Bin Lin

Homework HA Chapter 2

Exercise 2.8 1. For each of the following series (from the fma package), make a graph of the data. If transforming seems appropriate, do so and describe the effect.

a. Monthly total of people on unemployed benefits in Australia (January 1956-July 1992).

b. Monthly total of accidental deaths in the United States (January 1973-December 1978).

c. Quarterly production of bricks (in millions of units) at Portland, Australia (March 1956-September 1994).

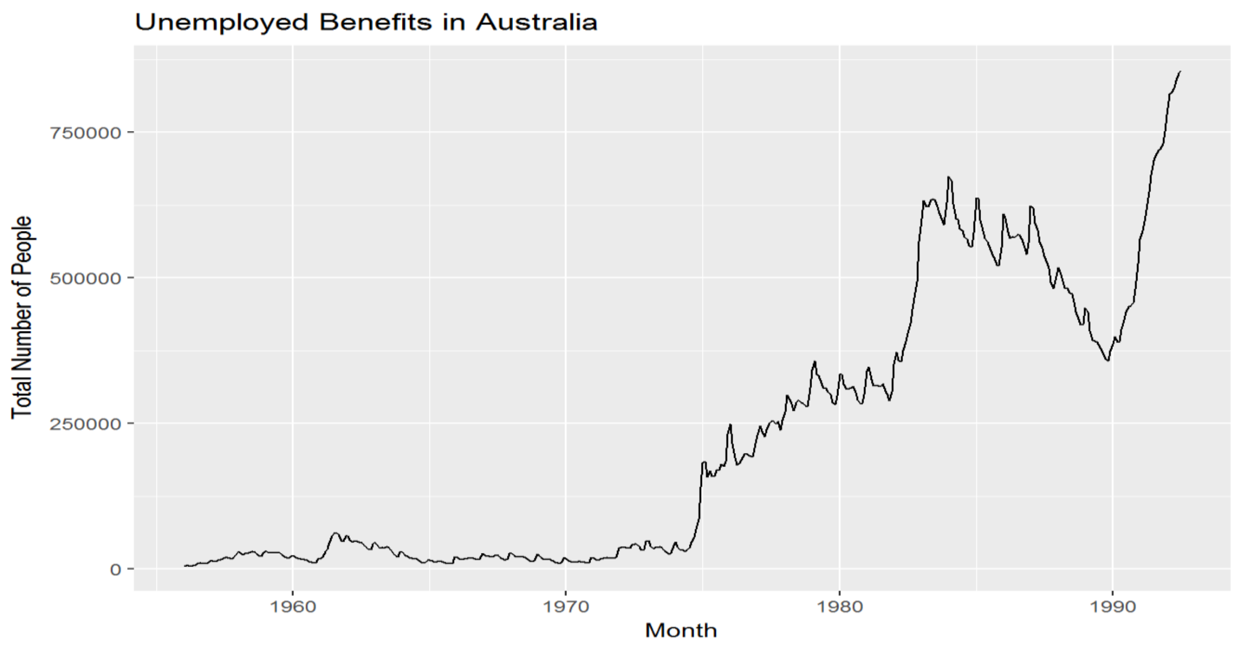
Hints: data(package="fma") will give a list of the available data. To plot a transformed data set, use plot(BoxCox(x,0.5)) where x is the name of the data set and 0.5 is the Box-Cox parameter.

a.

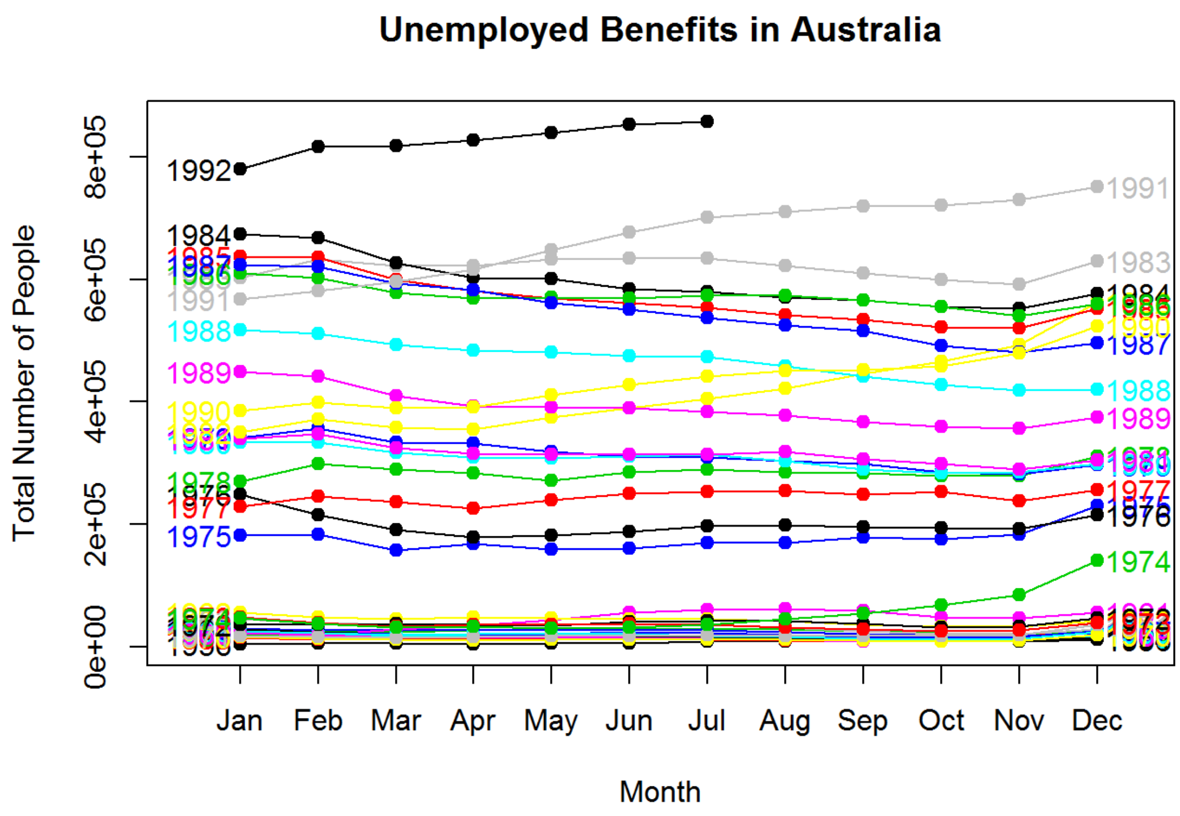
Approaches: After downloading the datasets, I printed out the Time Plot, Seasonal Plot, and Seasonal Subseries Plot before I make a decision that if data transforming is appropriate. The Time Plot is the graph that plots the observations against the time of observations, with consecutive observations joined by straight lines. Seasonal Plot is very similar to Time Plot, except its x-axis is corresponding to seasons. The Seasonal Subseries Plots a particular type of Seasonal Plots where data for each season are collected together in separate mini time plots.

Interpretations: The Time Plot a significant increase in terms of people who collect unemployment benefits in Australia, in the year of 1975, 1982-1983, and after 1990. In the meantime, there is significant decline between 1982 to 1990. The changes does not seem to be periodic or seasonal. The evidences are showed in Seasonal Plot and Seasonal Subseries Plot. The average number of people collecting unemployment benefit which is indicated by the horizontal line in Seasonal Subseries Plot shows throughout the year the observations decreases slightly with a slight increase in the month of December. Since the changes of the number of observations are huge. The differences tends to be exponential, I think it is appropriate to adopt Box-Cox transformations. The shape of the graph does not changed much, however, the scale of the y-interval falls within 0 to 300 right now.

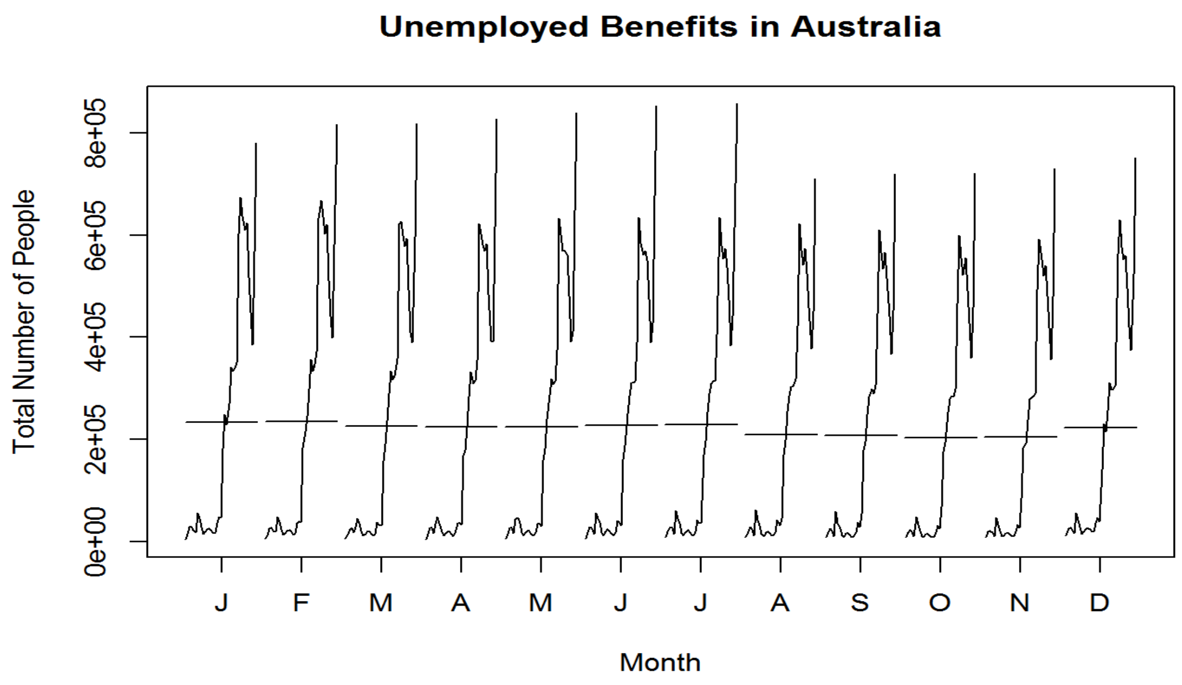
```
library("forecast")

library("xts")

library("fma")

library("ggplot2")

autoplot(dole, main="Unemployed Benefits in Australia", ylab="Total Number of People", xlab="Month")
```



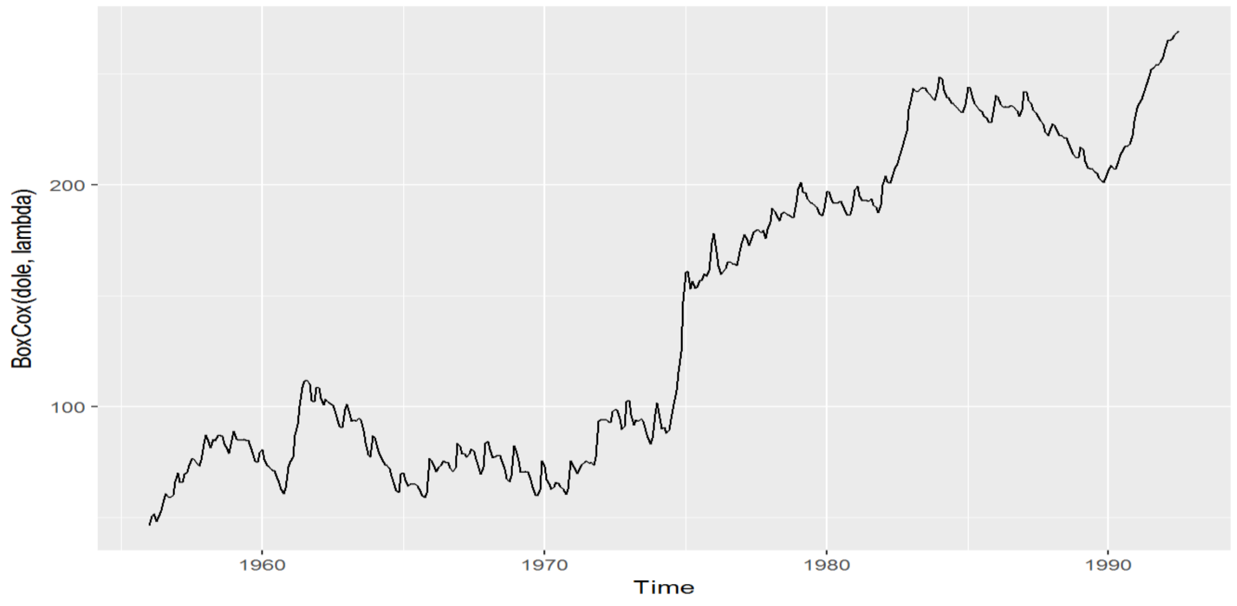
```
seasonplot(dole, main="Unemployed Benefits in Australia", ylab="Total Number
of People", xlab="Month",year.labels=TRUE, year.labels.left=TRUE, col=1:20,
pch=19)
```



```
monthplot(dole, main="Unemployed Benefits in Australia", ylab="Total Number
of People", xlab="Month")
```

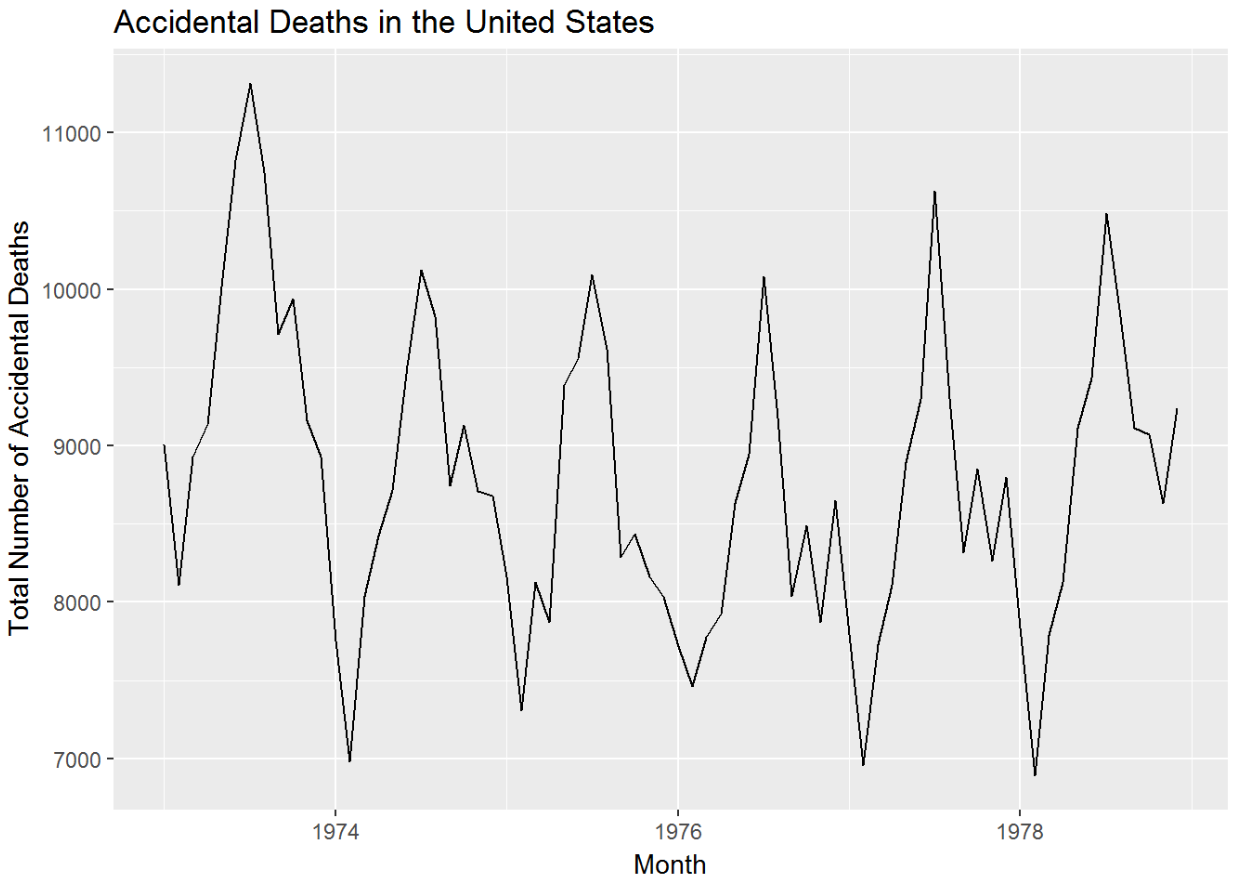


```
lambda <- BoxCox.lambda(dole)

autoplot(BoxCox(dole, lambda))
```

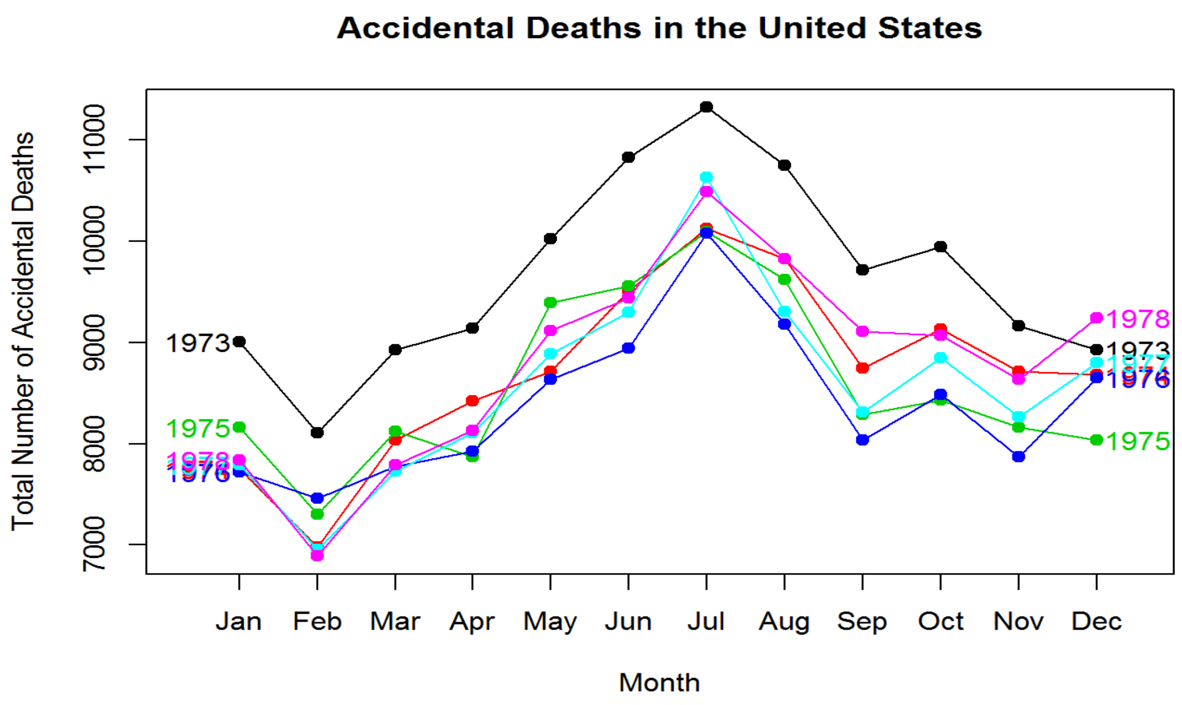


b. Interpretation: The Time Plot shows the observations oscillate over the years. There are peaks and troughs periodically. The Seasonal Plot shows for each year, the peak usually happen during the summer, especially the month of July, we will see the highest number of accidental death during that year. From the Seasonal Subseries Plot, we are able to tell that between 1973 and 1978, the accidental deaths goes down first then goes up again.

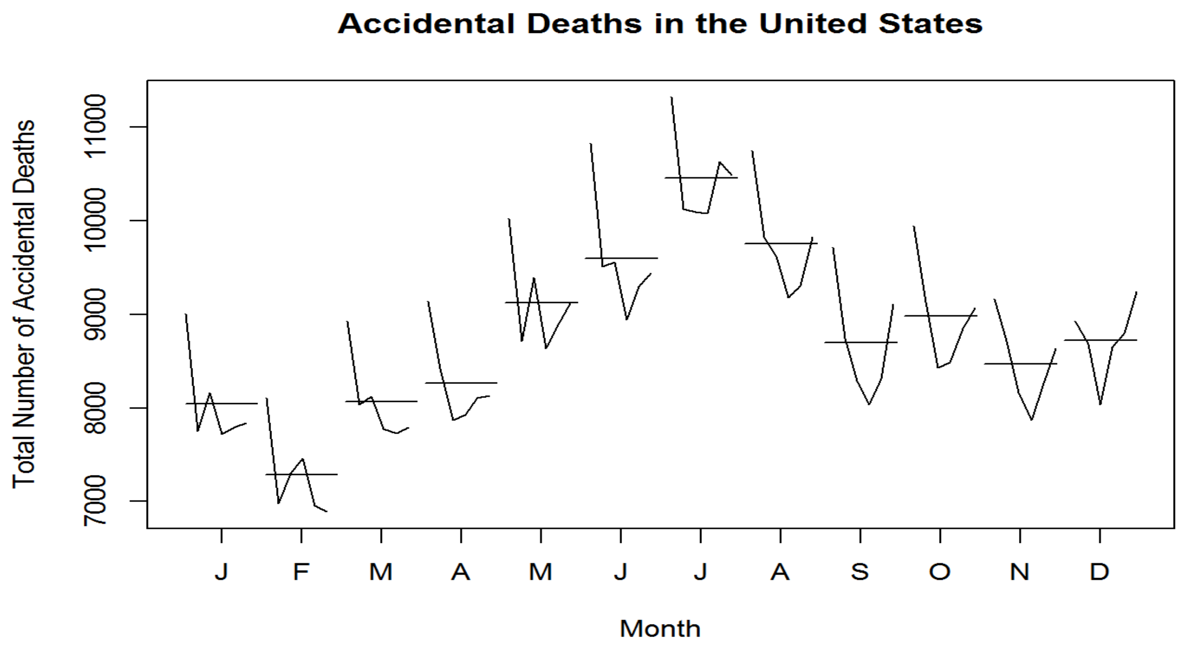
```
autoplot(usdeaths, main="Accidental Deaths in the United States", ylab="Total
Number of Accidental Deaths", xlab="Month")
```



```
seasonplot(usdeaths, main="Accidental Deaths in the United States",
ylab="Total Number of Accidental Deaths", xlab="Month",year.labels=TRUE,
year.labels.left=TRUE, col=1:20, pch=19)
```

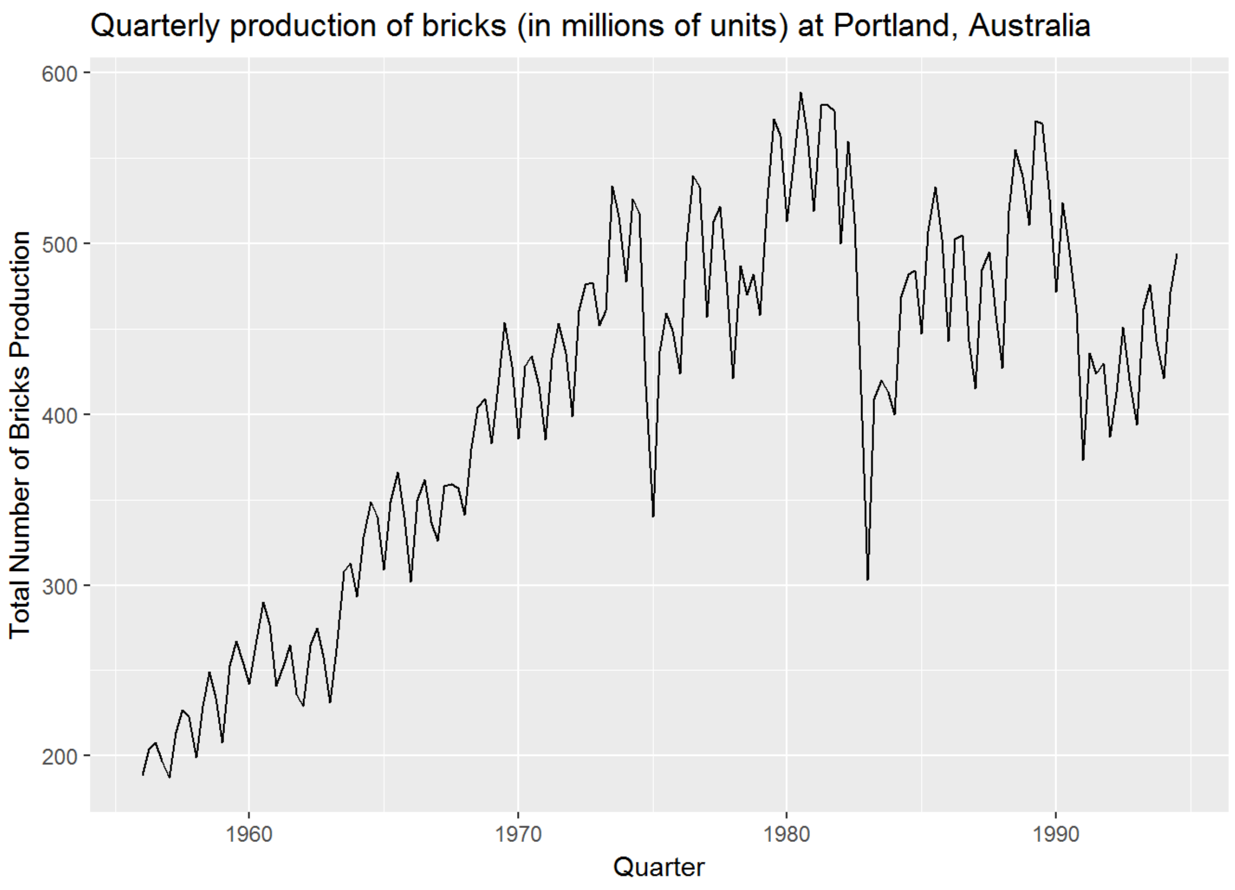


```
monthplot(usdeaths, main="Accidental Deaths in the United States",
ylab="Total Number of Accidental Deaths", xlab="Month")
```

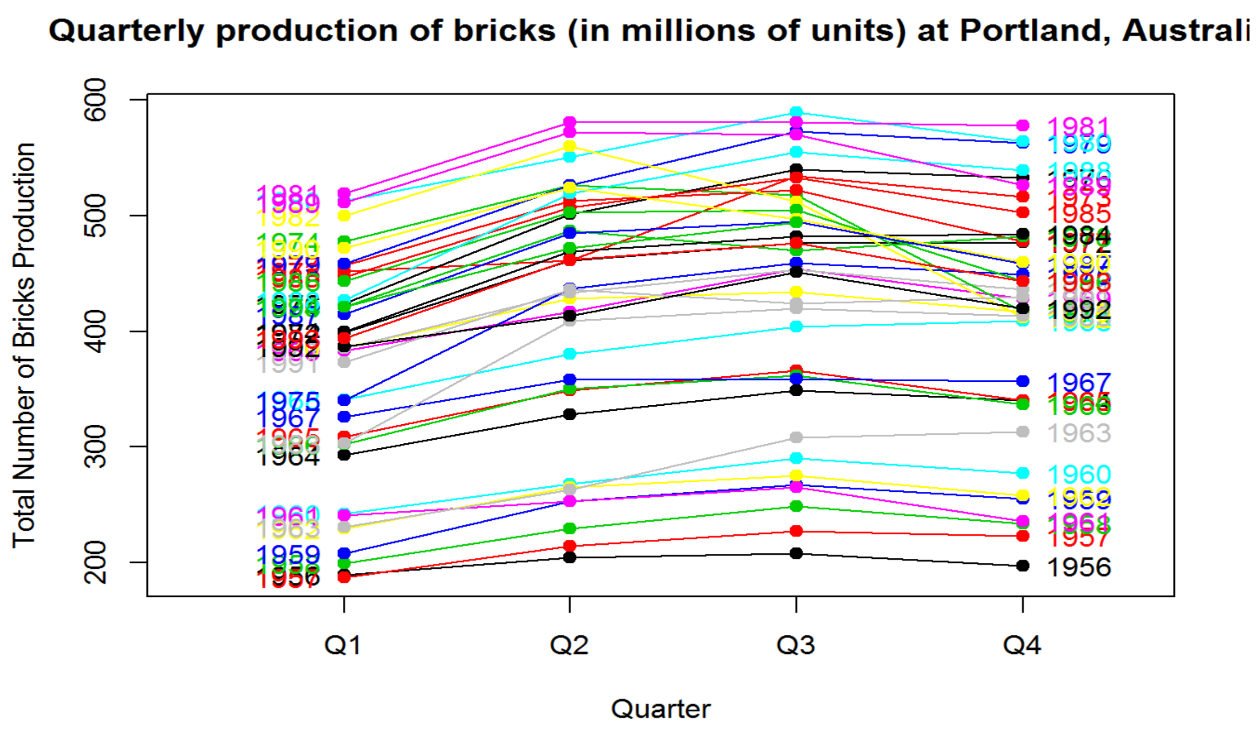


c. Interpretation: The Time Plot shows steady increase of quarterly production of bricks before the year of 1975, then Portland experienced sudden decrease and increase of production few times after 1975. The Season Plot shows that the first quarter tends to have weaker production, while Seasonal Subseries Plot also demonstrate that.

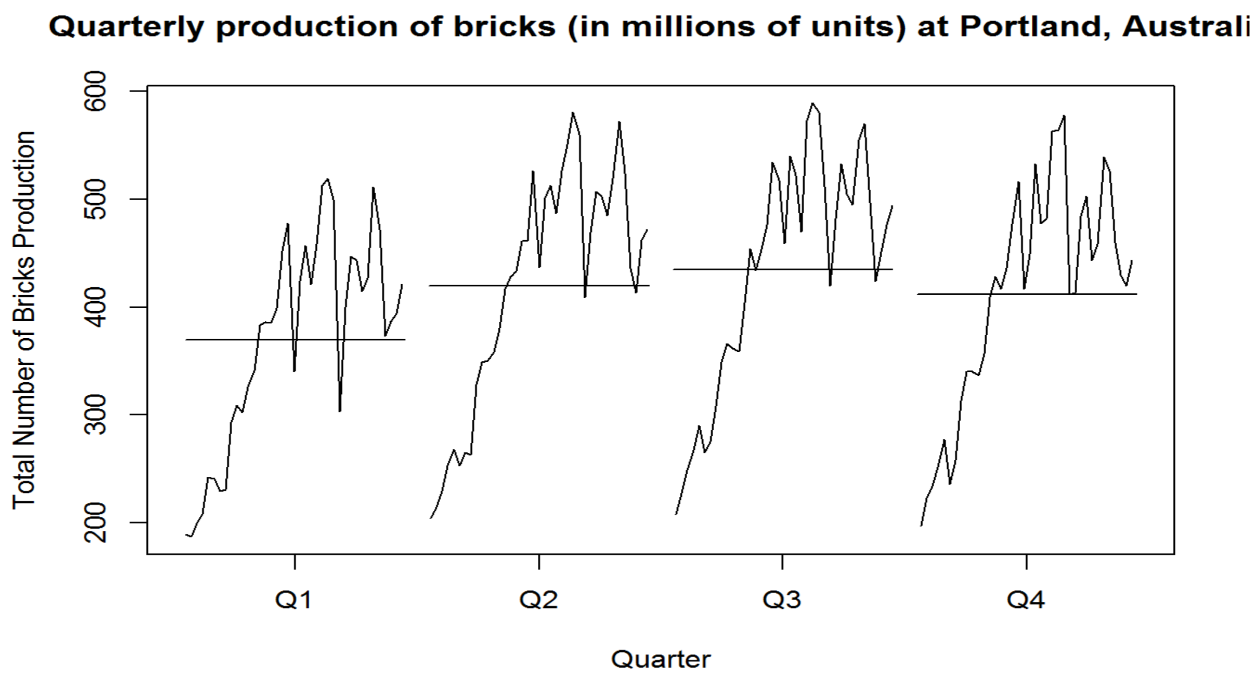
```
autoplot(bricksq, main="Quarterly production of bricks (in millions of units)
at Portland, Australia", ylab="Total Number of Bricks Production",
xlab="Quarter")
```



```
seasonplot(bricksq, main="Quarterly production of bricks (in millions of
units) at Portland, Australia", ylab="Total Number of Bricks Production",
xlab="Quarter",year.labels=TRUE, year.labels.left=TRUE, col=1:20, pch=19)
```



```
monthplot(bricksq, main="Quarterly production of bricks (in millions of
units) at Portland, Australia", ylab="Total Number of Bricks Production",
xlab="Quarter")
```

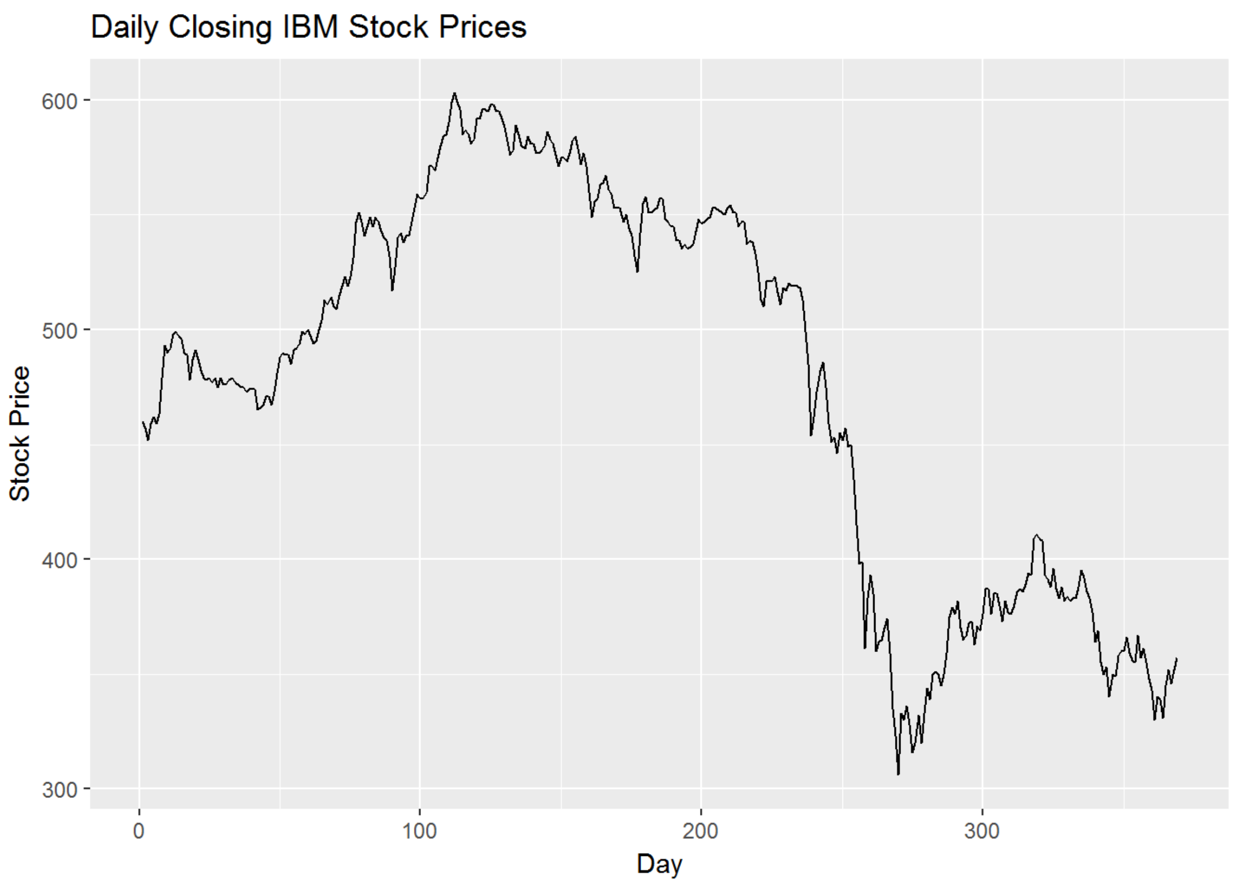


3. Consider the daily closing IBM stock prices (data set ibmclose).

a. Produce some plots of the data in order to become familiar with it.

The Time Plot shows that in the first 100 days, the stock price increase steadily. The following around 150 days, it has been flat. The latter 150 days, the stock price plummeted, and the daily price change has been dramatic.

```
autoplot(ibmclose, main="Daily Closing IBM Stock Prices", ylab="Stock Price",
xlab="Day")
```



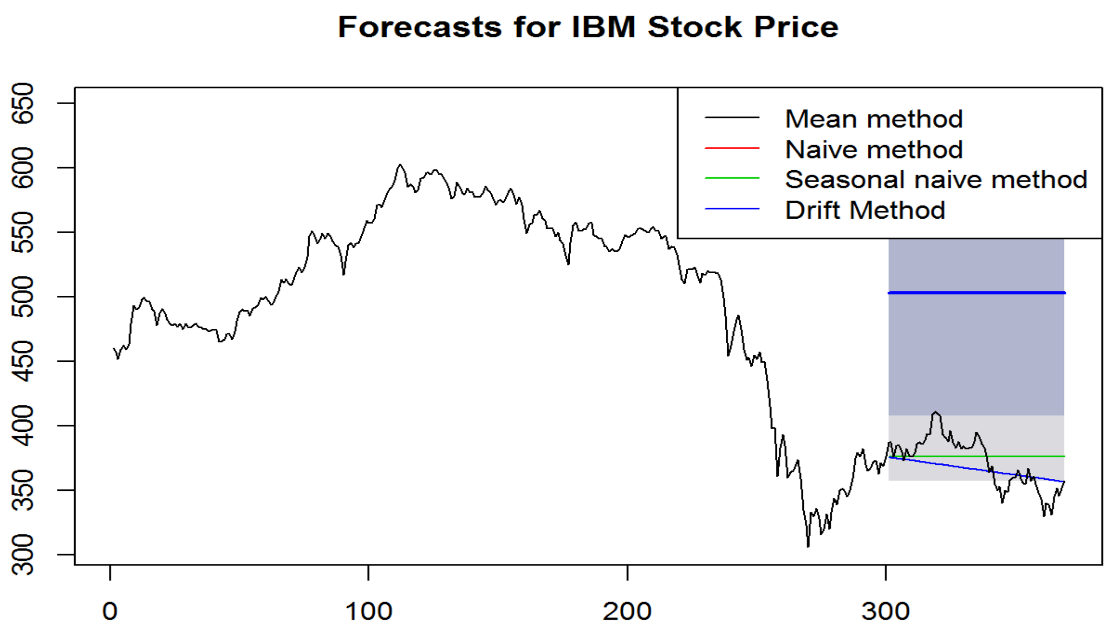
b. Split the data into a training set of 300 observations and a test set of 69 observations.

```
#training <- ibmclose[1: 300]

#test <- ibmclose[301:369]


training <- window(ibmclose, start = 1, end = 300)

test <- window(ibmclose, start = 301, end = 369)
```

c. Try various benchmark methods to forecast the training set and compare the results on the test set. Which method did best?

Approaches: There are several forecasting methods. Average method forecasts of all future values are equal to the mean of the historical data. Naive method forecasts all future value are simply the value of the last observation. Seasonal Naive Method forecasts all future values are be equal to the last observed value from the same season of the year. Drift Method forecasts the future values will increase or decrease over time, where the amount of change over time (called the drift) is set to be the average change seen in the historical data.

```
ibmfit1 <- meanf(training, h=69)

ibmfit2 <- naive(training, h=69)

ibmfit3 <- snaive(training, h=69)

ibmfit4 <- rwf(training, h = 69, drift=TRUE)


plot(ibmfit1, main="Forecasts for IBM Stock Price")

lines(ibmfit2$mean, col=2)

lines(ibmfit3$mean, col=3)

lines(ibmfit4$mean, col=4)

legend("topright", lty = 1,col = 1:4, legend=c("Mean method","Naive
method","Seasonal naive method", "Drift Method"))

lines(ibmclose)
```



Interpretation: From the following table, we are able to tell that out of all the methods, Drift Method is the best method in this case to predict the IBM stock price, because IBM has the lowest Mean Absolute Error(MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error(MAPE), Mean Absolute Scaled Error(MASE). The mean method is the worst, whiel both Naive Method and Seasonal Naive Method produce the same result.

```
ibm_accuracy <- data.frame(rbind(accuracy(ibmfit1, test)[2, c(2, 3, 5, 6)],
                          accuracy(ibmfit2, test)[2, c(2, 3, 5, 6)],
                          accuracy(ibmfit3, test)[2, c(2, 3, 5, 6)],
                          accuracy(ibmfit4, test)[2, c(2, 3, 5, 6)]
                          ))
row.names(ibm_accuracy) <- c("Mean method","Naive method","Seasonal naive
method", "Drift Method")


ibm_accuracy
```

```
##                            RMSE       MAE      MAPE      MASE
## Mean method           132.12557 130.61797 35.478819 25.626492
## Naive method           20.24810  17.02899  4.668186  3.340989
## Seasonal naive method  20.24810  17.02899  4.668186  3.340989
## Drift Method           17.06696  13.97475  3.707888  2.741765
```

Homework HA Chapter 6

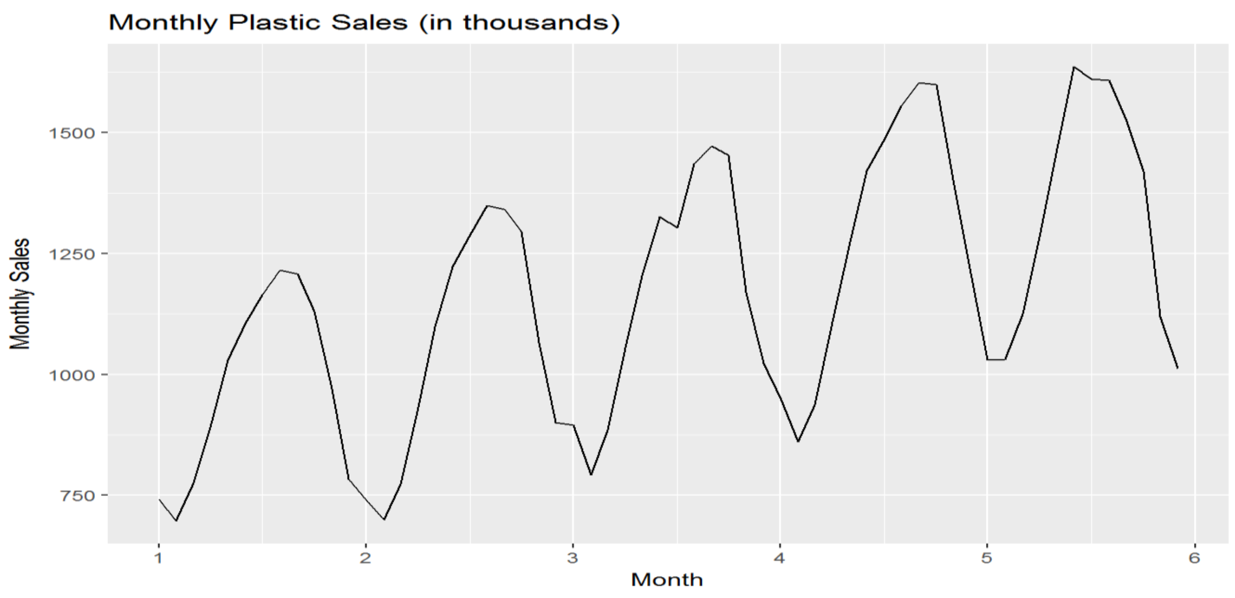
The data below represent the monthly sales (in thousands) of product A for a plastics manufacturer for years 1 through 5 (data set plastics).

	1	2	3	4	5
Jan	742	741	896	951	1030
Feb	697	700	793	861	1032
Mar	776	774	885	938	1126
Apr	898	932	1055	1109	1285
May	1030	1099	1204	1274	1468
Jun	1107	1223	1326	1422	1637
Jul	1165	1290	1303	1486	1611
Aug	1216	1349	1436	1555	1608
Sep	1208	1341	1473	1604	1528
Oct	1131	1296	1453	1600	1420
Nov	971	1066	1170	1403	1119
Dec	783	901	1023	1209	1013

a. Plot the time series of sales of product A. Can you identify seasonal fluctuations and/or a trend?

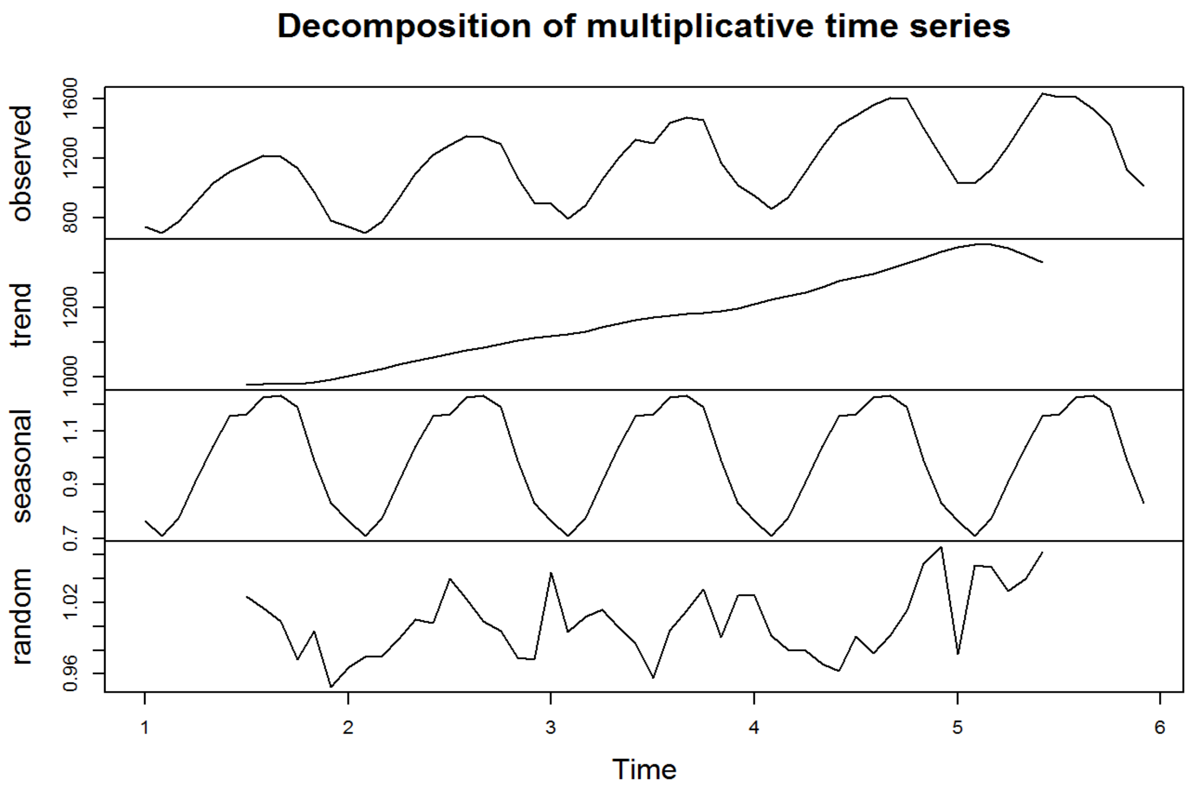
There is both seasonal fluctuation and a trend. The highest sale of the year always falls around July or August. Since January the sale will consistently goes up. After it reaches the peak during July or August it will consistently fall. Year over year, we observe rise in sales.

```r
library(fma)

library(ggplot2)

head(plastics)

autoplot(plastics, main="Monthly Plastic Sales (in thousands)", ylab="Monthly
Sales", xlab = "Month")
```



b. Use a classical multiplicative decomposition to calculate the trend-cycle and seasonal indices.

```
fit_decom <- decompose(plastics, type="multiplicative")
plot(fit_decom)
```



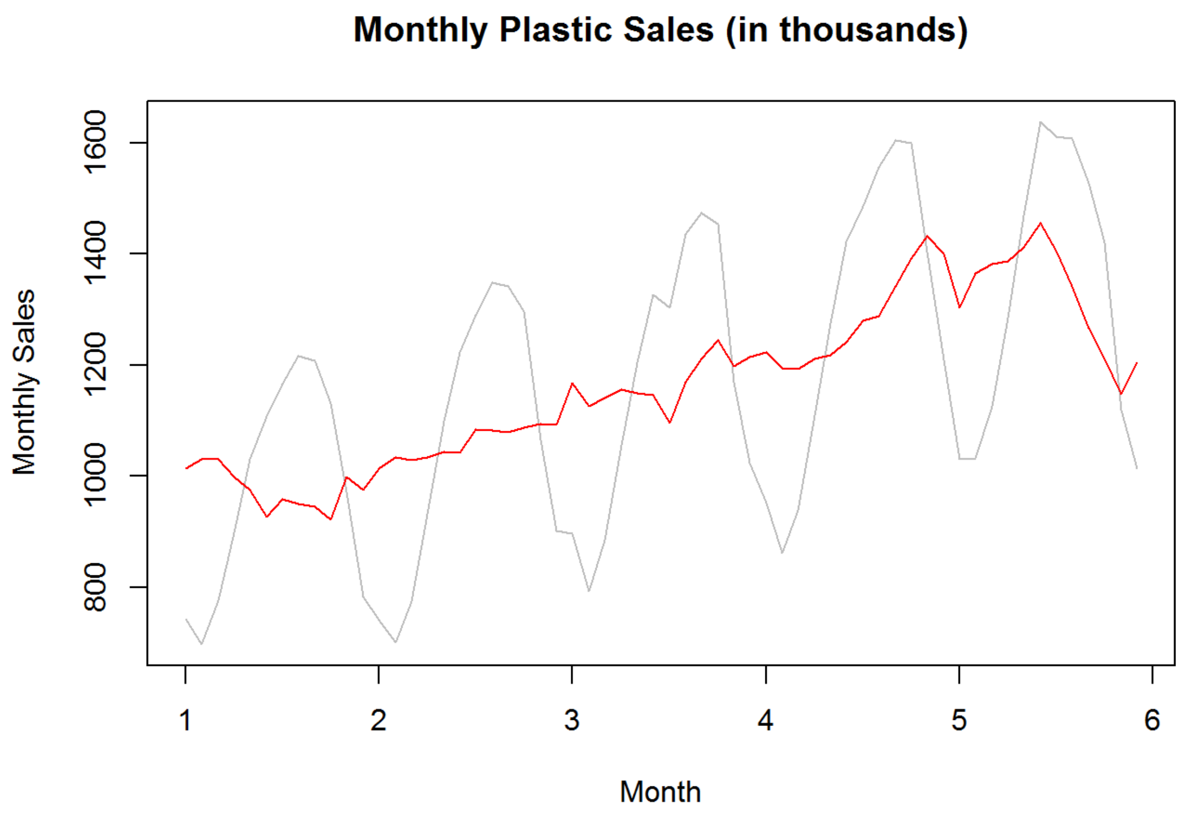
c. Do the results support the graphical interpretation from part (a)?

The trend-cycle and seasonal indices support the graphical interpretation from part (a). There is upward trend for the year over year sales. Furthermore, the season component shows periodic up and down movements on the sales in each individual year.

d. Compute and plot the seasonally adjusted data.

For multiplicative data, the seasonally adjusted values are obtained using yt/St. The red line in the following graph shows the seasonally adjusted data. After we removed the seasonal component, the monthly plastic sales actually goes down after year five.

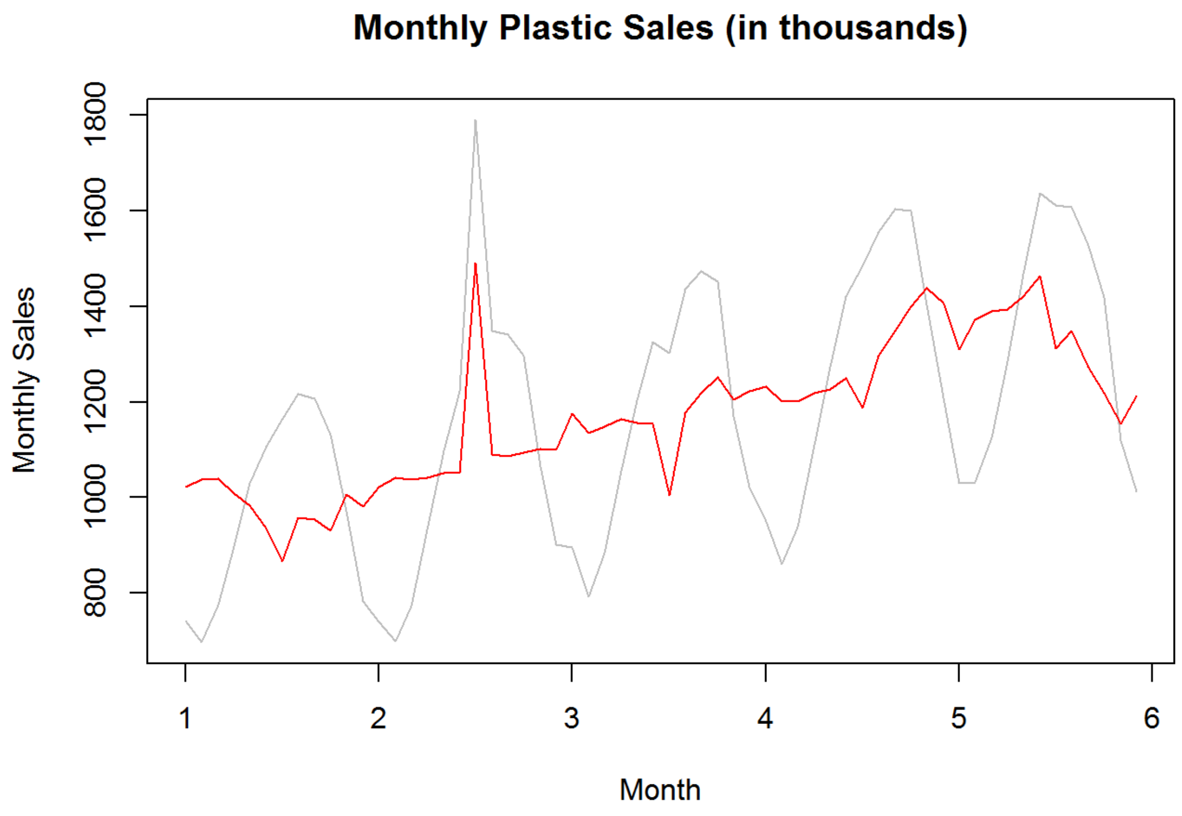
```r
fit <- stl(plastics, s.window = "periodic")

plot(plastics, col="grey", main = "Monthly Plastic Sales (in thousands)",
xlab="Month", ylab = "Monthly Sales")

lines(seasadj(fit), col="red", ylab="Seasonally adjusted")
```



e. Change one observation to be an outlier (e.g., add 500 to one observation), and recompute the seasonally adjusted data. What is the effect of the outlier?

The outlier changed the Time Plot of the plastic sales dramatically. In the meantime, it also changed the seasonally adjusted data. The magnitude seems to be smaller than the change seen on the Time Plot graph.

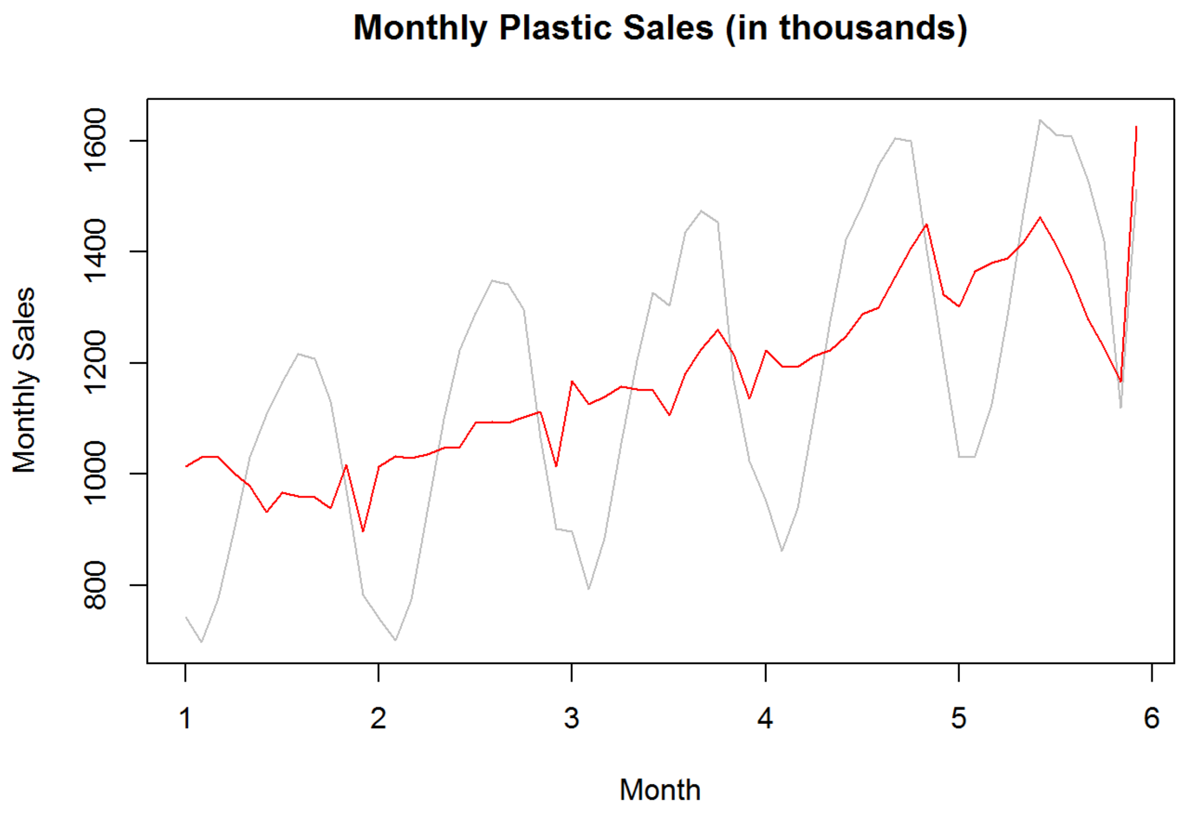
```
set.seed(100)

plastics2 <- plastics

x <- sample(1:length(plastics2), 1)


plastics2[x] <- plastics2[x] + 500

plot(plastics2, col="grey", main = "Monthly Plastic Sales (in thousands)",
xlab="Month", ylab = "Monthly Sales")


fit2 <- stl(plastics2, s.window = "periodic")

lines(seasadj(fit2), col="red", ylab="Seasonally adjusted")
```



f. Does it make any difference if the outlier is near the end rather than in the middle of the time series?

It seems like it does not make any difference if the outlier is near the end or in the middle of the time series. Both the Time Plot and seasonally adjusted data get changed about the same magnitude.

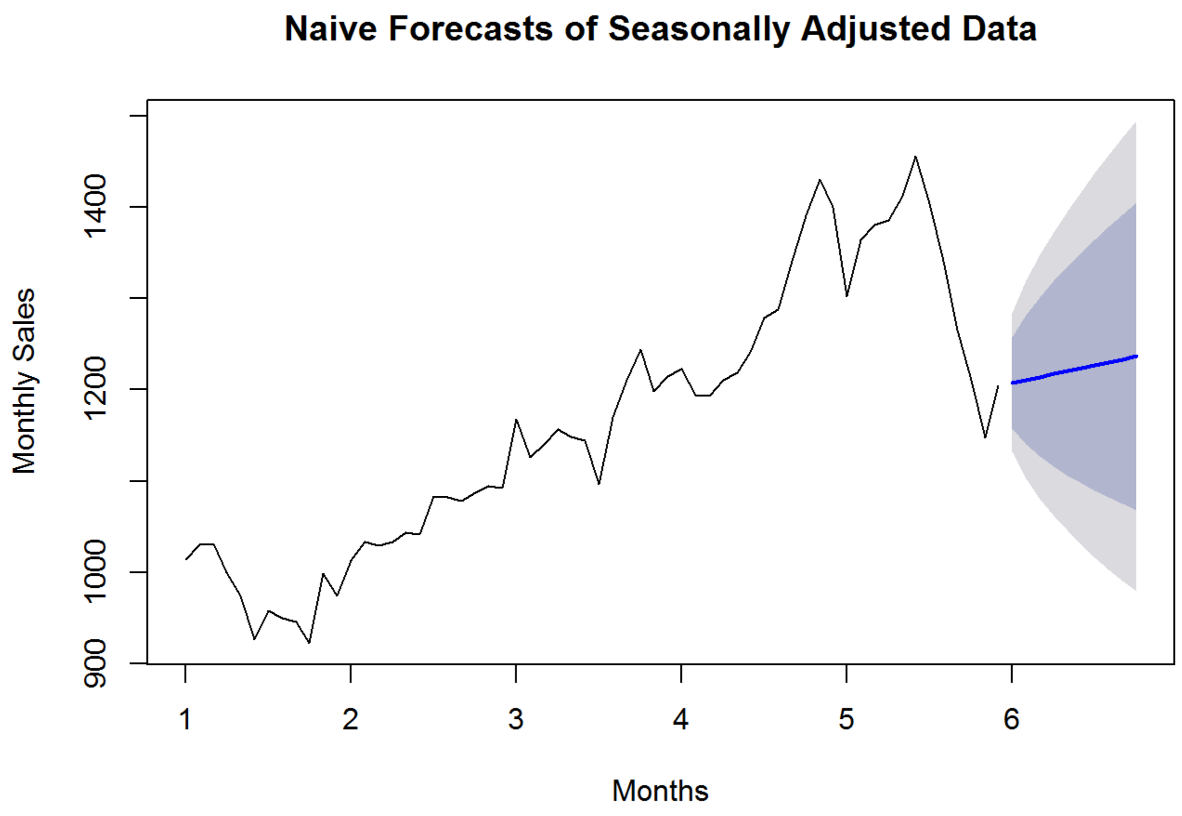
```
plastics3 <- plastics

plastics3[length(plastics3)] <- plastics3[length(plastics3)] + 500

plot(plastics3, col="grey", main = "Monthly Plastic Sales (in thousands)",
xlab="Month", ylab = "Monthly Sales")


fit3 <- stl(plastics3, s.window = "periodic")

lines(seasadj(fit3), col="red", ylab="Seasonally adjusted")
```



g. Use a random walk with drift to produce forecasts of the seasonally adjusted data.

Random walk with drift method is one of the non-seasonal forecasting method. It can be used to forecast seasonally adjusted data.

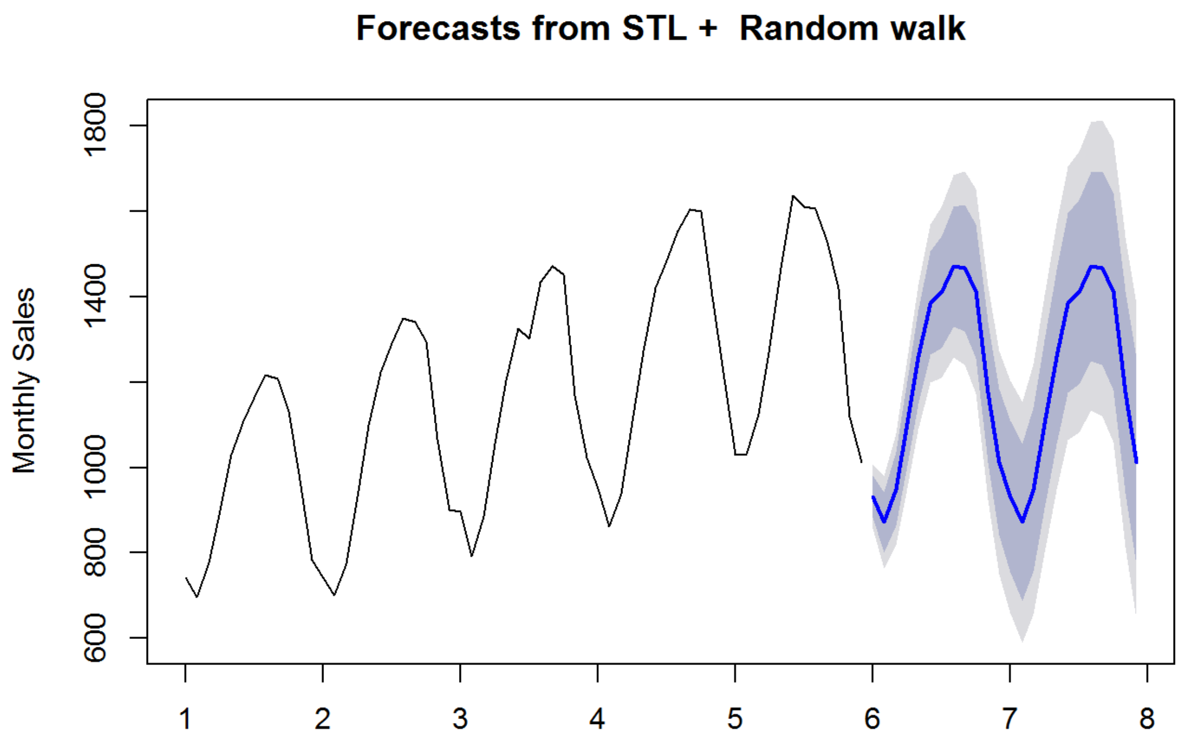
```r
plastics_rwf <- rwf(seasadj(fit), drift = TRUE)


plot(plastics_rwf, main = "Naive Forecasts of Seasonally Adjusted Data",
xlab="Months", ylab = "Monthly Sales")
```



h. Reseasonalize the results to give forecasts on the original scale.

Once we add the forecast of the seasonal component to the seasonally adjusted data (the process is called reseasonalized), the resulting forecast will be more accurately capture the actual future values. The method forecast is enable us to do that. The following figure produced the result for the year 7 and 8.

```r
fcast <- forecast(fit, method="naive")

plot(fcast, ylab="Monthly Sales")
```


Forecasts from STL + Random walk
Monthly Sales
600
800
1000
1400
1800
1
2
3
4
5
6
7
8

Homework KJ Chapter 3

3.1. The UC Irvine Machine Learning Repository6 contains a data set related to glass identification. The data consist of 214 glass samples labeled as one of seven class categories. There are nine predictors, including the refractive index and percentages of eight elements: Na, Mg, Al, Si, K, Ca, Ba, and Fe. The data can be accessed via:

```
library(mlbench)
library(kableExtra)
library(knitr)
```

Question A and B

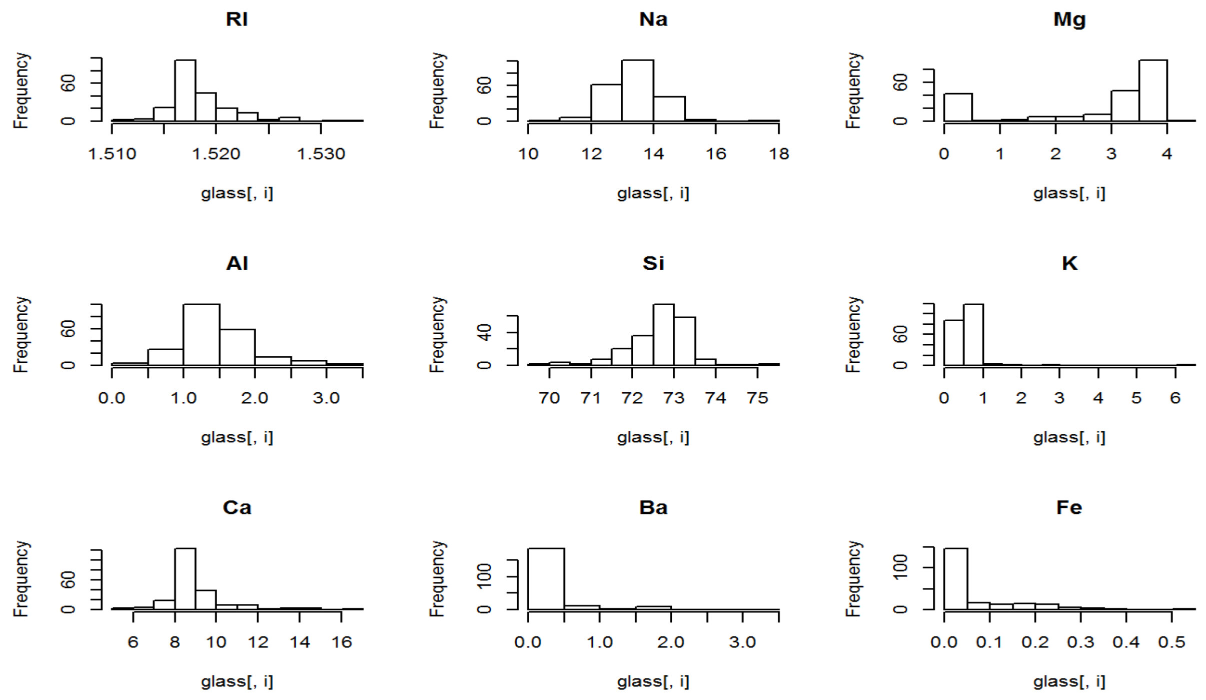
(a). Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors.

(b). Do there appear to be any outliers in the data? Are any predictors skewed?

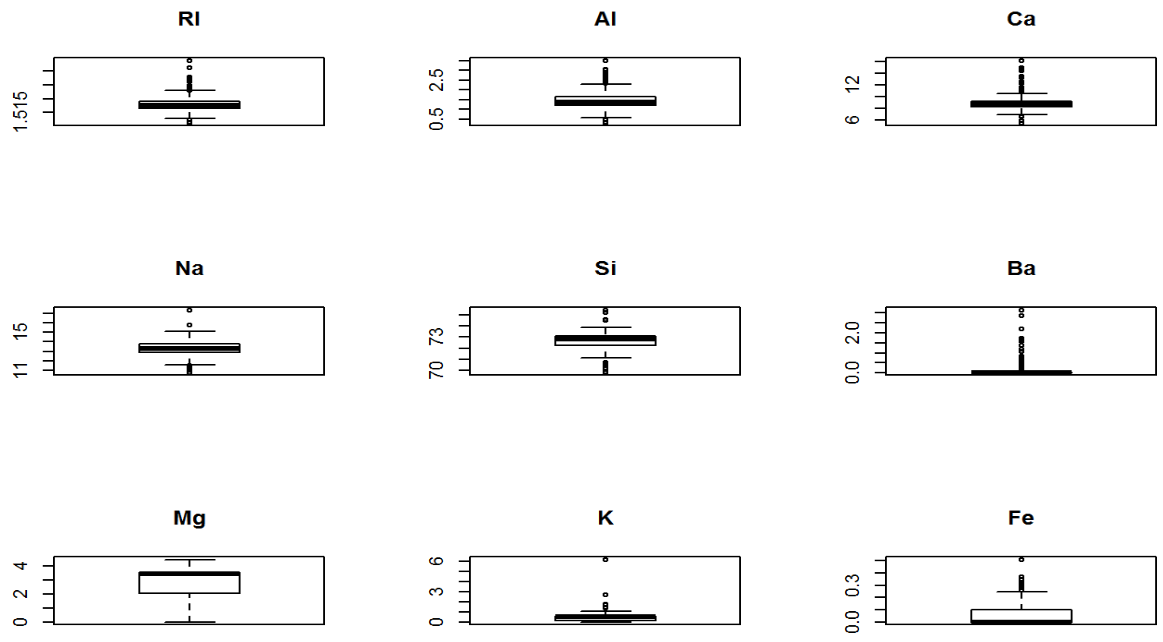
Approaches: In order to understand the distribution of the predictor variable, the most straight forward method is to use the histogram obtain the frequencies of each predictor variables. The boxplot and summary statistics are useful in terms of searching for outliers and quantiles. And the correlation table tells us the relationship between each variables.

Interpretation: From the histogram, it shows the Fe and Ba variable contains lots of zeros, which make their graphs highly skewed to the right (K too). Most of the variables including RI, NA, Al, SI, CA have peaks in the center of the distribution. They appear to be more normally distributed. One exception is Mg, which has a trough in the center, but peaks on both ends. The summary statistics and boxplot have the same pattern as what is shown on the histogram, in addition to that, is also tell us there is lots of outliers in variable Ri, Al, Ca, Ba, Fe. The correlation table tell us that most of the variables are not related to each other, except the pair between RI and Ca, the correlation coefficient appear to be 0.7, which is moderately strong.

```
glass <- subset(Glass, select = -Type)
predictors <- colnames(glass)

par(mfrow = c(3, 3))
for(i in 1:9)
  {
  hist(glass[,i], main = predictors[i])
}
```



```
par(mfcol=c(3, 3))
for (i in 1:9)
  {boxplot(glass[,i], main = predictors[i])}
```



```
kable(summary(glass))
```

RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
Min. :1.511	Min. :10.73	Min. :0.000	Min. :0.290	Min. :69.81	Min. :0.0000	Min. : 5.430	Min. :0.000	Min. :0.00000
1st Qu.:1.517	1st Qu.:12.91	1st Qu.:2.115	1st Qu.:1.190	1st Qu.:72.28	1st Qu.:0.1225	1st Qu.: 8.240	1st Qu.:0.000	1st Qu.:0.00000
Median :1.518	Median :13.30	Median :3.480	Median :1.360	Median :72.79	Median :0.5550	Median : 8.600	Median :0.000	Median :0.00000
Mean :1.518	Mean :13.41	Mean :2.685	Mean :1.445	Mean :72.65	Mean :0.4971	Mean : 8.957	Mean :0.175	Mean :0.05701
3rd Qu.:1.519	3rd Qu.:13.82	3rd Qu.:3.600	3rd Qu.:1.630	3rd Qu.:73.09	3rd Qu.:0.6100	3rd Qu.: 9.172	3rd Qu.:0.000	3rd Qu.:0.10000
Max. :1.534	Max. :17.38	Max. :4.490	Max. :3.500	Max. :75.41	Max. :6.2100	Max. :16.190	Max. :3.150	Max. :0.51000

```
cor.table <- cor(glass, use = "pairwise", method = "spearman")
kable(round(cor.table, 2))
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
RI	1.00	0.03	0.14	-0.49	-0.53	-0.29	0.70	-0.18	0.10
Na	0.03	1.00	-0.13	0.14	-0.27	-0.58	0.03	0.41	-0.22
Mg	0.14	-0.13	1.00	-0.51	-0.34	0.20	-0.29	-0.46	0.10
Al	-0.49	0.14	-0.51	1.00	0.20	0.15	-0.28	0.47	-0.08
Si	-0.53	-0.27	-0.34	0.20	1.00	0.00	-0.22	0.17	-0.07
K	-0.29	-0.58	0.20	0.15	0.00	1.00	-0.47	-0.26	0.09
Ca	0.70	0.03	-0.29	-0.28	-0.22	-0.47	1.00	-0.01	0.11
Ba	-0.18	0.41	-0.46	0.47	0.17	-0.26	-0.01	1.00	0.01
Fe	0.10	-0.22	0.10	-0.08	-0.07	0.09	0.11	0.01	1.00

c. Are there any relevant transformations of one or more predictors that might improve the classification model?

Box-Cox transformation can be used to normalize the data, which are highly skewed. While Spatial Sign Transformation can be used to lower the effects of outliers, if the model is considered to be sensitive to outliers.

3.2. The soybean data can also be found at the UC Irvine Machine Learning Repository. Data were collected to predict disease in 683 soybeans. The 35 predictors are mostly categorical and include information on the environmental conditions (e.g., temperature, precipitation) and plant conditions (e.g., left spots, mold growth). The outcome labels consist of 19 distinct classes.

The data can be loaded via:

```
library(mlbench)
library(ggplot2)
library(lattice)
```

a. Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?

Approaches: Predictors with degenerate distributions are those predictors whose variances tend to approach zero. . The fraction of unique values over the sample size is low (say 10%). . The ratio of the frequency of the most prevalent value to the frequency of the second most prevalent value is large (say around 20).

The caret package function nearZeroVar will return the column numbers of any predictors that fulfill the conditions stated above.

Interpretation: The following code shows that three variables leaf.mild, mycelium and sclerotia are predictors with degenerate distribution.

```
library(tidyr)
library(dplyr)
library(caret)

index <- nearZeroVar(Soybean)
colnames(Soybean)[index]

## [1] "leaf.mild" "mycelium"  "sclerotia"
```

b. Roughly 18% of the data are missing. Are there particular predictors that are more likely to be missing? Is the pattern of missing data related to the classes?

Approaches: I am using the dplyr package to handle the data tidying and transformation. In this package there are many methods which I can use. for example: summarize_all, mutate, arrange et cetera.

Interpretation: According to first list, we can tell that hail, sever, seed.tmt, and lodge are the variables that miss lots of data, While Class and leaves do not have missing data at all. Based on second graph, we can tell that the pattern of missing data is also related to the classes. phytophthora-rot has the most, followed by 2-4-d-injury , nematode, diaporthe-pod-&-stem-blight, herbicide-injury. Everything else have complete sets of data.

```
missing_values <- Soybean %>%
  select(everything()) %>%
  summarize_all(funs(sum(is.na(.))))

data.frame(sort(missing_values, decreasing = TRUE))
```

```
##   hail sever seed.tmt lodging germ leaf.mild fruiting.bodies fruit.spots
## 1  121   121      121     121  112       108             106         106
##   seed.discolor shriveling leaf.shread seed mold.growth seed.size
## 1           106        106         100   92          92        92
##   leaf.halo leaf.marg leaf.size leaf.malf fruit.pods precip stem.cankers
## 1        84        84        84        84         84     38           38
##   canker.lesion ext.decay mycelium int.discolor sclerotia plant.stand
## 1            38        38       38           38        38          36
##   roots temp crop.hist plant.growth stem date area.dam Class leaves
## 1    31   30        16           16   16    1        1     0      0
```

```
missing_classes <- Soybean %>%
  gather(key = predictors, value = value, -Class)%>%
  group_by(Class)%>%
  summarize(n = sum(is.na(value)))%>%
  mutate(n, missing = n/(nrow(Soybean) * 35))%>%
  arrange(desc(missing))
missing_classes
```

```
## # A tibble: 19 x 3
##    Class                           n missing
##    <fct>                       <int>   <dbl>
##  1 phytophthora-rot             1214 0.0508
##  2 2-4-d-injury                  450 0.0188
##  3 cyst-nematode                 336 0.0141
##  4 diaporthe-pod-&-stem-blight   177 0.00740
##  5 herbicide-injury              160 0.00669
##  6 alternarialeaf-spot             0 0.
##  7 anthracnose                     0 0.
##  8 bacterial-blight                0 0.
##  9 bacterial-pustule               0 0.
## 10 brown-spot                      0 0.
## 11 brown-stem-rot                  0 0.
## 12 charcoal-rot                    0 0.
## 13 diaporthe-stem-canker           0 0.
## 14 downy-mildew                    0 0.
## 15 frog-eye-leaf-spot              0 0.
## 16 phyllosticta-leaf-spot          0 0.
## 17 powdery-mildew                  0 0.
## 18 purple-seed-stain               0 0.
## 19 rhizoctonia-root-rot            0 0.
```

c. Develop a strategy for handling missing data, either by eliminating predictors or imputation.

Approaches: We can use Mice package to impute the missing values. Mice means multivariate imputation by chained equations. To improve the running time, I only run one time of imputation and use the default function ppm. Ppm means predictive mean matching. " It is similar to the regression method except that for each missing value, it imputes a value randomly from a set of observed values whose predicted values are closest to the predicted value for the missing value from the simulated regression model" (Heitjan and Little 1991; Schenker and Taylor 1996).

Interpretation:

The Mice package is very useful in terms of imputating values. As the end result shows, there is no more missing value in the data frame.

```
library(mice)

Soybean_impute <- mice(Soybean, m=1, method = "pmm", print = F)

Soybean_impute <- complete(Soybean_impute)


result<- Soybean_impute %>%

  select(everything()) %>%

  summarize_all(funs(sum(is.na(.))))


data.frame(sort(result, decreasing = TRUE))
```

```
##   Class date plant.stand precip temp hail crop.hist area.dam sever
## 1     0    0           0      0    0    0         0        0     0
##   seed.tmt germ plant.growth leaves leaf.halo leaf.marg leaf.size
## 1        0    0            0      0         0         0         0
##   leaf.shread leaf.malf leaf.mild stem lodging stem.cankers canker.lesion
## 1           0         0         0    0       0            0             0
##   fruiting.bodies ext.decay mycelium int.discolor sclerotia fruit.pods
## 1               0         0        0            0         0          0
##   fruit.spots seed mold.growth seed.discolor seed.size shriveling roots
## 1           0    0           0             0         0          0     0
```

Homework HA Chapter 7

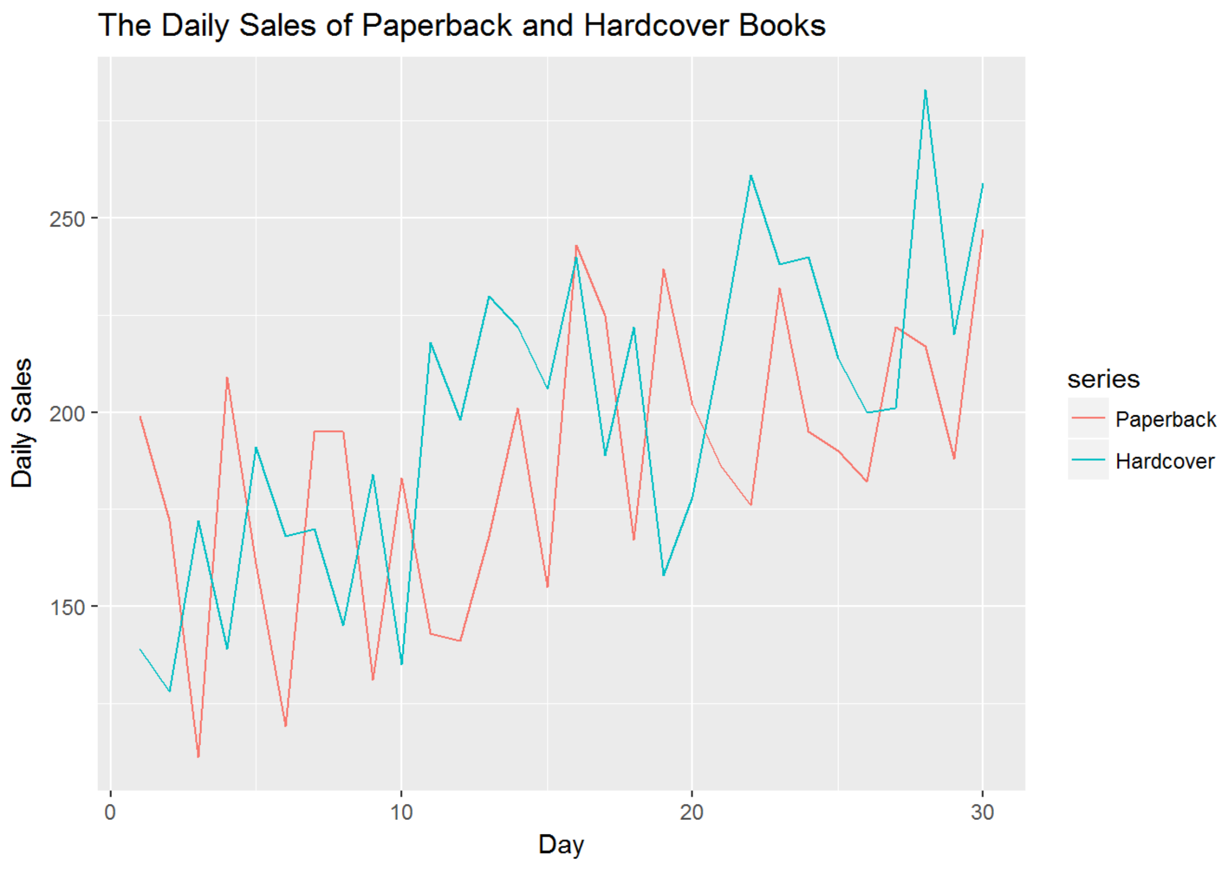
7.8 1. Data set books contains the daily sales of paperback and hardcover books at the same store. The task is to forecast the next four days' sales for paperback and hardcover books (data set books).

a. Plot the series and discuss the main features of the data.

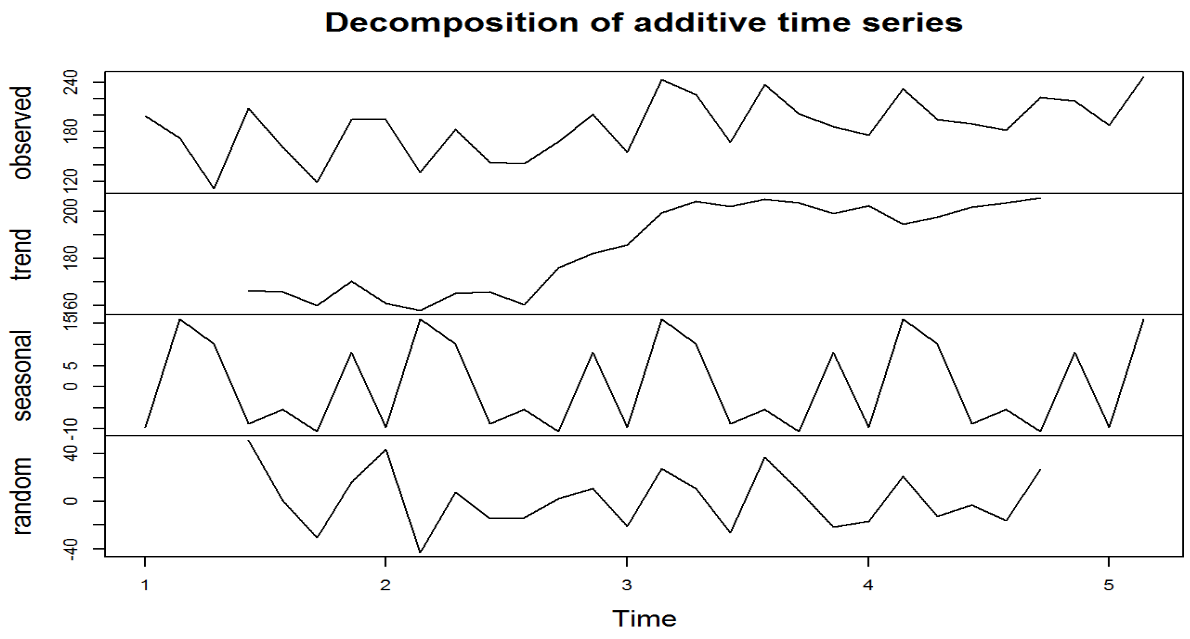
Approaches: I was using the Time Series plot to get a general graph of the daily sales for both paperback and hardcover books. Then I will use classical decomposition to investigate the trend, seasonal, and random components. Since books contains daily data, I would set the time frequency into 7 to search for any weekly seasonality

Interpretation: From the general Time Series graph, there is clear upward trend associated with both daily sales of paperback and hardcover books. The sale varies a lot from day to day. No apparent seasonality observed in this graph. In addition, it looks like there is a inverse relationship between two variables as sale of one peak, the sale of the other bottoms. From the decomposition graph, we can tell sales of both trending upwards. There is some sort of seasonality exist, as the sale then to peak once a week.

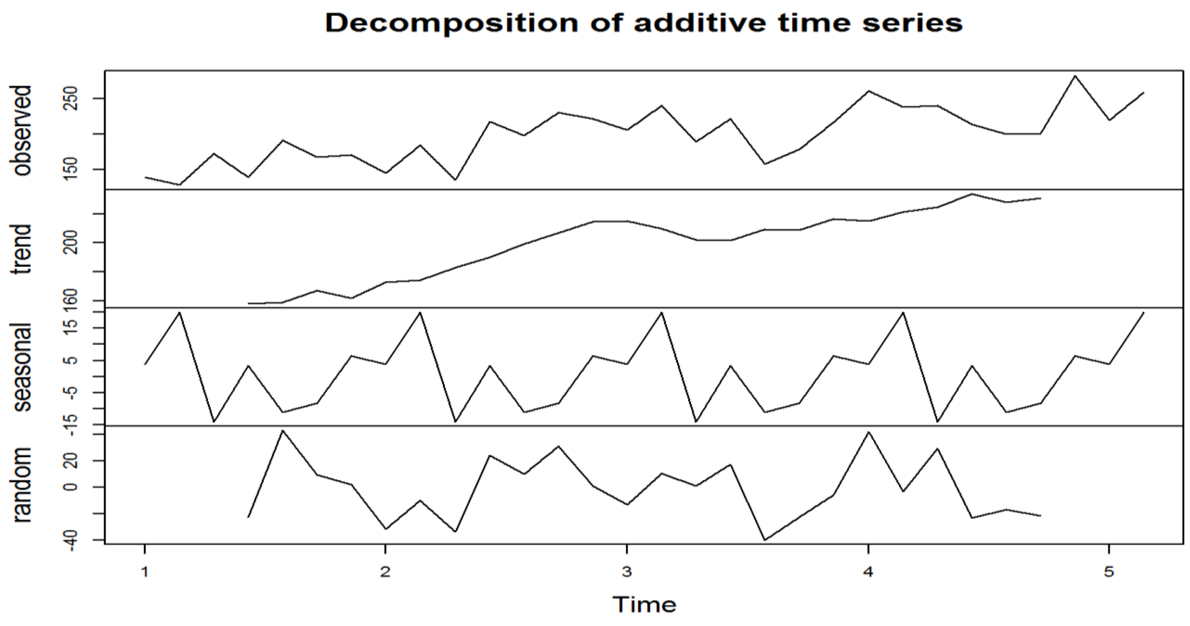
```r
library(fma)

library(ggplot2)

autoplot(books, main="The Daily Sales of Paperback and Hardcover Books",
ylab="Daily Sales", xlab="Day")
```



```
books_ts <- ts(books, frequency = 7)
fit_decom1 <- decompose(books_ts[, 1])
plot(fit_decom1)
```



```
fit_decom2 <- decompose(books_ts[, 2])
plot(fit_decom2)
```



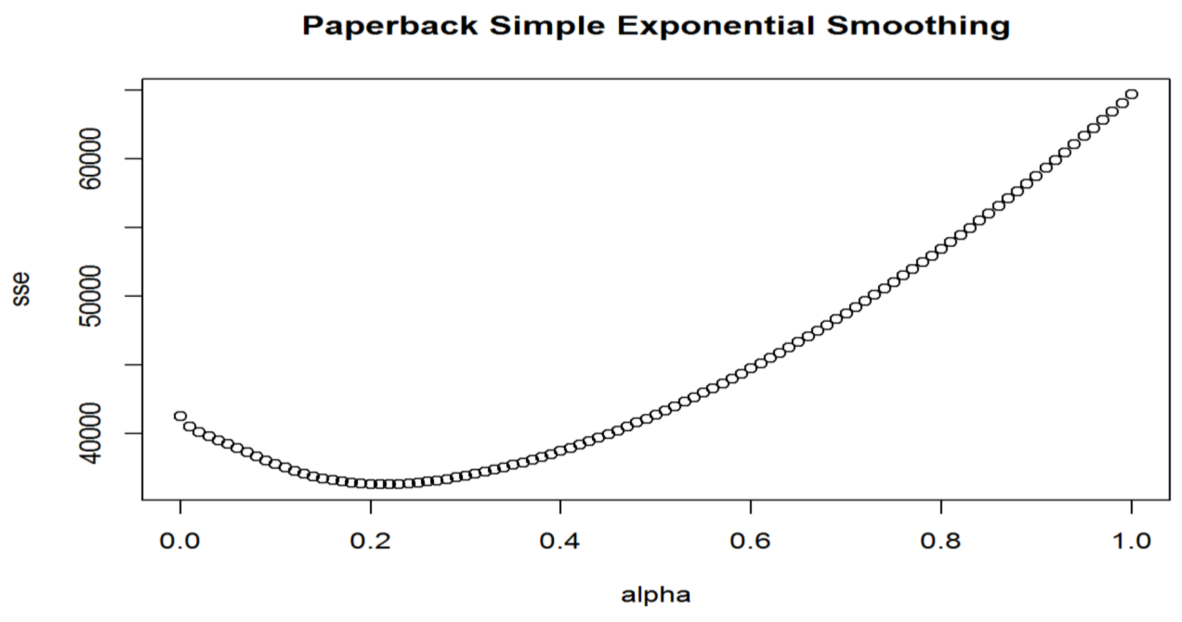
b. Use simple exponential smoothing with the ses function (setting initial=“simple”) and explore different values of alpha for the paperback series. Record the within-sample SSE for the one-step forecasts. Plot SSE against alpha and find which value of alpha works best. What is the effect of alpha on the forecasts?

Approaches: Simple exponential smoothing is method suitable for forecasting data with no trend or seasonal pattern. Forecasts are calculated using weighted averages where the weights decrease exponentially as observations come from further in the past.

Interpretation:

From the graph, we are able to tell that alpha the SSE reaches its minimum when alpha is around 0.21. Alpha is the smoothing parameter, it can only take values between 0 and 1. The larger the alpha value, more weight is given to the recent observation and vice versa.

```
alpha <- numeric()
sse <- numeric()

for(i in seq(0, 1, 0.01)) {
  fit <- ses(books[, 1], alpha = i, initial="simple", h = 4)
  alpha <- c(alpha, i)
  sse <- c(sse, fit$model$SSE)
}
result <- data.frame(alpha, sse)
plot(result, main="Paperback Simple Exponential Smoothing")
```



c. Now let ses select the optimal value of alpha. Use this value to generate forecasts for the next four days. Compare your results with 2.

We can use summary statistics of simple exponential smoothing to find out the optimal value of alpha, which minimize SSE. Or on the other way, we can extract the value from the model, we will get the optimal alpha value is 0.2125115.

```
fit1  <- ses(books[, 1], initial="simple", h = 4)

summary(fit1)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = books[, 1], h = 4, initial = "simple")
##
##   Smoothing parameters:
##     alpha = 0.2125
##
##   Initial states:
##     l = 199
##
##   sigma:  34.7918
## Error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 1.749509 34.79175 28.64424 -2.770157 16.56938 0.7223331
##                    ACF1
## Training set -0.1268119
##
## Forecasts:
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 31       210.1537 165.5663 254.7411 141.9631 278.3443
## 32       210.1537 164.5706 255.7368 140.4404 279.8671
## 33       210.1537 163.5962 256.7112 138.9501 281.3573
## 34       210.1537 162.6418 257.6657 137.4905 282.8170
```

```
optimal_alpha1 <- fit1$model$par["alpha"]

optimal_alpha1
```

```
##     alpha

## 0.2125115
```

d. Repeat but with initial="optimal". How much difference does an optimal initial level make?

If we repeat the process with initial = "optimal", the optimal of alpha value is 0.1685384. The SSE values we obtain using three different methods are very closed to each other.

```
fit2  <- ses(books[, 1], initial="optimal", h = 4)
summary(fit2)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = books[, 1], h = 4, initial = "optimal")
##
##   Smoothing parameters:
##     alpha = 0.1685
##
##   Initial states:
##     l = 170.8257
##
##   sigma:  33.6377
##
##      AIC     AICc      BIC
## 318.9747 319.8978 323.1783
##
## Error measures:
##                    ME     RMSE     MAE       MPE     MAPE      MASE
## Training set 7.176212 33.63769 27.8431 0.4737524 15.57782 0.7021303
##                    ACF1
## Training set -0.2117579
##
## Forecasts:
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 31       207.1098 164.0013 250.2182 141.1811 273.0384
## 32       207.1098 163.3934 250.8261 140.2513 273.9682
## 33       207.1098 162.7937 251.4258 139.3342 274.8853
## 34       207.1098 162.2021 252.0174 138.4294 275.7901
```

```
optimal_alpha2 <- fit2$model$par["alpha"]
optimal_alpha2
##     alpha
## 0.1685384
method <- c("Graph", "Simple", "Optimal")
```

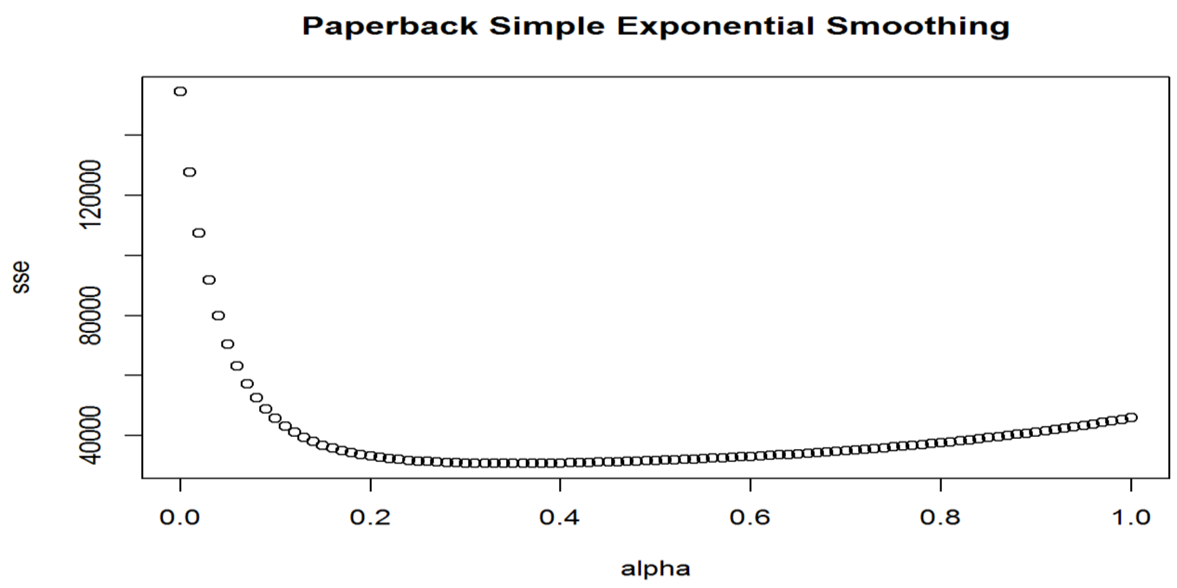
```
alphas <- c(0.21, optimal_alpha1, optimal_alpha2)
SSES <- c(min(result$sse), fit1$model$SSE, sum(residuals(fit2) ^ 2))

final <- data.frame(method,alphas, SSES)
final
```

```
##    method    alphas     SSES
## 1   Graph 0.2100000 36314.59
## 2  Simple 0.2125115 36313.98
## 3 Optimal 0.1685384 33944.82
```

e. Repeat steps (b)-(d) with the hardcover series.

After repeating the steps, we can tell three different methods end up with different values of alpha. However, the SSE values are all very similar.

```
alpha <- numeric()
sse <- numeric()

for(i in seq(0, 1, 0.01)) {

  fit <- ses(books[, 2], alpha = i, initial="simple", h = 4)
  alpha <- c(alpha, i)
  sse <- c(sse, fit$model$SSE)
}

result <- data.frame(alpha, sse)
plot(result, main="Paperback Simple Exponential Smoothing")
```



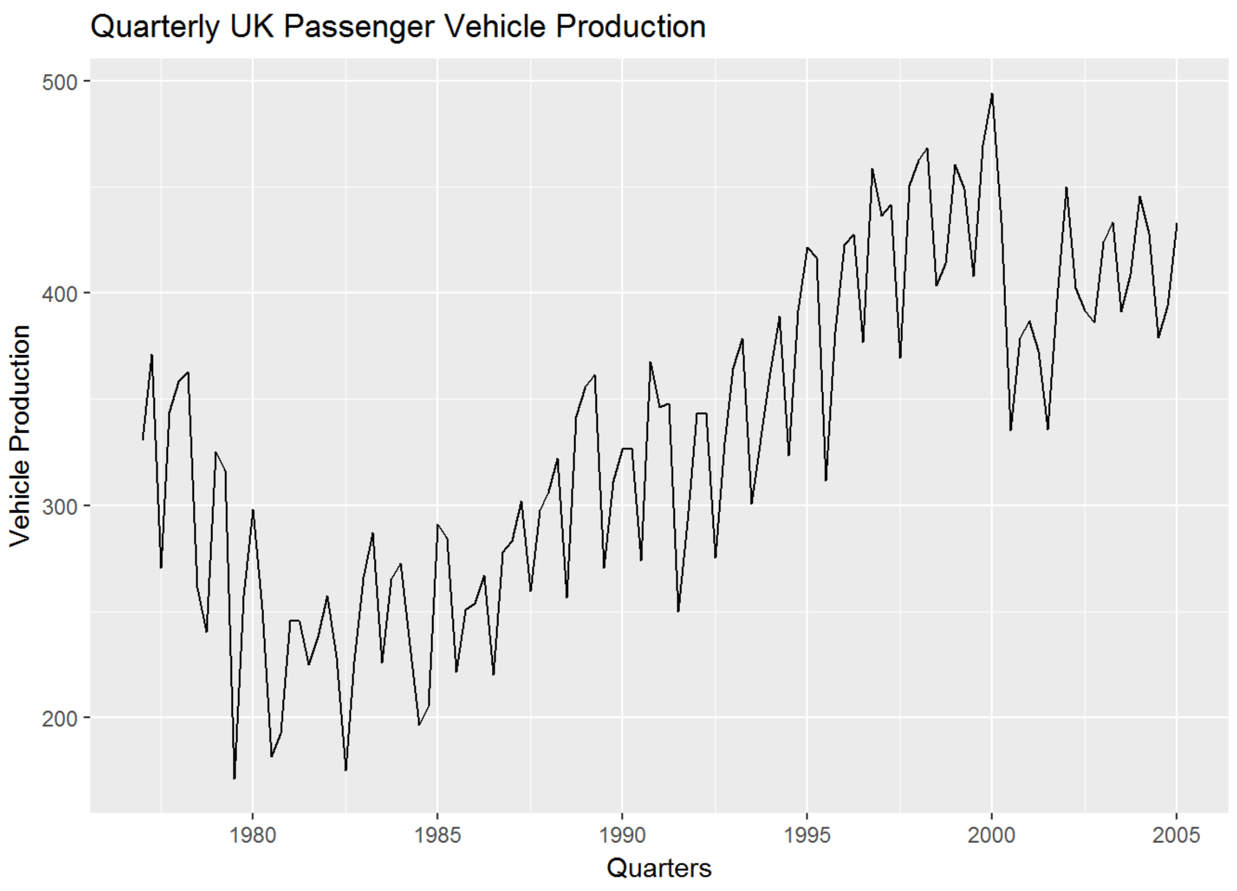
```
fit1  <- ses(books[, 2], initial="simple", h = 4)
optimal_alpha1 <- fit1$model$par["alpha"]
optimal_alpha1
##     alpha
## 0.3473305
fit2  <- ses(books[, 2], initial="optimal", h = 4)
optimal_alpha2 <- fit2$model$par["alpha"]
optimal_alpha2
##     alpha
## 0.3282696
method <- c( "Simple", "Optimal")
alphas <- c(optimal_alpha1, optimal_alpha2)
SSES <- c(fit1$model$SSE, sum(residuals(fit2) ^ 2))

final <- data.frame(method,alphas, SSES)
final
##    method    alphas     SSES
## 1  Simple 0.3473305 30758.07
## 2 Optimal 0.3282696 30587.69
```

3. For this exercise, use the quarterly UK passenger vehicle production data from 1977:1–2005:1 (data set ukcars).

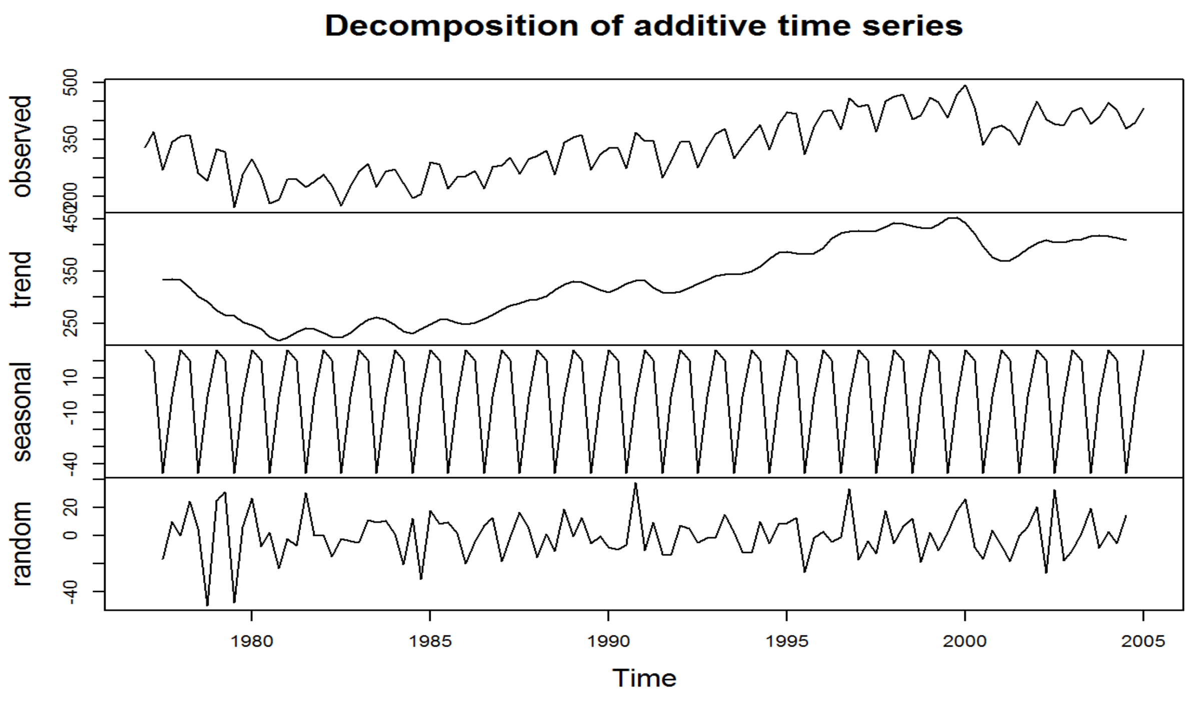
a. Plot the data and describe the main features of the series.

From the decomposition figure, we can tell that there is upward trend regarding to the vehicle production. The upward trend stopped starting the year of 2000. In addition, the vehicle production has very strong seasonality correlation.

```
library(fpp)

autoplot(ukcars, main ="Quarterly UK Passenger Vehicle Production", ylab =
"Vehicle Production", xlab = "Quarters")
```

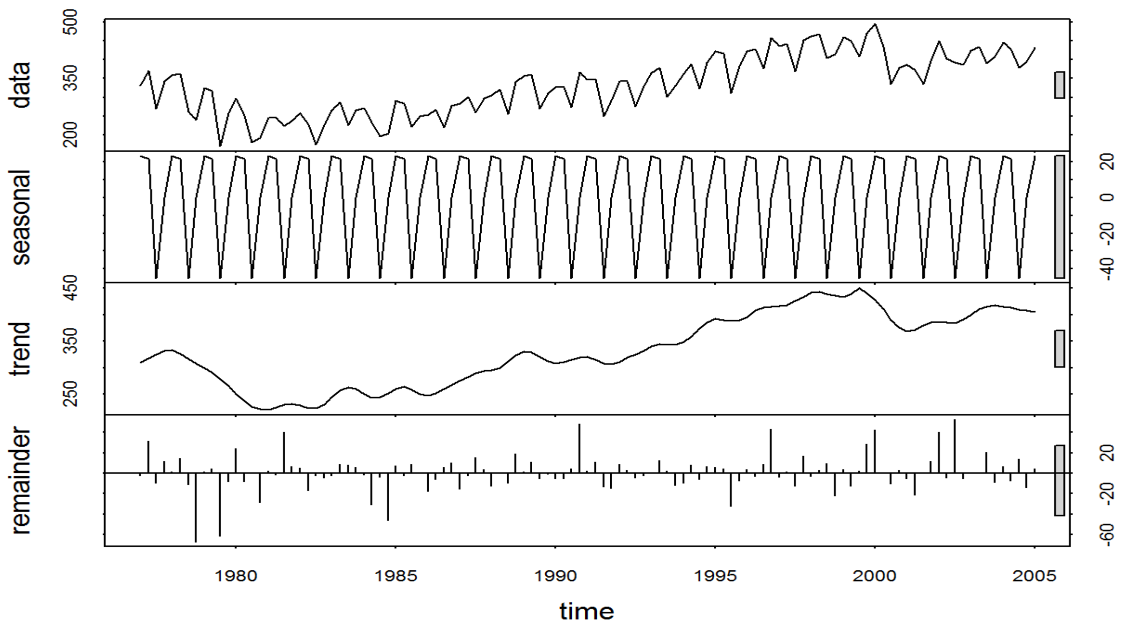


```
fit_decom <- decompose(ukcars)

plot(fit_decom)
```

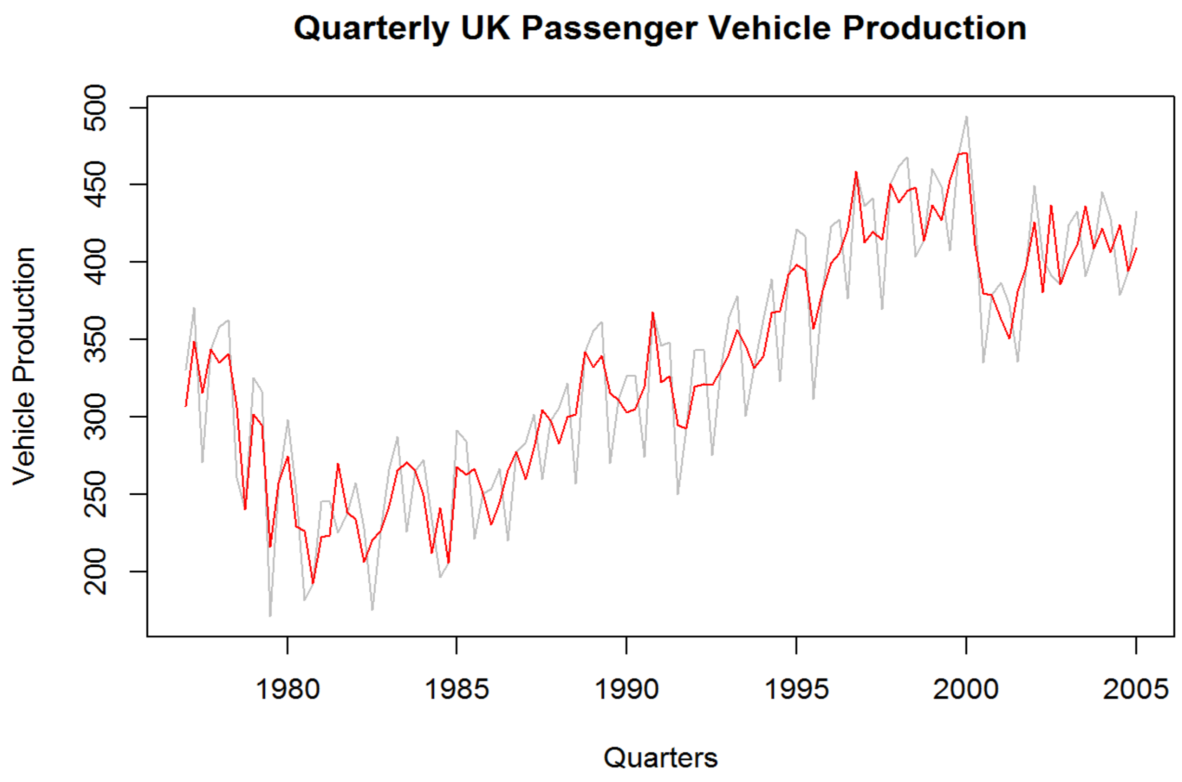


b. Decompose the series using STL and obtain the seasonally adjusted data.

```
fit <- stl(ukcars, s.window = "periodic", robust = TRUE)

plot(fit)
```



```
plot(ukcars, col="grey", main ="Quarterly UK Passenger Vehicle Production",
ylab = "Vehicle Production", xlab = "Quarters")

lines(seasadj(fit), col="red", ylab="Seasonally adjusted")
```

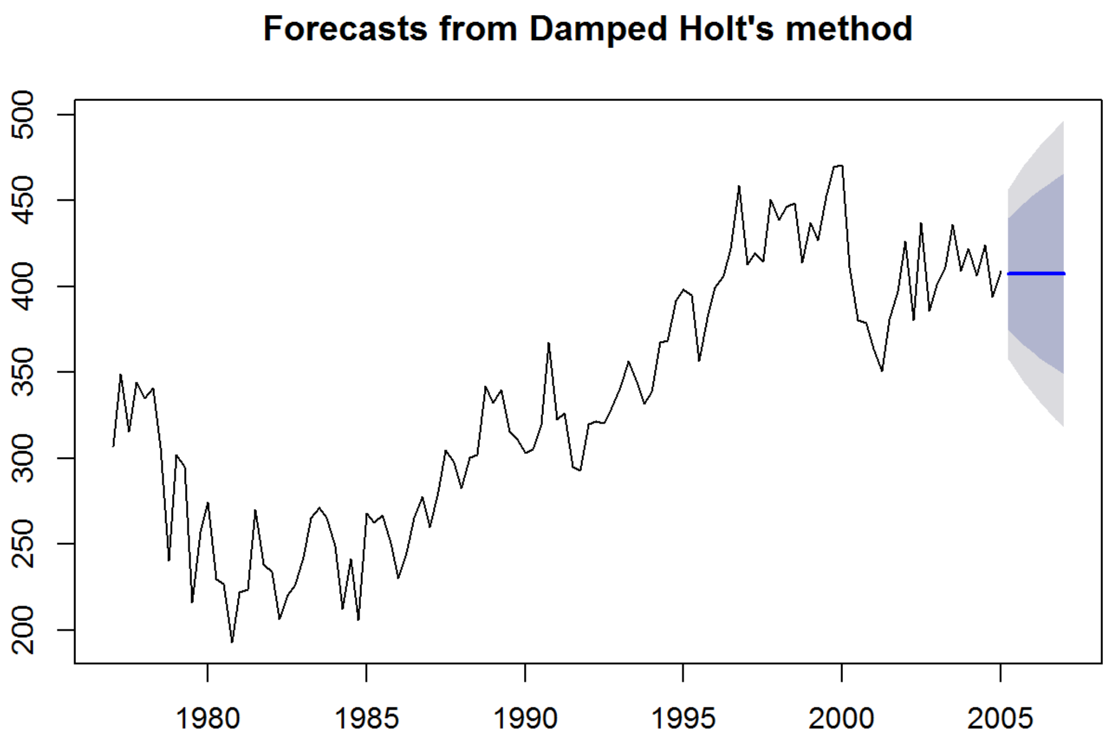


c. Forecast the next two years of the series using an additive damped trend method applied to the seasonally adjusted data. Then reseasonalize the forecasts. Record the parameters of the method and report the RMSE of the one-step forecasts from your method.

As showed in the following, I applied additive damped trend method to seasonally adjusted data. The RMSE is 25.20318. After the forecasts have been reseasonalized, the RMSE does not change.

```
fit2 <- holt(seasadj(fit), damped = TRUE, h = 8)

plot(fit2)

accuracy(fit2)

##                      ME     RMSE     MAE       MPE     MAPE      MASE

## Training set 2.518454 25.20318 20.5804 0.3038991 6.585405 0.6707052

##                     ACF1

## Training set 0.0353549

fit3 <- forecast(fit2, method = "naive")
```

```
plot(fit3)
```

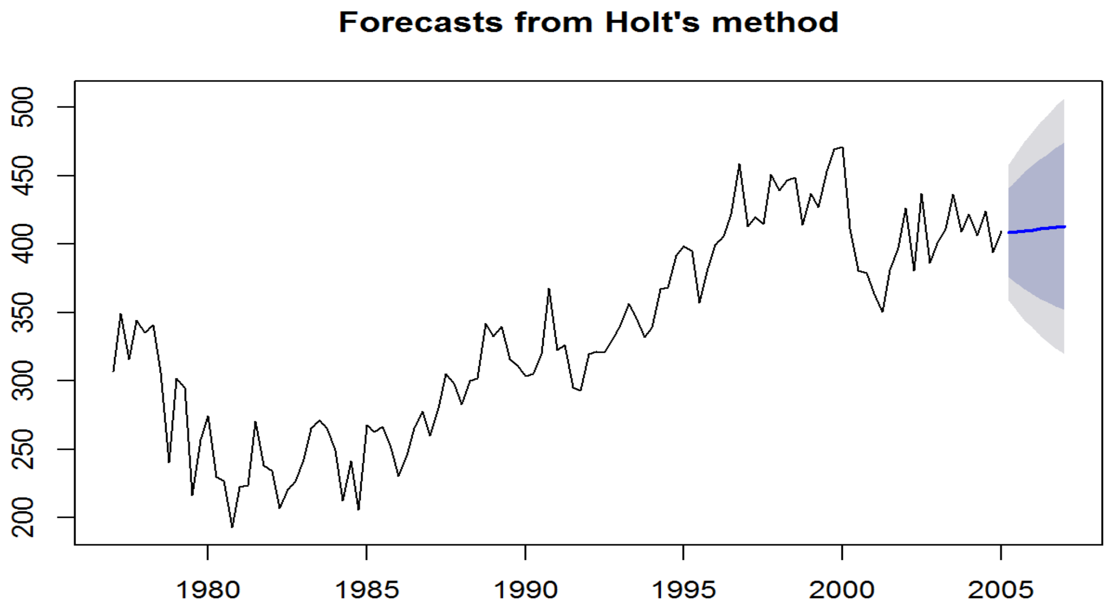


```
accuracy(fit3)
##                     ME     RMSE     MAE       MPE     MAPE      MASE
## Training set 2.518454 25.20318 20.5804 0.3038991 6.585405 0.6707052
##                   ACF1
## Training set 0.0353549
```

d. Forecast the next two years of the series using Holt's linear method applied to the seasonally adjusted data. Then reseasonalize the forecasts. Record the parameters of the method and report the RMSE of of the one-step forecasts from your method.

As showed in the following, I applied Holt's linear method to seasonally adjusted data. The RMSE is 25.39072. After the forecasts have been reseasonalized, the RMSE does not change.

```
fit4 <- holt(seasadj(fit), h = 8)
plot(fit4)
accuracy(fit4)
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.1407116 25.39072 20.14514 -0.5931913 6.500319 0.6565204
##                    ACF1
## Training set 0.02953472
fit5 <- forecast(fit4, method = "naive")
plot(fit5)
```



```
accuracy(fit5)
```

```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.1407116 25.39072 20.14514 -0.5931913 6.500319 0.6565204
##                    ACF1
## Training set 0.02953472
```

e. Now use ets() to choose a seasonal model for the data.

Approaches: There exist two types of models one with additive errors and one with multiplicative errors. Each model also consists of three components or states: error, trend, seasonal. Possibilities for each component are:

Error ={A,M}, Trend ={N,A,Ad,M,Md}, and Seasonal ={N,A,M}. Therefore, there are total 30 state space models. Function ets can be used to estimate the models just by customerzing the arguments.
Interpretation: According to the statistical summary, the estimated model is ETS(A,N,A) with RMSE value to be 25.25792. Model ETS(A,N,A) means this model had additive errors, and additive seasonality, without trend.

```
fit6 <- ets(ukcars)

summary(fit6)
```

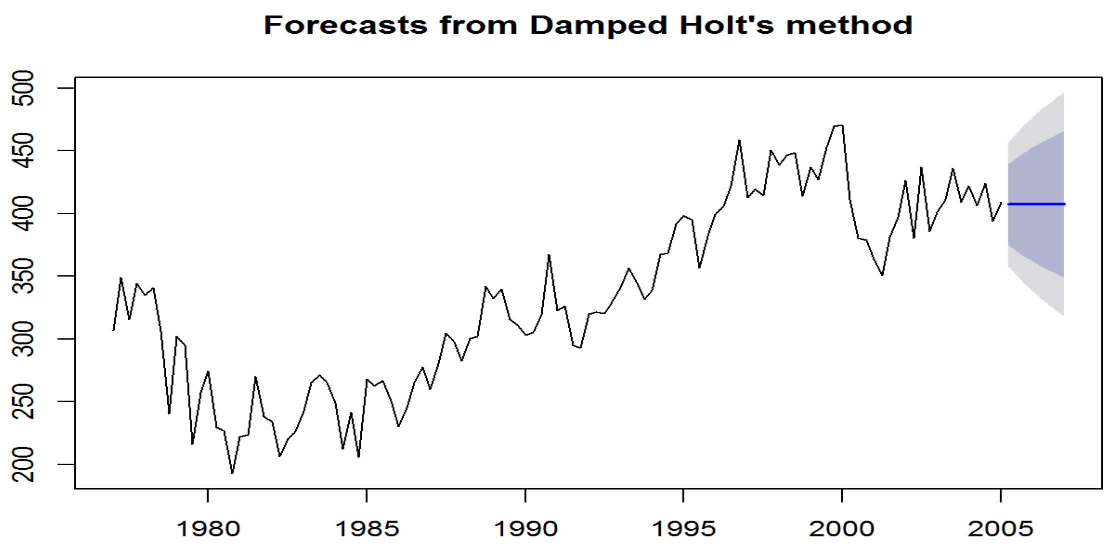
```
## ETS(A,N,A)
##
## Call:
##  ets(y = ukcars)
##
##   Smoothing parameters:
##     alpha = 0.6267
##     gamma = 1e-04
##
##   Initial states:
##     l = 313.0916
##     s=-1.1271 -44.606 21.5553 24.1778
##
##   sigma:  25.2579
##
##      AIC     AICc      BIC
## 1277.980 1279.047 1297.072
##
## Training set error measures:
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 1.324962 25.25792 20.12508 -0.1634983 6.609629 0.6558666
##                    ACF1
## Training set 0.01909295
```

f. Compare the RMSE of the fitted model with the RMSE of the model you obtained using an STL decomposition with Holt's method. Which gives the better in-sample fits?

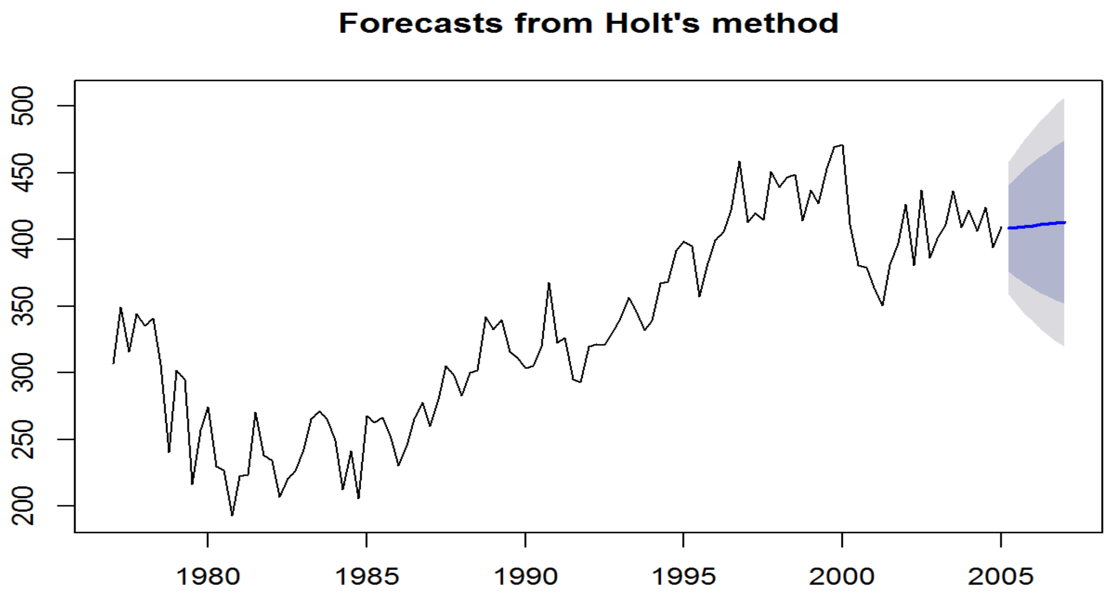
After we using additive damped trend method, Holt's linear method, and ets, we obtain three RMSE values. They are 25.20318, 25.39072, and 25.25792 respectively. All of them are not identical, but they are very closed to each other, which deem three models have similar accuracy to forecast the future vehicle productions.

g. Compare the forecasts from the two approaches? Which seems most reasonable?

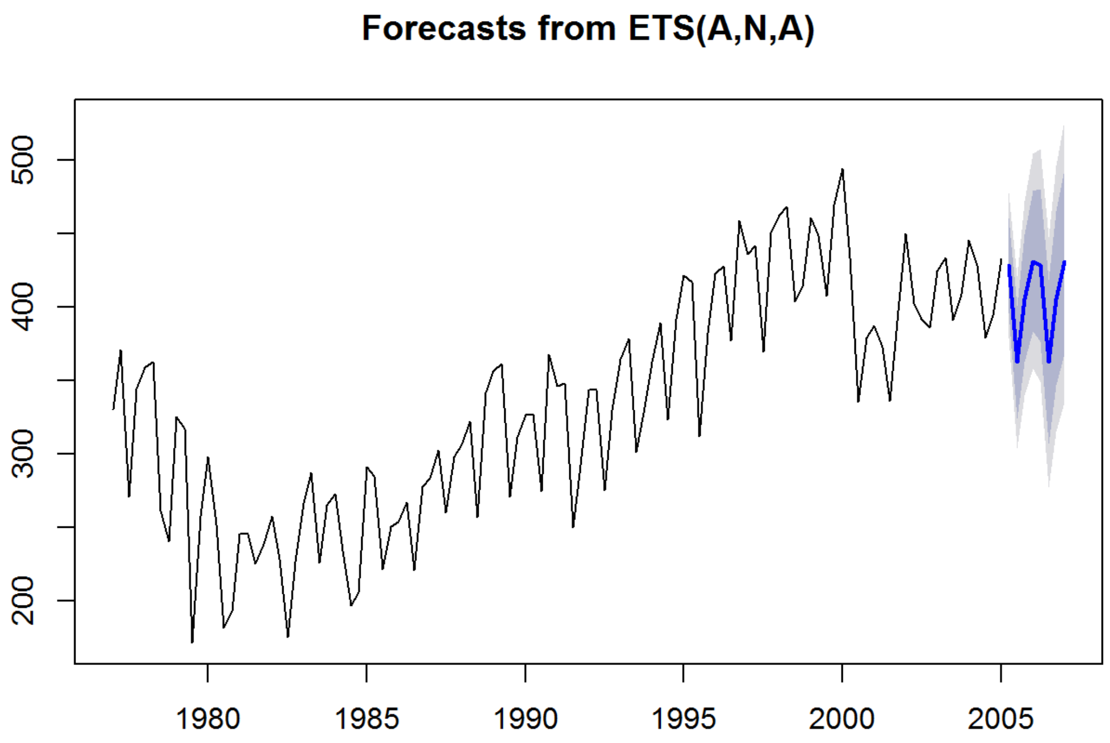
```r
plot(fit2)
```



```r
plot(fit4)
```



```r
plot(forecast(fit6,  h = 8))
```



I think ETS(A,N,A) is more reasonable, because it oscillate throughout the year which is a better presentation for the seasonality of the vehicle productions.

Homework HA Chapter 8

8.11 1. Figure 8.24 shows the ACFs for 36 random numbers, 360 random numbers and for 1,000 random numbers.

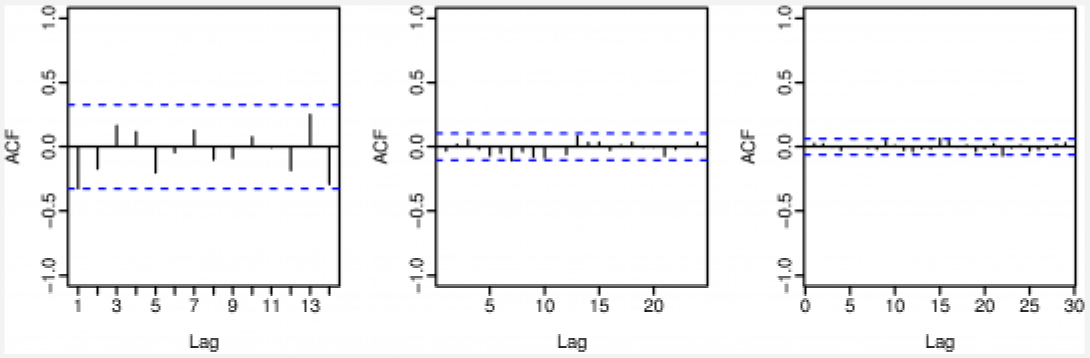


Figure 8.24: Left: ACF for a white noise series of 36 numbers. Middle: ACF for a white noise series of 360 numbers. Right: ACF for a white noise series of 1,000 numbers.

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

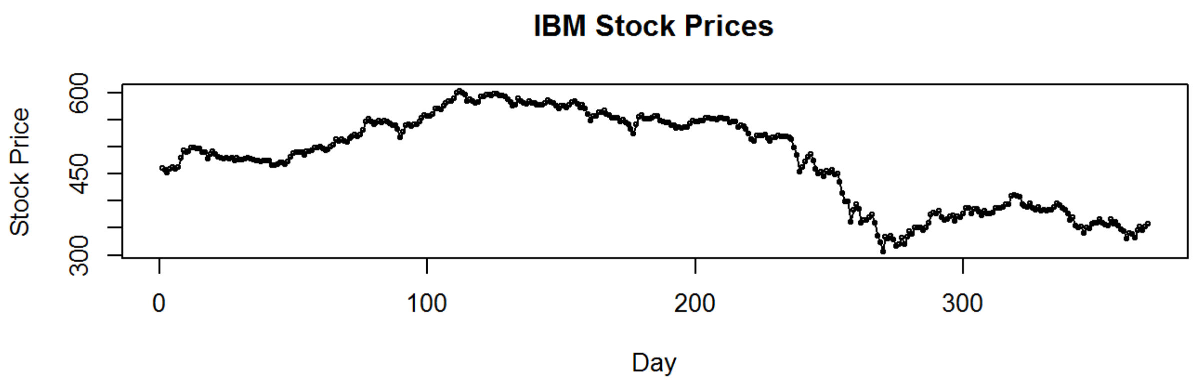
The autocorrelation coefficients are normally plotted to form the autocorrelation function or ACF. The plot is also known as a correlogram. Time series that show no autocorrelation are called “white noise”. If autocorrelations are small enough (less than the bounds), this is the evidence that the data are white noise. Therefore, they all indicate the data are white noise, since majority of the data (at least 95%) are under the bounds. The differences among these figures are the number of lags as shown on the x-axis.

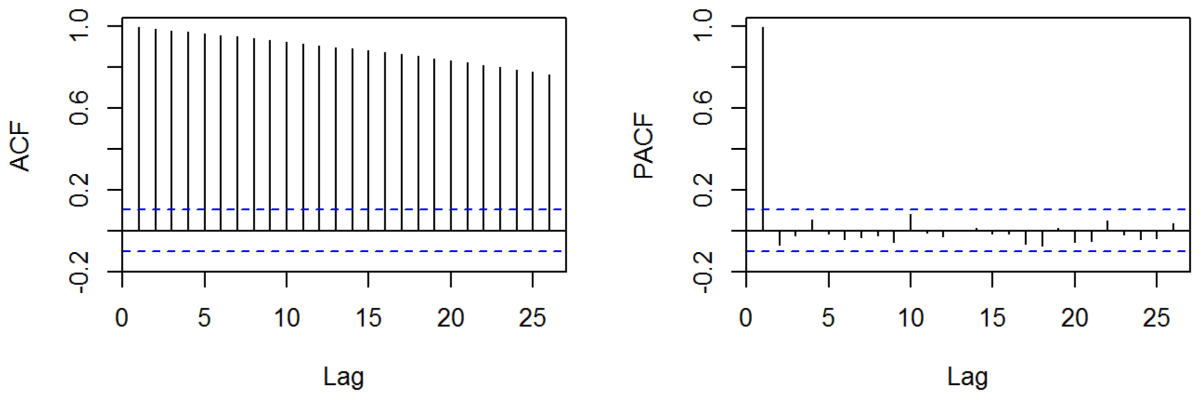
b. Why are the critical values at different distances from the mean of zero? Why are the autocorrelations different in each figure when they each refer to white noise?

The formula to calculate the bound is +/- (2/sqrt(T)), where T is the length of the time series. The larger the T, the smaller the bounds. For these three figures, they different number of data, that is why the critical values are at different distances from the mean of zero.

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

```r
library(forecast)

library(fma)

library(fpp)

library(ggplot2)

tsdisplay(ibmclose, main="IBM Stock Prices", ylab = "Stock Price",
xlab="Day")
```





Approaches: Autocorrelation measures the linear relationship between lagged values of a time series. ACF is useful to identify non-stationary time series. for non-stationary data, ACF decreases slowly, and vice versa. In addition, the value of ACF tends to be large and positive for non-stationary data. Differencing is one way to make a time series stationary. Basically it means to compute the differences between consecutive observations, so that eliminate trend and seasonality.

Interpretation:

Time Series with trends, or with seasonality, are not stationary. The Time Series graph above shows upward then downward trend. Therefore, it is non-stationary. Moreover, the ACF values are all out of bounds, all of which are decreasing over time. This further proves the data is non-stationary. The PACF means partial autocorrelation function. PACF only describes the direct relationship between an observation and its lag. Based on what is shown above, there is no obvious partial correlation exist (besides the partial correlation with itself).

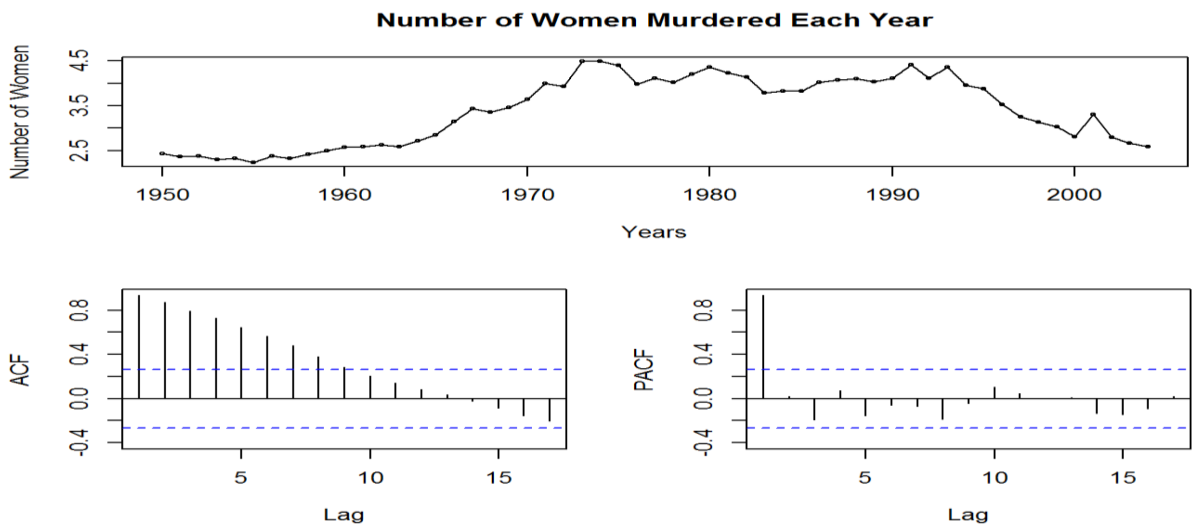
6. Consider the number of women murdered each year (per 100,000 standard population) in the United States (data set wmurders).

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

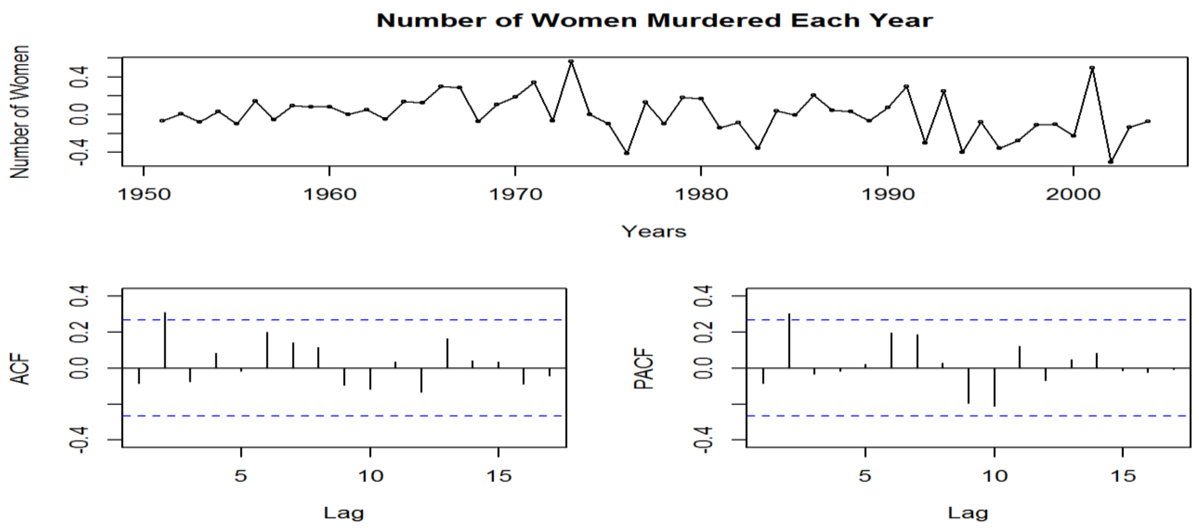
Approaches: Differencing can help stabilize the mean of a time series by removing changes in the level of a time series, and so eliminating trend and seasonality. This is one way to make a non-stationary time series stationary.

Interpretation: The Time Series graph has no apparent seasonality, but it has strong upward trend before the year 1970 and strong downward trend after 1990. This data is non-stationary. The first difference graph looks stationary, however, the ACF and PACF figure each one of them has on spike that is going out of the bounds. We need to perform Unit Root test and KPSS test to check if more differencing is necessary or not.

```
tsdisplay(wmurders, main="Number of Women Murdered Each Year", ylab = "Number of Women", xlab="Years")
```



```
tsdisplay(diff(wmurders), main="Number of Women Murdered Each Year", ylab = "Number of Women", xlab="Years")
```

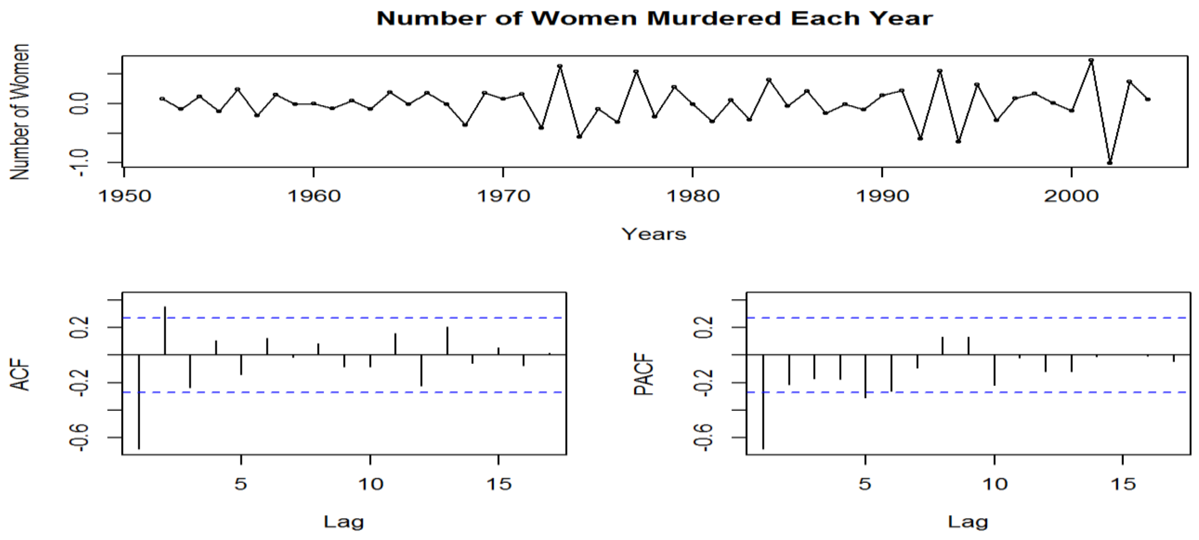


The p-value from the Unit Root Test is 0.02726, which is less than 5%, which indicates that the data is stationary. The p-value from the KPSS Test is 0.02379, which is less than 5%, which indicate that the data is non-stationary. We need to perform second differencing.

```
adf.test(diff(wmurders))
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(wmurders)
## Dickey-Fuller = -3.7688, Lag order = 3, p-value = 0.02726
## alternative hypothesis: stationary
kpss.test(diff(wmurders))
##
##  KPSS Test for Level Stationarity
##
## data:  diff(wmurders)
## KPSS Level = 0.58729, Truncation lag parameter = 1, p-value =
## 0.02379
```

The second differencing graph appear to be much more stationary. On PACF graph, there is a significant spike at lag 1, but none beyond lag 1, so that we can determine that the p = 1. On ACF graph, there is a significant spike at lag 1, but none beyond lag 1, so that we can determine that the q = 1. Therefore, the appropriate ARIMA(p,d,q) model is ARIMA(1, 2, 1)

```
tsdisplay(diff(diff(wmurders)), main="Number of Women Murdered Each Year",
ylab = "Number of Women", xlab="Years")
```



```
adf.test(diff(diff(wmurders)))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(diff(wmurders))
## Dickey-Fuller = -5.1646, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(diff(diff(wmurders)))
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  diff(diff(wmurders))
## KPSS Level = 0.030483, Truncation lag parameter = 1, p-value = 0.1
```

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

If c=0 and d=2, the long-term forecasts will follow a straight line. If c???0 and d=2, the long-term forecasts will follow a quadratic trend. From the Time Series figure, the data seem to follow a quadratic trend, therefore a constant need to be included.

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

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

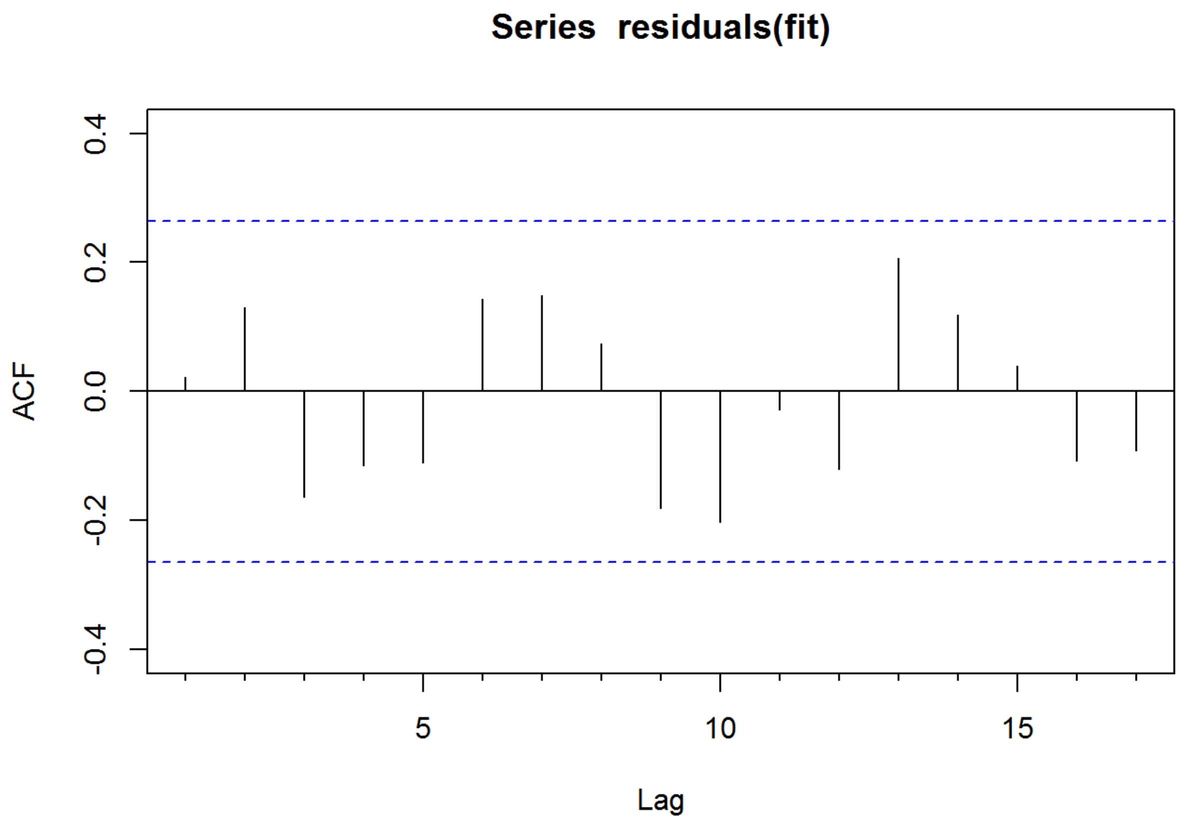
d. Fit the model using R and examine the residuals. Is the model satisfactory?

The ACF plot of the residuals shows all correlations within the threshold limits indicating that the residuals are behaving like white noise. A portmanteau test returns a p-value of 0.1039, which is also greater than the threshold 0.05. Therefore, the result indicates the residuals are white noise, so that the model is satisfactory.

```
fit <- Arima(wmurders, order = c(1, 2, 1), include.constant = TRUE)
summary(fit)
```

```
## 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
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set -0.01065956 0.2072523 0.1528734 -0.2149476 4.335214 0.9400996
##                    ACF1
## Training set 0.02176343
```

```
Acf(residuals(fit))
```



```
Box.test(residuals(fit), lag=24, fitdf=4, type="Ljung")
##
##  Box-Ljung test
##
## data:  residuals(fit)
## X-squared = 28.238, df = 20, p-value = 0.1039
```

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

Approached: 1. Expand the ARIMA equation so that yt is on the left hand side and all other terms are on the right. 2. Rewrite the equation by replacing t by T+h. 3. On the right hand side of the equation, replace future observations by their forecasts, future errors by zero, and past errors by the corresponding residuals.

```
murder_forecast <- forecast(fit, h = 3)
print(murder_forecast)
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2005       2.470660 2.194836 2.746484 2.048824 2.892496
## 2006       2.363106 1.986351 2.739862 1.786908 2.939304
## 2007       2.252833 1.765391 2.740276 1.507354 2.998313
```

```
summary(murder_forecast)
```

```
##
## Forecast method: ARIMA(1,2,1)
##
## Model Information:
## 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
##
## Error measures:
##                        ME      RMSE       MAE        MPE     MAPE      MASE
## Training set -0.01065956 0.2072523 0.1528734 -0.2149476 4.335214 0.9400996
##                    ACF1
## Training set 0.02176343
##
## Forecasts:
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2005       2.470660 2.194836 2.746484 2.048824 2.892496
## 2006       2.363106 1.986351 2.739862 1.786908 2.939304
## 2007       2.252833 1.765391 2.740276 1.507354 2.998313
```

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

Where $\phi_1 = -0.2434$ and $\theta_1 = -0.8261$

$$\left[1-(2+\phi_1)B+(1+2\phi_1)B^2-\phi_1 B^3\right] y_t = c + (1+\theta_1 B)e_t$$

$$[y_t-(2+\phi_1)y_{t-1}+(1+2\phi_1)y_{t-2}-\phi_1 y_{t-3}] = c + e_t + \theta_1 e_{t-1}$$

$$y_t = (2+\phi_1)y_{t-1}-(1+2\phi_1)y_{t-2}+\phi_1 y_{t-3}+e_t+\theta_1 e_{t-1}+c$$

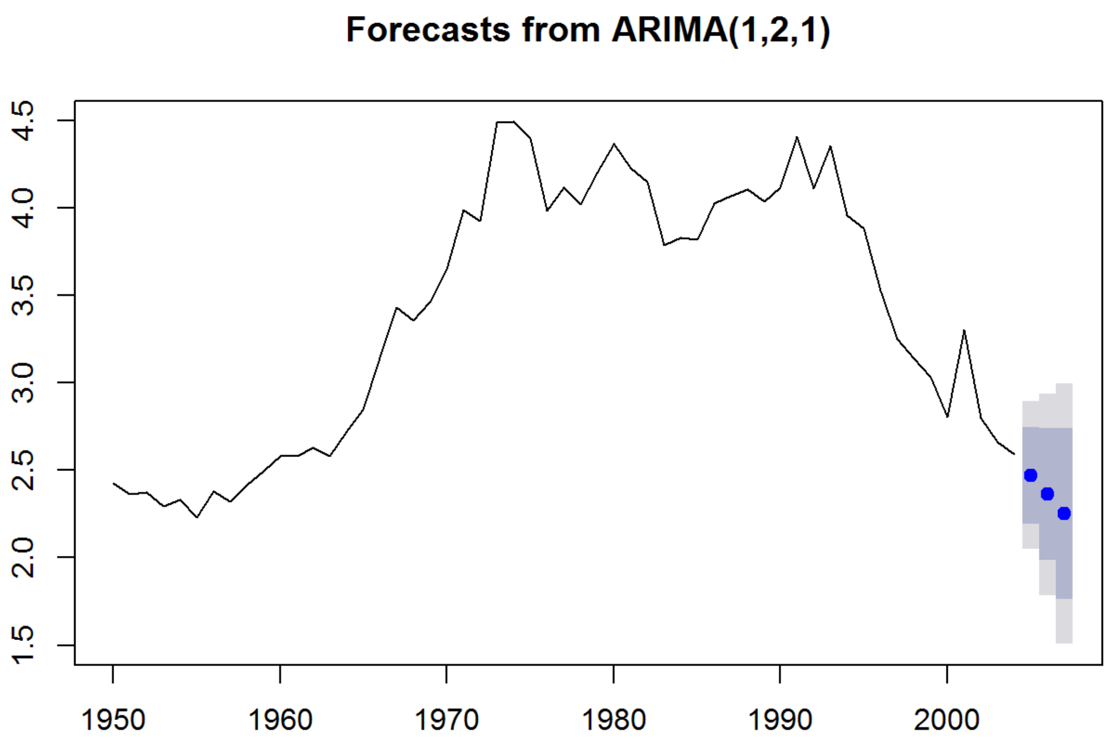
$$y_{T+1} = (2+\phi_1)y_T-(1+2\phi_1)y_{T-1}+\phi_1 y_{T-2}+e_{T+1}+\theta_1 e_T+c$$

$$y_{T+2} = (2+\phi_1)y_{T+1}-(1+2\phi_1)y_T+\phi_1 y_{T-1}+e_{T+2}+\theta_1 e_{T+1}+c$$

$$y_{T+3} = (2+\phi_1)y_{T+2}-(1+2\phi_1)y_{T+1}+\phi_1 y_T+e_{T+3}+\theta_1 e_{T+2}+c$$

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

```
plot(murder_forecast)
```



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

The auto.arima give the same model I have chosen. The better model is the model that can minimize the AIC and BIC, since both models are the same. They are equally well.

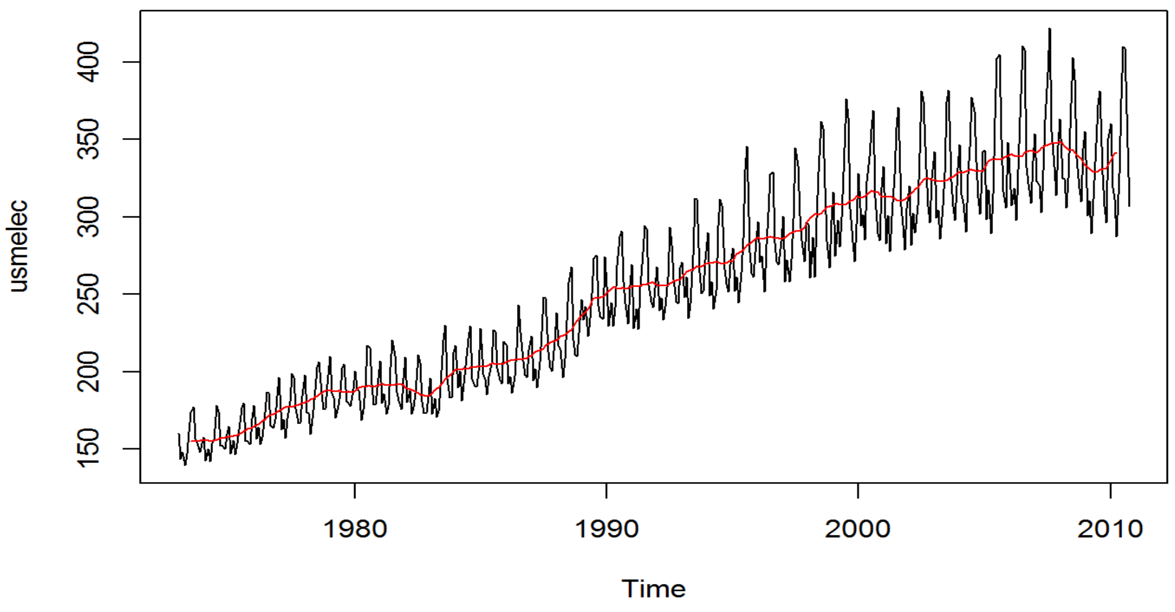
```
fit1 <- auto.arima(wmurders, seasonal=FALSE)
summary(fit1)
```

```
## 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
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set -0.01065956 0.2072523 0.1528734 -0.2149476 4.335214 0.9400996
##                    ACF1
## Training set 0.02176343
```

8. Consider the total net generation of electricity (in billion kilowatt hours) by the U.S. electric industry (monthly for the period 1985-1996). (Data set usmelec.) In general there are two peaks per year: in mid-summer and mid-winter.

a. Examine the 12-month moving average of this series to see what kind of trend is involved.

```
plot(usmelec)
lines(ma(usmelec, order = 12), col = "red")
```

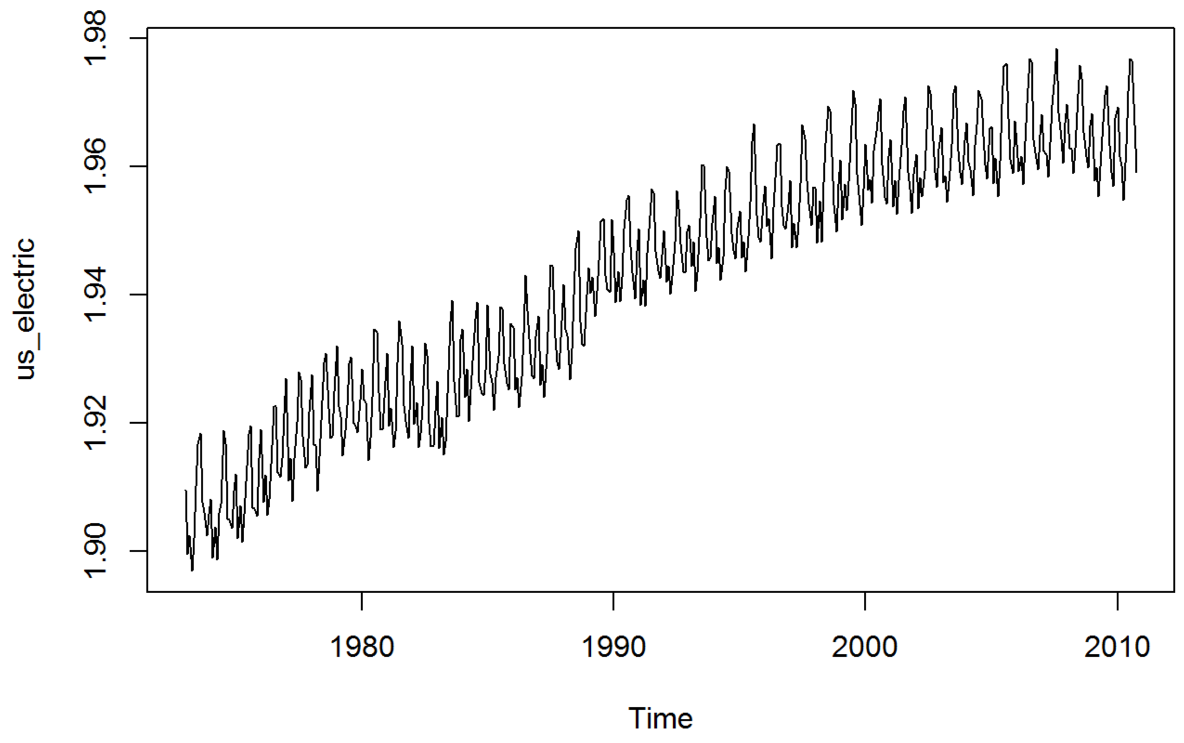


There is increase electricity net generation in the US over years. In addition, it is also associated with strong seasonality factors.

b. Do the data need transforming? If so, find a suitable transformation.

Yes, because the variance of the dataset keep increasing over time. Transformations is necessary to help stabilize the variance of a time series. The Box-Cox Transformation dramatically decrease its variances as shown in the figure. Therefore, it is a suitable transformation.

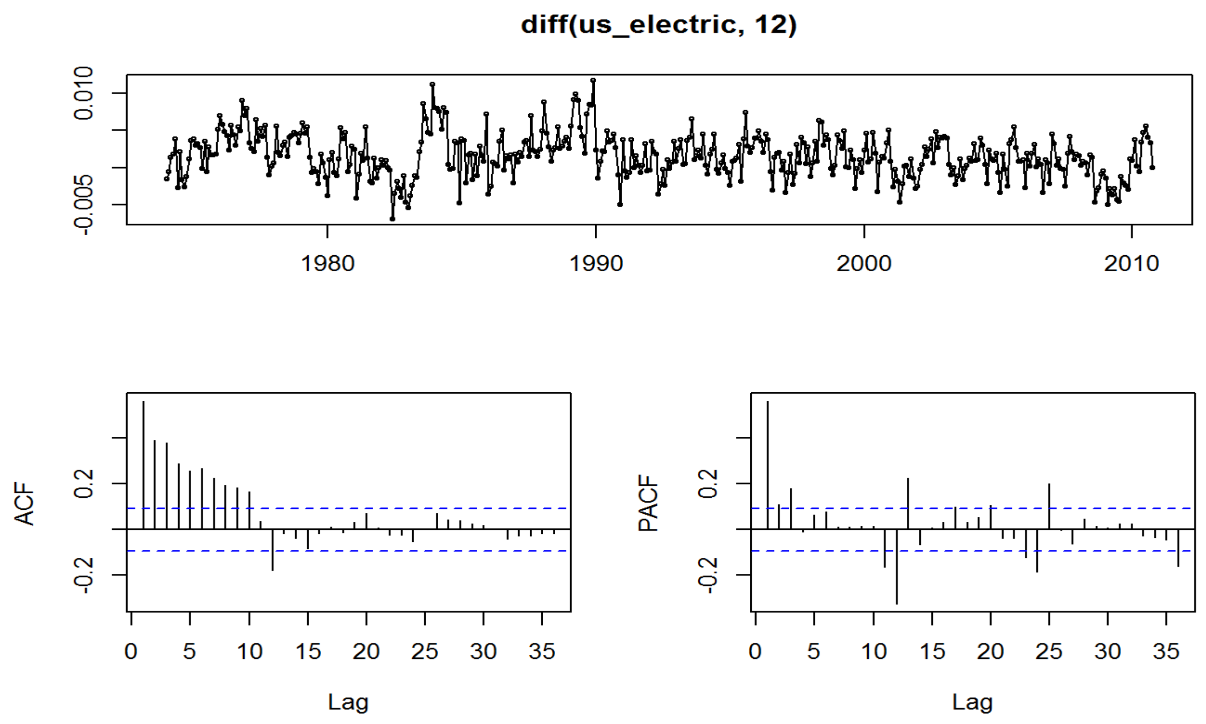
```
lambda <- BoxCox.lambda(usmelec)

us_electric <- BoxCox(usmelec, lambda = lambda)

plot(us_electric)
```



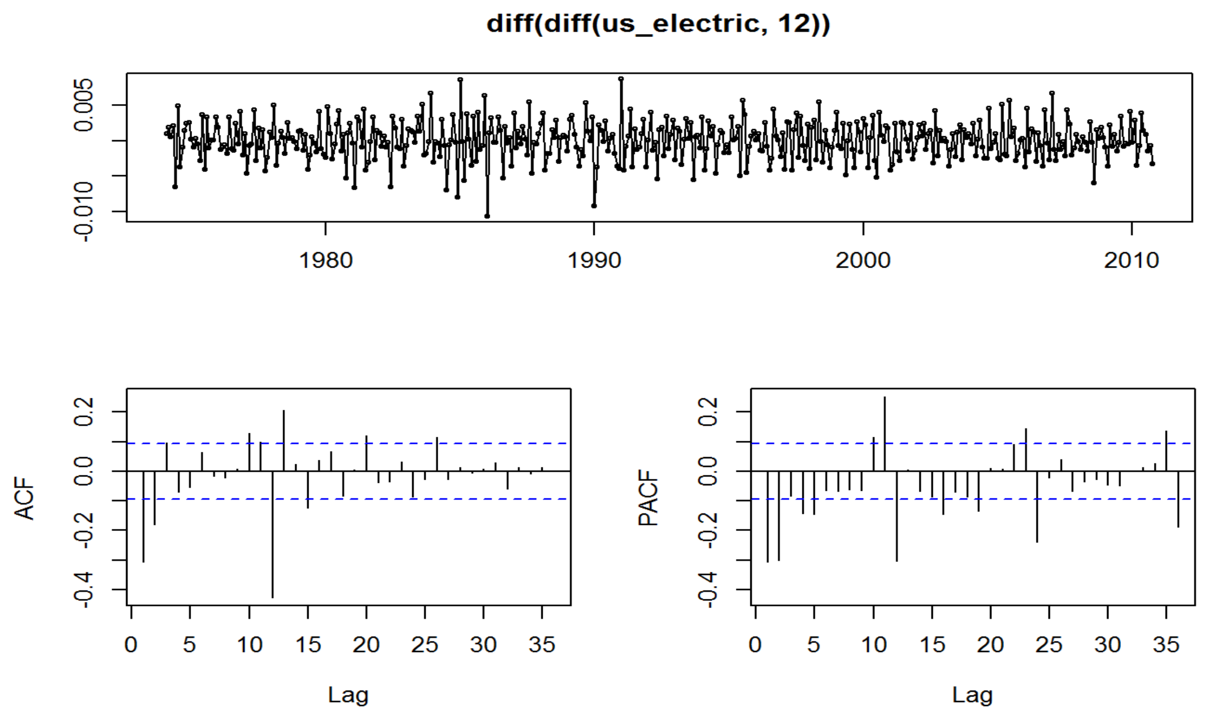
c. Are the data stationary? If not, find an appropriate differencing which yields stationary data.

Time series with trends or with seasonality are not stationary. Therefore, this dataset is not stationary neither. Since the dataset has strong seasonal pattern, I want to try the seasonal differencing first. The ACF of this differencing drops slowly towards zero. This means the differenced data is still non-stationary. Therefore, I need to perform second-order differencing. The resulting graph looks much better, Both Unit Root test (p<0.05) and KPSS (p>0.05) test proved that the resulting dataset is stationary now.

```r
tsdisplay(diff(us_electric, 12))
```



```r
tsdisplay(diff(diff(us_electric, 12)))
```



```
adf.test(diff(diff(us_electric, 12)))
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(diff(us_electric, 12))
## Dickey-Fuller = -10.551, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
kpss.test(diff(diff(us_electric, 12)))
##
##  KPSS Test for Level Stationarity
##
## data:  diff(diff(us_electric, 12))
## KPSS Level = 0.012984, Truncation lag parameter = 4, p-value = 0.1
```

d. Identify a couple of ARIMA models that might be useful in describing the time series. Which of your models is the best according to their AIC values?

The significant spike at lag 1 in the ACF suggests non-seasonal MA component, and the significant spike at lag 12 in the ACF suggests the seasonal MA component. On the other hand, The significant spike at lag 1 in the PACF suggests non-seasonal AR component, and the significant spike at lag 12 in the ACF suggests a seasonal AR component.

Therefore, we can test on few models as shown in the following. ARIMA(0,1,2)(0,1,1)12 is the best models because it has the lowest level of Aic (-4257.311)

```
fit1 <- Arima(usmelec, order=c(0,1,1), seasonal=c(0,1,1), lambda = lambda)
fit2 <- Arima(usmelec, order=c(0,1,2), seasonal=c(0,1,1), lambda = lambda)
fit3 <- Arima(usmelec, order=c(0,1,3), seasonal=c(0,1,1), lambda = lambda)
fit4 <- Arima(usmelec, order=c(1,1,0), seasonal=c(1,1,0), lambda = lambda)
fit5 <- Arima(usmelec, order=c(2,1,0), seasonal=c(1,1,0), lambda = lambda)
fit6 <- Arima(usmelec, order=c(3,1,0), seasonal=c(1,1,0), lambda = lambda)
```

```
fit1$aic
```

```
## [1] -4231.601
```

```
fit2$aic
```

```
## [1] -4257.311
```

```
fit3$aic
```

```
## [1] -4256.383
```

```
fit4$aic
```

```
## [1] -4073.213
```

```
fit5$aic
```

```
## [1] -4108.789
```

```
fit6$aic
```

```
## [1] -4109.758
```

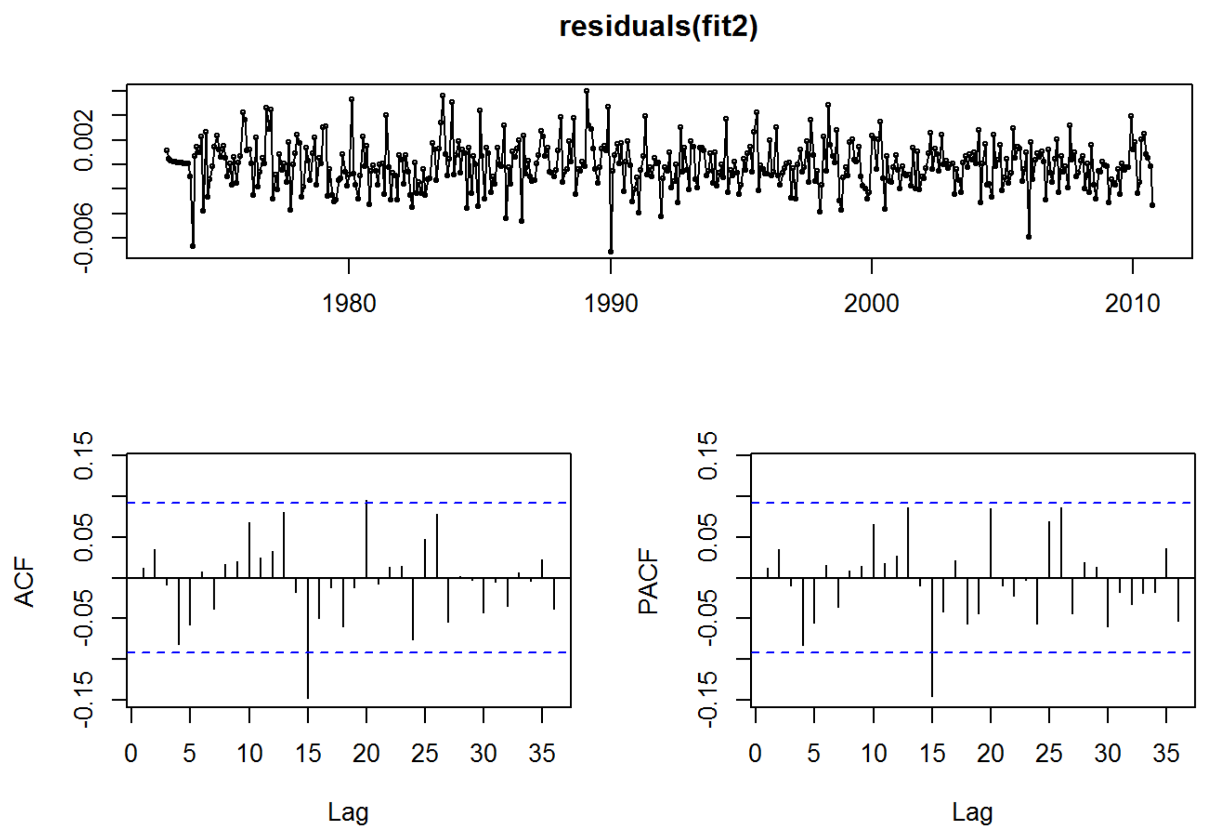
e. Estimate the parameters of your best model and do diagnostic testing on the residuals. Do the residuals resemble white noise? If not, try to find another ARIMA model which fits better.

The following shows the parameters of the model. The ACF and PACF plot of the residuals shows majority of correlations are within the threshold limits (<5%) indicating that the residuals are behaving like white noise. Although the portmanteau test returns a p-value of 0.03357, which is less than the threshold 0.05, the other models have even worse p-value after running each one of them. Therefore, the result indicates the residuals resemble white noise.

```
fit2$coef
```

```
##        ma1        ma2       sma1
## -0.4274796 -0.2570436 -0.8582555
```

```
tsdisplay(residuals(fit2))
```

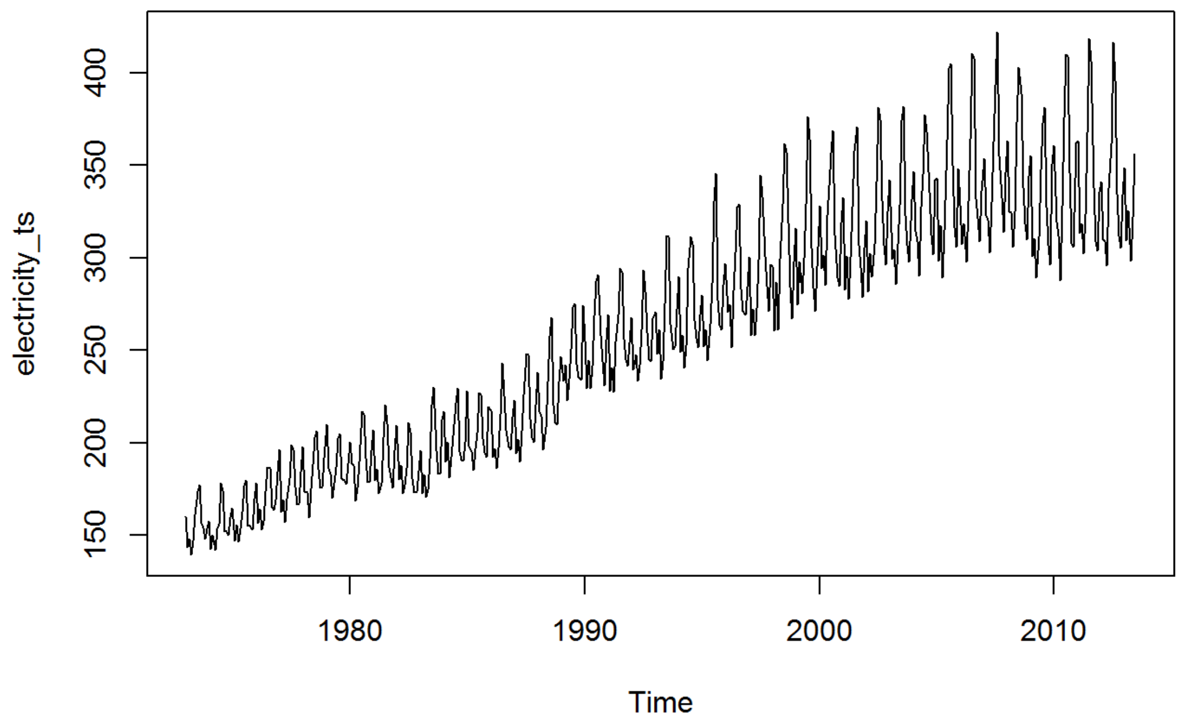


```
Box.test(residuals(fit2), lag=24, fitdf=4, type="Ljung")
```

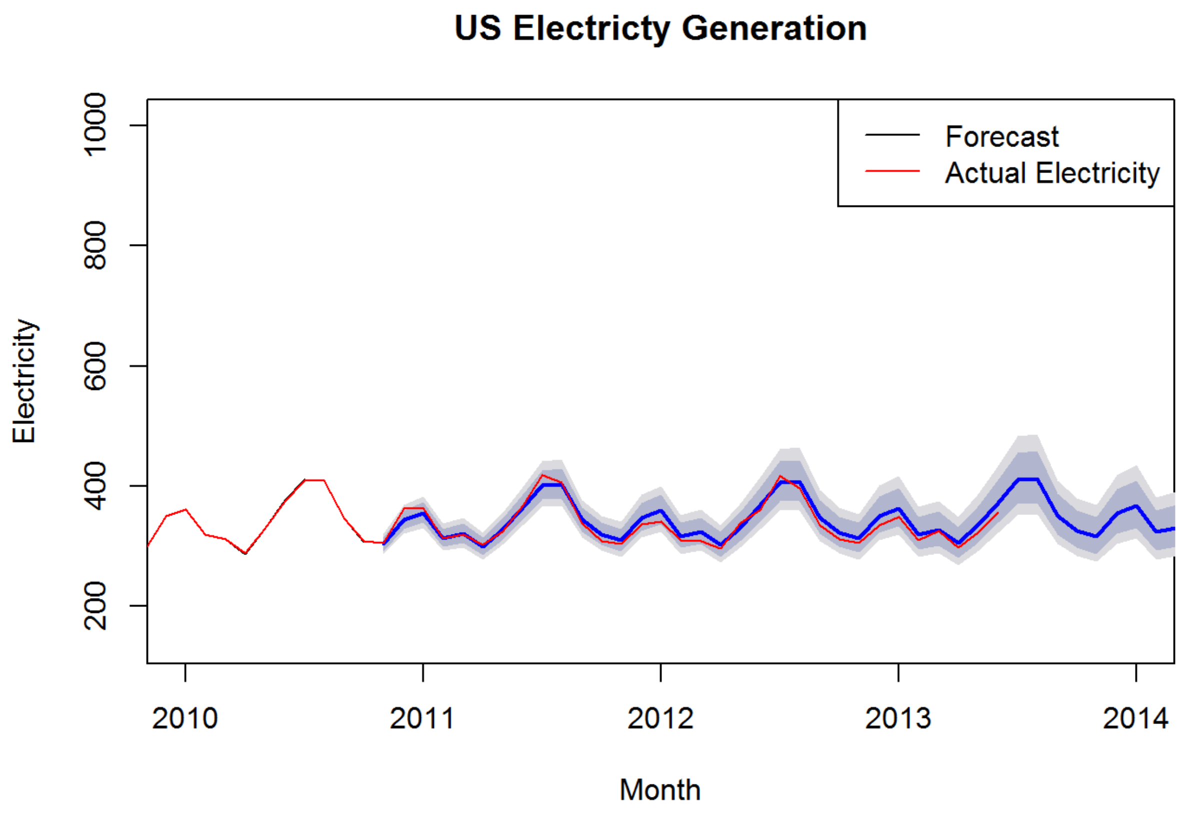
```
##
##  Box-Ljung test
##
## data:  residuals(fit2)
## X-squared = 33.02, df = 20, p-value = 0.03357
```

f. Forecast the next 15 years of generation of electricity by the U.S. electric industry. Get the latest figures from http://data.is/zgRWCO to check on the accuracy of your forecasts.

```
g. actual_data <- read.csv("electricity-overview.csv")

h. colnames(actual_data) <- c("Month", "Electricity")

i. electricity_ts <- ts(actual_data$Electricity, start = c(1973, 1),
   frequency = 12)

j. plot(electricity_ts)
```



```
e_forecast <- forecast(fit2, h = 180)

plot(e_forecast, main = "US Electricty Generation", ylab = "Electricity",
xlab = "Month", xlim = c(2010, 2014))

lines(electricity_ts, col = "red")

legend("topright", lty = 1, col = c(1, 2), c("Forecast", "Actual
Electricity"))
```



The forecast has been very accurate as shown above, also as shown below.

```
accuracy(e_forecast, electricity_ts)
```

```
##                      ME      RMSE      MAE       MPE     MAPE      MASE
## Training set -0.3499476  7.121762 5.194804 -0.150549 1.996946 0.5705842
## Test set     -4.3607938 10.349199 8.981722 -1.412878 2.644302 0.9865298
##                     ACF1 Theil's U
## Training set -0.01270233        NA
## Test set      0.45559428 0.3310075
```

g. How many years of forecasts do you think are sufficiently accurate to be usable?

Since the model only accurately predct the data from 10/2010 up to 06/2013 (the most recent available actual data), therefore, the forecast is only accurate for around 5 years.