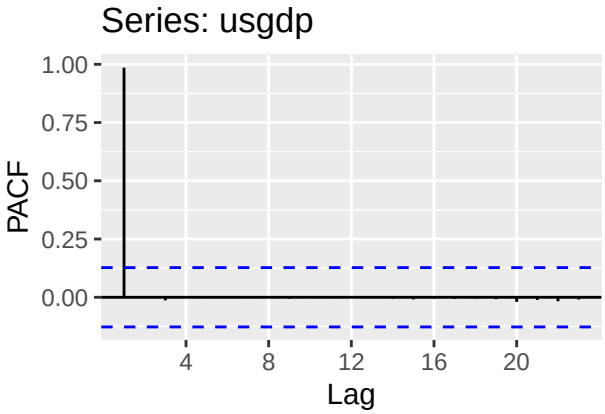
Answer b:

First plot of usgdp, without differencing.

```
grid.arrange(autoplot(usgdp), ggAcf(usgdp), ggPacf(usgdp), nrow = 2)
```







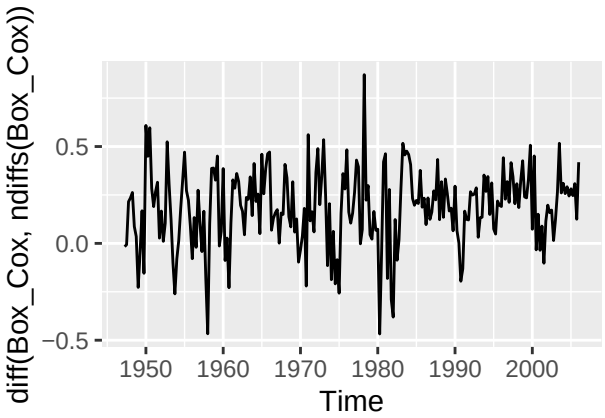
The graph is curving up, and has no seasonality. The ACF plot shows that majority of the spikes have crossed the critical values (blue dash-line) and are decreasing.

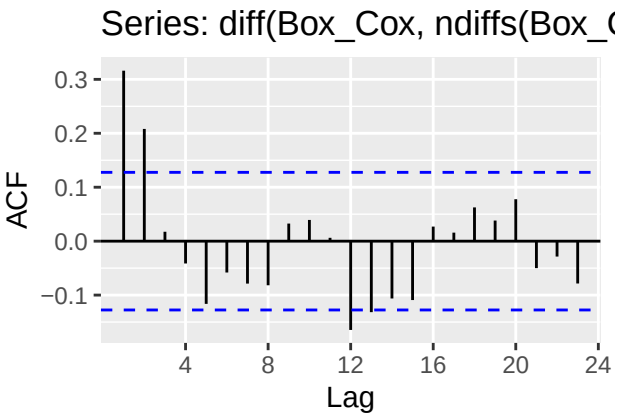
```
print(paste0('Number of differencing: ', ndiffs(usgdp)))
```

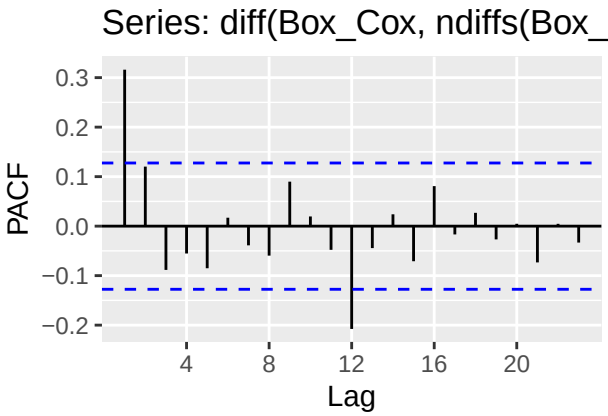
```
## [1] "Number of differencing: 2"
```

Now, I'll apply BoxCox transformation, with differencing.

```
Box_Cox_lambda <- BoxCox.lambda(usgdp)
Box_Cox <- BoxCox(usgdp, Box_Cox_lambda)
grid.arrange(autoplot(diff(Box_Cox, ndiffs(Box_Cox))), ggAcf(diff(Box_Cox, ndiffs(Box_Cox))), ggPacf(di
```







There are a few spikes that have crossed the ciritcal value.

So, let me check whether this is indeed stationary.

```
ifelse(kpss.test(diff(Box_Cox, ndiffs(Box_Cox)))$p.value > 0.05, "Stationary", "Not stationaey")
```

```
## Warning in kpss.test(diff(Box_Cox, ndiffs(Box_Cox))): p-value greater than
## printed p-value
```

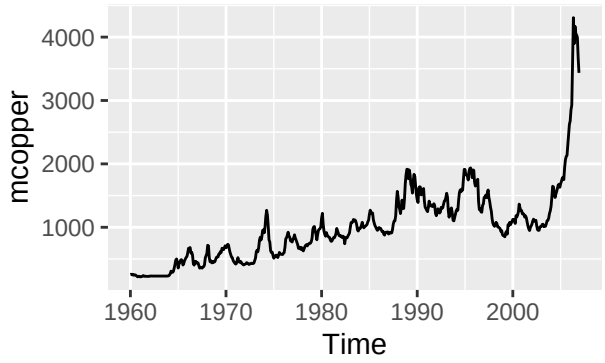
```
## [1] "Stationary"
```

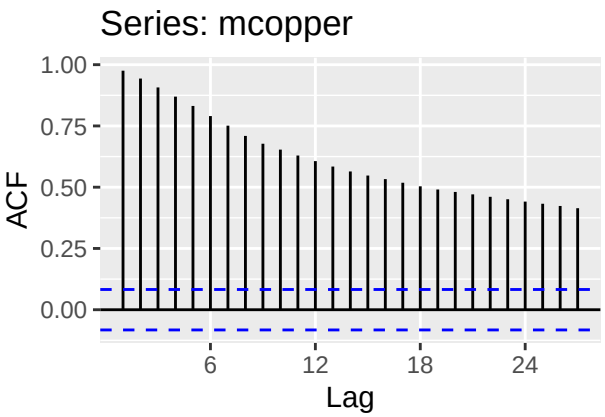
The lambda is 0.366352 and the order of differencing is 2.

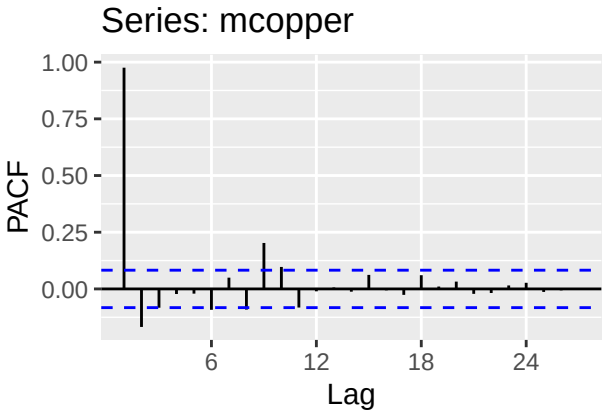
Answer c:

First plot of mcopper, without differencing.

```
grid.arrange(autoplot(mcopper), ggAcf(mcopper), ggPacf(mcopper), nrow = 2)
```







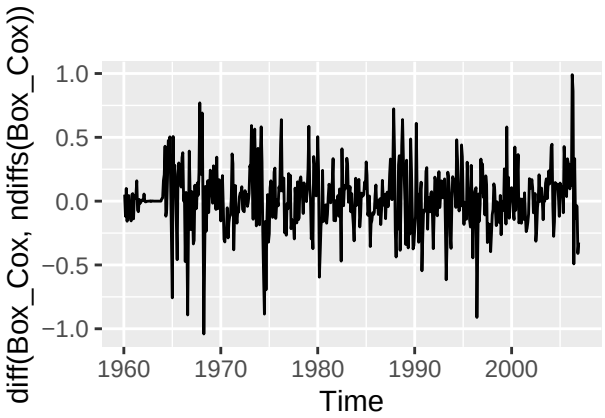
The graph is curving up, and has no seasonality. The ACF plot shows that majority of the spikes have crossed the critical values (blue dash-line) and are decreasing.

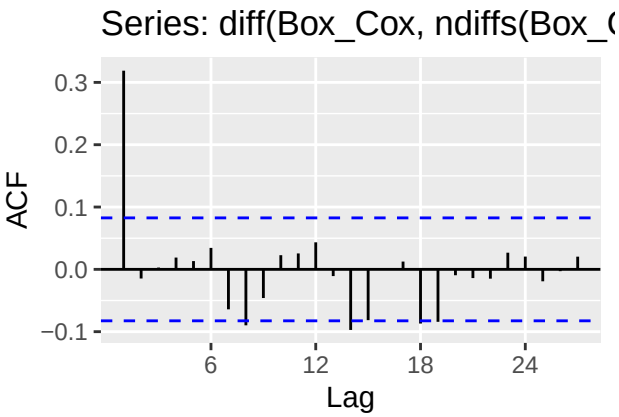
```
print(paste0('Number of differencing: ', ndiffs(mcopper)))
```

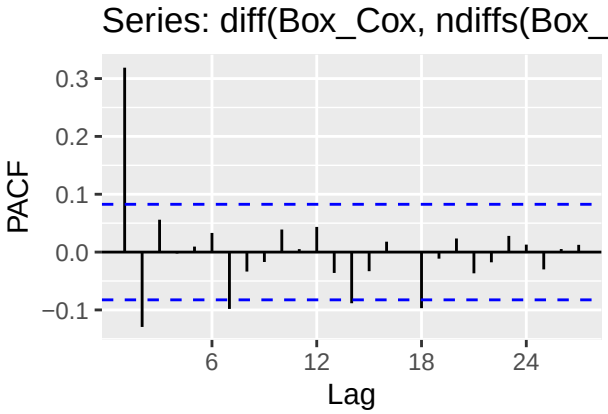
```
## [1] "Number of differencing: 1"
```

Now, I'll apply BoxCox transformation, with differencing.

```
Box_Cox_lambda <- BoxCox.lambda(mcopper)
Box_Cox <- BoxCox(mcopper, Box_Cox_lambda)
grid.arrange(autoplot(diff(Box_Cox, ndiffs(Box_Cox))), ggAcf(diff(Box_Cox, ndiffs(Box_Cox))), ggPacf(di
```







There are a few spikes that have crossed the ciritcal value.

So, let me check whether this is indeed stationary.

```
ifelse(kpss.test(diff(Box_Cox, ndiffs(Box_Cox)))$p.value > 0.05, "Stationary", "Not stationaey")
```

```
## Warning in kpss.test(diff(Box_Cox, ndiffs(Box_Cox))): p-value greater than
## printed p-value
```

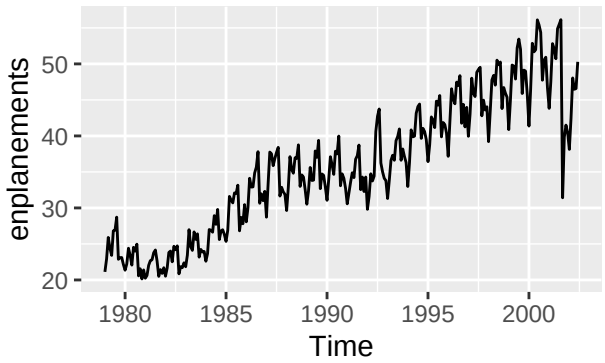
```
## [1] "Stationary"
```

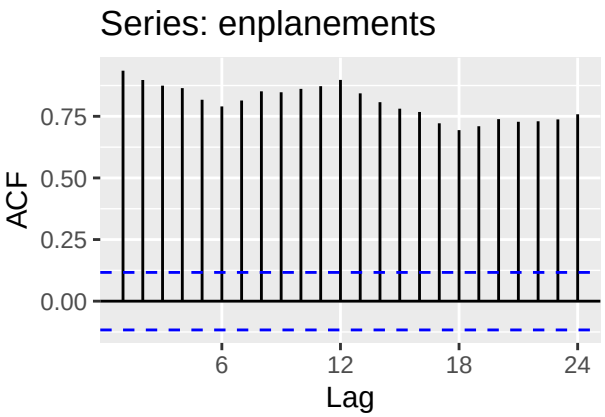
The lambda is 0.1919047 and the order of differencing is 1.

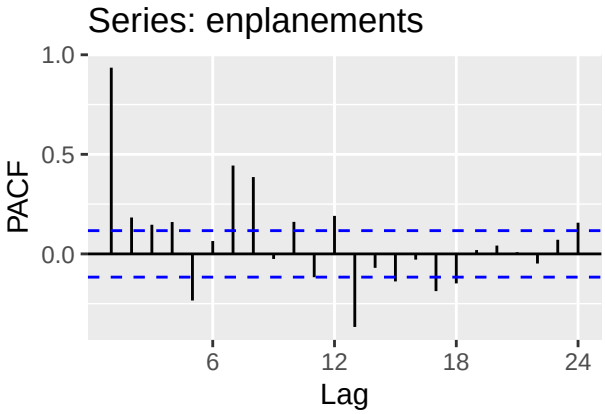
Answer d:

First plot of enplanements, without differencing.

```
grid.arrange(autoplot(enplanements), ggAcf(enplanements), ggPacf(enplanements), nrow = 2)
```







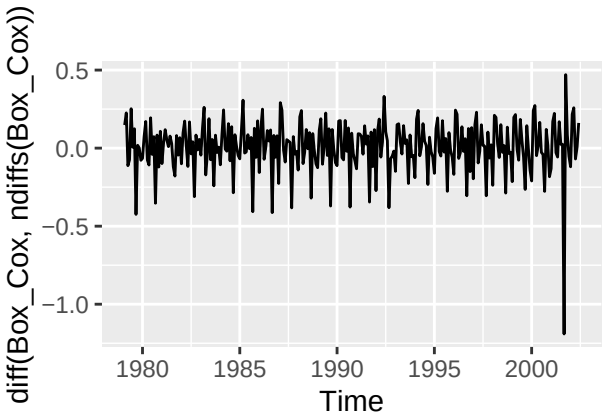
The graph is curving up, and has seasonality. The ACF plot shows that majority of the spikes have crossed the critical values (blue dash-line).

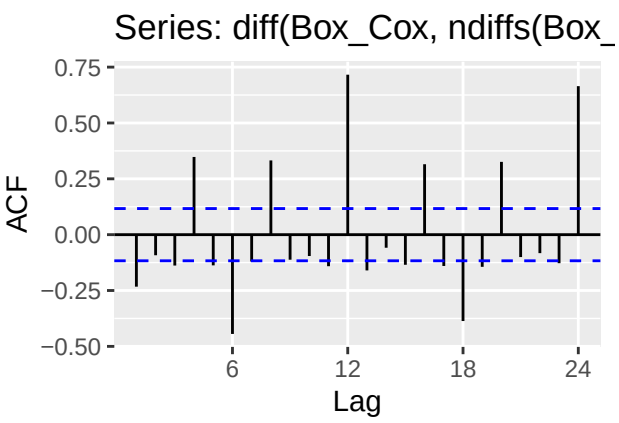
```
print(paste0('Number of differencing: ', ndiffs(enplanements)))
```

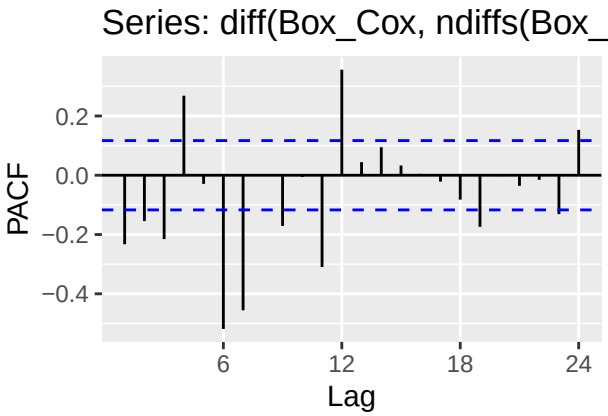
```
## [1] "Number of differencing: 1"
```

Now, I'll apply BoxCox transformation, with differencing.

```
Box_Cox_lambda <- BoxCox.lambda(mcopper)
Box_Cox <- BoxCox(enplanements, Box_Cox_lambda)
grid.arrange(autoplot(diff(Box_Cox, ndiffs(Box_Cox))), ggAcf(diff(Box_Cox, ndiffs(Box_Cox))), ggPacf(di
```







There are a few spikes that have crossed the ciritcal value.

So, let me check whether this is indeed stationary.

```
ifelse(kpss.test(diff(Box_Cox, ndiffs(Box_Cox)))$p.value > 0.05, "Stationary", "Not stationaey")
```

```
## Warning in kpss.test(diff(Box_Cox, ndiffs(Box_Cox))): p-value greater than
## printed p-value
```

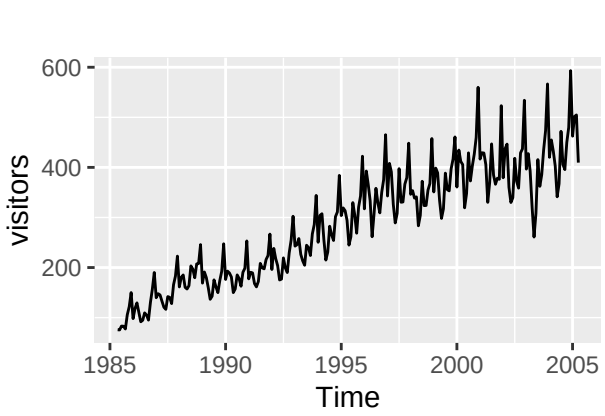
```
## [1] "Stationary"
```

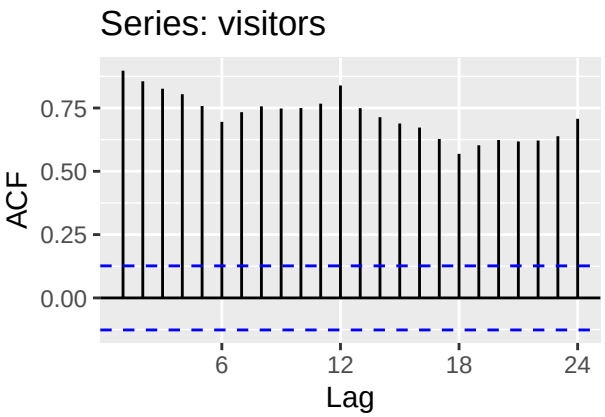
The lambda is 0.1919047 and the order of differencing is 1.

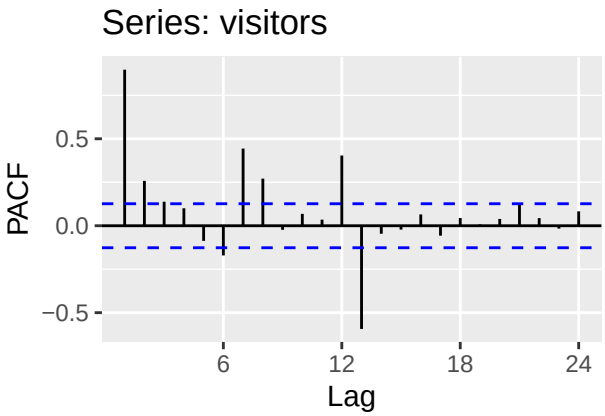
Answer e:

First plot of visitors, without differencing.

```
grid.arrange(autoplot(visitors), ggAcf(visitors), ggPacf(visitors), nrow = 2)
```







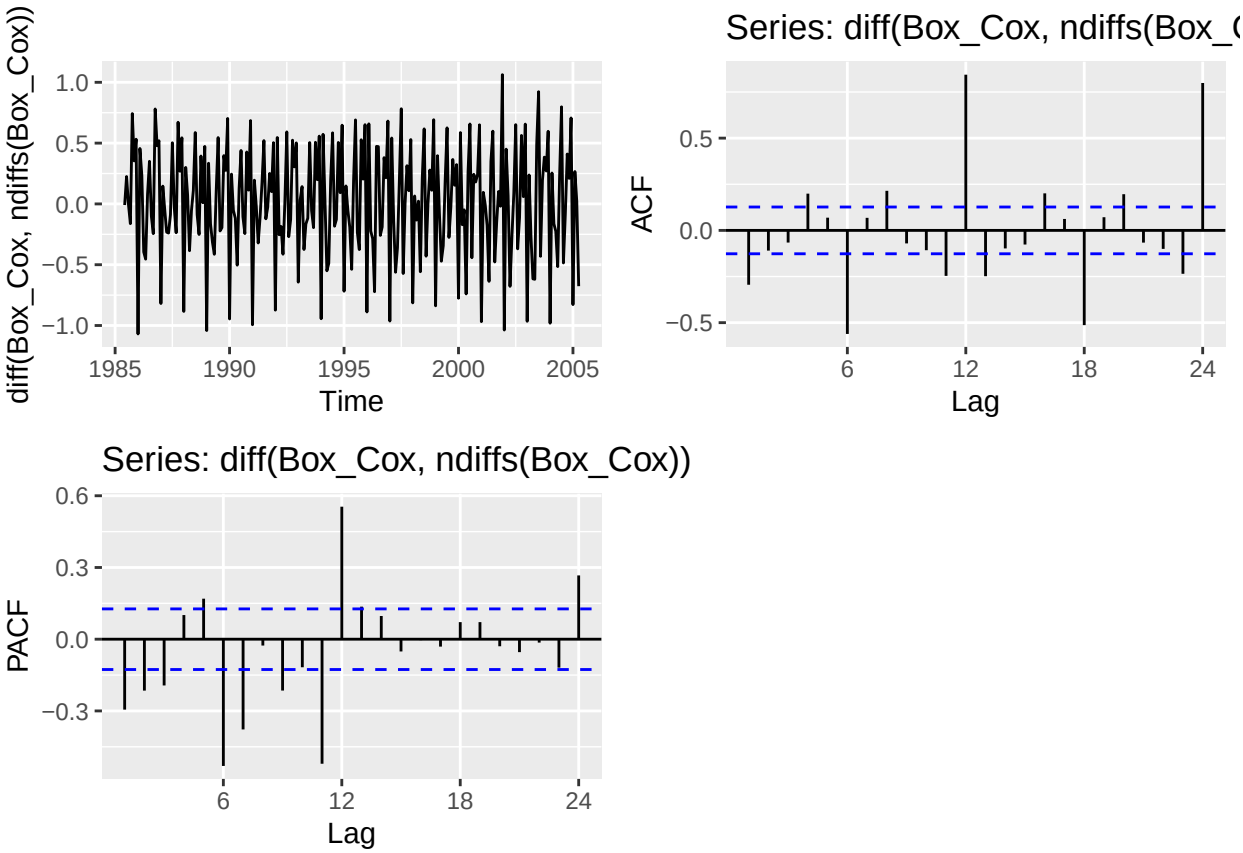
The graph is curving up, and has seasonality. The ACF plot shows that majority of the spikes have crossed the critical values (blue dash-line).

```
print(paste0('Number of differencing: ', ndiffs(visitors)))
```

```
## [1] "Number of differencing: 1"
```

Now, I'll apply BoxCox transformation, with differencing.

```
Box_Cox_lambda <- BoxCox.lambda(mcopper)
Box_Cox <- BoxCox(visitors, Box_Cox_lambda)
grid.arrange(autoplot(diff(Box_Cox, ndiffs(Box_Cox))), ggAcf(diff(Box_Cox, ndiffs(Box_Cox))), ggPacf(di
```



There are a few spikes that have crossed the ciritcal value.

So, let me check whether this is indeed stationary.

```
ifelse(kpss.test(diff(Box_Cox, ndiffs(Box_Cox)))$p.value > 0.05, "Stationary", "Not stationaey")
```

```
## Warning in kpss.test(diff(Box_Cox, ndiffs(Box_Cox))): p-value greater than
## printed p-value
```

```
## [1] "Stationary"
```

The lambda is 0.1919047 and the order of differencing is 1.

Exercise 8.5

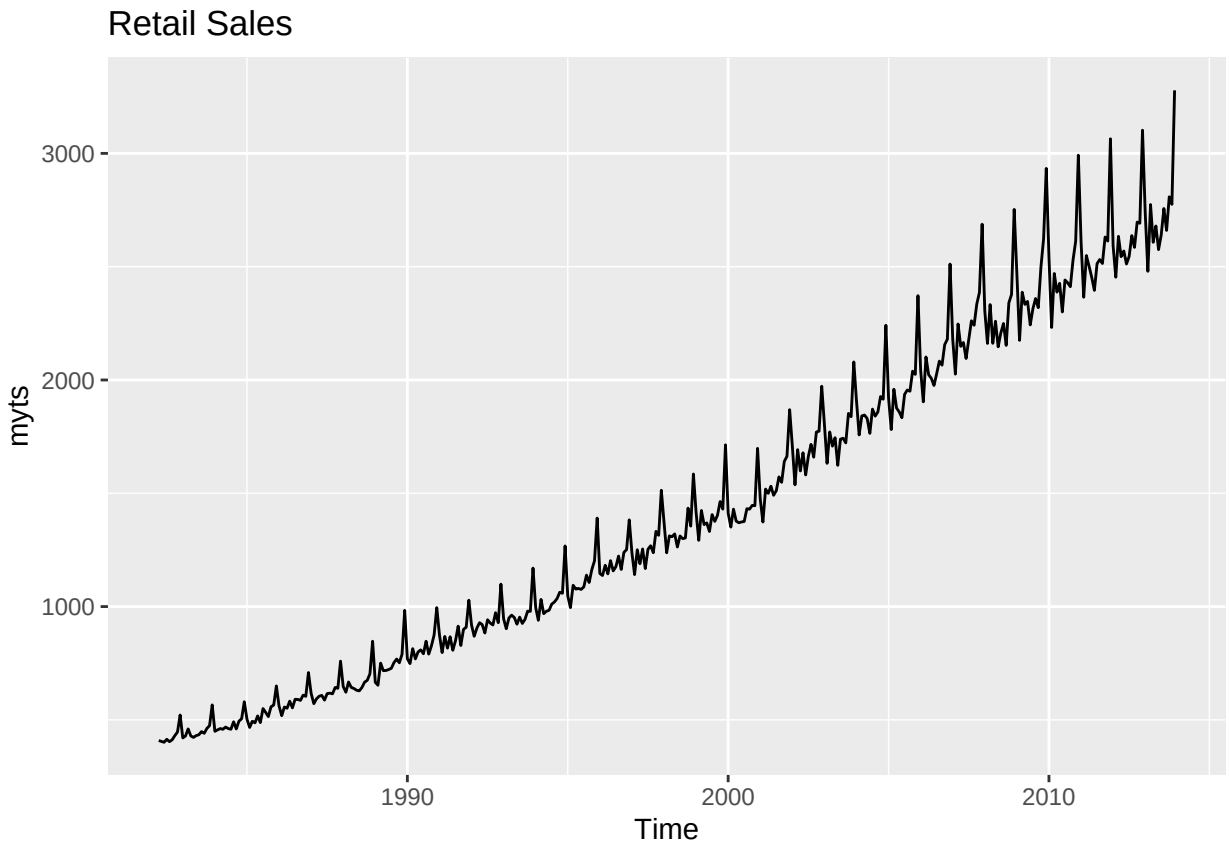
For your retail data (from Exercise 3 in Section 2.10), find the appropriate order of differencing (after transformation if necessary) to obtain stationary data.

Answer:

My code from from Exercise 3 in Section 2.10 are in the following code-chunk.

```
retail_data <- read_excel("retail.xlsx", skip = 1)
myts <- ts(retail_data[, "A3349398A"], frequency = 12, start = c(1982, 4))

autoplot(myts) + ggtitle("Retail Sales")
```

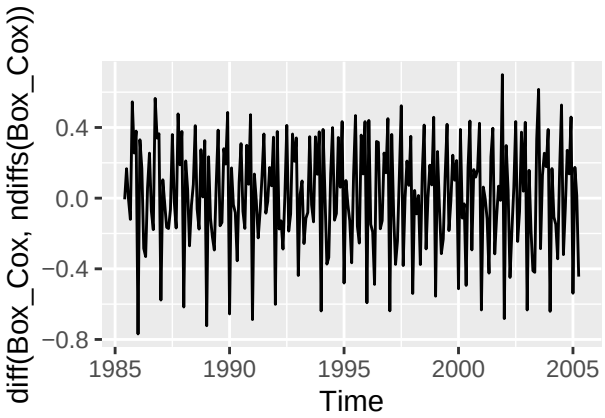


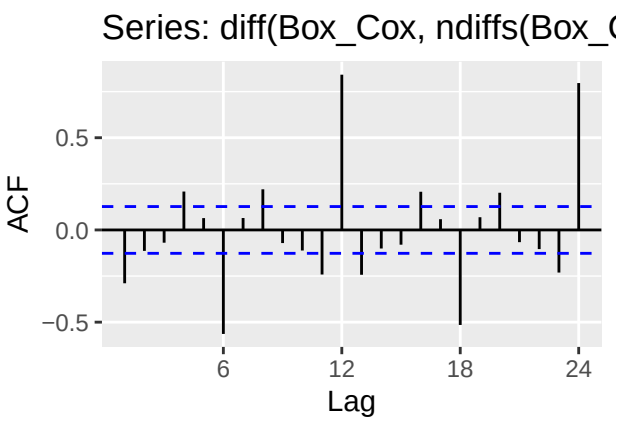
The number of differencing.

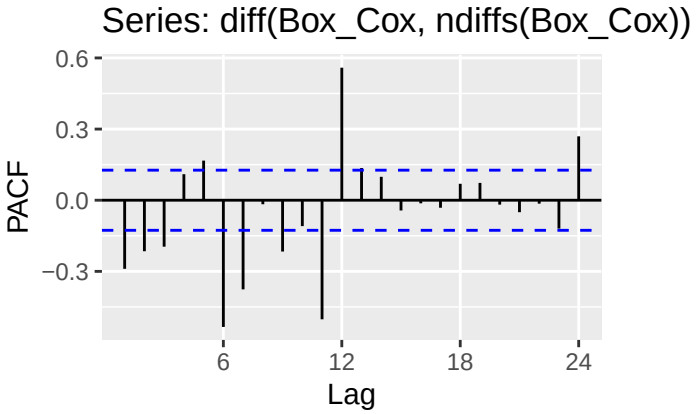
```
print(paste0('Number of differencing: ', ndiffs(myts)))
```

```
## [1] "Number of differencing: 1"
```

```
myts_lambda <- BoxCox.lambda(myts)
Box_Cox <- BoxCox(visitors, BoxCox.lambda(myts))
grid.arrange(autoplot(diff(Box_Cox, ndiffs(Box_Cox))), ggAcf(diff(Box_Cox, ndiffs(Box_Cox))), ggPacf(di
```







So, let me check whether this is indeed stationary.

```
ifelse(kpss.test(diff(Box_Cox, ndiffs(Box_Cox)))$p.value > 0.05, "Stationary", "Not stationaey")
```

```
## Warning in kpss.test(diff(Box_Cox, ndiffs(Box_Cox))): p-value greater than
## printed p-value
```

```
## [1] "Stationary"
```

The lambda is 0.1919047 and the order of differencing is 1.

Exercise 8.6

Use R to simulate and plot some data from simple ARIMA models.

a.Use the following R code to generate data from an AR(1) model with and. The process starts with $\phi_1 = 0.6$ and $\sigma^2 = 1$. The process starts with $y_1 = 0$.

```
y <- ts(numeric(100))
e <- rnorm(100)
for(i in 2:100)
y[i] <- 0.6 * y[i-1] + e[i]
```

- It's not clear where $\sigma^2 = 1$ figures in the above given code.*

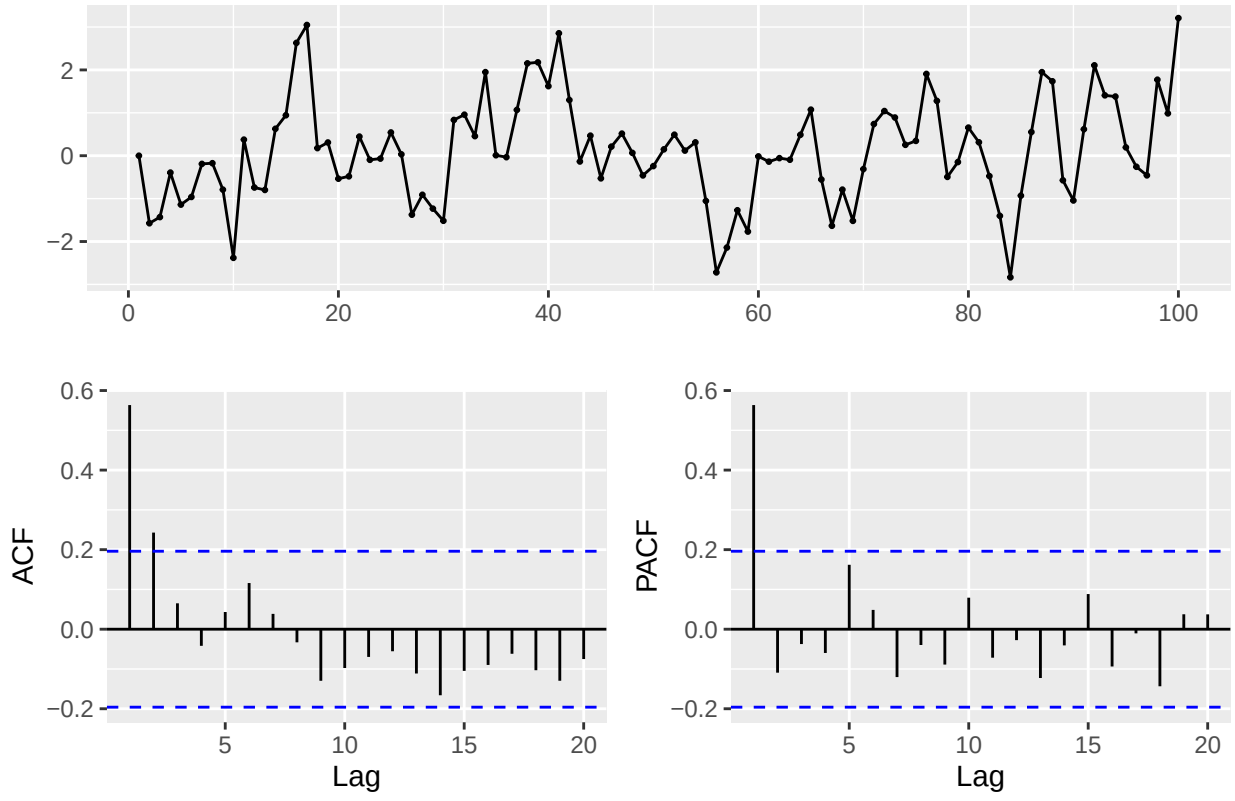
b.Produce a time plot for the series. How does the plot change as you change ϕ_1 ?

First let's create the R code to generate AR(1) data. In the following code-chunk, I am using the code given in the above code-chunk and creating a function, with phi as the input parameter.

```
set.seed(43)
ts1 <- function(phi) {
  y <- ts(numeric(100))
  e <- rnorm(100)
  for(i in 2:100) {
    y[i] <- phi * y[i - 1] + e[i]
  }
  return (y)
}
```

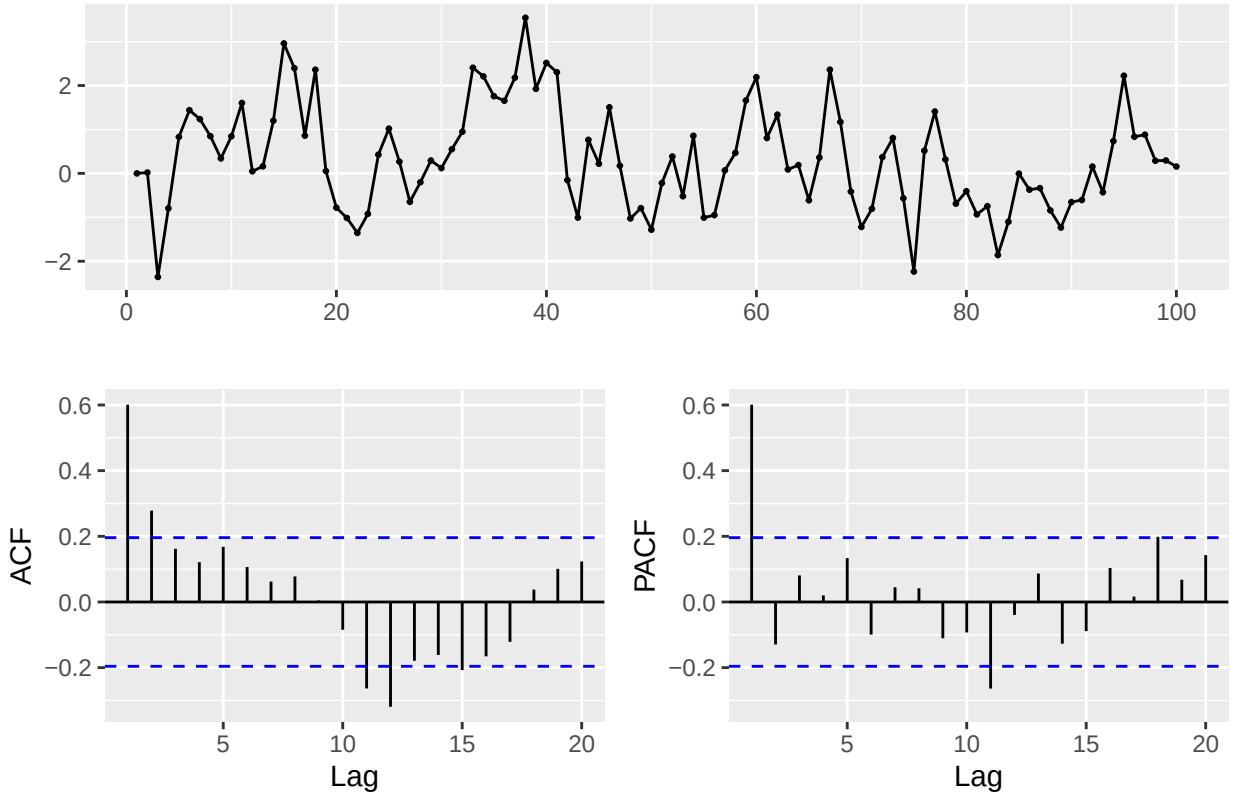
Time plot for the series with $\phi_1 = 0.6$

```
ts1(0.6) %>% ggtsdisplay()
```



Now, let's see how the plot changes, as ϕ_1 is changed (the second part of question b).

```
ts1(0.7) %>% ggtsdisplay()
```



With $\phi_1 = 0.7$, the plot changes, noticeably.

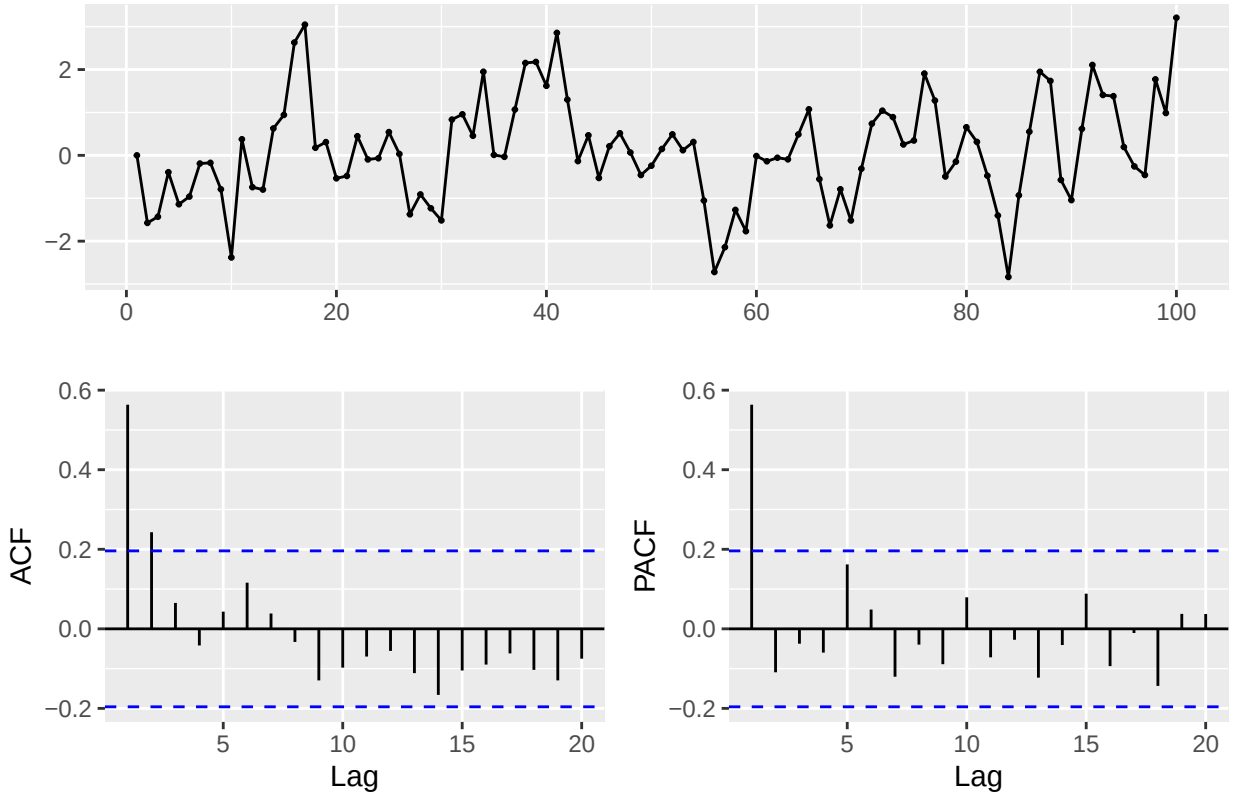
c.Write your own code to generate data from an MA(1) model with $\theta_1 = 0.6$ and $\sigma^2 = 1$.

This code should be no different from the earlier code in question b.

```
set.seed(43)
ts2 <- function(theta) {
  y <- ts(numeric(100))
  e <- rnorm(100)
  for(i in 2:100) {
    y[i] <- theta * y[i - 1] + e[i]
  }
  return (y)
}
```

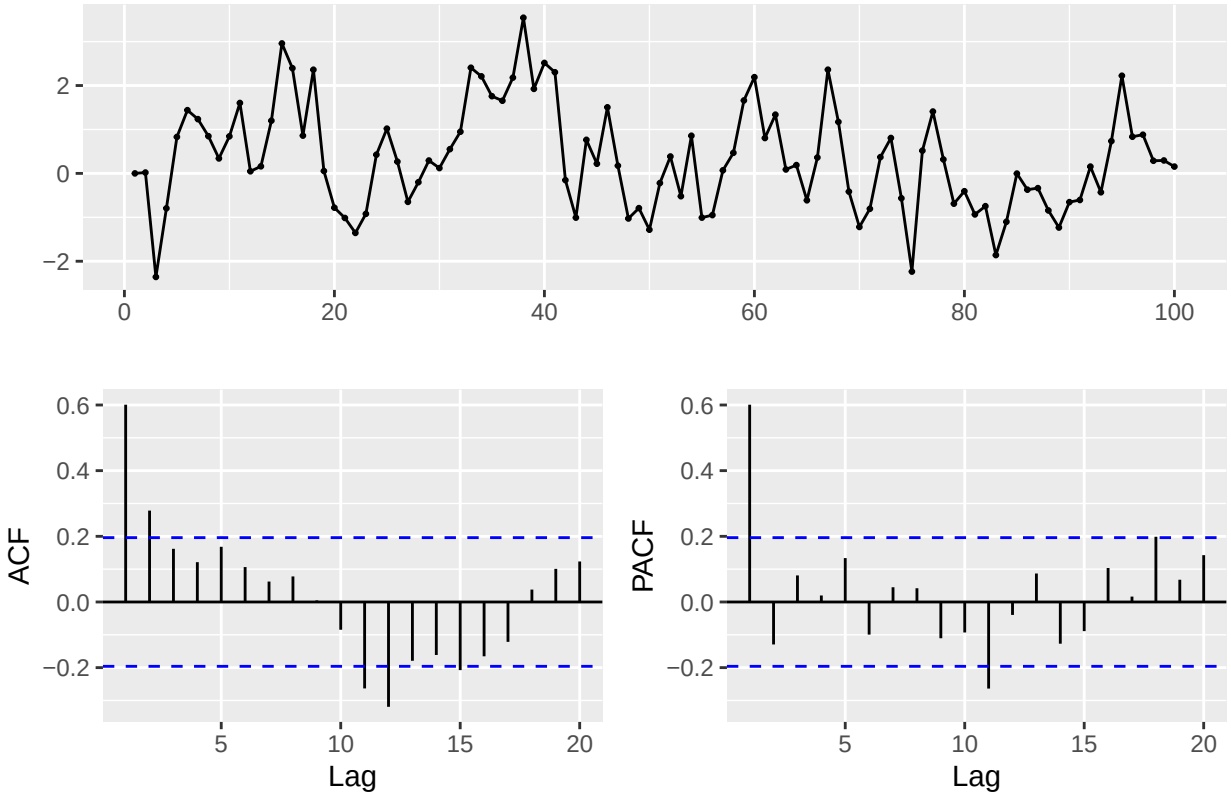
d.Produce a time plot for the series. How does the plot change as you change θ_1 ?

```
ts2(0.6) %>% ggtsdisplay()
```



Now, let's see how the plot changes, as θ_1 is changed (the second part of question d).

```
ts2(0.7) %>% ggtsdisplay()
```



With $\theta_1 = 0.7$, the plot changes, noticeably.

I really don't get the point of question c. It seems like repetition of quesion c. It's also not clear what the role of $\sigma^2 = 1$ is.

e.Generate data from an ARMA(1,1) model with $\phi_1 = 0.6$, $\theta_1 = 0.6$ and $\sigma^2 = 1$.

The code is as follows.

```
set.seed(43)
ts3 <- function(phi, theta) {
  y <- ts(numeric(100))
  e <- rnorm(100)
  e[1] <- 0
  for(i in 3:100) {
    y[i] <- phi * y[i - 1] + theta * y[i - 2] + e[i]
  }
  return (y)
}
```

f.Generate data from an AR(2) model with $\phi_1 = -0.8$ and $\phi_2 = 0.3$ and $\sigma^2 = 1$.

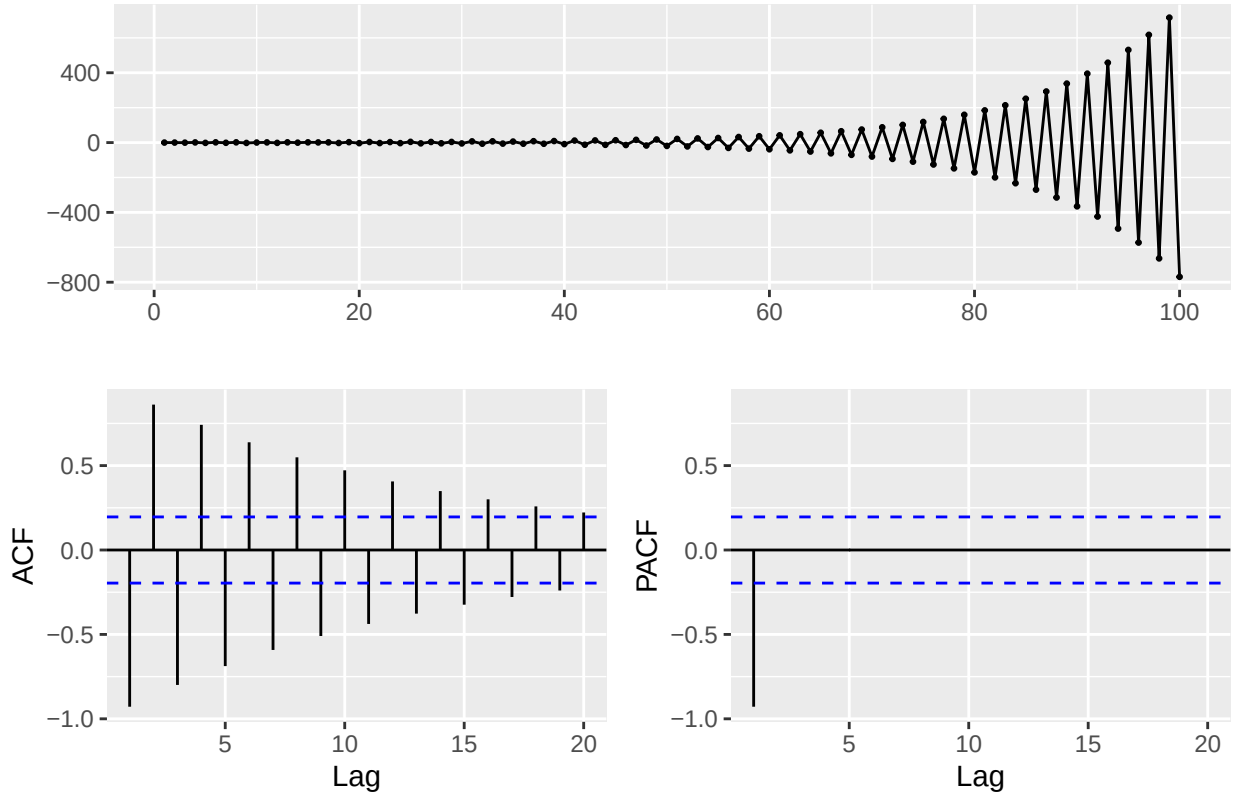
The code is as follows.

```
set.seed(43)
ts4 <- function(phi_1, phi_2) {
  y <- ts(numeric(100))
  e <- rnorm(100)
```

```
  e[1] <- 0
  for(i in 3:100) {
    y[i] <- phi_1 * y[i - 1] + phi_2 * y[i - 2] + e[i]
  }
  return (y)
}
```

g.Graph the latter two series and compare them.

```
ts4(-0.8, 0.3) %>% ggtsdisplay()
```



Exercise 8.7

Consider wmurders, the number of women murdered each year (per 100,000 standard population) in the United States.

a.By studying appropriate graphs of the series in R, find an appropriate ARIMA(p, d, q) model for these data.

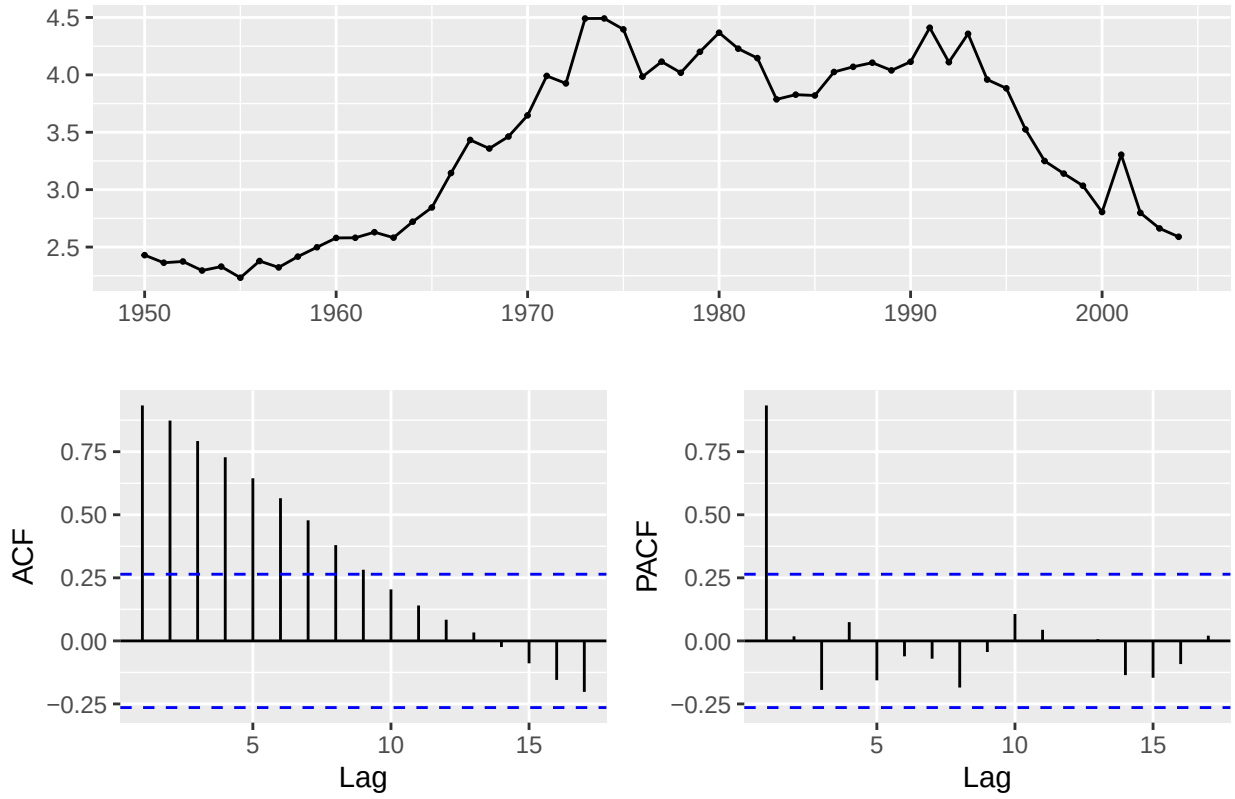
Of the 3 parameters, first let me determine d, the degree of differencing with ndiffs() function.

```
ndiffs(wmurders)
```

```
## [1] 2
```

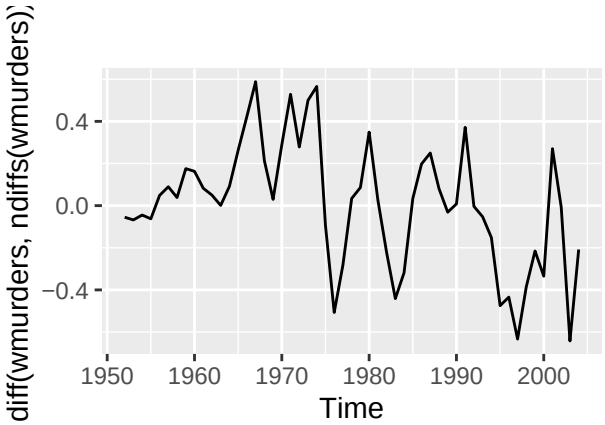
So, d = 2. That leaves are p and q, which I'll try to determine in the following.

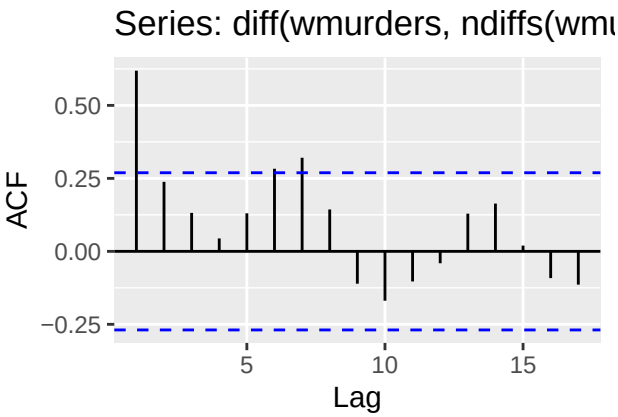
```
wmurders %>% ggtsdisplay()
```

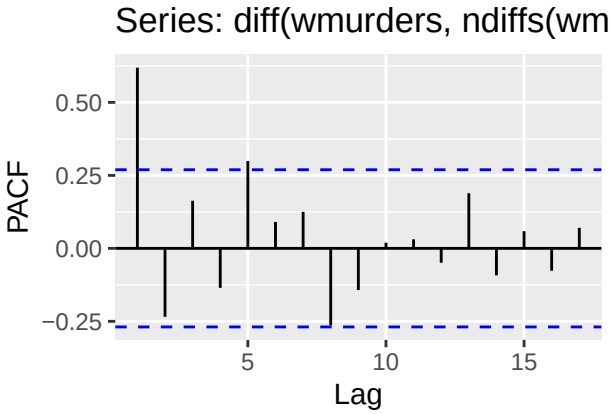


It's a uneven series, which needs to be differenced, to make the series stationary.

```
grid.arrange(autoplot(diff(wmurders, ndiffs(wmurders))), ggAcf(diff(wmurders, ndiffs(wmurders))), ggPac
```







High value of PACF indicates that p is 1.

So, in this ARIMA(p, d, q) model, p = 1, d = 2 and I can't determine q.

b.Should you include a constant in the model? Explain.

No. Explanation is on page 250 (printed copy), which says, "Inclusion of a constant in a non-stationary ARIMA model is equivalent to including a polynomial trend of order d in the forecast function. (If the constant is omitted, the forecast function includes a polynomial trend of order d - 1)".

c.Write this model in terms of the backshift operator.

Since degree of differencing equals 2, we need two backshift operations. So, the formula involving backshift operator is as follows:

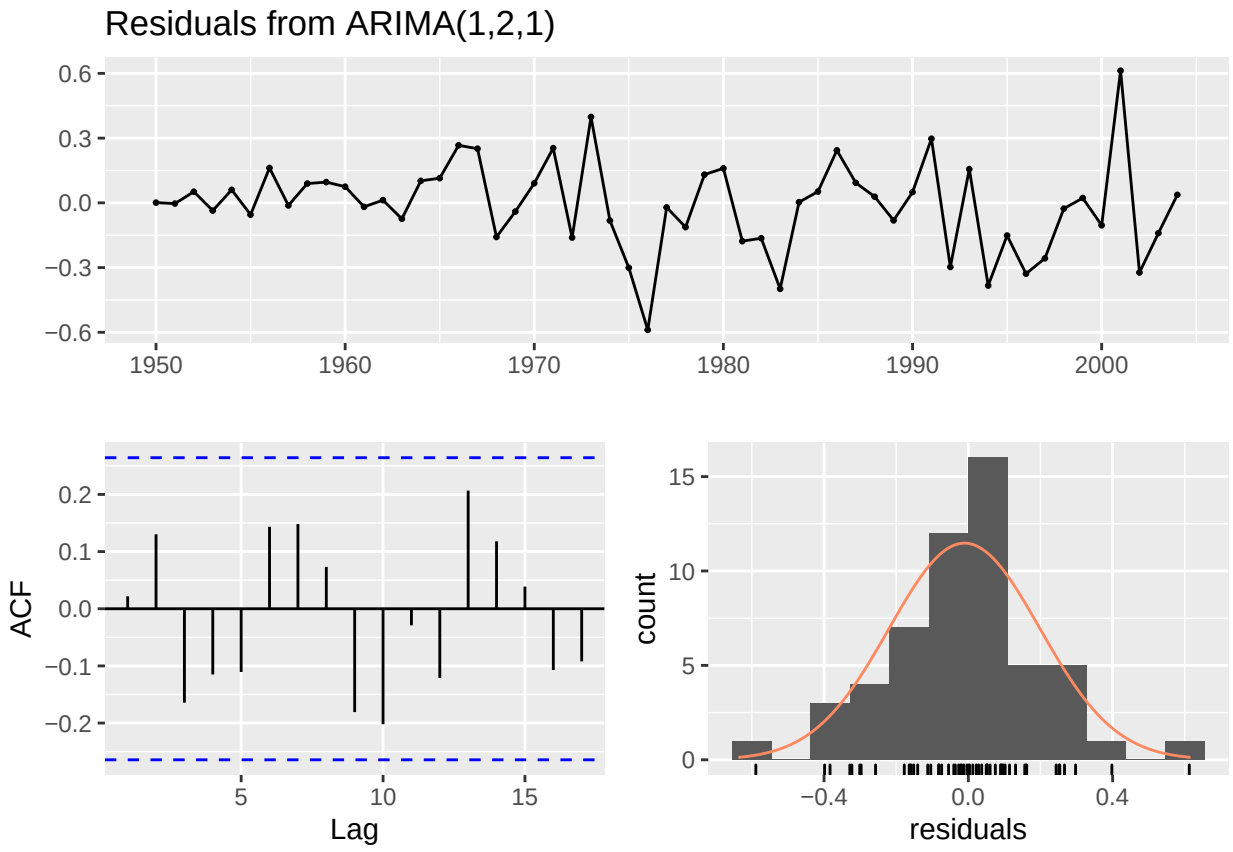
$(1 - \phi_1 B)(1 - B)^2 y_t = c + (1 + \theta_1 B)e_t$

This is deduced from the equation 8.2 on page 237 (printed copy).

d.Fit the model using R and examine the residuals. Is the model satisfactory?

Earlier I obtained d = 2 and p = 1 and q was not determined. So, assuming q = 1, we get the following model.

```
fit_arima <- arima(wmurders, order = c(1, 2, 1))
checkresiduals(fit_arima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,2,1)
## Q* = 12.419, df = 8, p-value = 0.1335
##
## Model df: 2.   Total lags used: 10
```

Observations:
- The spikes are within the boundaries (blue dash-lines), which means the residuals are White Noise.
- The p-value is 0.1335.
- The histogram is approximately Normal.

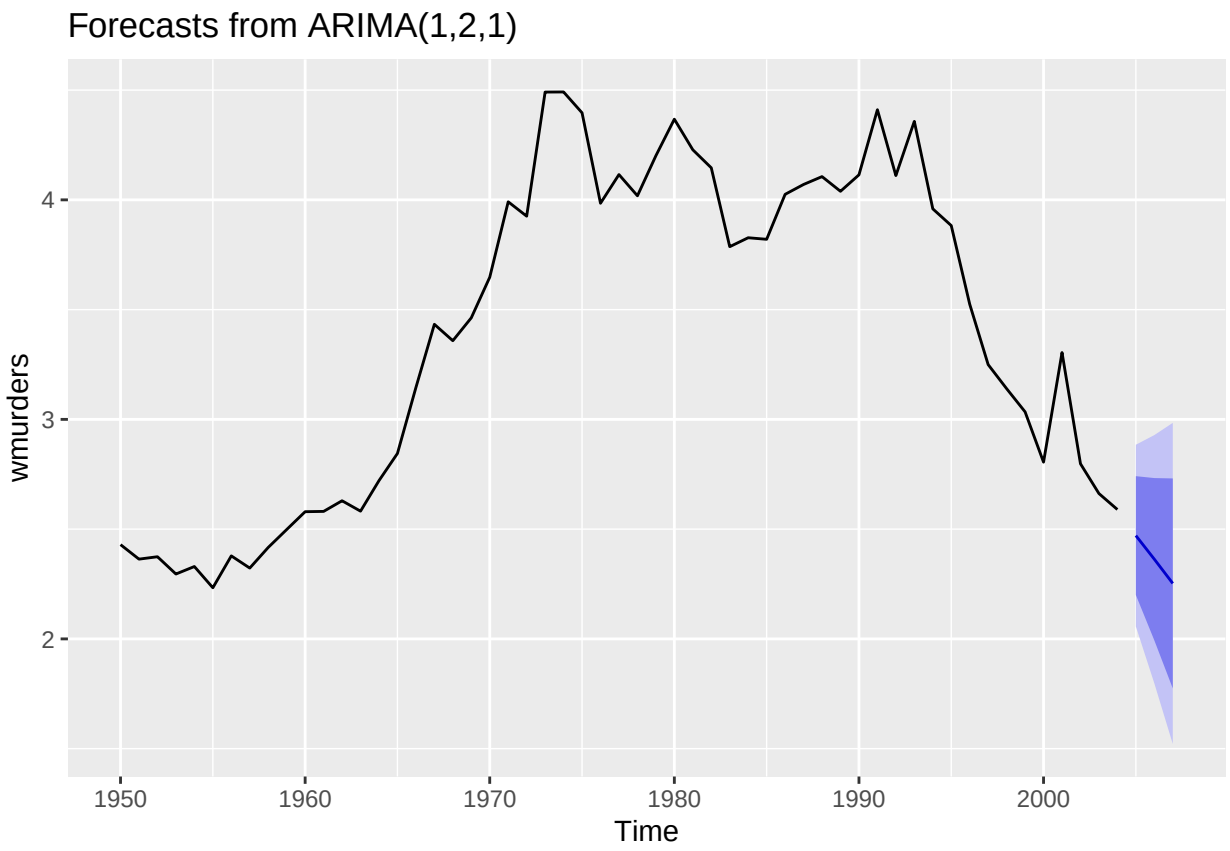
e.Forecast three times ahead. Check your forecasts by hand to make sure that you know how they have been calculated.

```
forecast(fit_arima, h = 3)
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2005       2.470660 2.200091 2.741229 2.056861 2.884459
## 2006       2.363106 1.993529 2.732684 1.797886 2.928327
## 2007       2.252833 1.774677 2.730989 1.521557 2.984110
```

f.Create a plot of the series with forecasts and prediction intervals for the next three periods shown.

```
autoplot(forecast(fit_arima, h = 3))
```



g.Does auto.arima() give the same model you have chosen? If not, which model do you think is better?

```
auto.arima(wmurders)
```

```
## Series: wmurders
## ARIMA(1,2,1)
##
## Coefficients:
##           ar1      ma1
##       -0.2434  -0.8261
## s.e.   0.1553   0.1143
##
## sigma^2 estimated as 0.04632:  log likelihood=6.44
## AIC=-6.88   AICc=-6.39   BIC=-0.97
```

auto.arima(wmurders) gives the same model ARIMA(1, 2, 1).

Marker: 624-06