Predict_413_Sec55_Homework_1

Singh, Gurjeet

January 28, 2018

Contents

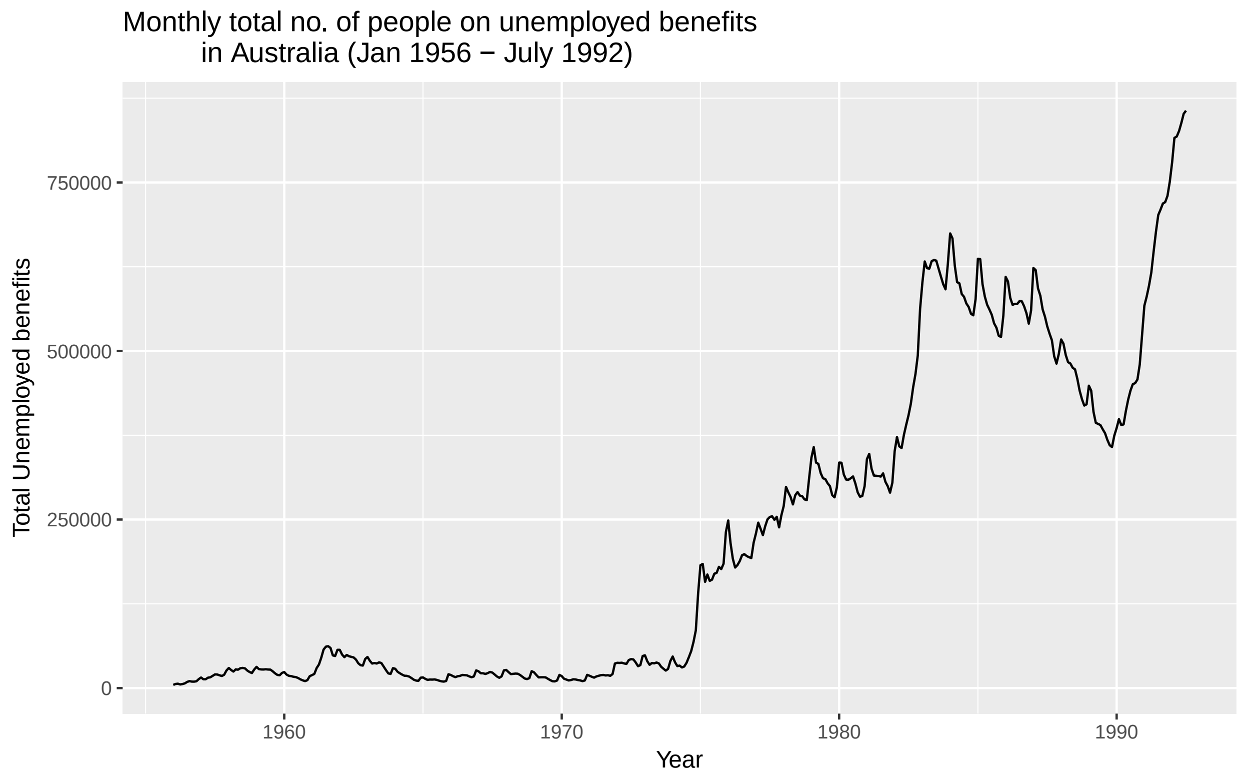
Chapter 2

Question 1

Ch2.Q1.a)

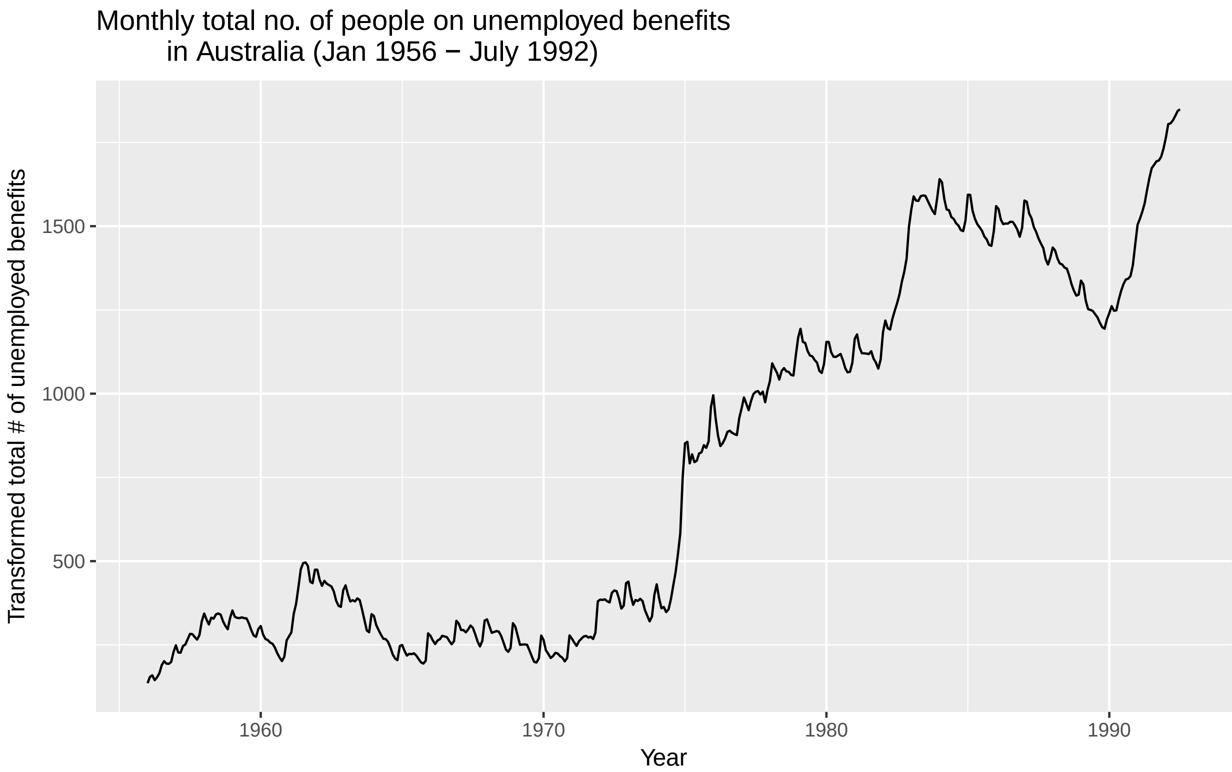
The figure below shows the plot of the monthly total of people on unemployment benefits in Australia from January 1956 to July 1992.

```
data("dole")
autoplot(dole) +
  ggtitle("Monthly total no. of people on unemployed benefits
          in Australia (Jan 1956 - July 1992)") +
  xlab("Year") + ylab("Total Unemployed benefits")
```



The next figure shows the plot with the BoxCox transformation with the 0.5 parameters. The plot seems to smooth a little after the transformation. It is not easy to identify the pattern between the years 1956 and 1975.

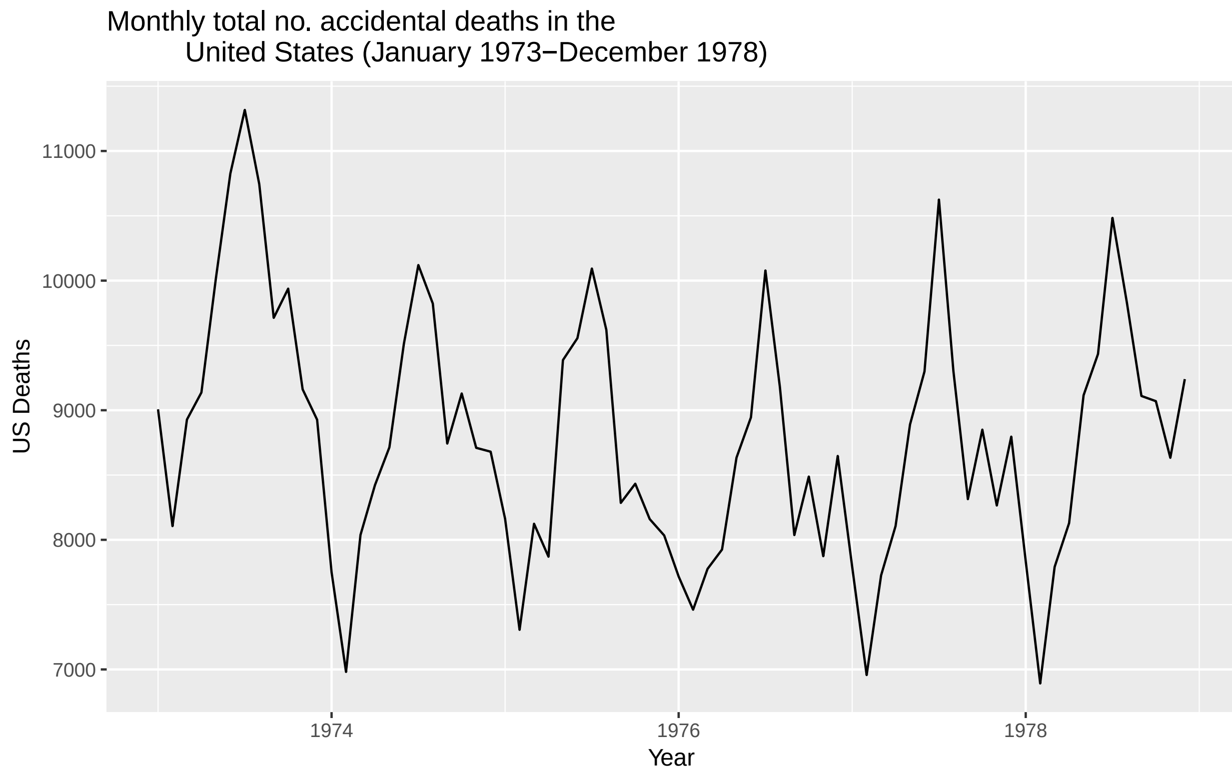
```
autoplot(BoxCox(dole,0.5)) +
  ggtitle("Monthly total no. of people on unemployed benefits
          in Australia (Jan 1956 - July 1992)") +
  xlab("Year") + ylab("Transformed total # of unemployed benefits")
```



Ch2.Q1.b)

The figure below shows the plot of the monthly total of accidental deaths in the United States from January 1973 to December 1978.

```r
data("usdeaths")
autoplot(usdeaths) +
  ggtitle("Monthly total no. accidental deaths in the
          United States (January 1973-December 1978)") +
  xlab("Year") + ylab("US Deaths")
```

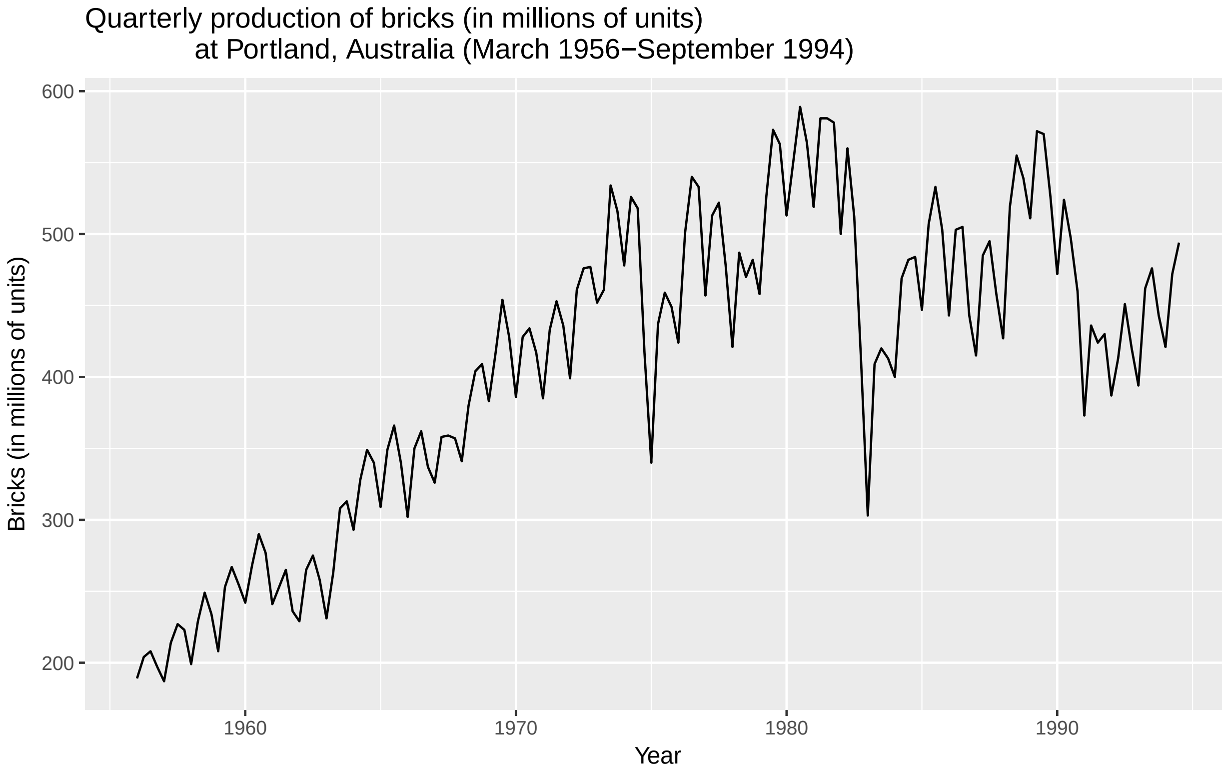



```r
  # autoplot(BoxCox(usdeaths,0.5)) +
  #   ggtitle("Monthly total no. accidental deaths in the
  #           United States (January 1973-December 1978)") +
  #   xlab("Year") + ylab("US Deaths")
```

Ch2.Q1.c)

The figure below shows the plot of the quarterly production of bricks (in millions of units) at Portland, Australia from March 1956 to September 1994.

```r
data("bricksq")
  # head(bricksq)
autoplot(bricksq) +
  ggtitle("Quarterly production of bricks (in millions of units)
              at Portland, Australia (March 1956-September 1994)") +
  xlab("Year") + ylab("Bricks (in millions of units)")
```



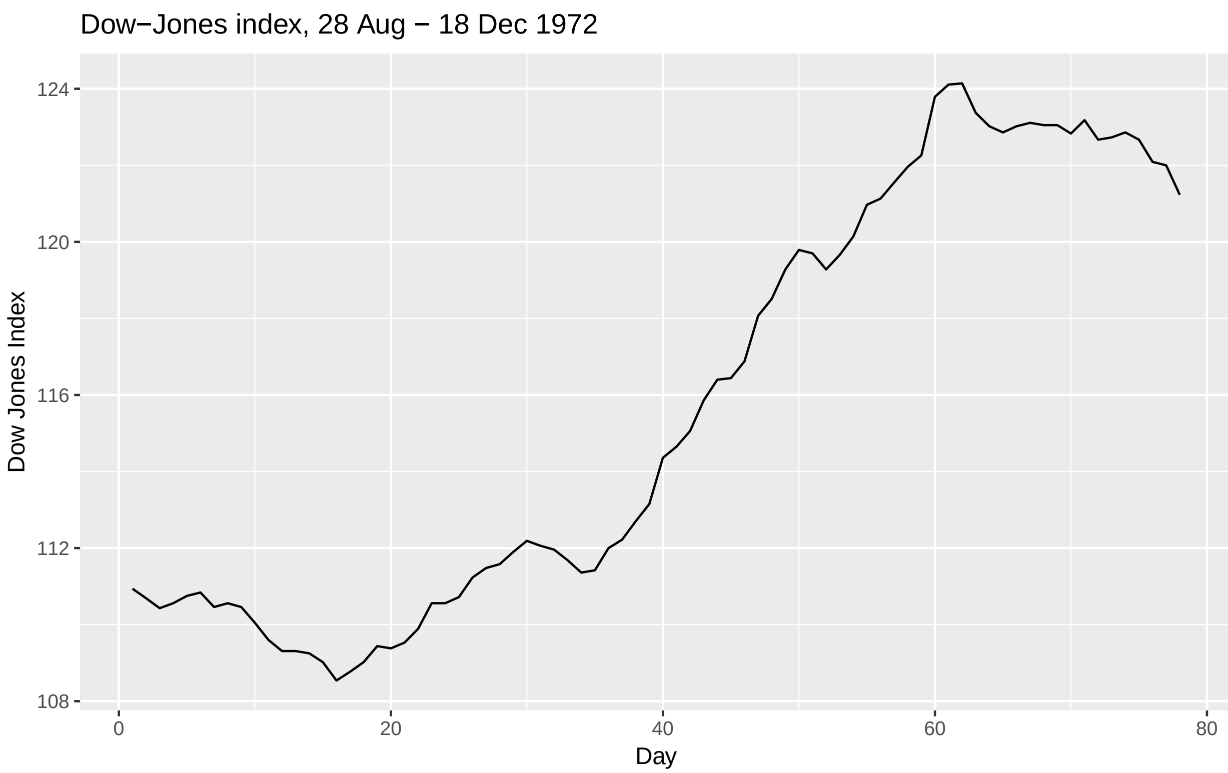

```r
# autoplot(BoxCox(bricksq, 0.5)) +
#   ggtitle("Quarterly production of bricks (in millions of units)
#               at Portland, Australia (March 1956-September 1994)") +
#   xlab("Year") + ylab("Bricks (in millions of units)")
```

Question 2

Ch2.Q2.a)

The figure below shows the time series plot of Dow Jones index from August 28, 1972 to December 18, 1972.

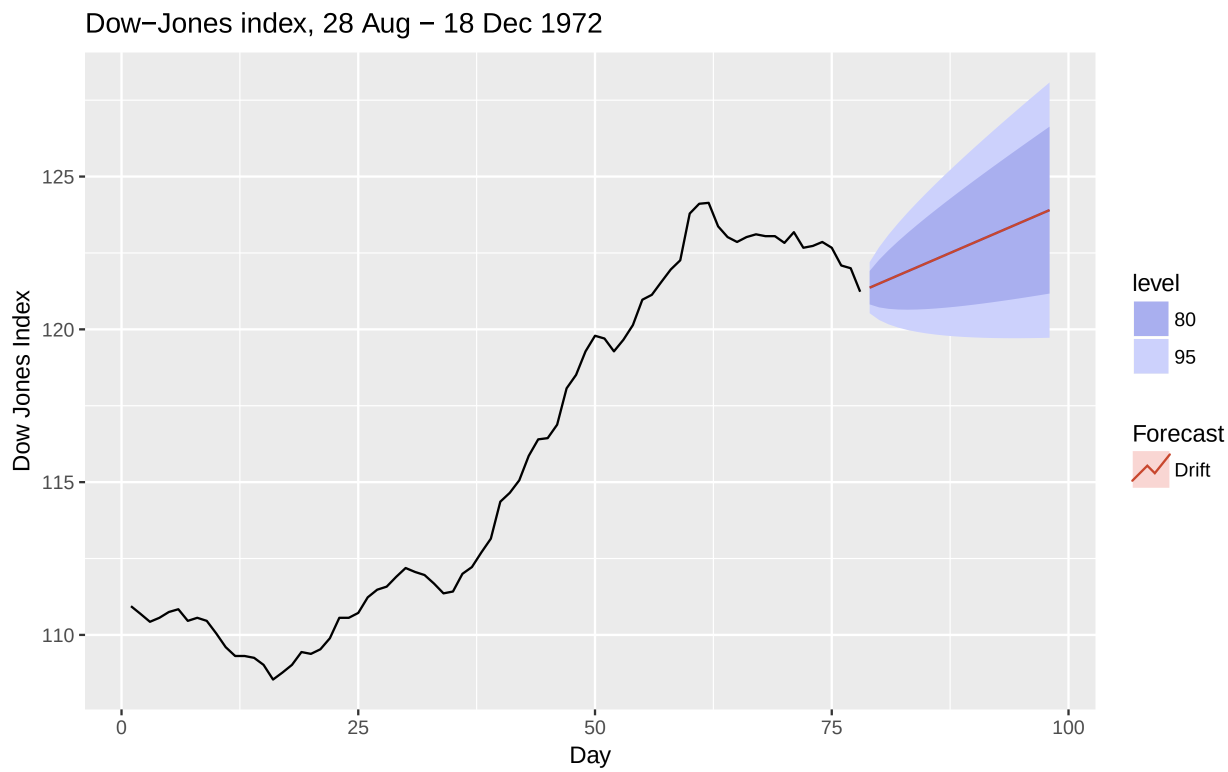
```
data("dowjones")
autoplot(dowjones) + xlab("Day") + ylab("Dow Jones Index") +
  ggtitle("Dow-Jones index, 28 Aug - 18 Dec 1972")
```



Ch2.Q2.b)

The figure below shows the forecast of the Dow Jones Index data using the drift method.

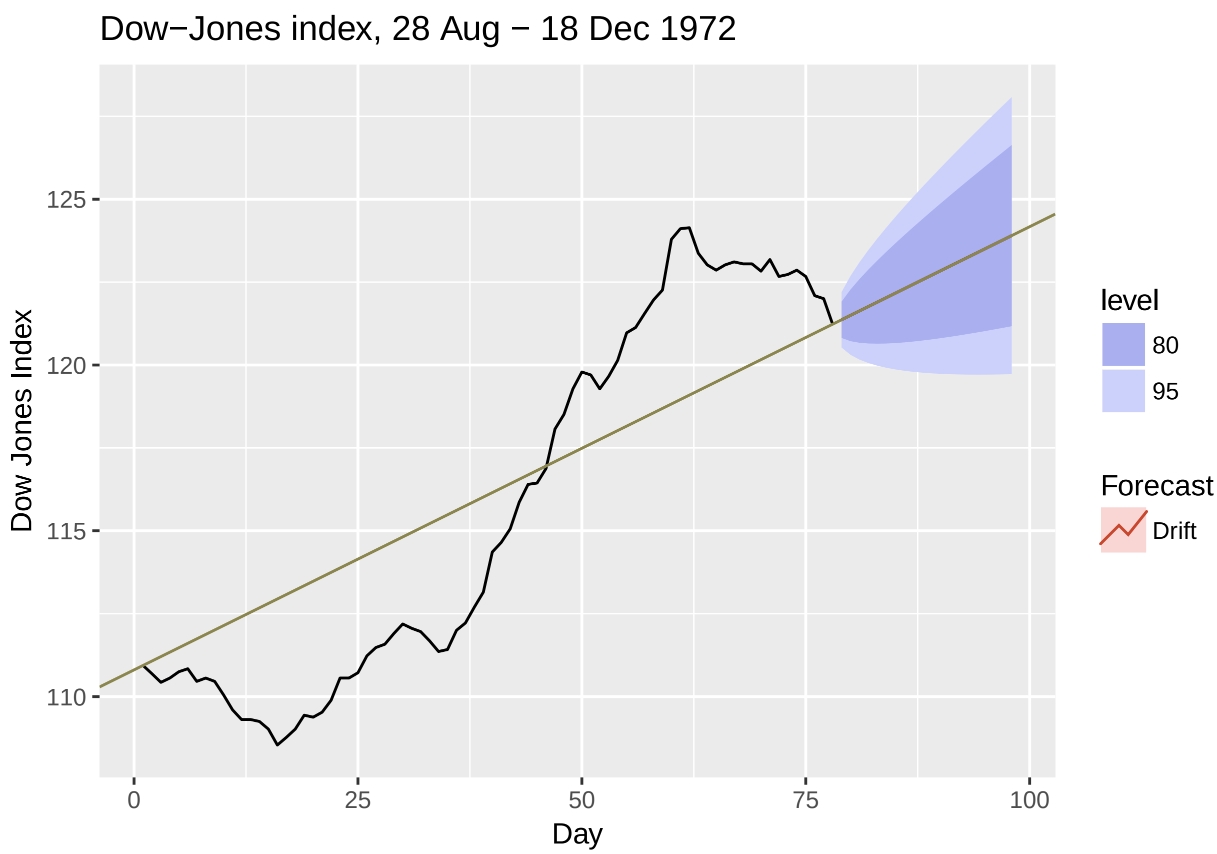
```
autoplot(rwf(dowjones, drift=TRUE, h=20)) +
  forecast::autolayer(rwf(dowjones, drift=TRUE, h=20), PI=FALSE, series="Drift") +
  ggtitle("Dow-Jones index, 28 Aug - 18 Dec 1972") +
  xlab("Day") + ylab("Dow Jones Index") +
  guides(colour=guide_legend(title="Forecast"))
```



Ch2.Q2.c)

The figure below shows that the forecasts are identical to extending the line drawn between the first and last observations.

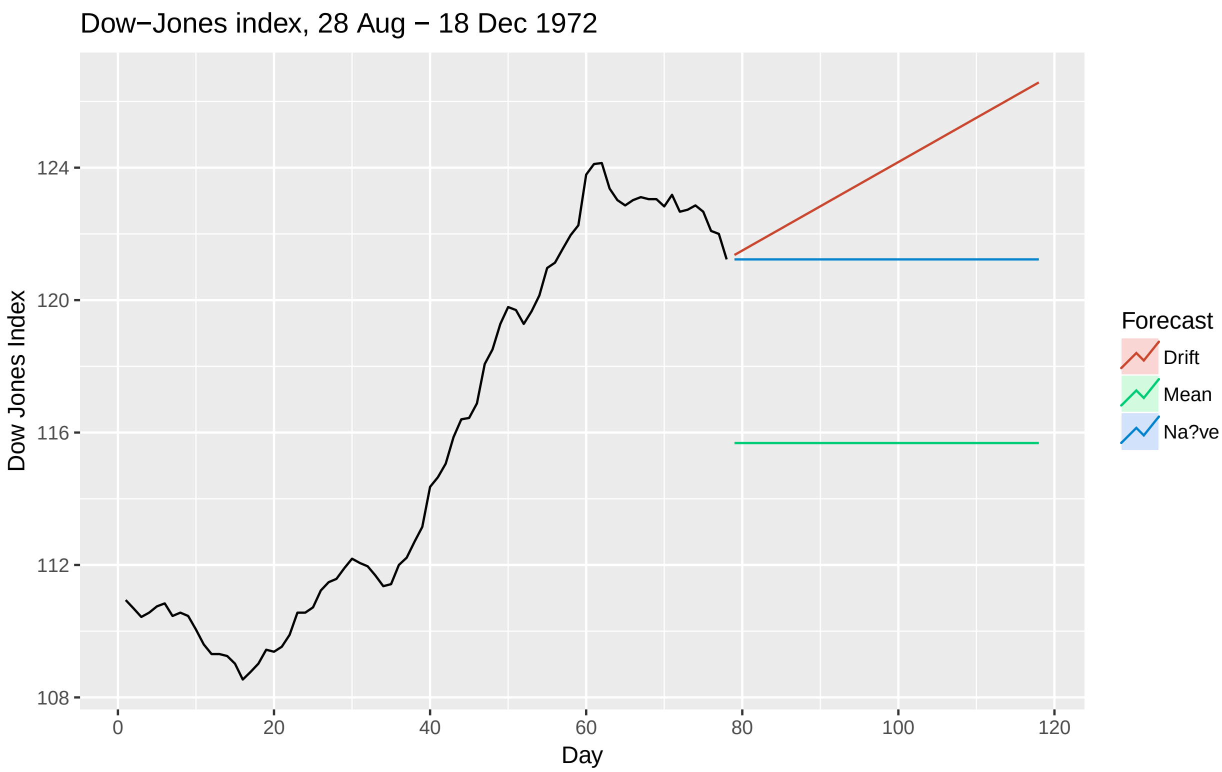
```
Xk <- length(dowjones)
X1 <- 1
Yk  <- dowjones[Xk]
Y1 <- dowjones[1]

slope <- (Yk - Y1)/(Xk - X1)
intercept <- Y1 - slope

autoplot(rwf(dowjones, drift=TRUE, h=20)) +
  forecast::autolayer(rwf(dowjones, drift=TRUE, h=20), PI=FALSE, series="Drift") +
  ggtitle("Dow-Jones index, 28 Aug - 18 Dec 1972") +
  xlab("Day") + ylab("Dow Jones Index") +
  guides(colour=guide_legend(title="Forecast")) +
  geom_abline(slope = slope, intercept = intercept, col = "khaki4")
```



Ch2.Q2.d)

The figure below shows some of the benchmark functions such as Mean, Drift, and Naive methods to forecast the Dow Jones Index data. It appears that drift method is the clear winner here.

```
autoplot(dowjones) +
  forecast::autolayer(meanf(dowjones, h=40), PI=FALSE, series="Mean") +
  forecast::autolayer(rwf(dowjones, h=40), PI=FALSE, series="Na?ve") +
  forecast::autolayer(rwf(dowjones, drift=TRUE, h=40), PI=FALSE, series="Drift") +
  ggtitle("Dow-Jones index, 28 Aug - 18 Dec 1972") +
  xlab("Day") + ylab("Dow Jones Index") +
  guides(colour=guide_legend(title="Forecast"))
```



The table below shows the metrics from each benchmark functions used to forecast the data. Looking at the results, it is quite evident that drift method performed really well and is the best method to select. However, the Naive method did not perform so poorly either. It appears that Naive method prediction is within 95% prediction intervals. Therefore, the metric results are very close to what drift method.

```
meanfcast <- accuracy(meanf(dowjones, h=40))
row.names(meanfcast) <- "Mean"
naivefcast <- accuracy(rwf(dowjones, h=40))
row.names(naivefcast) <- "Na?ve"
driftfcast <- accuracy(rwf(dowjones, drift=TRUE, h=40))
row.names(driftfcast) <- "Drift"

knitr::kable(rbind(meanfcast,naivefcast, driftfcast), caption = "Benchmark Results")
```

Table 1: Benchmark Results

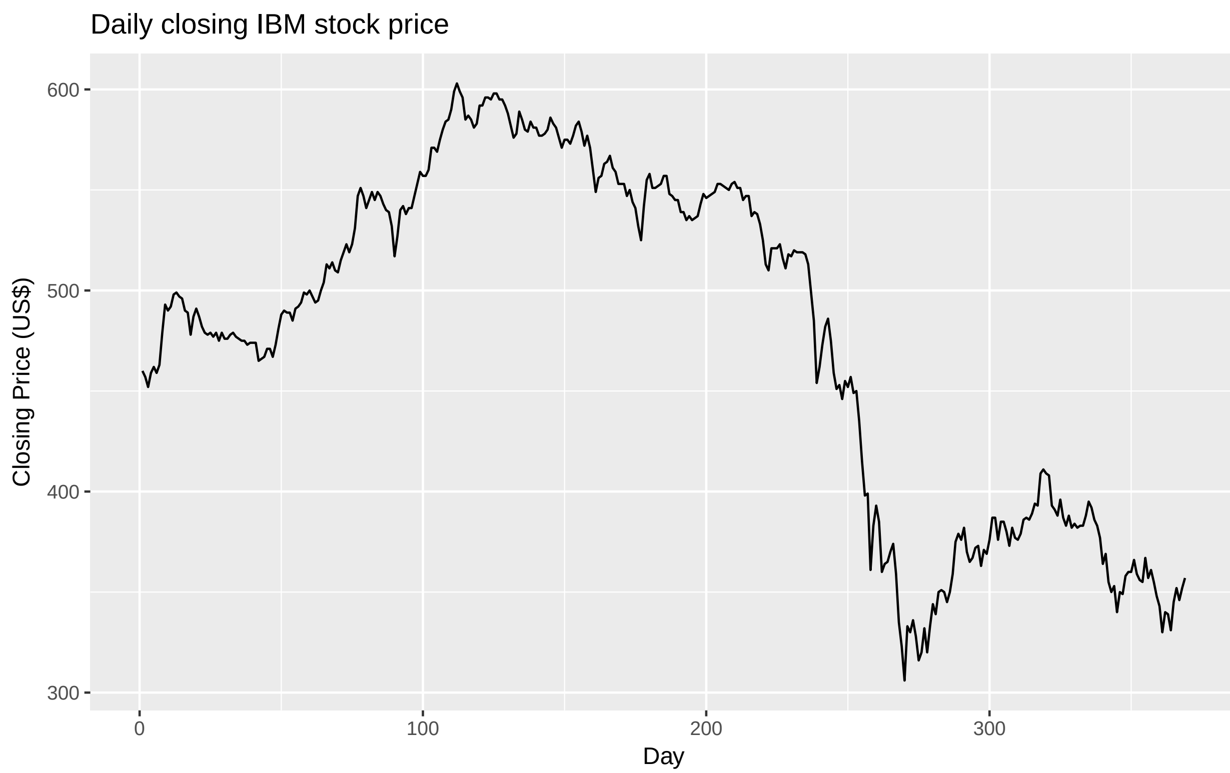
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Mean	0.0000000	5.4706531	5.1041026	-0.2216950	4.3998889	14.937890	0.9853294
Na?ve	0.1336364	0.4447223	0.3416883	0.1144757	0.2936792	1.000000	0.4218786
Drift	0.0000000	0.4241689	0.3253365	-0.0012387	0.2796548	0.952144	0.4218786

Question 3

Ch2.Q3.a)

The figure below shows the time series plot of the daily closing IBM stock prices.

```r
data("ibmclose")
autoplot(ibmclose) + xlab("Day") + ylab("Closing Price (US$)") +
  ggtitle("Daily closing IBM stock price")
```

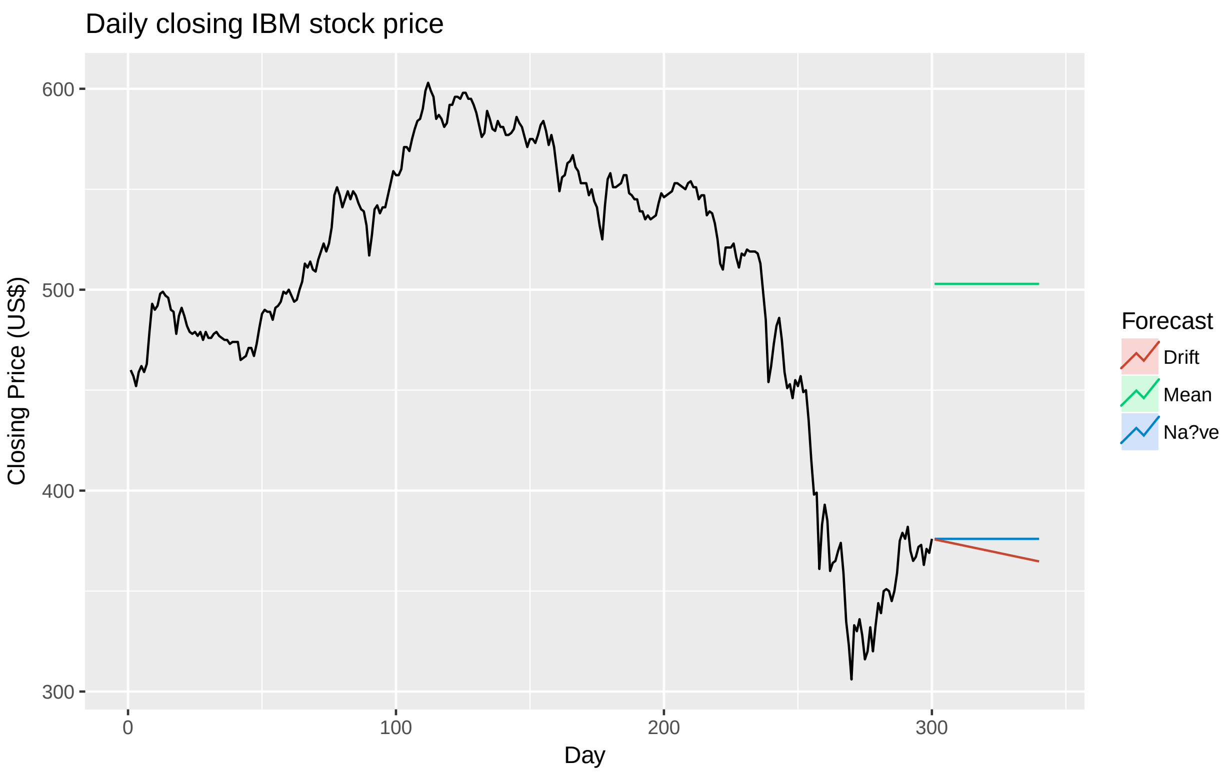


Ch2.Q3.b&c)

The figure below shows various benchmark functions such as Mean, Drift, and Naive methods used to forecast the daily closing IBM stock prices. It appears that naive and drift methods performed well here. They are well within the 80 and 95% prediction interval.

```r
  #str(ibmclose)
train.IBM.ts <- window(ibmclose, end =  300)
test.IBM.ts <-  window(ibmclose, start = 301 , end =  369)
  # str(train.IBM.ts)
  # str(test.IBM.ts)
Fcast.IBM.mean <- meanf(train.IBM.ts, h=40)
Fcast.IBM.Naive <- rwf(train.IBM.ts, h=40)
Fcast.IBM.Drift <- rwf(train.IBM.ts, drift=TRUE, h=40)

autoplot(train.IBM.ts) +
  forecast::autolayer(Fcast.IBM.mean, PI=FALSE, series="Mean") +
```

```
forecast::autolayer(Fcast.IBM.Naive, PI=FALSE, series="Na?ve") +
forecast::autolayer(Fcast.IBM.Drift, PI=FALSE, series="Drift") +
ggtitle("Daily closing IBM stock price") +
xlab("Day") + ylab("Closing Price (US$)") +
guides(colour=guide_legend(title="Forecast"))
```



The tables below list all the metrics from each benchmark functions used to forecast the data. Looking at the results, it appears that naive method is the best option here. MAPE and RMSE for 'Test Set' for naive method beats the other two.

```
acc.Mean <- accuracy(Fcast.IBM.mean, test.IBM.ts)
acc.Naive <- accuracy(Fcast.IBM.Naive, test.IBM.ts)
acc.Drift <-accuracy(Fcast.IBM.Drift, test.IBM.ts)

knitr::kable(acc.Mean[,1:6], caption = "Metrics for Mean Method")
```

Table 2: Metrics for Mean Method

	ME	RMSE	MAE	MPE	MAPE	MASE
Training set	0.0000	73.61532	58.72231	-2.642058	13.03019	11.52098
Test set	-115.7933	116.19611	115.79333	-29.992869	29.99287	22.71798

```
knitr::kable(acc.Naive[,1:6], caption = "Metrics for Naive Method")
```

Table 3: Metrics for Naive Method

	ME	RMSE	MAE	MPE	MAPE	MASE
Training set	-0.2809365	7.302815	5.09699	-0.0826287	1.115844	1.000000
Test set	11.1000000	14.719035	11.85000	2.8077817	3.012831	2.324902

```
knitr::kable(acc.Drift[,1:6], caption = "Metrics for Drift Method")
```

Table 4: Metrics for Drift Method

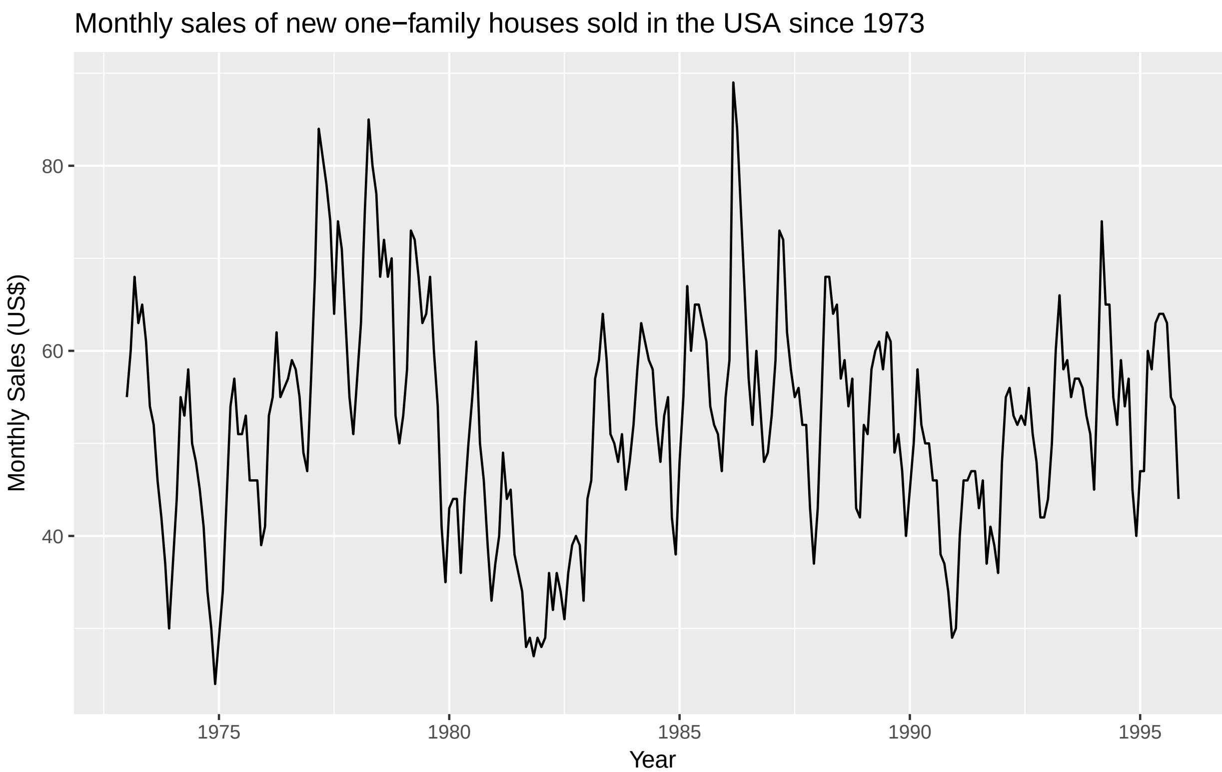
	ME	RMSE	MAE	MPE	MAPE	MASE
Training set	0.0000	7.297409	5.127996	-0.0253012	1.121650	1.006083
Test set	16.8592	19.733866	16.948997	4.2961173	4.320445	3.325295

Question 4

Ch2.Q4.a)

The figure below shows the time series plot of the sales of new one-family houses in the USA from January 1973 to November 1995.

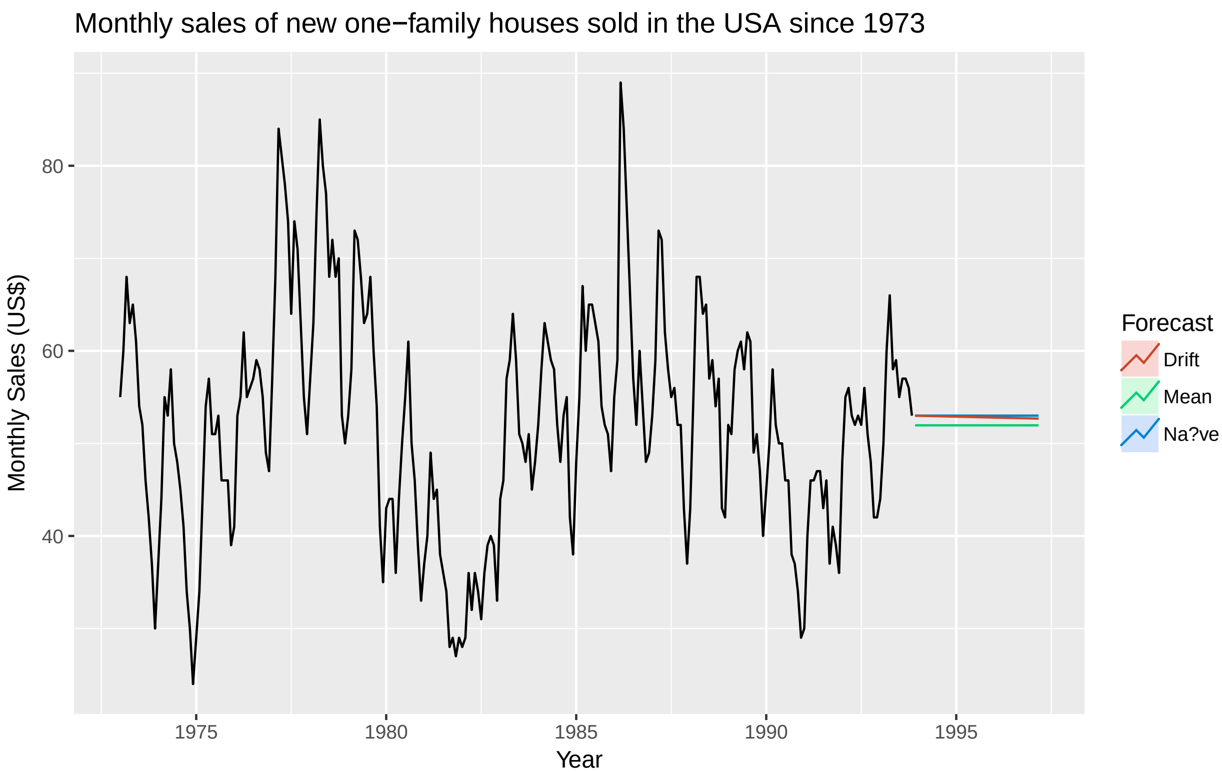
```
data("hsales")
autoplot(hsales) + xlab("Year") + ylab("Monthly Sales (US$)") +
  ggtitle("Monthly sales of new one-family houses sold in the USA since 1973")
```



Ch2.Q4.b&c)

The figure below shows various benchmark functions such as Mean, Drift, and Naive methods used to forecast the sales of new one-family houses in the USA from January 1973 to November 1995. It appears that all three methods forecasted really close. It would ideal to see the metrics result of each method to select the best method.

```
  #str(hsales)
train.HS.ts <- window(hsales, Start = 1973, end = c(1993,11))
test.HS.ts <-  window(hsales, start = c(1993,12))
  # str(train.HS.ts)
  # str(test.HS.ts)
Fcast.HS.mean <- meanf(train.HS.ts, h=40)
Fcast.HS.Naive <- rwf(train.HS.ts, h=40)
Fcast.HS.Drift <- rwf(train.HS.ts, drift=TRUE, h=40)

autoplot(train.HS.ts) +
  forecast::autolayer(Fcast.HS.mean, PI=FALSE, series="Mean") +
  forecast::autolayer(Fcast.HS.Naive, PI=FALSE, series="Na?ve") +
  forecast::autolayer(Fcast.HS.Drift, PI=FALSE, series="Drift") +
  ggtitle("Monthly sales of new one-family houses sold in the USA since 1973") +
  xlab("Year") + ylab("Monthly Sales (US$)") +
  guides(colour=guide_legend(title="Forecast"))
```


Monthly sales of new one−family houses sold in the USA since 1973
Monthly Sales (US$)
80
60
40
1975
1980
1985
1990
1995
Year
Forecast
Drift
Mean
Na?ve

The tables below list all the metrics from each benchmark functions used to forecast the data. Looking at the results, it appears that naive method is the best option here. MAPE and RMSE for 'Test set' for naive method beat the other two.

```
acc.Mean <- accuracy(Fcast.HS.mean, test.HS.ts)
acc.Naive <- accuracy(Fcast.HS.Naive, test.HS.ts)
acc.Drift <-accuracy(Fcast.HS.Drift, test.HS.ts)

knitr::kable(acc.Mean[,1:6], caption = "Metrics for Mean Method")
```

Table 5: Metrics for Mean Method

	ME	RMSE	MAE	MPE	MAPE	MASE
Training set	0.000000	12.162811	9.532738	-6.144876	20.38306	1.1234341
Test set	3.839475	9.022555	7.561587	4.779121	13.26183	0.8911338

```
knitr::kable(acc.Naive[,1:6], caption = "Metrics for Naive Method")
```

Table 6: Metrics for Naive Method

	ME	RMSE	MAE	MPE	MAPE	MASE
Training set	-0.008000	6.301111	5.000000	-0.767457	9.903992	0.5892505
Test set	2.791667	8.628924	7.208333	2.858639	12.849194	0.8495028

```
knitr::kable(acc.Drift[,1:6], caption = "Metrics for Drift Method")
```

Table 7: Metrics for Drift Method

	ME	RMSE	MAE	MPE	MAPE	MASE
Training set	0.000000	6.301106	4.999872	-0.7511048	9.903063	0.5892354
Test set	2.891667	8.658795	7.249000	3.0426108	12.901696	0.8542954

Chapter 4

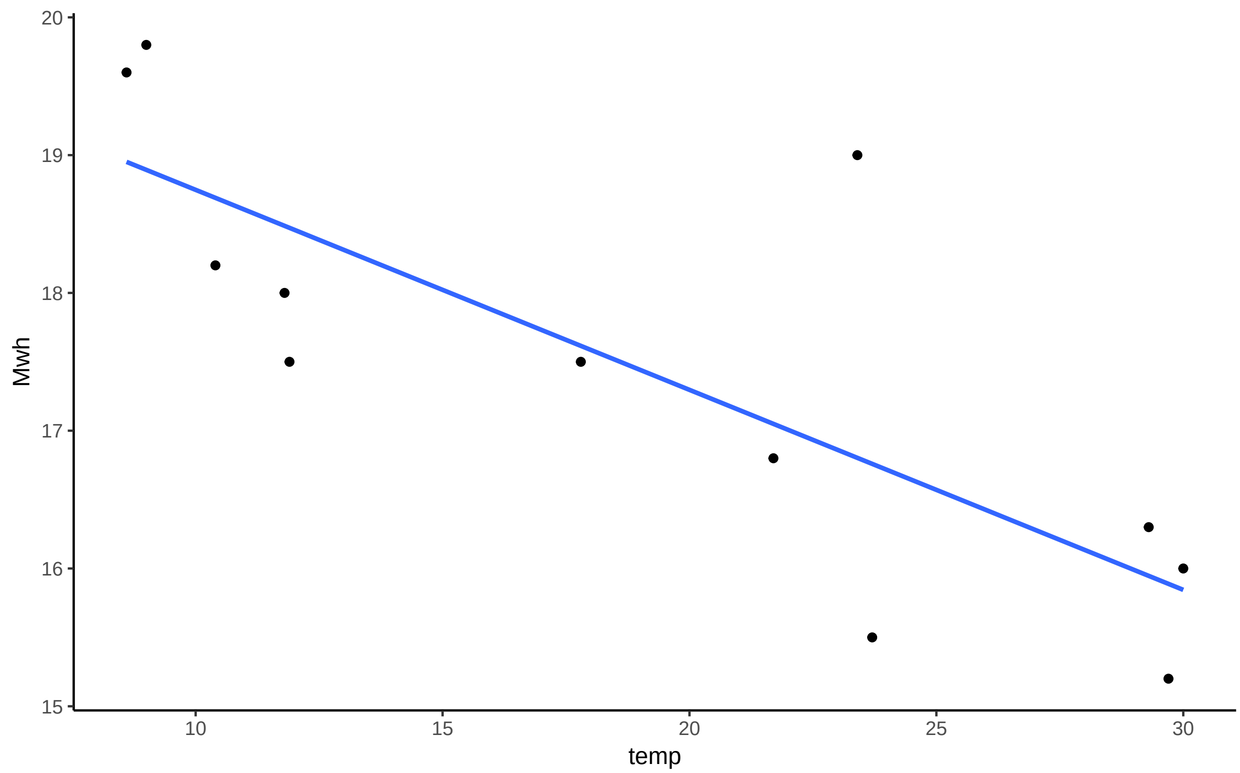
Question 1

Ch4.Q1.a)

There appears to be a negative relationship between temp and Mwh. However, the relationship is not perfect. As the temperature increases, the electricity consumption decreases. The negative relationship exists because of the less use of heater/heat due to the increase in the temperature. Therefore, the electricity consumption decreases.

```
data("econsumption")
ggplot(data = econsumption, mapping = aes(x = temp, y = Mwh)) +
      geom_point() +
      geom_smooth(method = 'lm',se = F) +
```

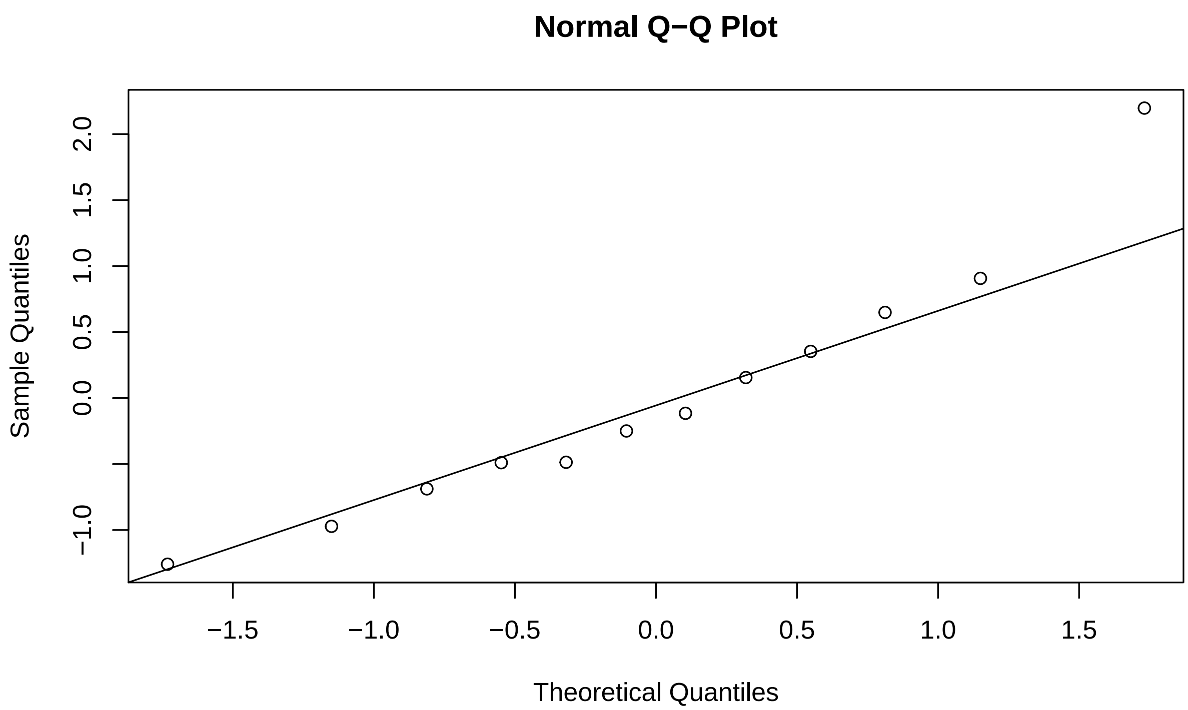
```r
    theme_classic()
```



Ch4.Q1.b)

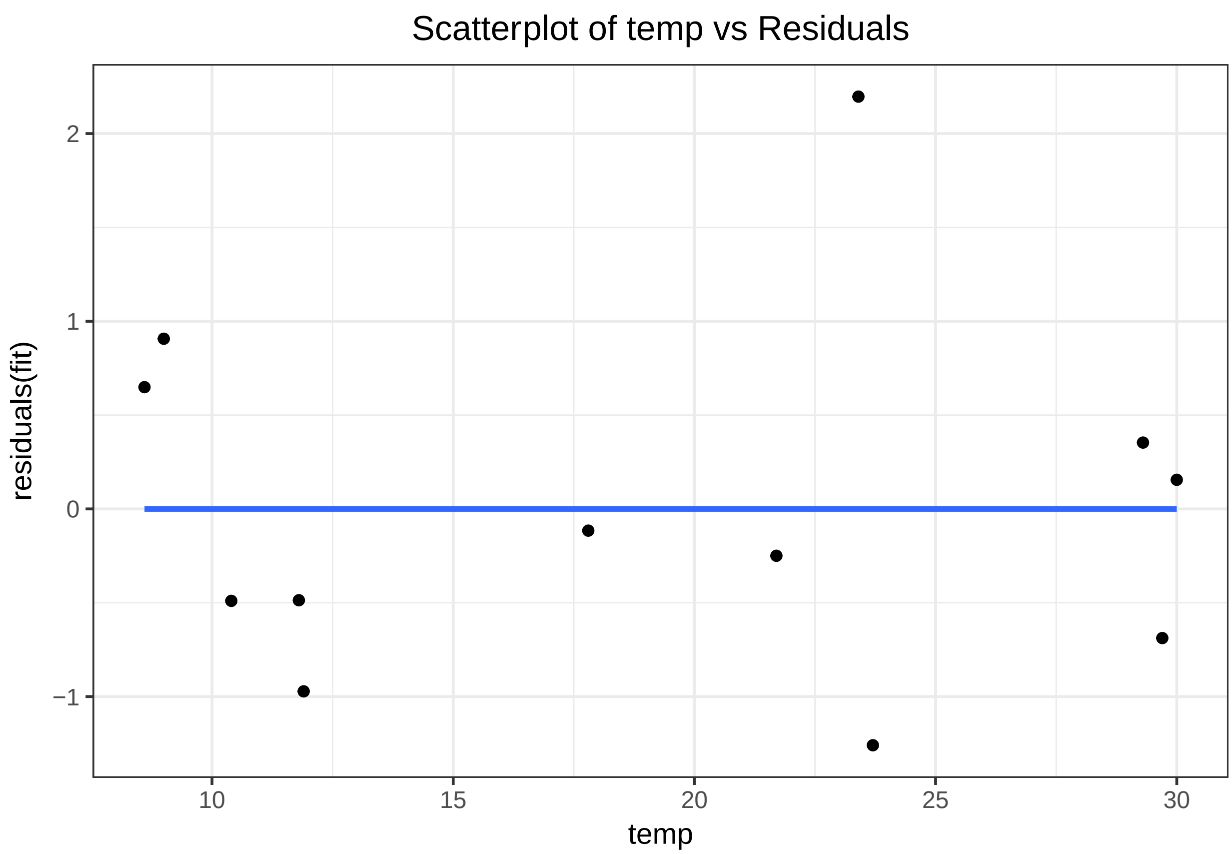
The figure below shows the normal QQ plot. It appears that most of the points fall on the line or close to it. It also shows that there is one outlier present (top right) in the data.

```r
fit <- with(econsumption, lm(Mwh ~ temp))
  #summary(fit)
  #plot(with(econsumption, residuals(fit) ~ temp))
qqnorm(fit$residuals)
qqline(fit$residuals)
```



The figure below shows the residual plots confirming that the errors are random and have constant variance.

```
ggplot(data = econsumption, mapping = aes(x = temp, y =  residuals(fit))) +
      geom_point() +
      geom_smooth(method = 'lm',se = F) +
      ggtitle("Scatterplot of temp vs Residuals") +
      theme_bw() +
      theme(plot.title = element_text(hjust=0.5))
```



Ch4.Q1.c)

The table below shows the forecasted values. The prediction seems to be accurate. For instance, in the source data, when the temperature was 10.4 degrees, the electricity consumption was 18.2 Mwh. For 10 degrees, our model has predicted 18.75 Mwh which to me seems quite close.

```
tempLab <- data.frame(temp = c(10,35))
fcast <- forecast(fit, newdata = tempLab)
attCol <- cbind.data.frame(tempLab, data.frame(fcast))
knitr::kable(attCol, caption = "Electricity Consumption Prediction
                                 when temp = 10 and temp = 35")
```

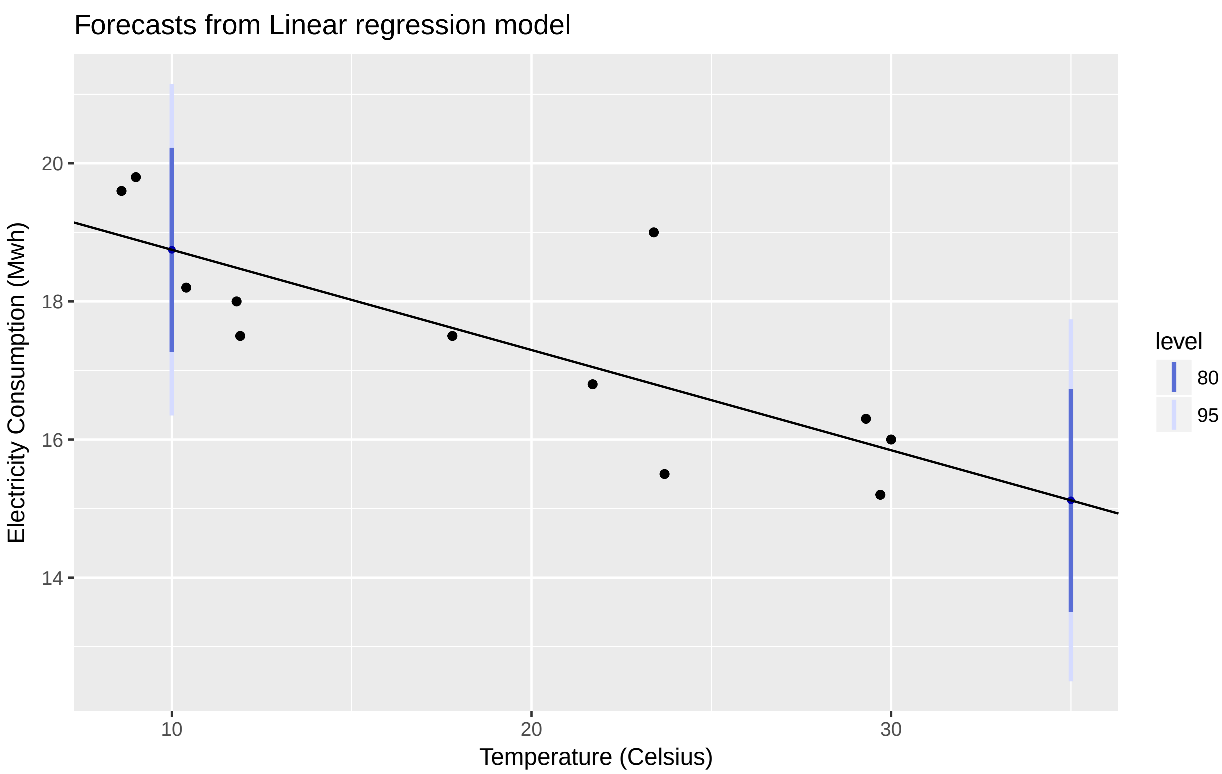
Table 8: Electricity Consumption Prediction when temp = 10 and temp = 35

temp	Point.Forecast	Lo.80	Hi.80	Lo.95	Hi.95
10	18.74795	17.27010	20.22579	16.34824	21.14766
35	15.11902	13.50469	16.73335	12.49768	17.74035

Ch4.Q1.d)

The figure below shows the forecast with 80% and 95% prediction intervals for electricity consumption with temperature = 10 degrees and temperature = 35 degrees.

```r
autoplot(fcast) +
  xlab("Temperature (Celsius)") +
  ylab("Electricity Consumption (Mwh)")
```



Question 2

Ch4.Q2.a)

The Winning times (in seconds) for men's 400 meters final for each of the last few Olympic games (2000 - 2012) has been included in the original dataset. The table below shows the summary statistic of the new dataset.

```r
data("olympic")
lastfewoly <- data.frame(Year = c(2000, 2004, 2008, 2012),
                         time = c(43.84, 44.00, 43.75, 43.94))
olympic <- rbind.data.frame(olympic, lastfewoly)
summ <- cbind(summary(olympic$Year),
               summary(olympic$time))
colnames(summ) <- c("Year","Time")
knitr::kable(summ,
             caption = "Summary Statistic")
```

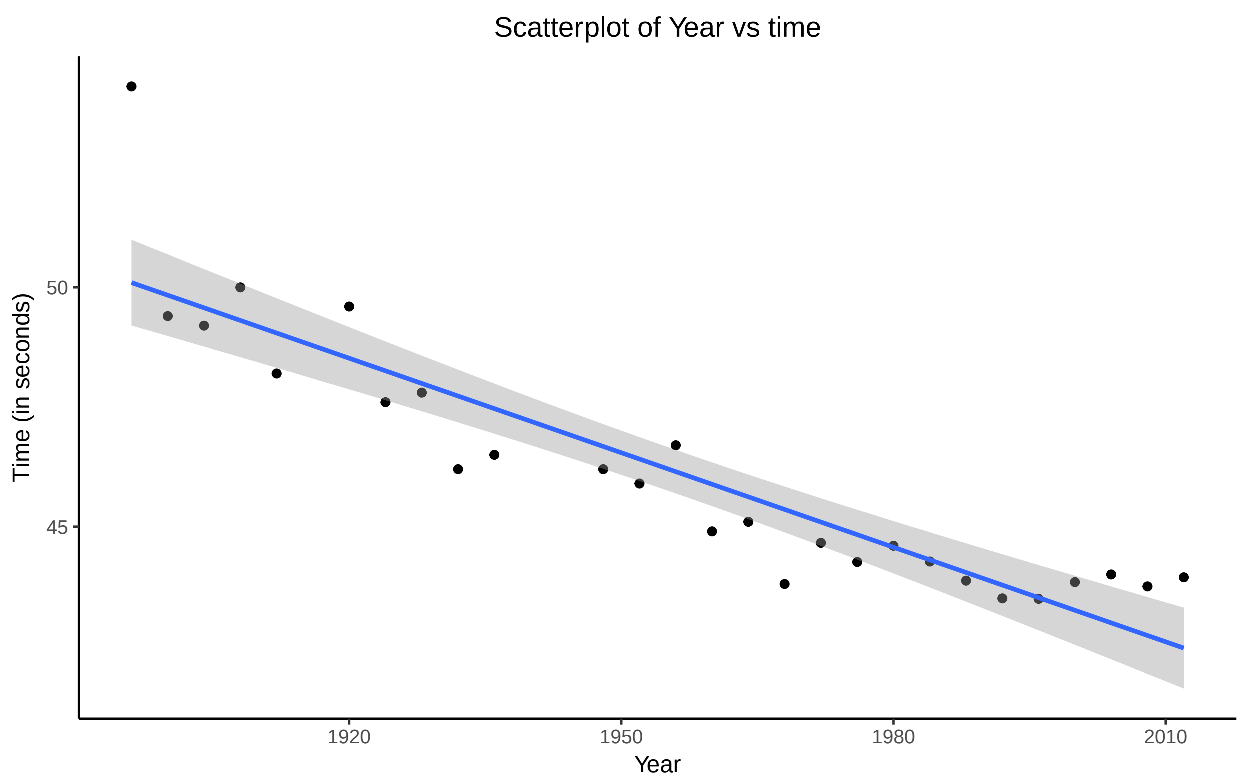
Table 9: Summary Statistic

	Year	Time
Min.	1896	43.49
1st Qu.	1926	43.97
Median	1960	45.10
Mean	1956	46.13
3rd Qu.	1986	47.70
Max.	2012	54.20

Ch4.Q2.b)

The figure below shows the scatterplot of Year vs time with the regression line. The plot shows that the winning times have significantly reduced in the recent years as compared to prior to 1950. There is also an outlier present (top left) in the data.

```
ggplot(data = olympic, mapping = aes(x = Year, y =  time)) +
      geom_point() +
      geom_smooth(method = 'lm',se = T) +
      ggtitle("Scatterplot of Year vs time") +
      ylab("Time (in seconds)") +
      theme_classic() +
      theme(plot.title = element_text(hjust=0.5))
```



Ch4.Q2.c)

The result below shows the summary statistic of our linear regression model. The dataset was split into training and test set. We use the 70/30 split approach. We then used the training set to create our regression model. We see that for every one unit increase in the year, the winning times will decrease on average by 0.088.

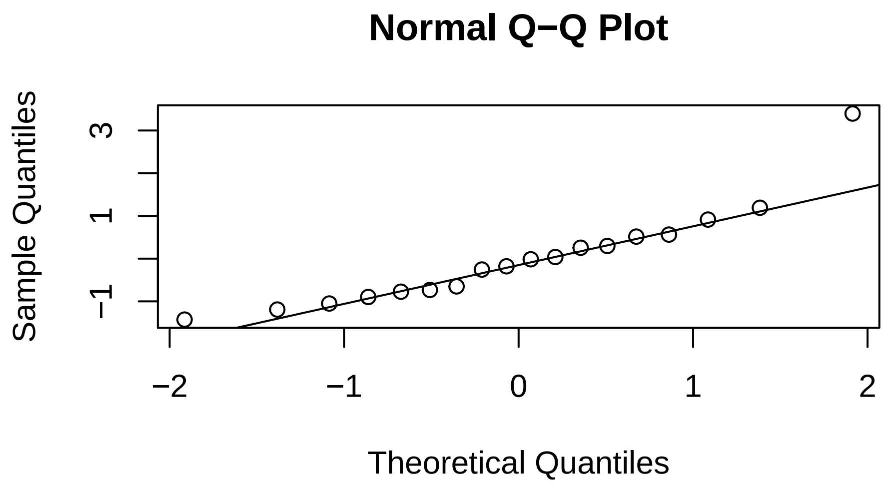
```
train.df <- olympic[1:18,]
test.df  <- olympic[19:27,]
fitReg <- lm(time ~ Year, data = train.df)
summary(fitReg)
```

```
##
## Call:
## lm(formula = time ~ Year, data = train.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4266 -0.7634 -0.0972  0.4614  3.3965
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 218.11738   20.65829  10.558 1.28e-08 ***
## Year         -0.08825    0.01067  -8.273 3.58e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.161 on 16 degrees of freedom
## Multiple R-squared:  0.8105, Adjusted R-squared:  0.7987
## F-statistic: 68.44 on 1 and 16 DF,  p-value: 3.581e-07
```

Ch4.Q2.d)

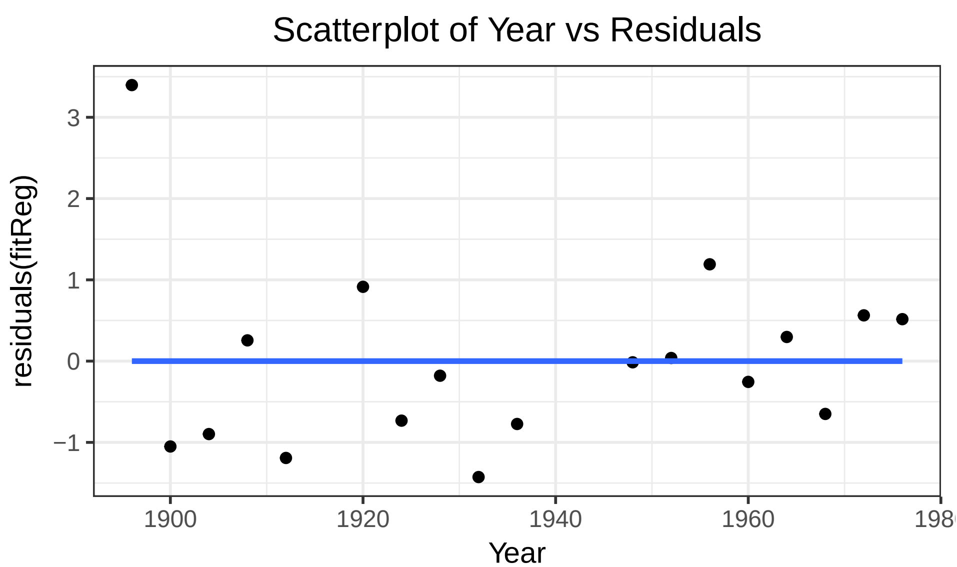
The Normal QQ plot below shows that most of the points fall on the line or close to it. It also shows that there are two outliers (one in the top right and one in the bottom left) in the data.

```
qqnorm(fitReg$residuals)
qqline(fitReg$residuals)
```



The residual plot shows that the errors are random. It does seem to me that there is any pattern to it. Hence, we conclude that residuals show constant variance.

```r
ggplot(data = train.df, mapping = aes(x = Year, y =  residuals(fitReg))) +
    geom_point() +
    geom_smooth(method = 'lm',se = F) +
    ggtitle("Scatterplot of Year vs Residuals") +
    theme_bw() +
    theme(plot.title = element_text(hjust=0.5))
```



Ch4.Q2.e&f)

The figure below shows the forecast with 80% and 95% prediction intervals for winning times with year greater than 1976.

```
fcastReg <- forecast(fitReg, newdata = test.df)
YearLab <- data.frame(Year = test.df$Year)
attCol <- cbind.data.frame(YearLab, Obs_Time = test.df$time, data.frame(fcastReg))
autoplot(fcastReg) +
  xlab("Year") +
  ylab("Winning Times (in seconds)")
```

The table below shows the observed winning time and predicted winning time for four Olympics games held in years 2000, 2004, 2008, and 2014. Looks like our model did not project very well as compared to the reality. However, the prediction is within the 95% interval.

```
knitr::kable(attCol[attCol$Year>1996, c('Year','Obs_Time', "Point.Forecast")],
            caption = "Olympic Games Winning
                      Times Prediction", row.names = F)
```

Table 10: Olympic Games Winning Times Prediction

Year	Obs_Time	Point.Forecast
2000	43.84	41.62594
2004	44.00	41.27296
2008	43.75	40.91997
2012	43.94	40.56699

Question 3

The steps below show the derivation of the log-log model to prove that coefficient B1 is the elasticity coefficient.

Step 1: log y = B0 + B1*log x + e (log-log model)*
*Step 2: ddy[log y] = ddy[B0] + ddy[B1 * log x] + ddy [e] (taking derivatives on each side)*
Step 3: 1/y = 0 + B1 ddy[log x] + 0 (ddy values ddy[log y] = 1/y; ddy[log x] = 1/x; ddy[B0] = 0 = ddy[e])
Step 4: 1/y = B1 * 1/x
Step 5: x = B1y
Step 6: x/y = B1

Chapter 6

Question 1

The following transformation shows that 3 X 5 MA is equivalent to a 7-term weighted moving average with weights of 0.067, 0.133, 0.200, 0.200, 0.200, 0.133, and 0.067

= 1/3[1/5(Yt-3 + Yt-2 + Yt-1 + Yt + Yt+1) + 1/5(Yt-2 + Yt-1 + Yt + Yt+1 + Yt+2) + 1/5(Yt-1 + Yt + Yt + 1 + Yt+2 + Yt+3)]
= 1/3[1/5{ Yt-3 + Yt-2 + Yt-1 + Yt + Yt+1 + Yt-2 + Yt-1 + Yt + Yt+1 + Yt+2 + Yt-1 + Yt + Yt + 1 + Yt+2 + Yt+3}]
= 1/15[Yt - 3 + 2(Yt-2) + 3(Yt-1) + 3(Yt) +3(Yt + 1) + 2(Yt + 2) + Yt+3]
= 1/15(Yt - 3) + 2/15(Yt - 2) + 1/5(Yt-1) + 1/5(Yt) + 1/5(Yt+1) + 2/15(Yt + 2) + 1/15(Yt+3)
= 0.067(Yt - 3) + 0.133(Yt - 2) + 0.2(Yt-1) + 0.2 (Yt) + 0.2 (Yt+1) + 0.133 (Yt + 2) + 0.067 (Yt+3)

Question 2

Ch6.Q2.a)

Yes, we identify the seasonal fluctuations and the trend in the plot below. Every year there's a slight decrease in the sale, in the beginning, sale picks up in the mid of the year, and then starts to decrease again. Also, the sale has been increasing year over year. Hence, there's also an increasing trend in the sale of product A.

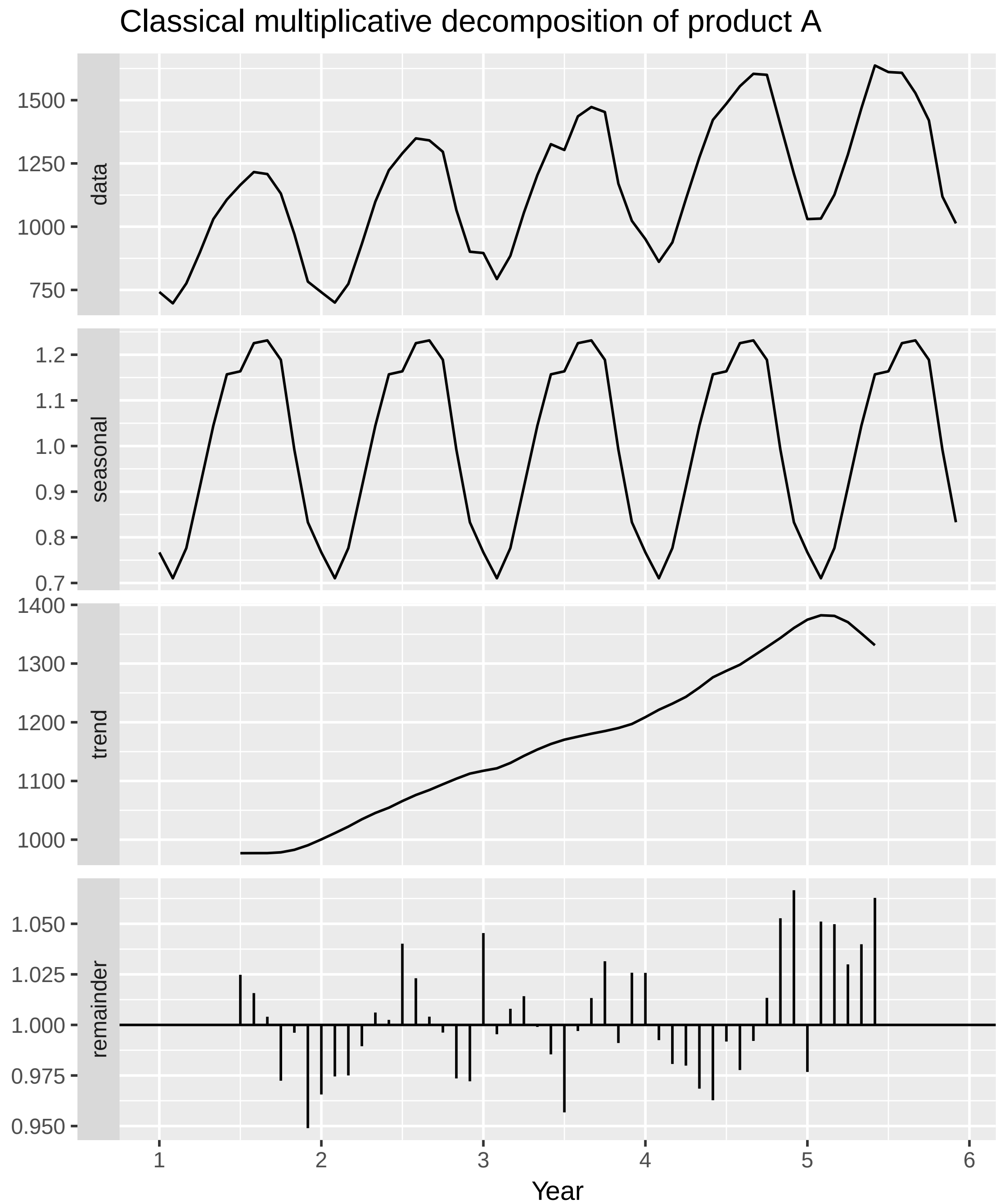
```
data("plastics")
autoplot(plastics) +
      ylab("Sale Amount (in thousands)") +
      xlab("Year") +
      ggtitle("Monthly sales of product A
              for a plastics manufacturer")
```

Ch6.Q2.b&c)

The figure below shows trend-cycle and seasonal indices of the classical multiplicative decomposition.

Yes, the results in the figure below support the graphical interpretation of part(a). With this decomposition, we confirm that the seasonal fluctuations and the increasing trend are present in the data.

```
plastics %>%
      decompose(type="multiplicative") %>%
      autoplot() + xlab("Year") +
      ggtitle("Classical multiplicative decomposition of product A")
```

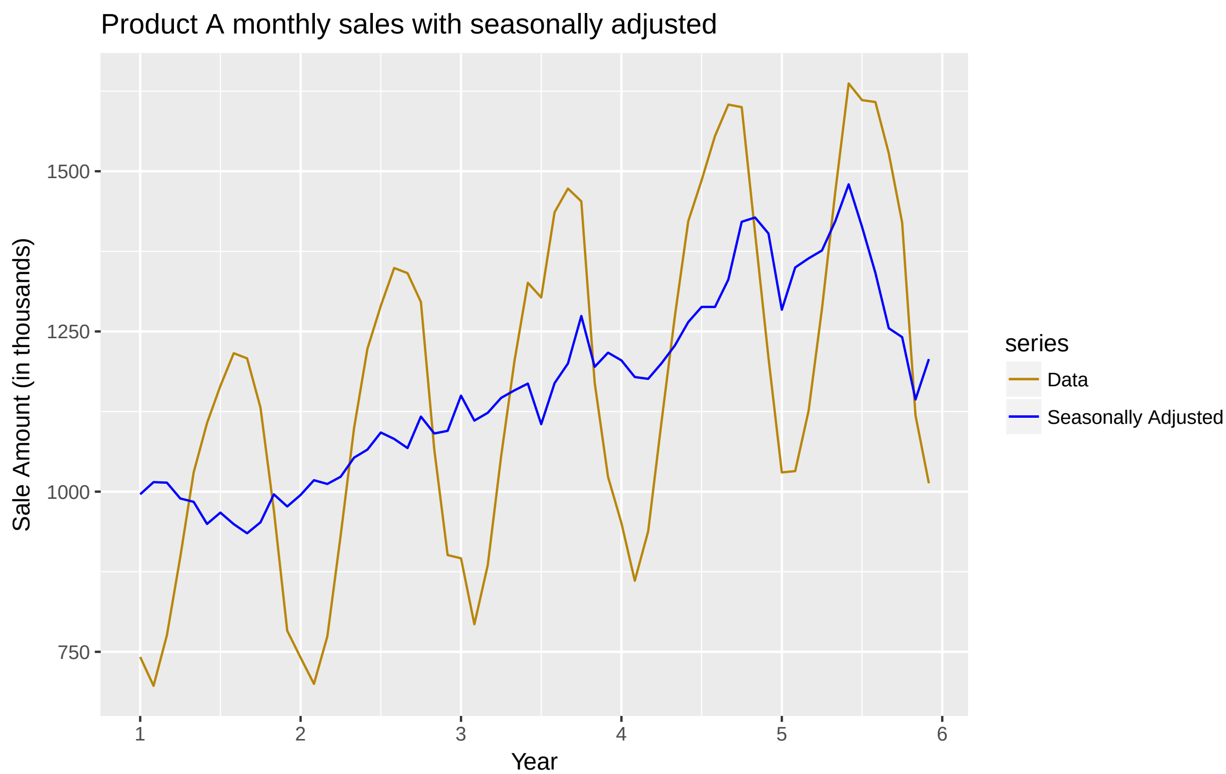


Ch6.Q2.d)

The figure below shows the plot of the original data and seasonally adjusted data.

```
fit <- stl(plastics, s.window="periodic", robust = T)
autoplot(plastics, series = "Data") +
    forecast::autolayer(seasadj(fit), series="Seasonally Adjusted") +
    xlab("Year") + ylab("Sale Amount (in thousands)") +
    ggtitle("Product A monthly sales with seasonally adjusted") +
```

```r
    scale_colour_manual(values=c("darkgoldenrod","blue"),
                     breaks=c("Data","Seasonally Adjusted"))
```



Ch6.Q2.e)

In the original data, we replaced the amount to 2,023 for the month of December and year 3. The tables below show the changes in the seasonally adjusted data. The first table shows the result from the original data set and second table shows the result from the data set with an outlier.

```r
#Seasonally adjusted data from original data set
seasFit <- seasadj(fit)
seasFit
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 1   995.8392 1014.8402 1013.9941  989.3123  984.0645  949.6119  967.2812
## 2   994.8392 1017.8402 1011.9941 1023.3123 1053.0645 1065.6119 1092.2812
## 3  1149.8392 1110.8402 1122.9941 1146.3123 1158.0645 1168.6119 1105.2812
## 4  1204.8392 1178.8402 1175.9941 1200.3123 1228.0645 1264.6119 1288.2812
## 5  1283.8392 1349.8402 1363.9941 1376.3123 1422.0645 1479.6119 1413.2812
##            Aug       Sep       Oct       Nov       Dec
## 1   949.2518  935.0209  952.1319  995.7861  976.8658
## 2  1082.2518 1068.0209 1117.1319 1090.7861 1094.8658
## 3  1169.2518 1200.0209 1274.1319 1194.7861 1216.8658
## 4  1288.2518 1331.0209 1421.1319 1427.7861 1402.8658
## 5  1341.2518 1255.0209 1241.1319 1143.7861 1206.8658
```

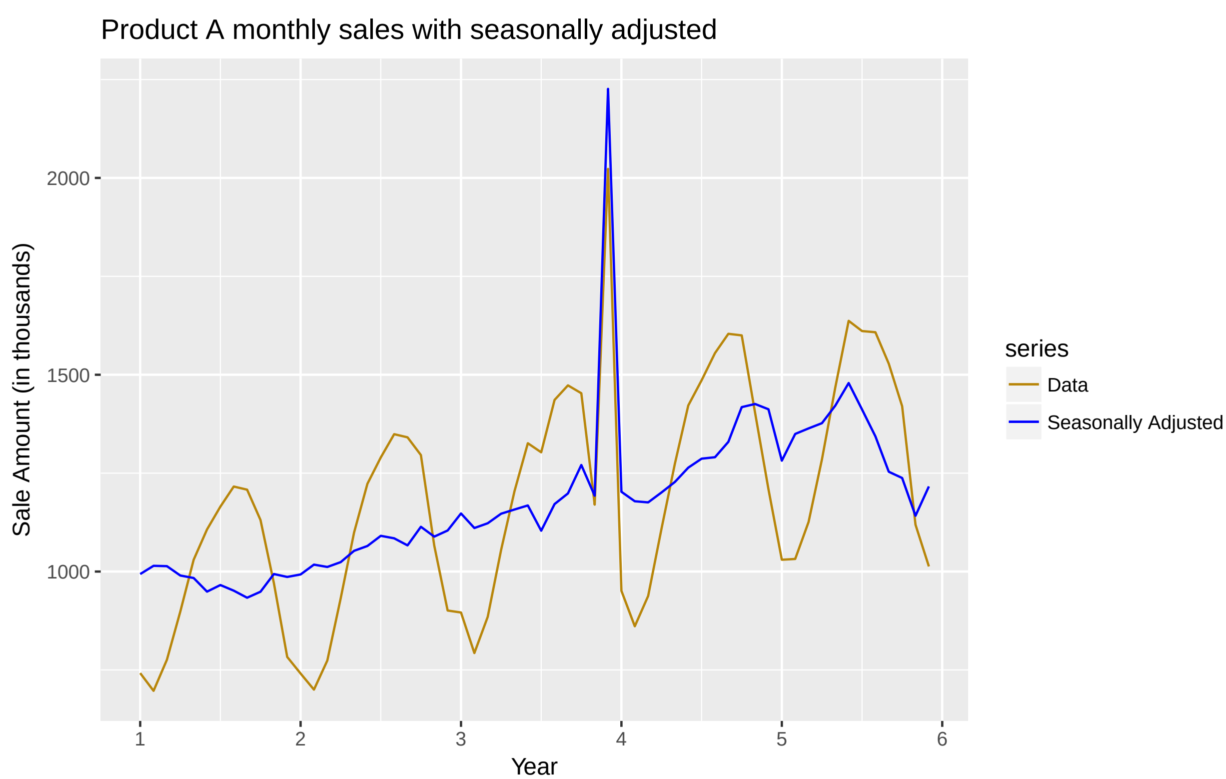
```r
plastics_outlier <- as.xts(plastics)
plastics_outlier["000312"] <- 2023  #added 1000 to the original value
```

```r
fit_outier <- stl(as.ts(plastics_outlier), s.window="periodic", robust = T)
#Seasonally adjusted data from outlier data set
seasFit.outlier <- seasadj(fit_outier)
seasFit.outlier
```

```
##         Jan       Feb       Mar       Apr       May       Jun       Jul
## 1  993.6494 1014.5969 1013.6897  989.9799  983.6071  948.9579  965.7852
## 2  992.6494 1017.5969 1011.6897 1023.9799 1052.6071 1064.9579 1090.7852
## 3 1147.6494 1110.5969 1122.6897 1146.9799 1157.6071 1167.9579 1103.7852
## 4 1202.6494 1178.5969 1175.6897 1200.9799 1227.6071 1263.9579 1286.7852
## 5 1281.6494 1349.5969 1363.6897 1376.9799 1421.6071 1478.9579 1411.7852
##         Aug       Sep       Oct       Nov       Dec
## 1  951.4535  933.5234  948.7386  993.7208  986.2975
## 2 1084.4535 1066.5234 1113.7386 1088.7208 1104.2975
## 3 1171.4535 1198.5234 1270.7386 1192.7208 2226.2975
## 4 1290.4535 1329.5234 1417.7386 1425.7208 1412.2975
## 5 1343.4535 1253.5234 1237.7386 1141.7208 1216.2975
```

The figure below shows the changes due to the outlier. We see the peak in the data and seasonally adjusted plots where the outlier exists. However, I cannot determine if it there's any other effect on the graph due to the outlier.

```r
autoplot(as.ts(plastics_outlier), series = "Data") +
    forecast::autolayer(seasFit.outlier, series="Seasonally Adjusted") +
    xlab("Year") + ylab("Sale Amount (in thousands)") +
    ggtitle("Product A monthly sales with seasonally adjusted") +
    scale_colour_manual(values=c("darkgoldenrod","blue"),
                     breaks=c("Data","Seasonally Adjusted"))
```



Ch6.Q2.f)

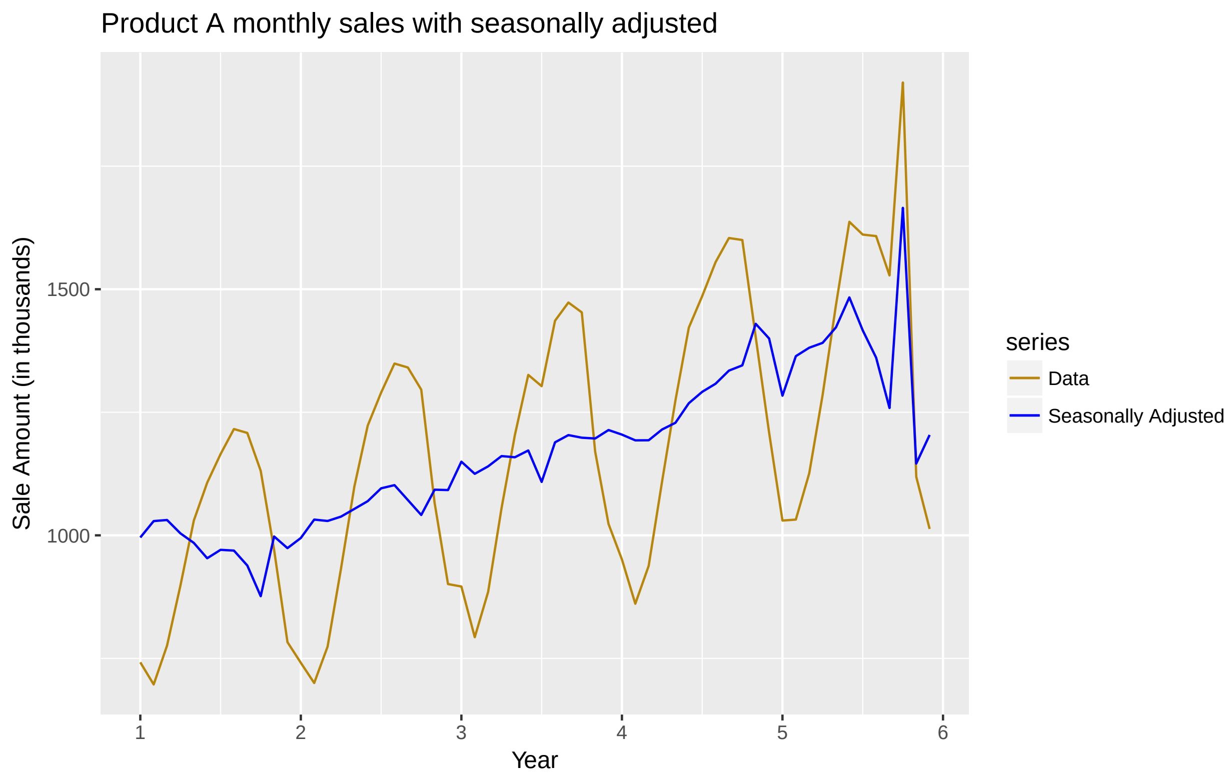
Based on the values and the plot, it appears that there's no difference if the outlier is near the end rather than in the middle of the time series.

```
plastics_outlier <- as.xts(plastics)
plastics_outlier["000510"] <- 1920  #added 500 to the original value
fit_outier_End <- stl(as.ts(plastics_outlier), s.window="periodic", robust = T)
seasFit.outlier.End <- seasadj(fit_outier_End)
seasFit.outlier.End
```

```
##         Jan       Feb       Mar       Apr       May       Jun       Jul
## 1  995.6289 1028.9663 1031.1597 1004.0011  984.6320  953.4195  970.5085
## 2  994.6289 1031.9663 1029.1597 1038.0011 1053.6320 1069.4195 1095.5085
## 3 1149.6289 1124.9663 1140.1597 1161.0011 1158.6320 1172.4195 1108.5085
## 4 1204.6289 1192.9663 1193.1597 1215.0011 1228.6320 1268.4195 1291.5085
## 5 1283.6289 1363.9663 1381.1597 1391.0011 1422.6320 1483.4195 1416.5085
##         Aug       Sep       Oct       Nov       Dec
## 1  968.9721  938.6788  876.3292  997.7260  973.9780
## 2 1101.9721 1071.6788 1041.3292 1092.7260 1091.9780
## 3 1188.9721 1203.6788 1198.3292 1196.7260 1213.9780
## 4 1307.9721 1334.6788 1345.3292 1429.7260 1399.9780
## 5 1360.9721 1258.6788 1665.3292 1145.7260 1203.9780
```

Again, I cannot determine any difference in the graph below. Not sure, if it there's any effect on the graph due to the outlier apart from the peak at the outlier point.

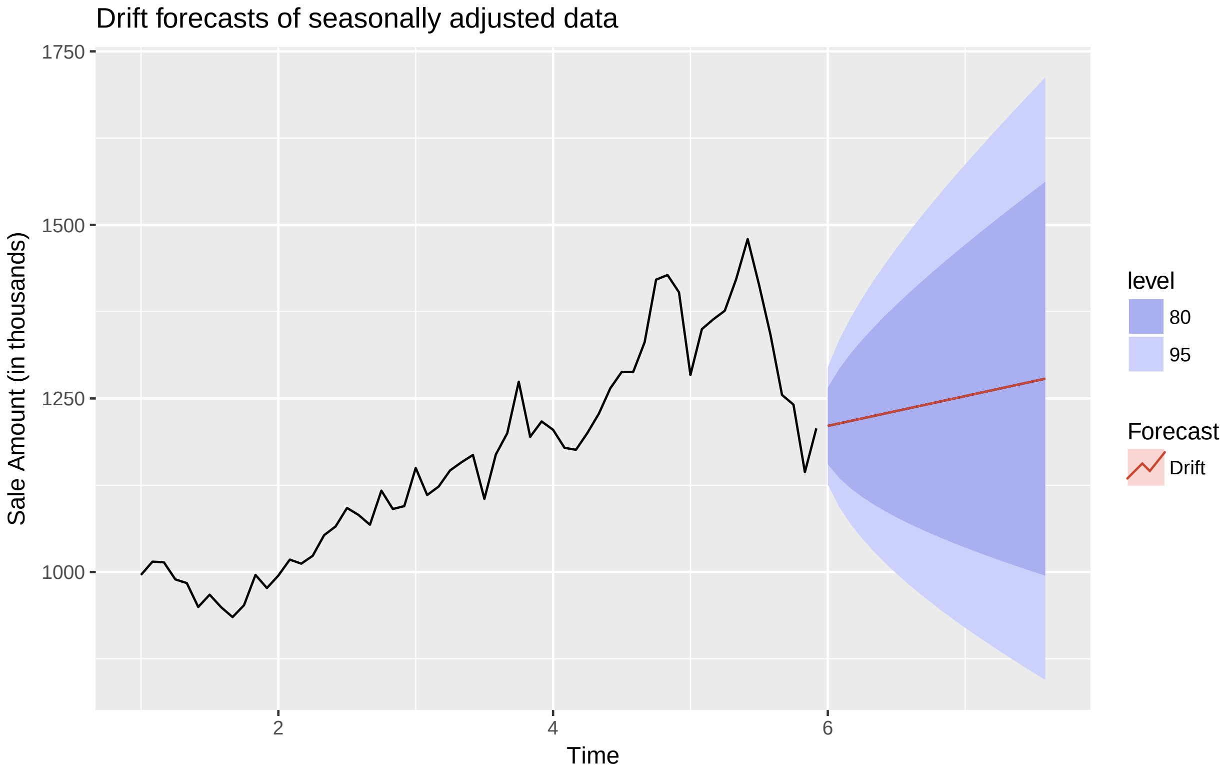
```
autoplot(as.ts(plastics_outlier), series = "Data") +
    forecast::autolayer(seasFit.outlier.End, series="Seasonally Adjusted") +
    xlab("Year") + ylab("Sale Amount (in thousands)") +
    ggtitle("Product A monthly sales with seasonally adjusted") +
    scale_colour_manual(values=c("darkgoldenrod","blue"),
                        breaks=c("Data","Seasonally Adjusted"))
```



Ch6.Q2.g)

The figure below shows the forecast of the seasonally adjusted data using the random walk with drift.

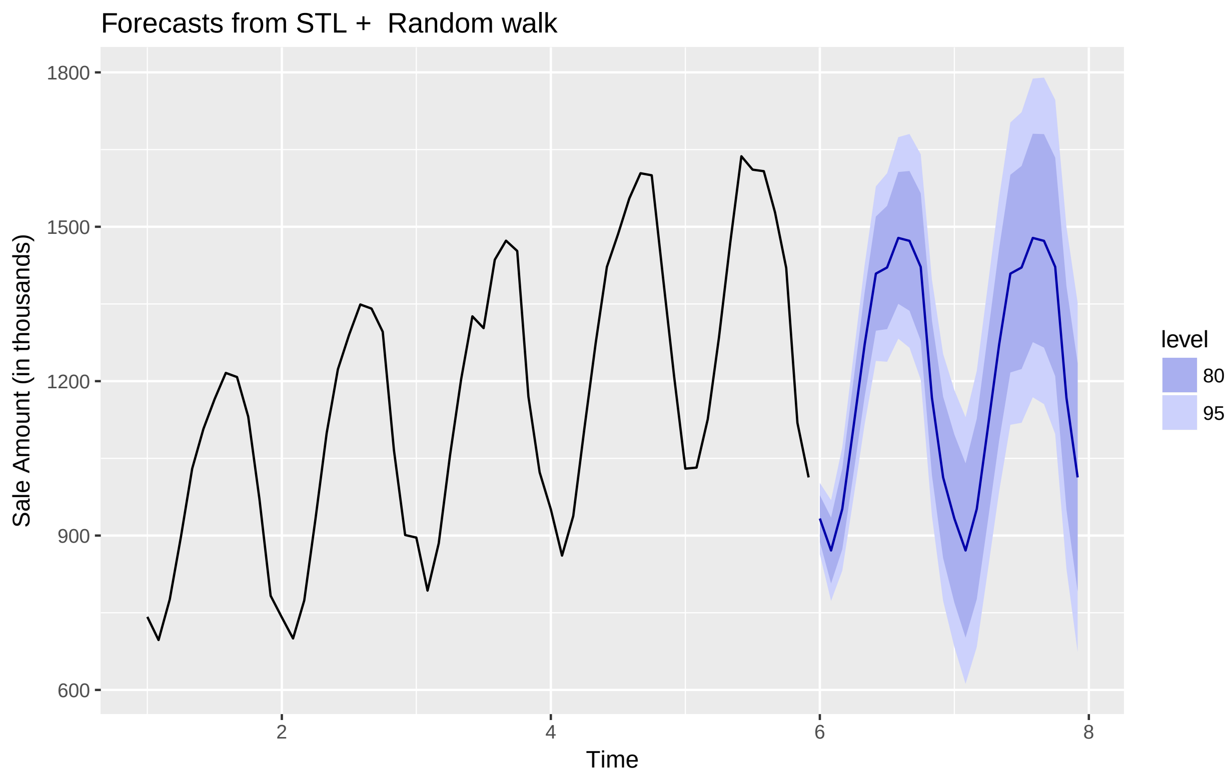
```
autoplot(rwf(seasadj(fit), drift = TRUE, h=20)) +
  forecast::autolayer(rwf(seasadj(fit), drift = TRUE, h=20), PI=FALSE, series="Drift") +
      ylab("Sale Amount (in thousands)") +
      ggtitle("Drift forecasts of seasonally adjusted data") +
      guides(colour=guide_legend(title="Forecast"))
```



Ch6.Q2.h)

The figure below shows the forecast of the product A sales data based on the naive forecast of the seasonally adjusted data and a seasonal naive forecast of the seasonal component, after an STL decomposition of the data. The forecast is between 80% and 95% prediction intervals.

```
stlf(plastics, method = "naive") %>%
          autoplot() + ylab("Sale Amount (in thousands)")
```



Question 3

Ch6.Q3.a)

In Figure 6.13, the 'data' panel is the actual observed data. The three components ('seasonal', 'trend', and 'remainder') in the bottom panels can be added together to reconstruct the data shown in the 'data' panel. In the figure, we notice that the seasonal component changes very slowly over time. The trend-cycle is linear and increasing. the remainder component is shown in the bottom panel is what is left over when the seasonal and trend-cycle components have been subtracted from the data.

The large grey bar in the second panel shows that the variation in the seasonal component is small as compared to the variation in the data.

In Figure 6.13, the bottom graph shows the seasonal sub-series plot of the seasonal component. This kind of graph helps us to visualize the variation in the seasonal component over time. In this case, we notice there are only very small changes over time.

Ch6.Q3.b)

Yes, the recession of 1991/1992 is visible in the estimated components. This can be observed in the fourth panel i.e. remainder component. The is a negative peak in the remainder component for the year 1991 and 1992.