HOMEWORK 1

Data 624

Group 1
Angrand, Burke, Deboch, Groysman, Karr

Table of Contents

Data 624: Week 1 Homework

Angrand, Burke, Deboch, Groysman, Karr

October 15, 2019

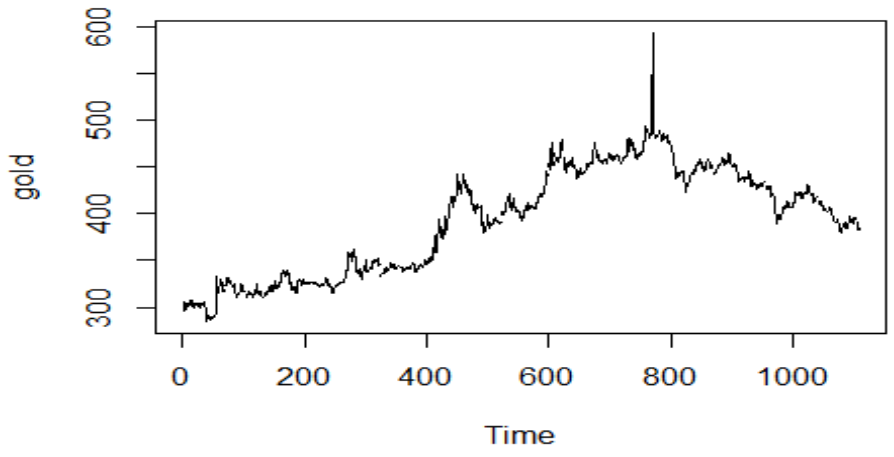
Week 1 Assignment

HW - Chapter 2 HA 2.1, 2.3

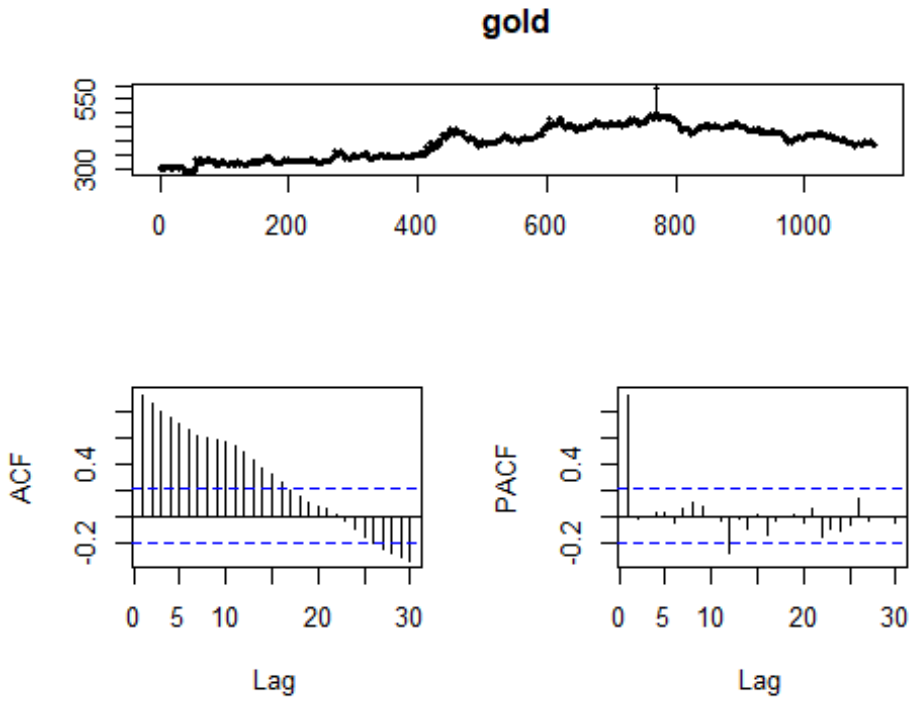
*2.1 Use the help function to explore what the series **gold**, **woolyrnq** and **gas** represent.*

Evaluation of *gold*:

```
help(gold)

## starting httpd help server ... done

#describe(gold)
head(gold)

## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1] 306.25 299.50 303.45 296.75 304.40 298.35

plot(gold)
```



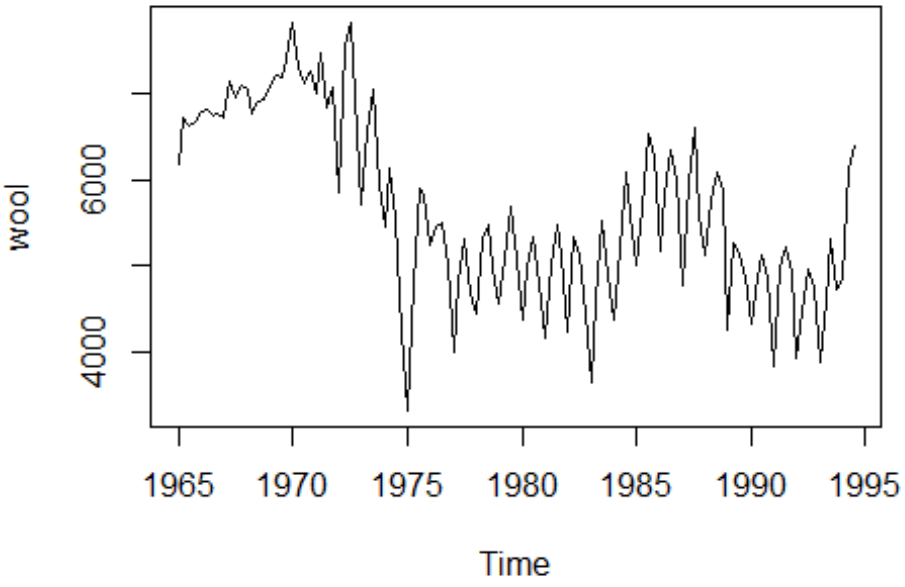

```
tsdisplay(gold)
```



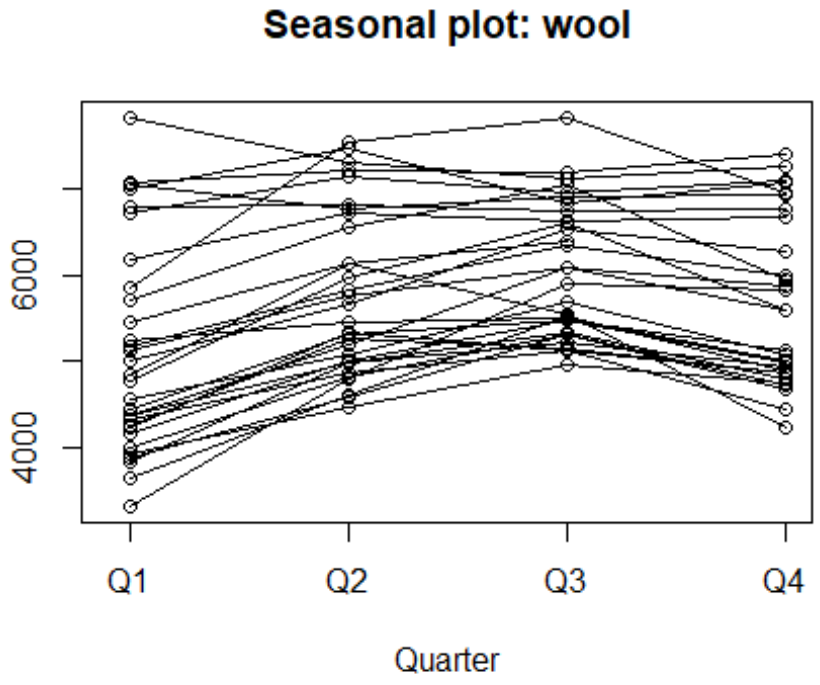
- The `gold` dataset is structured as a timeseries object. The description notes that the dimensions being compared are time in days 1 January 1985 - 31 March 1989 against price in gold of US dollar. Based on the plot, the price of gold steadily increases until ~800 days pass, there is a notable 30% spike and then a steady dropoff in price for the remaining 250 days.
- The time dimension of this dataset being daily, isn't setup to identify seasonality or year-over-year trends, so this type of evaluation has not been done. It is possible to transform the dataset with different granuality of the time dimension in order to reveal such trends.
- The lag scales simply show a trend of decreasing price at a somewhat constant rate crossing into decreasing price at a constant rate. The crossover occurs significantly at the point of the spike in price. Most of the PACF lag is within the range for being attributed to white noi

Evaluation of *wool*

```
wool <- ts(woolyrnq, start=1965, frequency=4)
#describe(wool)
head(wool)

##      Qtr1 Qtr2 Qtr3 Qtr4
## 1965 6172 6709 6633 6660
## 1966 6786 6800

plot(wool)
```

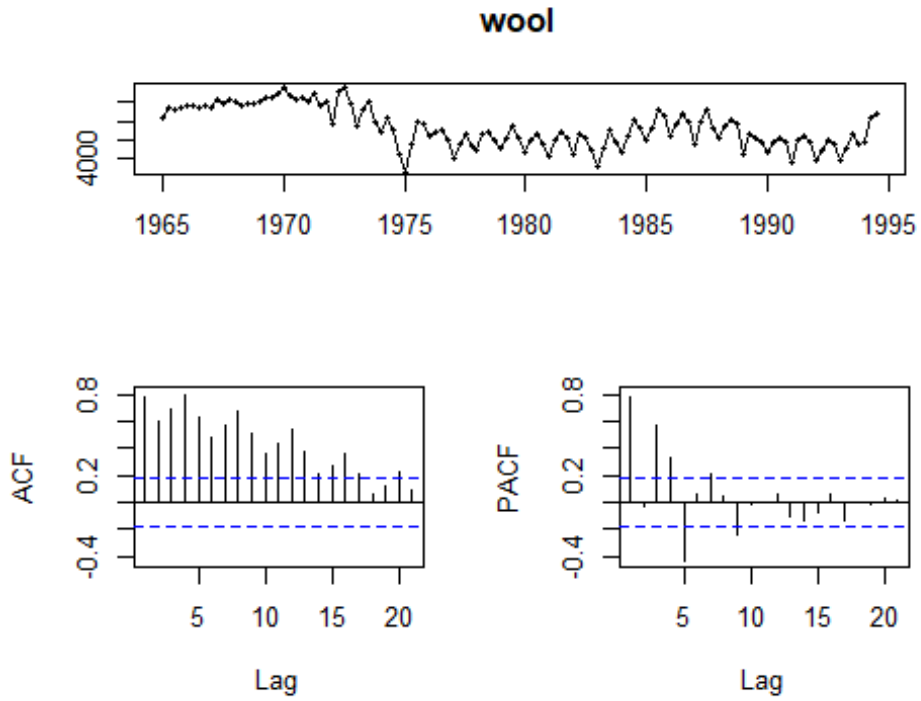



```
seasonplot(wool)
```




```
help(wool)
```

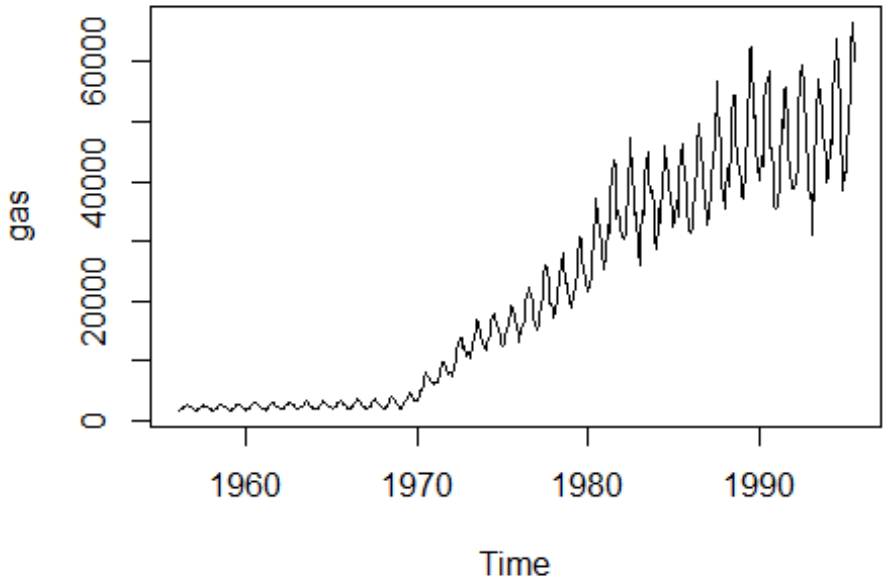
```
## No documentation for 'wool' in specified packages and libraries:
## you could try '??wool'

tsdisplay(wool)
```



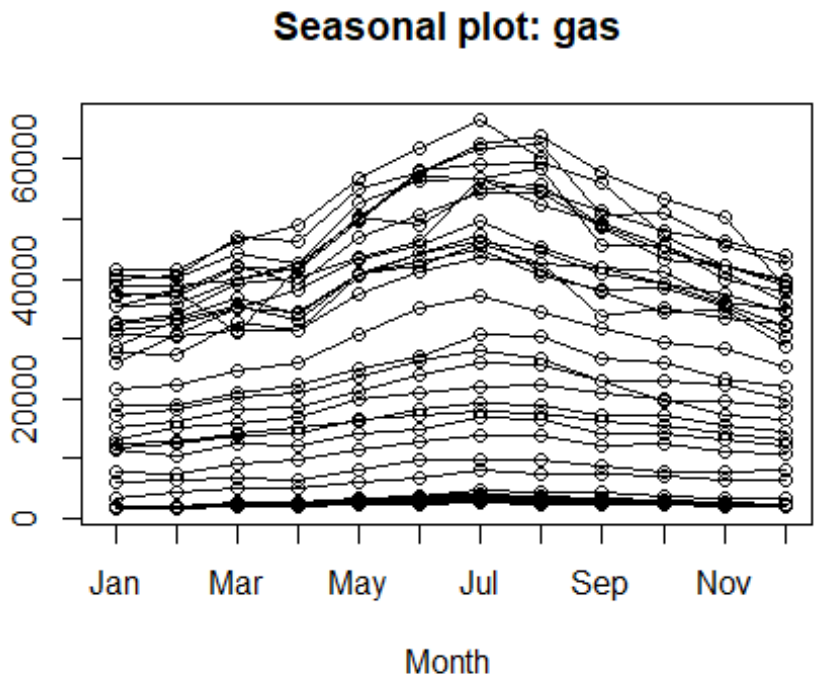
- The `wool` dataset is not structured as a timeseries object but this is easily remedied. It is clear from inspecting the data that the time dimension is quarterly beginning in 1965. The ts (timeseries) function converts the data so seasonality or year-over-year analysis is possible. It isn't clear what the `measure` dimension is capturing but perhaps it represents unit price or amount produced. In any case the trend shows an inital rise during the late 1960's followed by a greater drop during the 1970's and early 1980's, some fluctuation in the later 1980's early 1990's followed by a spike in the mid 1990's. The seasonality plots show peaks in Q3 and nadirs in Q1.
- The lag plots show slight diminishing ACF trend with cyclical seasonal fluctuation and cyclical PACF with a diminishing magnitude.

Evaluation of *gas*

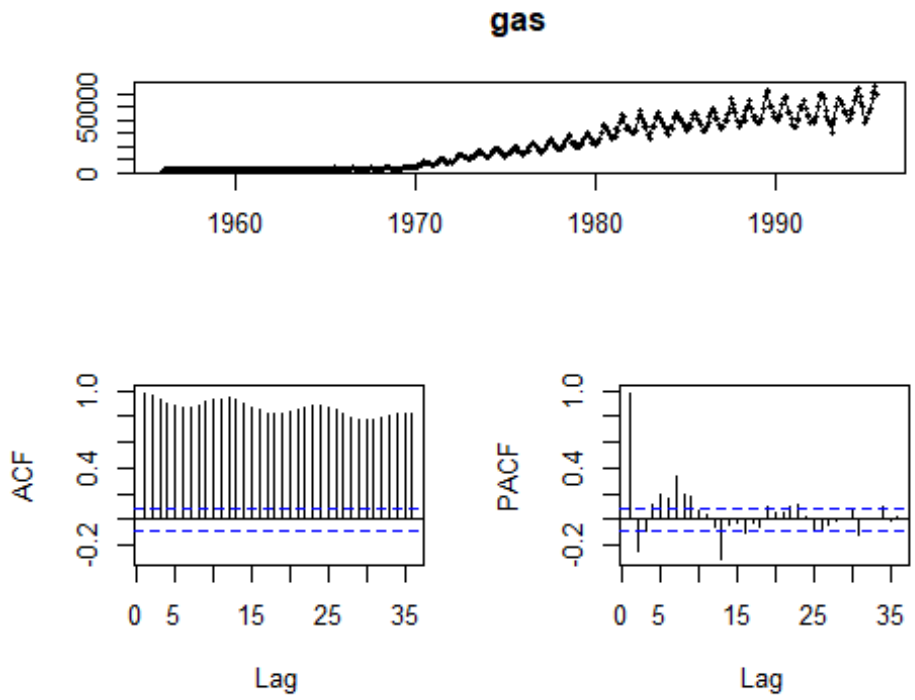
```
help(gas)
#describe(gas)
head(gas)

##       Jan  Feb  Mar  Apr  May  Jun
## 1956 1709 1646 1794 1878 2173 2321

plot(gas)
```




```
seasonplot(gas)
```



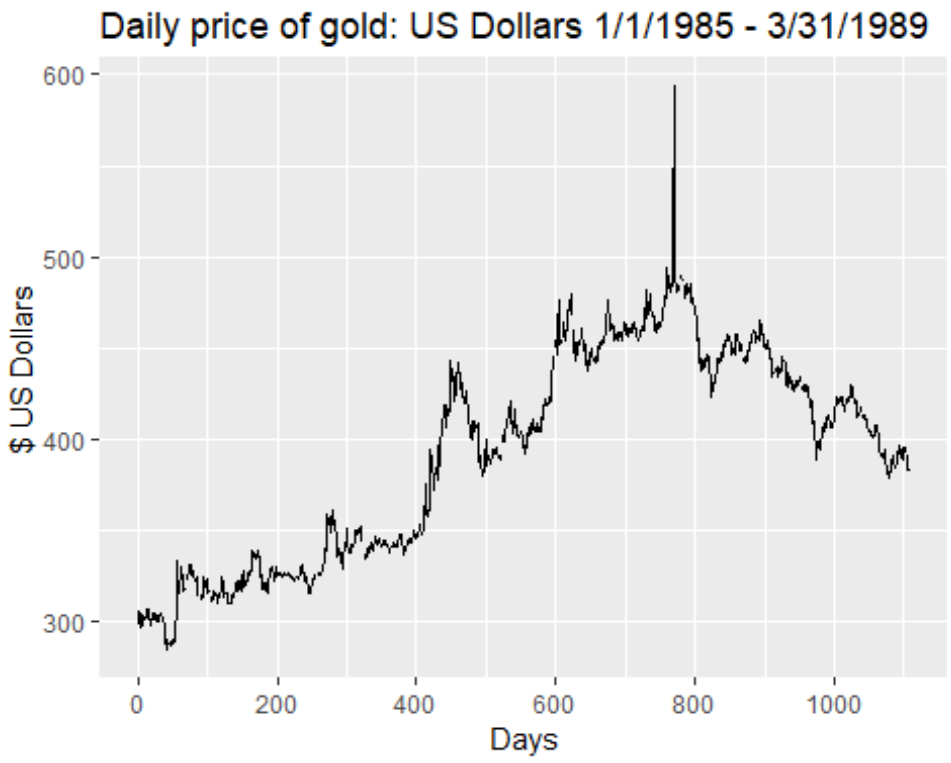

```
tsdisplay(gas)
```



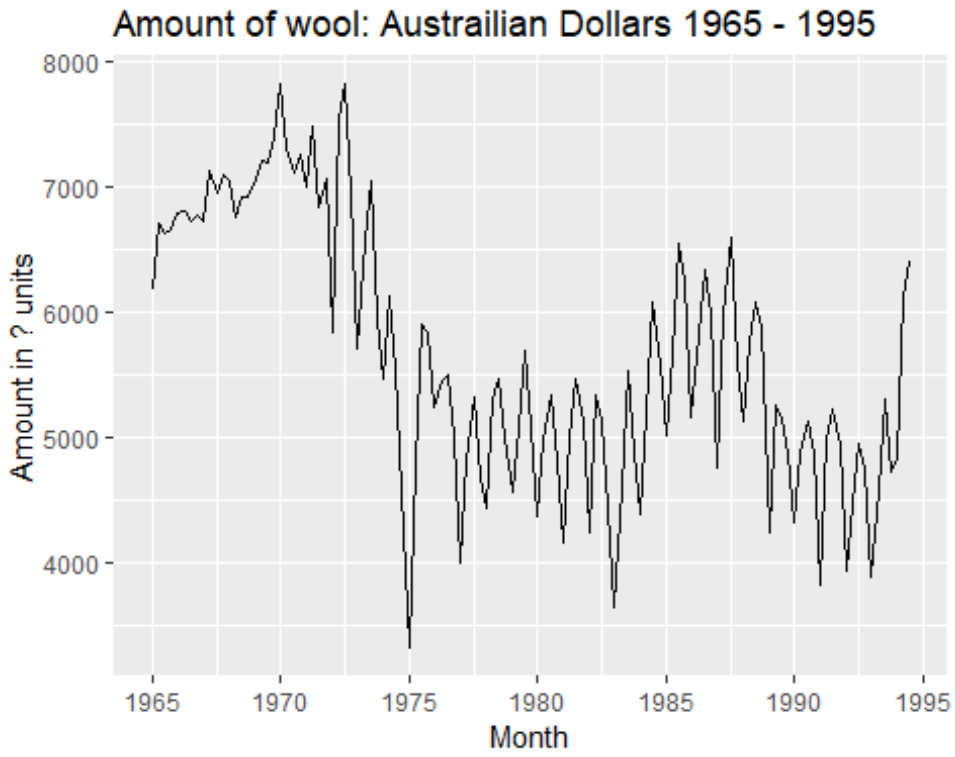
- The `gas` dataset is structured as a timeseries object. The description notes that the dimensions being compared are time months 1956- 1995 against production of Austrailian gas in 10K increments of unit volume.
- Based on the plot, the production of gas is flat up until the 1970s with slight seasonal fluctuation. From 1970 onward however, the trend shows increasing production at an increasing rate coupled with an increase in seasonal fluctuation.
- The seasonal plot shows peak production during July–winter in Austrailia–with a nadir during December and January–summer month.
- The lag plots show slight diminishing ACF trend with cyclical seasonal fluctuation and cyclical PACF with a diminishing magnitude. The magnitude becomes small enough toward the end that it could be attributed to white noise.

a. Use autoplot() to plot each of these in separate plots.

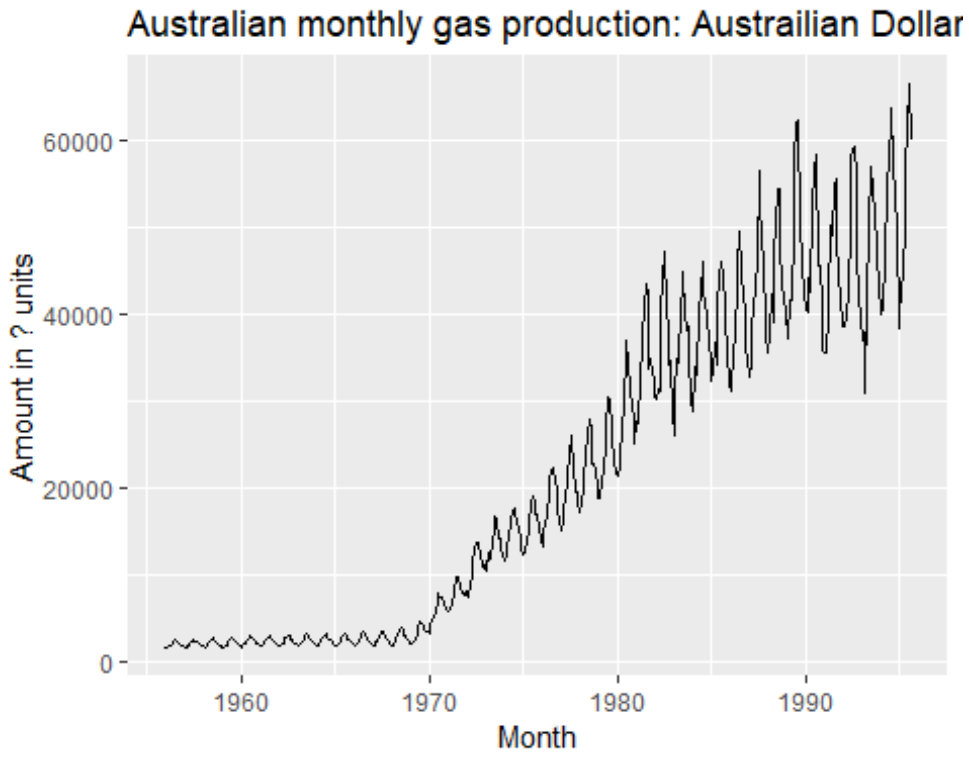
```
autoplot(gold) +
ggtitle("Daily price of gold: US Dollars 1/1/1985 - 3/31/1989") +
xlab("Days") +
ylab("$ US Dollars")
```




```
autoplot(wool) +
ggtitle("Amount of wool: Austrailian Dollars 1965 - 1995") +
xlab("Month") +
ylab("Amount in ? units")
```




```
  autoplot(gas) +
  ggtitle("Australian monthly gas production: Austrailian Dollar 1956â  1995.
") +
  xlab("Month") +
  ylab("Amount in ? units")
```



b. What is the frequency of each series? Hint: apply the frequency() function.

```
freq.df <- data.frame(frequency(gold),frequency(woolyrnq),frequency(gas))
names(freq.df) <- c("Frequency Gold", "Frequency Woolyrnq", "Frequency Gas")
freq.df

##   Frequency Gold Frequency Woolyrnq Frequency Gas
## 1              1                 4             12
```

- Gold:For some reasons, the function indicates that our data is annual basis, while in reality it is on daily basis. It seems to be a glitch.
- Woolyrnq: the function correctly shows that our data is on quaterly basis.
- Gas: Gas data is provided on monthly basis.

c. Use which.max() to spot the outlier in the gold series. Which observation was it?

```
paste0("Maximum Gold (Outlier Detection): ", which.max(gold))

## [1] "Maximum Gold (Outlier Detection): 770"
```

- Spike for gold prices has happened on day 770.

2.3 Download some monthly Australian retail data from the book website. These represent retail sales in various categories for different Australian states, and are stored in a MS-Excel file.

a. You can read the data into R with the following script:

```
temp = tempfile(fileext = ".xlsx")
dataURL <- "https://otexts.com/fpp2/extrafiles/retail.xlsx"
download.file(dataURL, destfile=temp, mode='wb')

retaildata <- readxl::read_excel(temp, sheet =1, skip =1)
kable(head(retaildata[1:6,1:6]))
```

Series ID	A3349335T	A3349627V	A3349338X	A3349398A	A3349468W
1982-04-01	303.1	41.7	63.9	408.7	65.8
1982-05-01	297.8	43.1	64.0	404.9	65.8
1982-06-01	298.0	40.3	62.7	401.0	62.3
1982-07-01	307.9	40.9	65.6	414.4	68.2
1982-08-01	299.2	42.1	62.6	403.8	66.0
1982-09-01	305.4	42.0	64.4	411.8	62.3

b. Select one of the time series as follows (but replace the column name with your own chosen column):

COlumn "A3349337W": "Turnover ; New South Wales ; Hardware, building and garden supplies retailing"

```
myts <- ts(retaildata[,"A3349337W"], frequency=12, start=c(1982,4))
myts
```

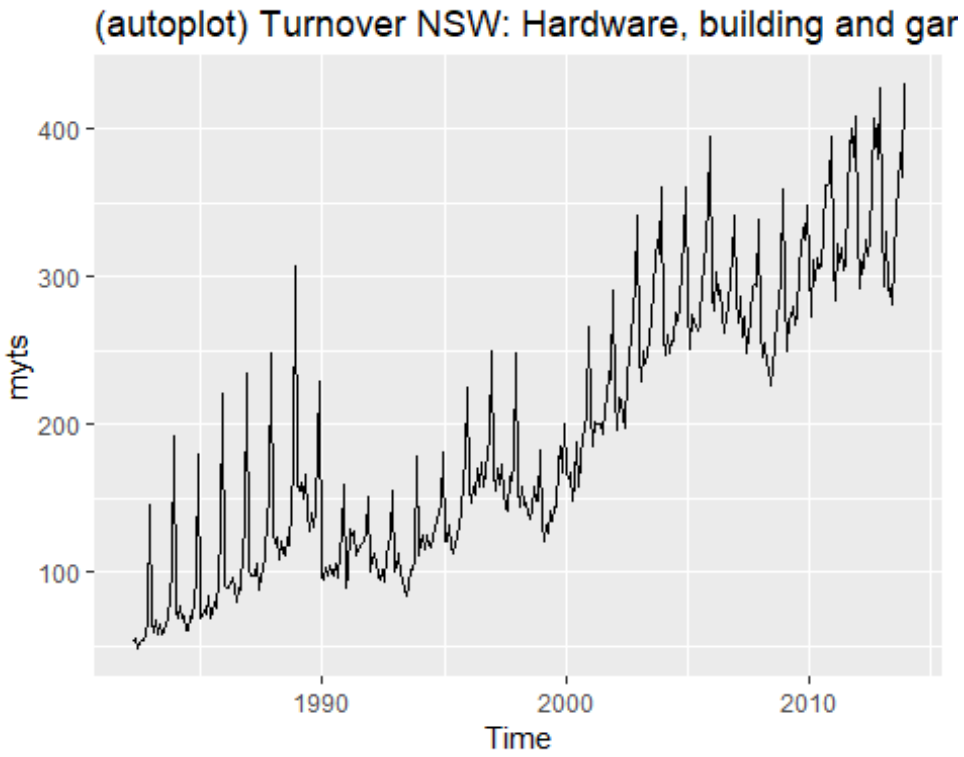
```
##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov
## 1982                     53.6  55.4  48.4  52.1  54.2  53.6  58.0  67.2
## 1983  66.6  59.2  67.3  57.7  64.9  58.6  58.8  64.8  68.7  84.1 101.2
## 1984  73.7  69.6  77.7  68.5  70.0  60.5  60.2  70.0  69.5  81.5  96.5
## 1985  69.4  69.8  74.1  71.9  83.6  68.8  71.8  79.4  76.0  97.0 126.8
## 1986  90.3  89.8  89.6  91.9  96.0  89.3  79.4  89.1  88.1 116.8 128.6
## 1987 103.9  97.3  97.9  97.2 106.5  88.2  97.7 100.2 110.8 137.3 150.5
## 1988 126.6 119.4 123.6 108.8 121.0 113.9 110.9 124.3 118.5 143.9 172.1
## 1989 160.7 155.2 161.0 149.3 165.6 140.1 128.2 140.4 130.2 143.3 185.3
## 1990  96.4  95.0 103.8  97.1 104.6 100.7  98.2 106.6  96.7 113.3 126.2
## 1991  89.1  99.6 129.0 125.6 127.3 111.7 114.1 118.0 119.6 121.5 128.5
## 1992 100.1 108.2 113.2 108.0  98.2  95.2 101.4  93.5 112.0 118.9 125.7
## 1993 100.7 102.8 113.5  99.2  95.4  89.3  84.4  91.1 102.2 101.4 108.5
## 1994 111.0 121.4 125.6 116.2 125.1 119.1 117.5 123.8 134.5 141.0 145.2
## 1995 120.8 121.0 132.6 116.3 113.2 120.2 124.3 134.0 140.6 163.7 176.2
## 1996 157.5 147.7 158.1 152.4 171.0 158.0 174.0 157.5 167.0 181.0 189.6
## 1997 168.0 154.9 169.9 159.8 172.7 154.1 144.9 141.3 164.3 162.7 172.8
## 1998 157.0 145.0 158.6 145.9 146.8 140.2 135.8 141.7 158.7 148.4 148.0
## 1999 133.1 120.5 132.2 126.0 141.0 135.0 143.7 144.4 171.7 185.5 167.9
## 2000 169.7 163.2 167.6 148.7 161.4 188.5 158.3 174.5 193.2 194.5 209.7
## 2001 209.6 185.2 202.2 200.0 200.3 200.3 193.6 211.4 218.2 236.3 230.6
## 2002 219.9 196.6 218.7 216.8 205.5 198.2 233.9 246.2 259.8 277.3 294.3
## 2003 247.0 229.3 250.3 241.6 247.0 258.7 271.3 291.1 312.7 324.6 315.2
## 2004 258.9 246.5 260.9 249.0 256.5 257.4 275.4 269.8 279.8 307.3 323.9
```

```
## 2005 281.8 250.6 274.1 270.3 268.2 264.0 266.9 298.6 303.1 329.4 345.6
## 2006 288.0 277.3 302.8 288.5 290.4 275.4 262.4 272.9 279.7 299.3 313.3
## 2007 286.4 268.4 286.6 260.0 273.0 248.5 259.7 272.2 293.6 294.9 294.3
## 2008 263.0 246.2 255.2 240.2 239.6 226.9 238.7 253.1 271.3 283.1 299.0
## 2009 289.3 249.6 272.1 272.9 279.4 267.8 273.1 307.7 318.2 334.0 325.0
## 2010 309.2 272.6 311.1 298.2 313.1 305.8 307.3 330.9 362.8 361.7 364.2
## 2011 311.6 283.7 322.2 310.8 319.5 305.1 308.9 355.6 384.9 401.1 382.1
## 2012 334.0 292.1 309.6 305.8 325.0 314.2 327.2 363.7 406.9 397.1 379.6
## 2013 340.0 293.9 330.7 290.7 291.8 281.1 309.8 344.6 360.7 384.7 367.9
##        Dec
## 1982 146.3
## 1983 192.3
## 1984 179.4
## 1985 221.2
## 1986 235.4
## 1987 248.8
## 1988 307.4
## 1989 228.9
## 1990 159.5
## 1991 151.4
## 1992 154.7
## 1993 179.0
## 1994 180.7
## 1995 225.4
## 1996 249.8
## 1997 248.7
## 1998 183.0
## 1999 200.7
## 2000 266.3
## 2001 291.0
## 2002 341.9
## 2003 360.8
## 2004 361.1
## 2005 395.2
## 2006 341.6
## 2007 339.3
## 2008 360.2
## 2009 348.9
## 2010 395.4
## 2011 409.0
## 2012 428.0
## 2013 430.7
```

c. Explore your chosen retail time series using the following functions

-autoplot() A trend exists when there is a long-term increase or decrease in the data. As per the below (autoplot) there seems to be a general upward trend in the data

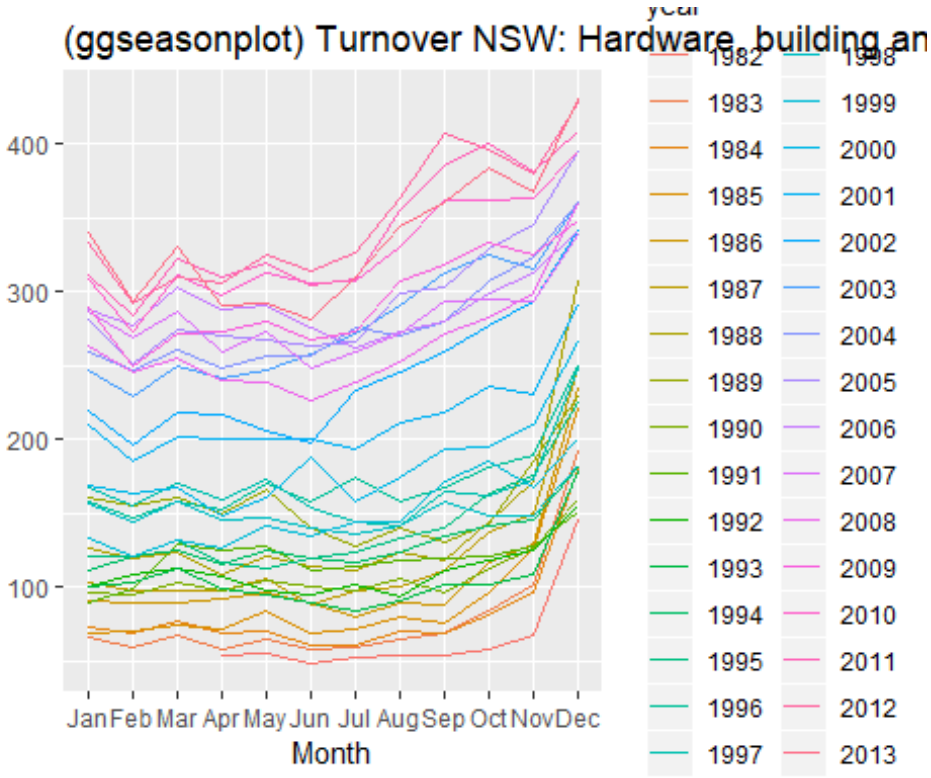
```
autoplot<- autoplot(myts) + ggtitle("(autoplot) Turnover NSW: Hardware, build
ing and garden supplies retailing")
autoplot
```



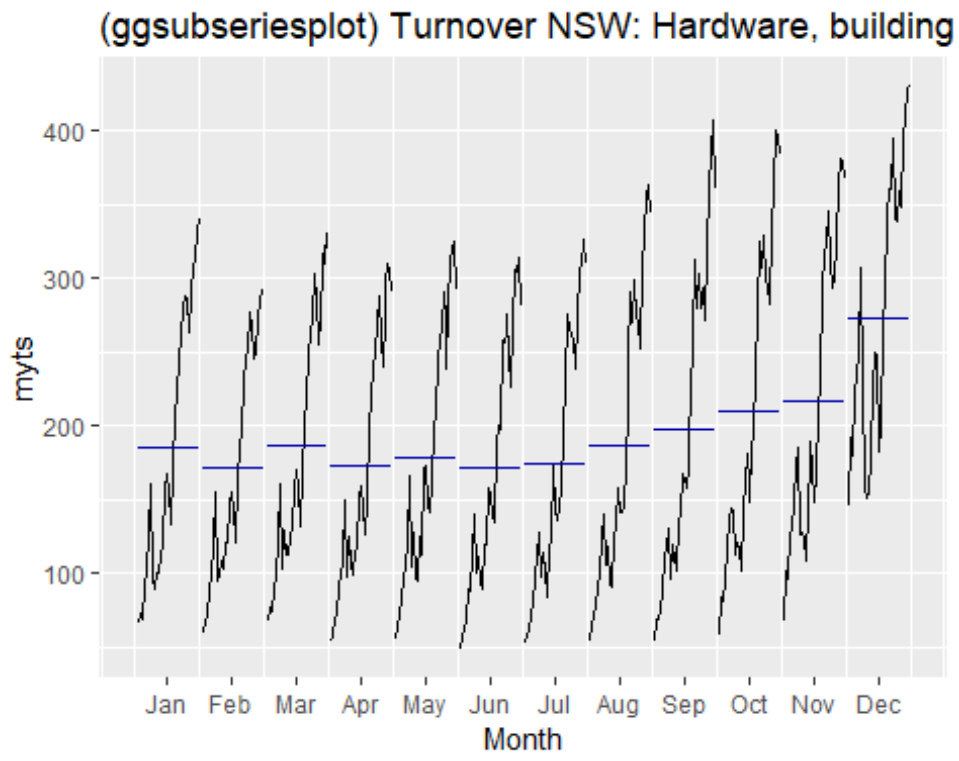
- Seasonality: A seasonal pattern occurs when a time series is affected by seasonal factors such as the time of the year or the day of the week. The seasonal plots are similar to a time plot except that the data are plotted against the individual “seasons” in which the data were observed. Below, the turnover in NSW for Hardware, building and garden supplies retailing is plotted using the seasonal plots to test seasonality. The data shows that the turnover is higher in the September, October, November and December months. This makes sense as Australia spring season starts in September and ends in late December.This is especially exemplified in the lag plots with the largest lags 12 months apart and the polar seasonplot
  - ggseasonplot(): It seems our data tend to increase slightly in December. February, the shortest month, seems to dip a little bit
  - ggsubseriesplot(): Again, we see that December is the highest month and February is the lowest
  - gglagplot(): If we look at Lag 12 we can see very strong indication of autocorrelation.
  - ggAcf(): Consistent decrease due to trend and very slight “scalloped” shape due to slight seasonality.

```
#seasonality

ggseasonplot<- ggseasonplot(myts)+ ggtitle("(ggseasonplot) Turnover NSW: Hard
ware, building and garden supplies retailing")
```

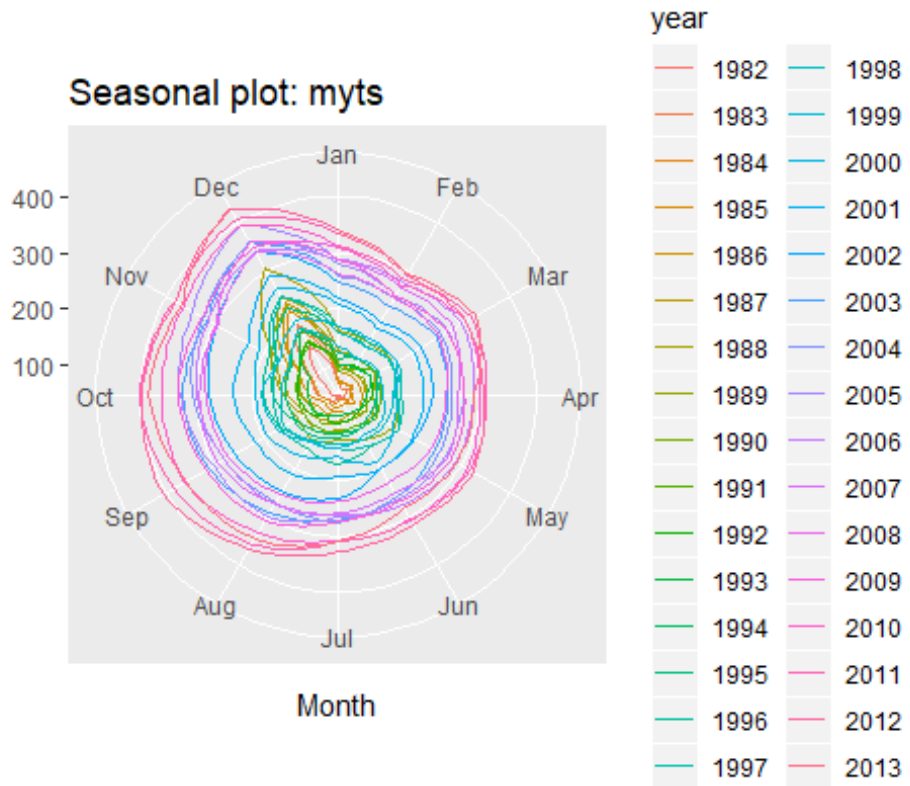
```
ggsubseriesplot<- ggsubseriesplot(myts)+ ggtitle("(ggsubseriesplot) Turnover
NSW: Hardware, building and garden supplies retailing")

ggseasonplotpolar<-ggseasonplot(myts, polar=TRUE)
gglagplot<- gglagplot(myts)+ ggtitle("(gglagplot) Turnover NSW: Hardware, bui
lding and garden supplies retailing")
ggAcf<- ggAcf(myts)+ ggtitle("(ggAcf) Turnover NSW: Hardware, building and ga
rden supplies retailing")

ggseasonplot
```



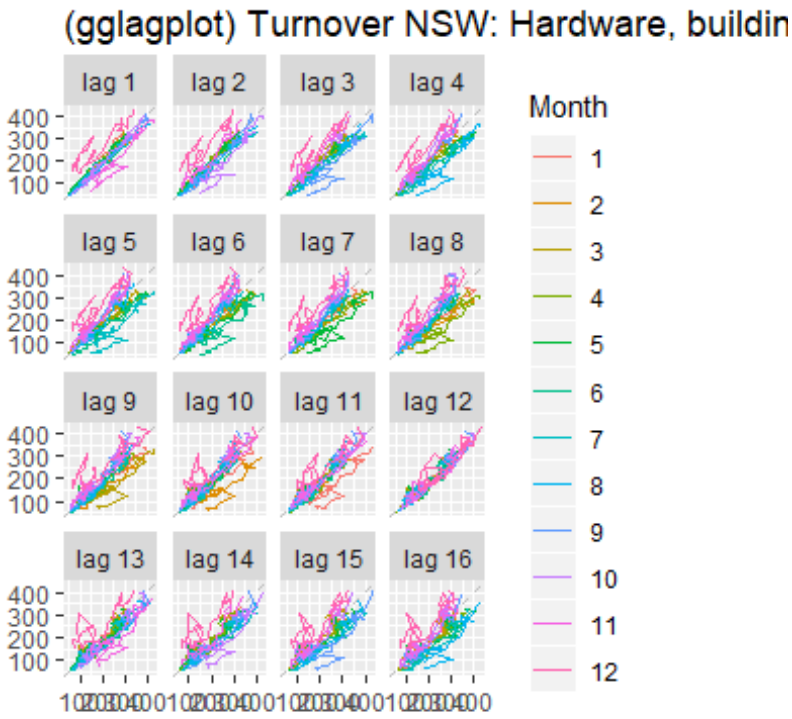

```
ggsubseriesplot
```



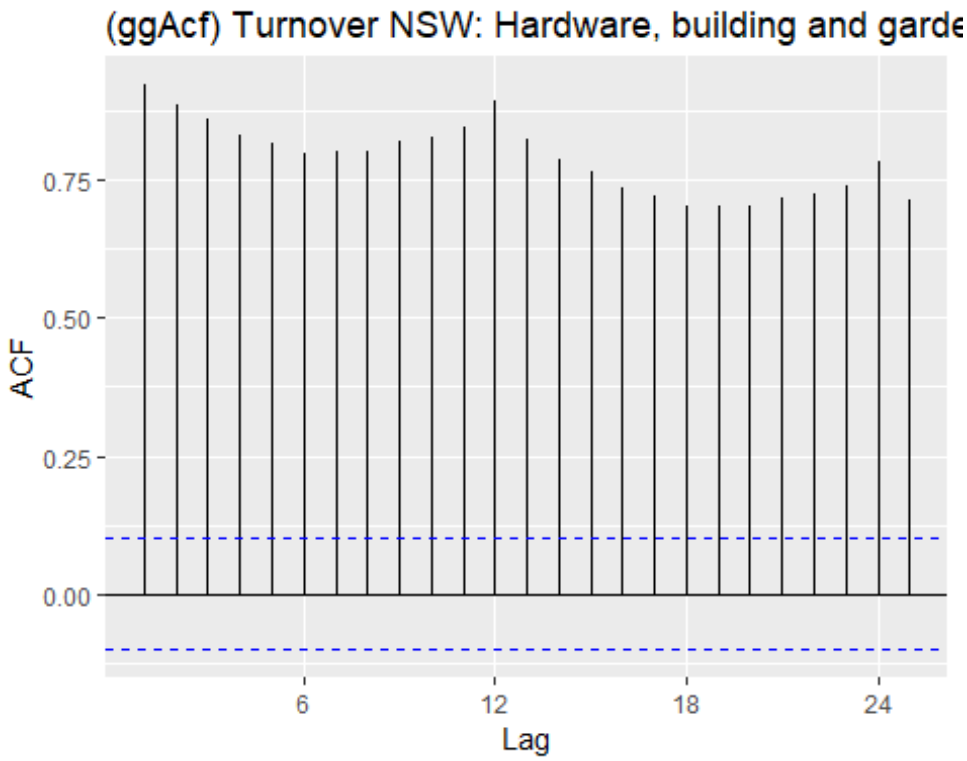

```
ggseasonplotpolar
```




```
gglagplot
```


(gglagplot) Turnover NSW: Hardware, buildin
lag 1
lag 2
lag 3
lag 4
lag 5
lag 6
lag 7
lag 8
lag 9
lag 10
lag 11
lag 12
lag 13
lag 14
lag 15
lag 16
400
300
200
100
Month
1
2
3
4
5
6
7
8
9
10
11
12

```
ggAcf
```


(ggAcf) Turnover NSW: Hardware, building and garde
ACF
0.75
0.50
0.25
0.00
6
12
18
24
Lag

Data 624: Week 2 Homework

Angrand, Burke, Deboch, Groysman, Karr

October 15, 2019

Week 2 Assignment

Chapter 6 HA 6.2

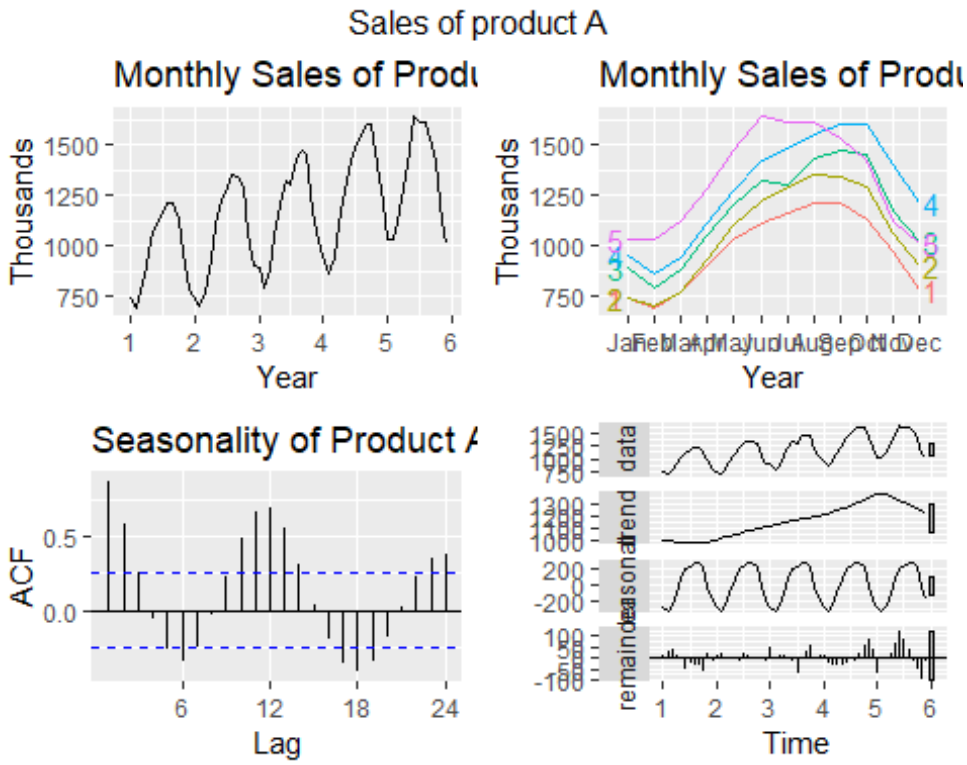
6.2 The plastics data set consists of the monthly sales (in thousands) of product A for a plastics manufacturer for five years.

```
#preliminary EDA
plastics
```

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

a. Plot the time series of sales of product A. Can you identify seasonal fluctuations and/or a trend-cycle? - Trend: Increasing over 1 year. Appears to be some seasonality(lag graph), length 1 year.Data is heavily seasonal.Dropping in winter and peaking in summer. Overall trend is positive.

```
general.plot <- autoplot(plastics) +ggtitle("Monthly Sales of Product A") +xl
ab("Year") +ylab("Thousands")
seasonal.plot <-ggseasonplot(plastics, year.labels=TRUE, year.labels.left=TRU
E) +
  ggtitle("Monthly Sales of Product A") +
  xlab("Year") +
  ylab("Thousands")

acf.plot <-ggAcf(plastics)+ ggtitle("Seasonality of Product A")
decomp.plot <- autoplot(stl(plastics,"periodic"))
grid.arrange(general.plot,
             seasonal.plot,
             acf.plot,
             decomp.plot,
             nrow = 2,
             top = "Sales of product A")
```



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

- The multiplicative decomposition shows a general upward trend from year 2 through year 5. After year 5, the trend appears to be decreasing. The decomposition graph also displays a seasonal shift (up/down) in one year increments.

```
plastics_decomp<-decompose(plastics,type="multiplicative")
head(plastics_decomp)
```

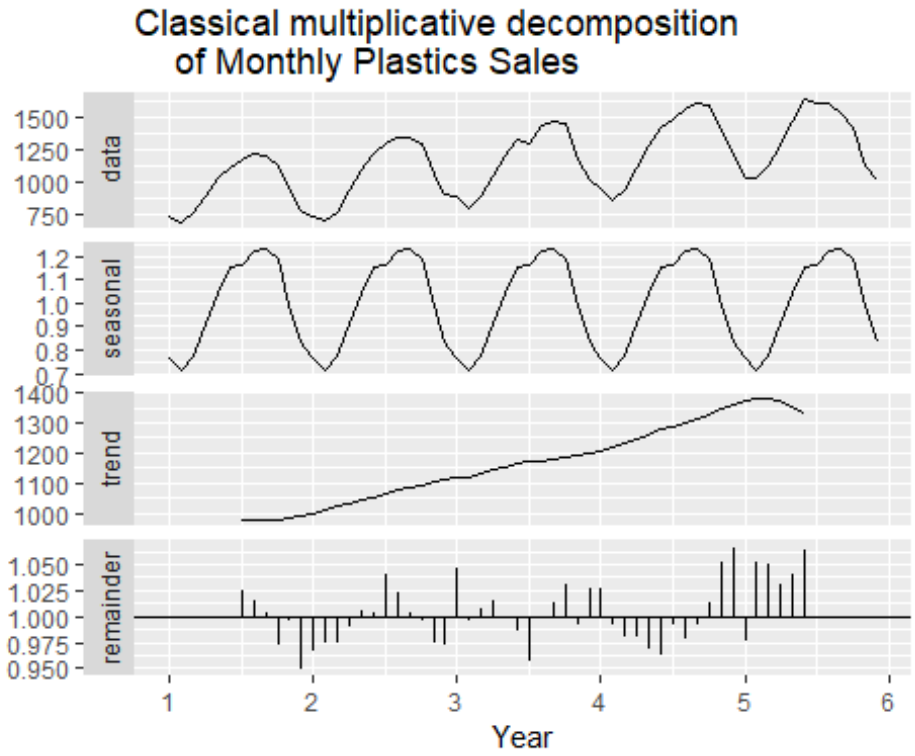
```
## $x
##    Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 1  742  697  776  898 1030 1107 1165 1216 1208 1131  971  783
## 2  741  700  774  932 1099 1223 1290 1349 1341 1296 1066  901
## 3  896  793  885 1055 1204 1326 1303 1436 1473 1453 1170 1023
## 4  951  861  938 1109 1274 1422 1486 1555 1604 1600 1403 1209
## 5 1030 1032 1126 1285 1468 1637 1611 1608 1528 1420 1119 1013
##
## $seasonal
##         Jan       Feb       Mar       Apr       May       Jun       Jul
## 1 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
## 2 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
## 3 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
## 4 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
## 5 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
##         Aug       Sep       Oct       Nov       Dec
## 1 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
## 2 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
```

```
## 3 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
## 4 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
## 5 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
##
## $trend
##         Jan       Feb       Mar       Apr       May       Jun       Jul
## 1        NA        NA        NA        NA        NA        NA  976.9583
## 2 1000.4583 1011.2083 1022.2917 1034.7083 1045.5417 1054.4167 1065.7917
## 3 1117.3750 1121.5417 1130.6667 1142.7083 1153.5833 1163.0000 1170.3750
## 4 1208.7083 1221.2917 1231.7083 1243.2917 1259.1250 1276.5833 1287.6250
## 5 1374.7917 1382.2083 1381.2500 1370.5833 1351.2500 1331.2500        NA
##         Aug       Sep       Oct       Nov       Dec
## 1  977.0417  977.0833  978.4167  982.7083  990.4167
## 2 1076.1250 1084.6250 1094.3750 1103.8750 1112.5417
## 3 1175.5000 1180.5417 1185.0000 1190.1667 1197.0833
## 4 1298.0417 1313.0000 1328.1667 1343.5833 1360.6250
## 5        NA        NA        NA        NA        NA
##
## $random
##         Jan       Feb       Mar       Apr       May       Jun       Jul
## 1        NA        NA        NA        NA        NA        NA 1.0247887
## 2 0.9656005 0.9745267 0.9750081 0.9894824 1.0061175 1.0024895 1.0401641
## 3 1.0454117 0.9953920 1.0079773 1.0142083 0.9990100 0.9854384 0.9567618
## 4 1.0257400 0.9924762 0.9807020 0.9798704 0.9684851 0.9627557 0.9917766
## 5 0.9767392 1.0510964 1.0498039 1.0299302 1.0398787 1.0628077        NA
##         Aug       Sep       Oct       Nov       Dec
## 1 1.0157335 1.0040354 0.9724119 0.9961368 0.9489762
## 2 1.0230774 1.0040674 0.9962088 0.9735577 0.9721203
## 3 0.9969907 1.0132932 1.0314752 0.9910657 1.0258002
## 4 0.9776897 0.9920952 1.0133954 1.0527311 1.0665946
## 5        NA        NA        NA        NA        NA
##
## $figure
##  [1] 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
##  [8] 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
##
## $type
## [1] "multiplicative"
```

```
summary(plastics_decomp)
```

```
##          Length Class  Mode
## x        60     ts     numeric
## seasonal 60     ts     numeric
## trend    60     ts     numeric
## random   60     ts     numeric
## figure   12     -none- numeric
## type      1     -none- character
```

```r
plastics %>% decompose(type="multiplicative") %>%
  autoplot() + xlab("Year") +
  ggtitle("Classical multiplicative decomposition
    of Monthly Plastics Sales")
```



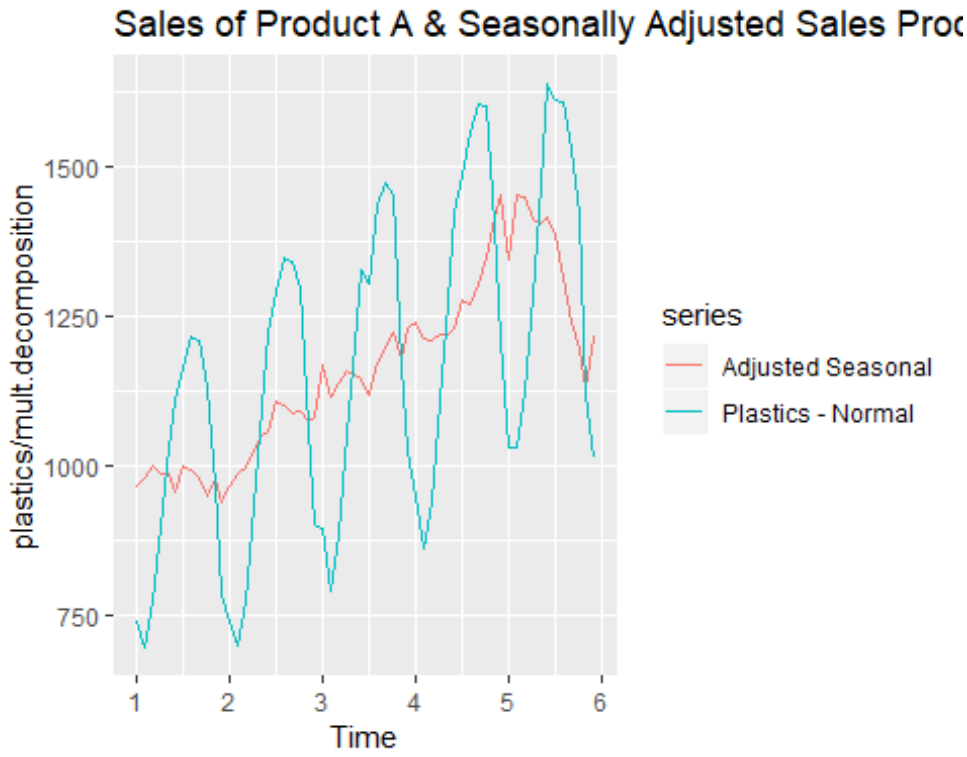
c. Do the results support the graphical interpretation from part a?

- Both part a and part b display a seasonal component of 1 year and a trend that is increasing over time.The decomposition shows strong seasonality in middle of year(summer). The decomposition does show a decline after year 5 because it relies on moving averages.

d. Compute and plot the seasonally adjusted data.

- The "Adjusted Seasonal" plot shows the sales with the seasonality removed, making the upward trend is more visable. Seasonal Adjustment plot also shows that the first year sales were flat and then they grew for the next 3 years. The 5th year saw a drop in sales.

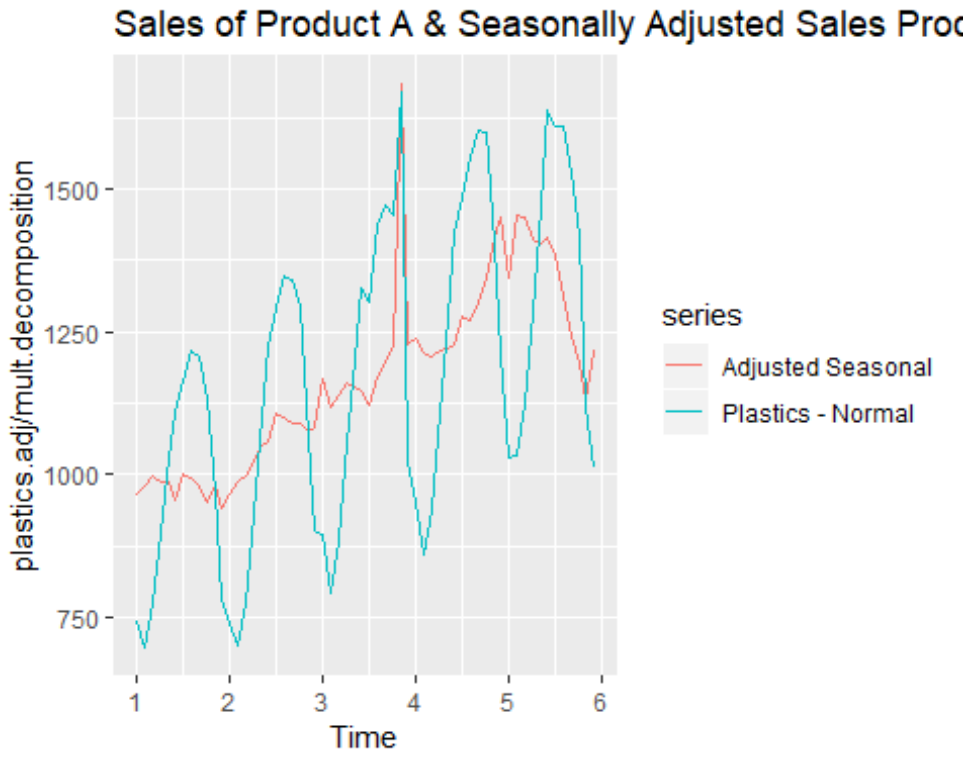
```r
mult.decomposition <- plastics %>%
  decompose(type="multiplicative") %>%
  seasonal


autoplot(plastics /mult.decomposition, series = "Adjusted Seasonal")+
  autolayer(plastics, series = "Plastics - Normal") +
  ggtitle("Sales of Product A & Seasonally Adjusted Sales Product A")
```



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?

- After 500 was added to the 35th datapoint in "plastics.adj" dataset, a large spike occurred in the middle of the data series and the seasonally adjusted data series. This can be observed in the chart below.

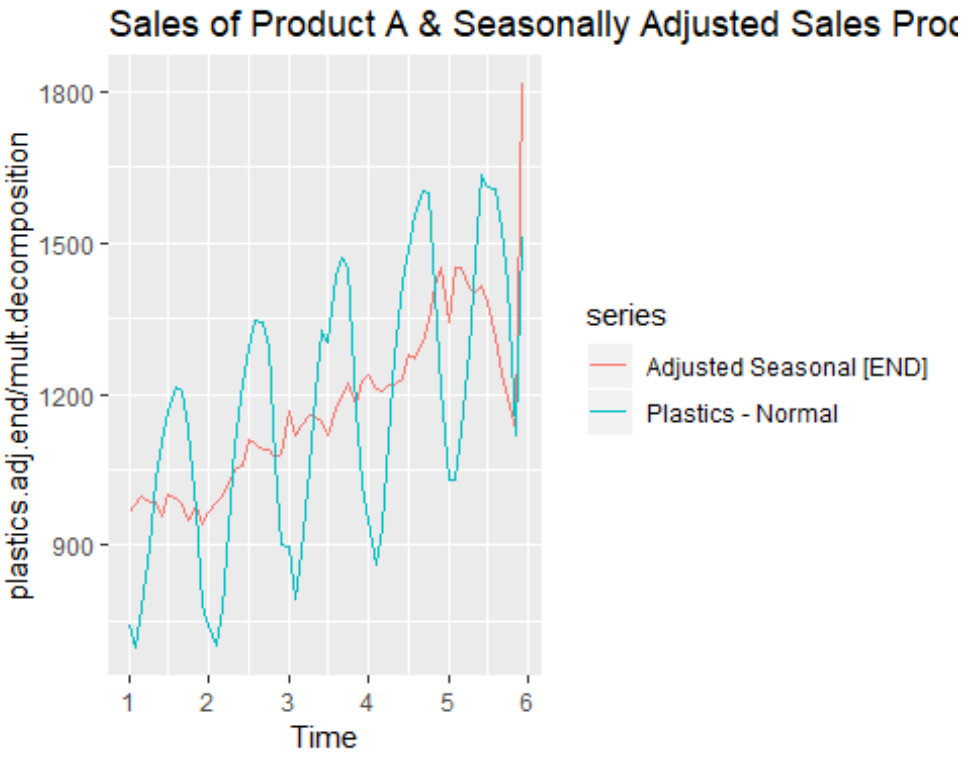
```
plastics.adj <- plastics
plastics.adj[35] <- plastics.adj[35]+500


mult.decomposition.adj <- plastics.adj %>%
  decompose(type="multiplicative") %>%
  seasonal


autoplot(plastics.adj /mult.decomposition, series = "Adjusted Seasonal")+
  autolayer(plastics.adj, series = "Plastics - Normal") +
  ggtitle("Sales of Product A & Seasonally Adjusted Sales Product A (Adjusted
 Series [MIDDLE])")
```



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

- After 500 was added to the 60th datapoint in "plastics.adj.end" dataset, a large spike occurred at the end of the dataset. However, the seasonality seems to more closely mirror that of the original chart.

```
end.series<- length(plastics)
plastics.adj.end <- plastics
plastics.adj.end[end.series] <- plastics.adj.end[end.series]+500


mult.decomposition.adj <- plastics.adj.end %>%
  decompose(type="multiplicative") %>%
  seasonal


autoplot(plastics.adj.end /mult.decomposition, series = "Adjusted Seasonal [E
ND]")+
  autolayer(plastics.adj.end, series = "Plastics - Normal") +
  ggtitle("Sales of Product A & Seasonally Adjusted Sales Product A (Adjusted
 Series [END])")
```


Sales of Product A & Seasonally Adjusted Sales Proc
plastics.adj.end/mult.decomposition
1800
1500
1200
900
1
2
3
4
5
6
Time
series
Adjusted Seasonal [END]
Plastics - Normal

Data 624: Week 3 Homework

Angrand, Burke, Deboch, Groysman, Karr

October 15, 2019

Week 3 Assignment

Chapter 3 KJ 1 and 2

3.1 The UC Irvine Machine Learning Repository 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:

```
data(Glass)
describe(Glass)
```

```
##       vars   n  mean   sd median trimmed  mad   min   max range  skew
## RI       1 214  1.52 0.00   1.52    1.52 0.00  1.51  1.53  0.02  1.60
## Na       2 214 13.41 0.82  13.30   13.38 0.64 10.73 17.38  6.65  0.45
## Mg       3 214  2.68 1.44   3.48    2.87 0.30  0.00  4.49  4.49 -1.14
## Al       4 214  1.44 0.50   1.36    1.41 0.31  0.29  3.50  3.21  0.89
## Si       5 214 72.65 0.77  72.79   72.71 0.57 69.81 75.41  5.60 -0.72
## K        6 214  0.50 0.65   0.56    0.43 0.17  0.00  6.21  6.21  6.46
## Ca       7 214  8.96 1.42   8.60    8.74 0.66  5.43 16.19 10.76  2.02
## Ba       8 214  0.18 0.50   0.00    0.03 0.00  0.00  3.15  3.15  3.37
## Fe       9 214  0.06 0.10   0.00    0.04 0.00  0.00  0.51  0.51  1.73
## Type*   10 214  2.54 1.71   2.00    2.31 1.48  1.00  6.00  5.00  1.04
##       kurtosis   se
## RI        4.72 0.00
## Na        2.90 0.06
## Mg       -0.45 0.10
## Al        1.94 0.03
## Si        2.82 0.05
## K        52.87 0.04
## Ca        6.41 0.10
## Ba       12.08 0.03
## Fe        2.52 0.01
## Type*    -0.29 0.12
```

```
str(Glass)
```

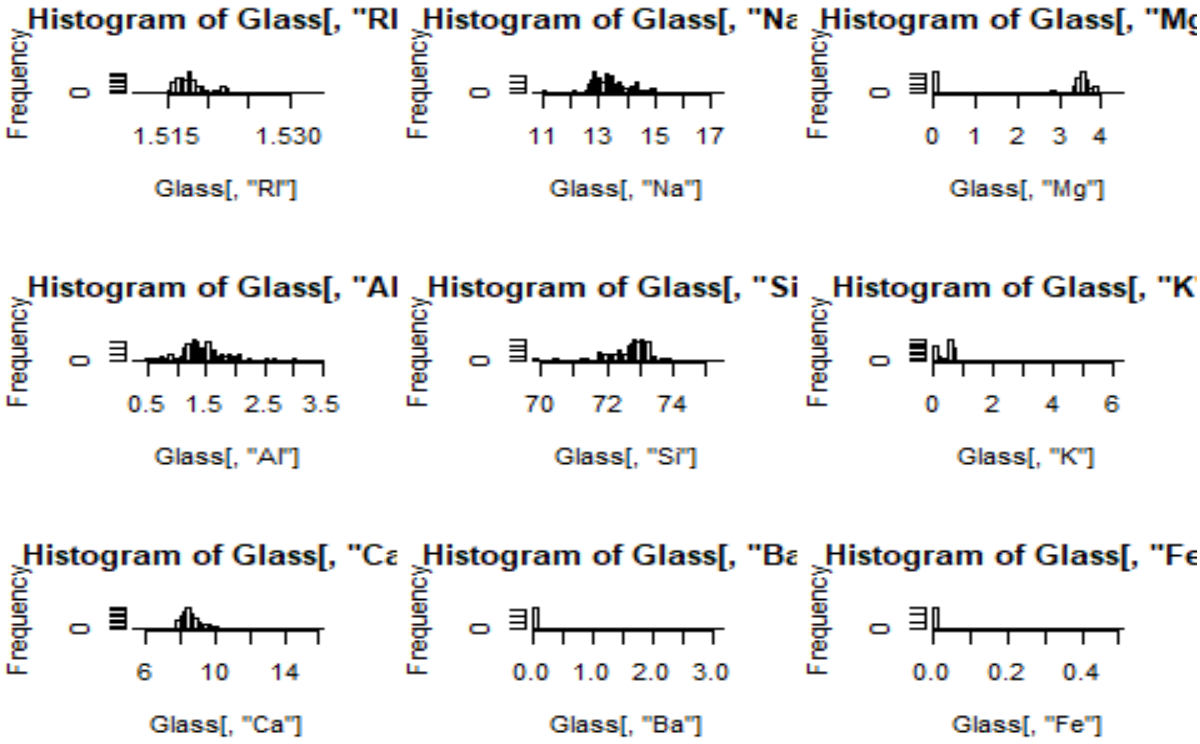
```
## 'data.frame':    214 obs. of  10 variables:
##  $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
##  $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
##  $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
##  $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
##  $ Si  : num  71.8 72.7 73 72.6 73.1 ...
##  $ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
##  $ Ca  : num  8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...
```

```
##  $ Ba  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
##  $ Type: Factor w/ 6 levels "1","2","3","5",..: 1 1 1 1 1 1 1 1 1 1 ...

my_df <- data.frame(Glass[,1:9])

cor(my_df)

##                RI          Na           Mg          Al           Si
## RI  1.0000000000 -0.19188538 -0.122274039 -0.40732603 -0.54205220
## Na -0.1918853790  1.00000000 -0.273731961  0.15679367 -0.06980881
## Mg -0.1222740393 -0.27373196  1.000000000 -0.48179851 -0.16592672
## Al -0.4073260341  0.15679367 -0.481798509  1.00000000 -0.00552372
## Si -0.5420521997 -0.06980881 -0.165926723 -0.00552372  1.00000000
## K  -0.2898327111 -0.26608650  0.005395667  0.32595845 -0.19333085
## Ca  0.8104026963 -0.27544249 -0.443750026 -0.25959201 -0.20873215
## Ba -0.0003860189  0.32660288 -0.492262118  0.47940390 -0.10215131
## Fe  0.1430096093 -0.24134641  0.083059529 -0.07440215 -0.09420073
##                K          Ca            Ba           Fe
## RI -0.289832711  0.8104027 -0.0003860189  0.143009609
## Na -0.266086504 -0.2754425  0.3266028795 -0.241346411
## Mg  0.005395667 -0.4437500 -0.4922621178  0.083059529
## Al  0.325958446 -0.2595920  0.4794039017 -0.074402151
## Si -0.193330854 -0.2087322 -0.1021513105 -0.094200731
## K   1.000000000 -0.3178362 -0.0426180594 -0.007719049
## Ca -0.317836155  1.0000000 -0.1128409671  0.124968219
## Ba -0.042618059 -0.1128410  1.0000000000 -0.058691755
## Fe -0.007719049  0.1249682 -0.0586917554  1.000000000
```

- A data frame with 214 observation containing examples of the chemical analysis of 7 different types of glass.

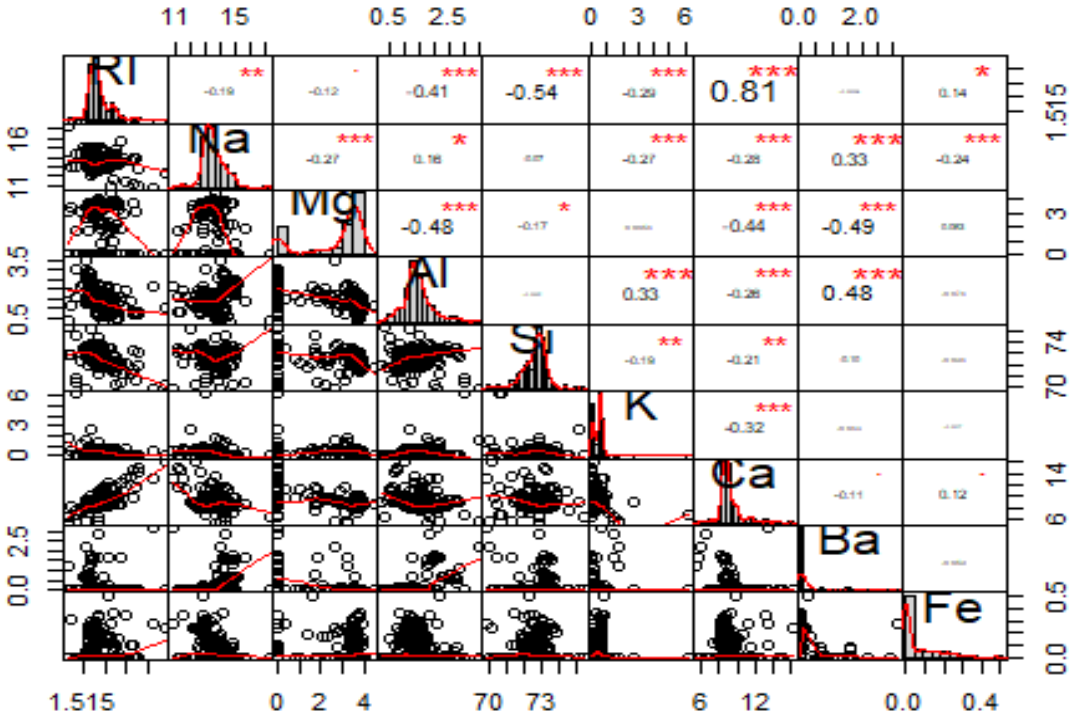
a. Using visualizations explore the predictor variables to understand their distributions as well as the relationships between predictors.

```
#histograms for each
#ZERO-INFLATED NEGATIVE BINOMIAL for Mg, Ba & Fe or is it a nuanced distribut
ion
par(mfrow = c(3,3))
hist(Glass[,'RI'],breaks=50)
hist(Glass[,'Na'],breaks=50)
hist(Glass[,'Mg'],breaks=50)
hist(Glass[,'Al'],breaks=50)
hist(Glass[,'Si'],breaks=50)
hist(Glass[,'K'],breaks=50)
hist(Glass[,'Ca'],breaks=50)
hist(Glass[,'Ba'],breaks=50)
hist(Glass[,'Fe'],breaks=50)
```



- There are a total of 214 glass samples taken with no instances of missing data for any of the predictor variables. Based upon their histograms and skewness, the predictors RI, Na, Al, Si & Ca display either either a normal distribution pattern or a distribution that could be transformed into a normal distribution pattern i.e. division by sqrt(s). The remaining predictor variables Mg, K, Ba & Fe display concentrations of 0 frequency.

```
my_df <- data.frame(Glass[,1:9])
chart.Correlation(my_df, histogram=TRUE, pch=19)
```



- From correlation we can see that:
  - RI is significantly positively correlated with CA and negatively correlated with AL,Si,K.
  - Na is Significantly positively correlated with Ba and negatively correlated with Mg,Al,K,Ca,Fe.
  - Mg is significantly negatively correlated with Ca,Ba ,Al.
  - Al is significantly positively correlated with K,Ba and negatively correlated with Ca.
  - Si is weakly negatively correlated with K and Cal.

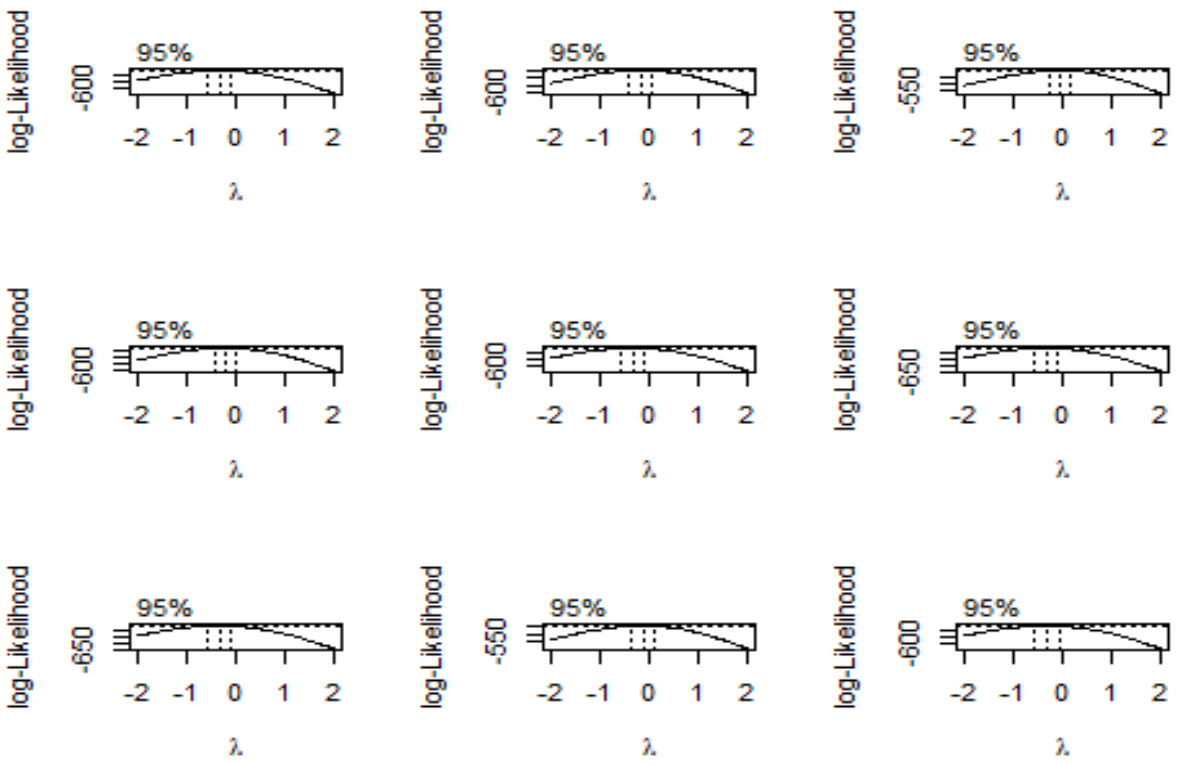
b. Do there appear to be any outliers in the data? Are any predictors skewed?

- From the above plot of histograms we can see that Mg,Si,K,Ca,Ba and Fe has outliers. Fe,Ba,Ca,K,Na,RI are positively skewed and Mg,Si are negatively skewed.

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

```
Glass$Type <- as.numeric(Glass$Type)
par(mfrow = c(3,3))
boxcox(Type~RI, data = Glass)
boxcox(Type~Na, data = Glass)
boxcox(Type~Mg, data = Glass)
boxcox(Type~Al, data = Glass)
boxcox(Type~Si, data = Glass)
```

```
boxcox(Type~K, data = Glass)
boxcox(Type~Ca, data = Glass)
boxcox(Type~Ba, data = Glass)
boxcox(Type~Fe, data = Glass)
```



- A better solution to handling the predictors with concentrations of 0 frequency is to use a zero-inflated binary distribution for continuous data. The two predictors with the greatest correlation are RI and Ca suggesting that in a multivariable regression model, one of these explanatory variables could be removed because it is strongly co-linear with the other thus having little to no loss of predictive ability to the model. Also, from the box cox transformation plot we can see that log transformation of Na, Mg and Ba will improve the model

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.

```
#Preliminary EDA
#Data Access
data(Soybean)
#Sampling
glimpse(Soybean)
```

```
## Observations: 683
## Variables: 36
## $ Class           <fct> diaporthe-stem-canker, diaporthe-stem-canker, ...
## $ date            <fct> 6, 4, 3, 3, 6, 5, 5, 4, 6, 4, 6, 4, 3, 6, 6, 5...
## $ plant.stand     <ord> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ precip          <ord> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0...
## $ temp            <ord> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 2...
## $ hail            <fct> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1...
## $ crop.hist       <fct> 1, 2, 1, 1, 2, 3, 2, 1, 3, 2, 1, 1, 1, 3, 1, 3...
## $ area.dam        <fct> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 2, 3, 3, 3...
## $ sever           <fct> 1, 2, 2, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1...
## $ seed.tmt        <fct> 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1...
## $ germ            <ord> 0, 1, 2, 1, 2, 1, 0, 2, 1, 2, 0, 1, 0, 0, 1, 2...
## $ plant.growth    <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ leaves          <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ leaf.halo       <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ leaf.marg       <fct> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2...
## $ leaf.size       <ord> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2...
## $ leaf.shread     <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ leaf.malf       <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ leaf.mild       <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ stem            <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ lodging         <fct> 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0...
## $ stem.cankers    <fct> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0...
## $ canker.lesion   <fct> 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 3, 3, 3, 3, 3, 3...
## $ fruiting.bodies <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0...
## $ ext.decay       <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0...
## $ mycelium        <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ int.discolor    <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2...
## $ sclerotia       <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1...
## $ fruit.pods      <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ fruit.spots     <fct> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
## $ seed            <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ mold.growth     <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ seed.discolor   <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ seed.size       <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ shriveling      <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ roots           <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
```

```
#Shape
dim(Soybean)
```

```
## [1] 683  36
```

```
#Stats
describe(Soybean)
```

```
##                  vars   n mean   sd median trimmed  mad min max range
## Class*              1 683 9.30 5.51      8    9.18 7.41   1  19    18
## date*               2 682 4.55 1.69      5    4.62 1.48   1   7     6
## plant.stand*        3 647 1.45 0.50      1    1.44 0.00   1   2     1
```

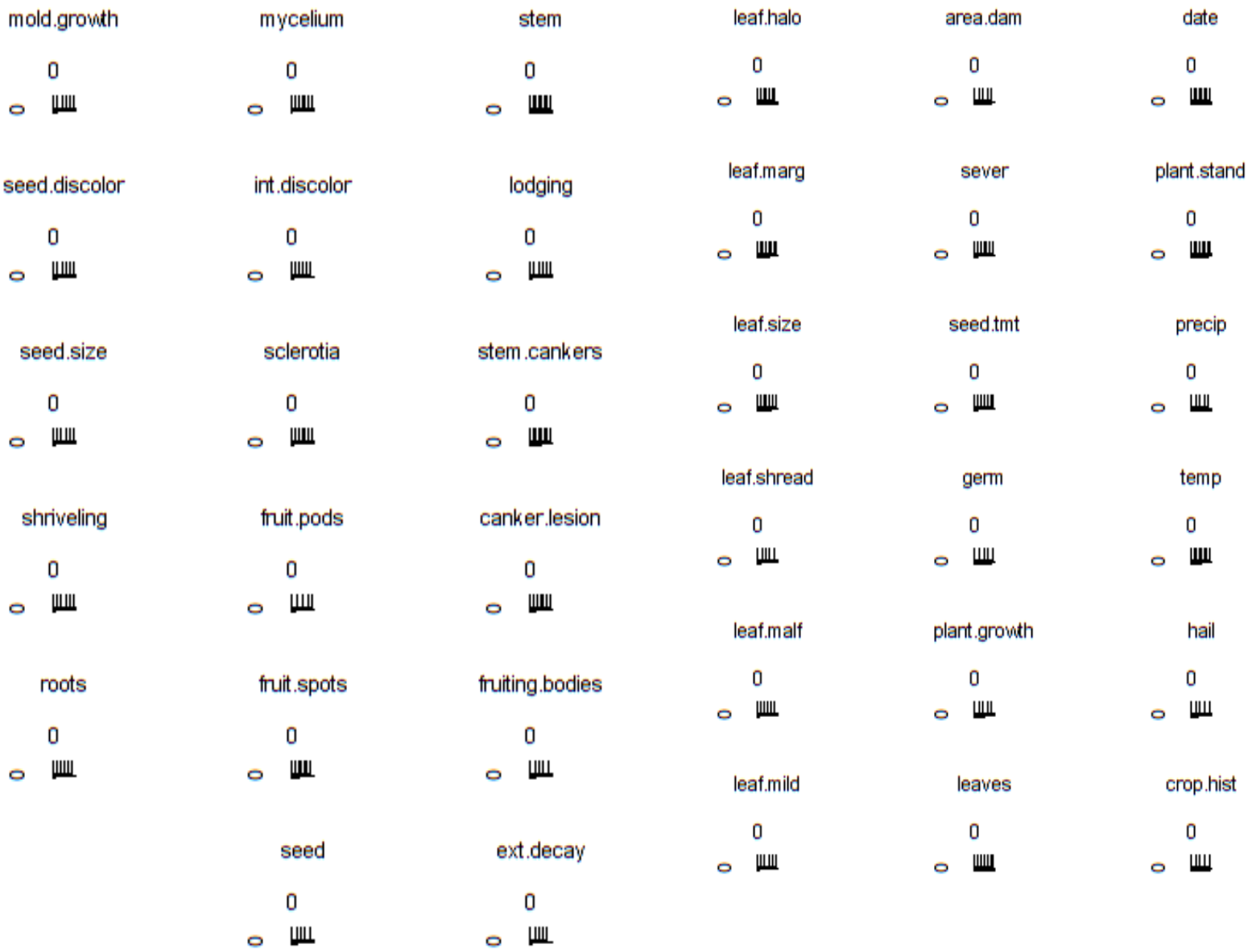
```
## precip*             4 645 2.60 0.69      3    2.74 0.00   1   3     2
## temp*               5 653 2.18 0.63      2    2.23 0.00   1   3     2
## hail*               6 562 1.23 0.42      1    1.16 0.00   1   2     1
## crop.hist*          7 667 2.88 0.98      3    2.98 1.48   1   4     3
## area.dam*           8 682 2.58 1.07      2    2.60 1.48   1   4     3
## sever*              9 562 1.73 0.60      2    1.69 0.00   1   3     2
## seed.tmt*          10 562 1.52 0.61      1    1.45 0.00   1   3     2
## germ*              11 571 2.05 0.79      2    2.06 1.48   1   3     2
## plant.growth*      12 667 1.34 0.47      1    1.30 0.00   1   2     1
## leaves*            13 683 1.89 0.32      2    1.98 0.00   1   2     1
## leaf.halo*         14 599 2.20 0.95      3    2.25 0.00   1   3     2
## leaf.marg*         15 599 1.77 0.96      1    1.72 0.00   1   3     2
## leaf.size*         16 599 2.28 0.61      2    2.34 0.00   1   3     2
## leaf.shread*       17 583 1.16 0.37      1    1.08 0.00   1   2     1
## leaf.malf*         18 599 1.08 0.26      1    1.00 0.00   1   2     1
## leaf.mild*         19 575 1.10 0.40      1    1.00 0.00   1   3     2
## stem*              20 667 1.56 0.50      2    1.57 0.00   1   2     1
## lodging*           21 562 1.07 0.26      1    1.00 0.00   1   2     1
## stem.cankers*      22 645 2.06 1.35      1    1.95 0.00   1   4     3
## canker.lesion*     23 645 1.98 1.08      2    1.85 1.48   1   4     3
## fruiting.bodies*   24 577 1.18 0.38      1    1.10 0.00   1   2     1
## ext.decay*         25 645 1.25 0.48      1    1.16 0.00   1   3     2
## mycelium*          26 645 1.01 0.10      1    1.00 0.00   1   2     1
## int.discolor*      27 645 1.13 0.42      1    1.00 0.00   1   3     2
## sclerotia*         28 645 1.03 0.17      1    1.00 0.00   1   2     1
## fruit.pods*        29 599 1.50 0.88      1    1.28 0.00   1   4     3
## fruit.spots*       30 577 1.85 1.17      1    1.69 0.00   1   4     3
## seed*              31 591 1.19 0.40      1    1.12 0.00   1   2     1
## mold.growth*       32 591 1.11 0.32      1    1.02 0.00   1   2     1
## seed.discolor*     33 577 1.11 0.31      1    1.02 0.00   1   2     1
## seed.size*         34 591 1.10 0.30      1    1.00 0.00   1   2     1
## shriveling*        35 577 1.07 0.25      1    1.00 0.00   1   2     1
## roots*             36 652 1.18 0.44      1    1.07 0.00   1   3     2
##                    skew kurtosis   se
## Class*             0.11    -1.38 0.21
## date*             -0.30    -0.90 0.06
## plant.stand*       0.19    -1.97 0.02
## precip*           -1.42     0.55 0.03
## temp*             -0.16    -0.58 0.02
## hail*              1.31    -0.29 0.02
## crop.hist*        -0.40    -0.92 0.04
## area.dam*          0.02    -1.29 0.04
## sever*             0.17    -0.56 0.03
## seed.tmt*          0.74    -0.44 0.03
## germ*             -0.09    -1.40 0.03
## plant.growth*      0.68    -1.54 0.02
## leaves*           -2.44     3.98 0.01
## leaf.halo*        -0.41    -1.76 0.04
## leaf.marg*         0.46    -1.75 0.04
## leaf.size*        -0.25    -0.63 0.02
```

```
## leaf.shread*      1.80     1.26 0.02
## leaf.malf*        3.22     8.35 0.01
## leaf.mild*        3.95    14.68 0.02
## stem*            -0.23    -1.95 0.02
## lodging*          3.23     8.42 0.01
## stem.cankers*     0.61    -1.51 0.05
## canker.lesion*    0.51    -1.24 0.04
## fruiting.bodies*  1.66     0.75 0.02
## ext.decay*        1.70     1.98 0.02
## mycelium*        10.20   102.18 0.00
## int.discolor*     3.34    10.57 0.02
## sclerotia*        5.40    27.19 0.01
## fruit.pods*       1.84     2.41 0.04
## fruit.spots*      0.95    -0.76 0.05
## seed*             1.54     0.37 0.02
## mold.growth*      2.43     3.93 0.01
## seed.discolor*    2.47     4.12 0.01
## seed.size*        2.66     5.10 0.01
## shriveling*       3.49    10.21 0.01
## roots*            2.46     5.49 0.02
```

- There are 19 classes, only the first 15 of which have been used in prior work. There are 35 categorical attributes, some nominal and some ordered. The value “dna” means does not apply. The values for attributes are encoded numerically, with the first value encoded as “0,” the second as “1,” etc.

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

```
df <- Soybean[,2:36]
par(mfrow = c(3, 6))
for (i in 1:ncol(df)) {
  barplot(table(df[,i]),ylab = names(df[i]))
}
```


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

nearZeroVar in R for the categorical variables

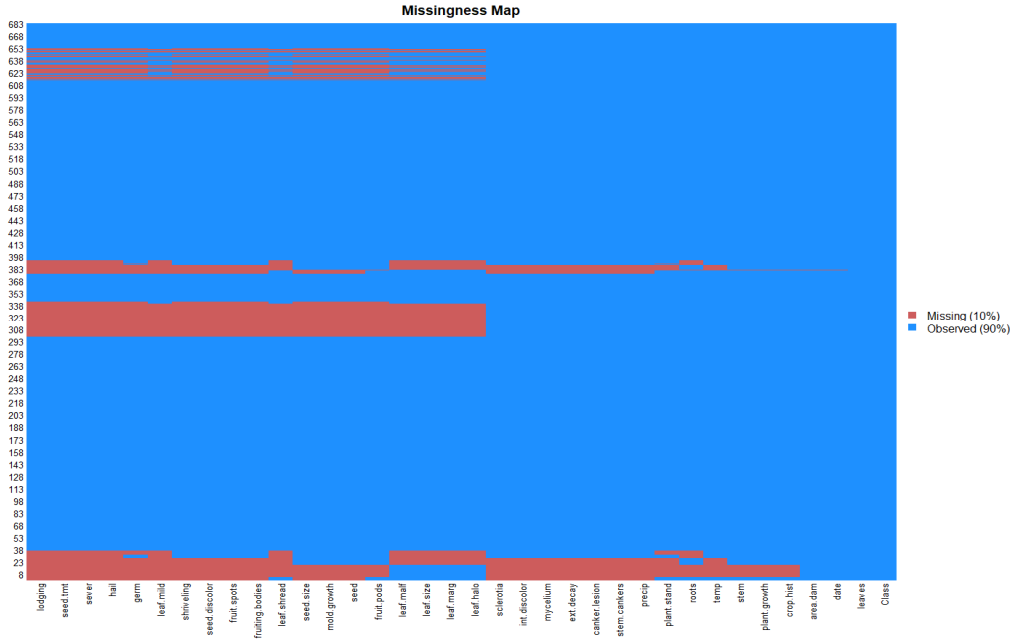
```
nearZeroVar(df,names = TRUE, saveMetrics=T)
```

```
##                   freqRatio percentUnique zeroVar   nzv
## date               1.137405     1.0248902   FALSE FALSE
## plant.stand        1.208191     0.2928258   FALSE FALSE
## precip             4.098214     0.4392387   FALSE FALSE
## temp               1.879397     0.4392387   FALSE FALSE
## hail               3.425197     0.2928258   FALSE FALSE
## crop.hist          1.004587     0.5856515   FALSE FALSE
## area.dam           1.213904     0.5856515   FALSE FALSE
## sever              1.651282     0.4392387   FALSE FALSE
## seed.tmt           1.373874     0.4392387   FALSE FALSE
## germ               1.103627     0.4392387   FALSE FALSE
## plant.growth       1.951327     0.2928258   FALSE FALSE
## leaves             7.870130     0.2928258   FALSE FALSE
## leaf.halo          1.547511     0.4392387   FALSE FALSE
## leaf.marg          1.615385     0.4392387   FALSE FALSE
## leaf.size          1.479638     0.4392387   FALSE FALSE
## leaf.shread        5.072917     0.2928258   FALSE FALSE
## leaf.malf         12.311111     0.2928258   FALSE FALSE
## leaf.mild         26.750000     0.4392387   FALSE  TRUE
## stem               1.253378     0.2928258   FALSE FALSE
## lodging           12.380952     0.2928258   FALSE FALSE
## stem.cankers       1.984293     0.5856515   FALSE FALSE
## canker.lesion      1.807910     0.5856515   FALSE FALSE
## fruiting.bodies    4.548077     0.2928258   FALSE FALSE
## ext.decay          3.681481     0.4392387   FALSE FALSE
## mycelium         106.500000     0.2928258   FALSE  TRUE
## int.discolor      13.204545     0.4392387   FALSE FALSE
## sclerotia         31.250000     0.2928258   FALSE  TRUE
## fruit.pods         3.130769     0.5856515   FALSE FALSE
## fruit.spots        3.450000     0.5856515   FALSE FALSE
## seed               4.139130     0.2928258   FALSE FALSE
## mold.growth        7.820896     0.2928258   FALSE FALSE
## seed.discolor      8.015625     0.2928258   FALSE FALSE
## seed.size          9.016949     0.2928258   FALSE FALSE
## shriveling        14.184211     0.2928258   FALSE FALSE
## roots              6.406977     0.4392387   FALSE FALSE
```

- There are few distributions degenerate . Specifically leaf.mild,mycelium and 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?

```
missmap(Soybean)
```




```
sort(colMeans(is.na(Soybean)),decreasing = T)
```

```
##            hail           sever        seed.tmt         lodging
##     0.177159590     0.177159590     0.177159590     0.177159590
##           germ       leaf.mild fruiting.bodies     fruit.spots
##     0.163982430     0.158125915     0.155197657     0.155197657
##   seed.discolor      shriveling     leaf.shread            seed
##     0.155197657     0.155197657     0.146412884     0.134699854
##     mold.growth       seed.size       leaf.halo       leaf.marg
##     0.134699854     0.134699854     0.122986823     0.122986823
##       leaf.size       leaf.malf      fruit.pods           precip
##     0.122986823     0.122986823     0.122986823     0.055636896
##    stem.cankers   canker.lesion       ext.decay        mycelium
##     0.055636896     0.055636896     0.055636896     0.055636896
##    int.discolor       sclerotia     plant.stand           roots
##     0.055636896     0.055636896     0.052708638     0.045387994
##           temp       crop.hist    plant.growth            stem
##     0.043923865     0.023426061     0.023426061     0.023426061
##            date        area.dam           Class          leaves
##     0.001464129     0.001464129     0.000000000     0.000000000
```

- Particularly hail, sever,seed.tmt,lodging, germ,leaf.mild fruiting.bodies, fruit.spots,seed.discolor,shriveling, leaf.shread, seed,mold.growth,seed.size,leaf.halo,are more likelly to be missing.

```
Soybean %>%
mutate(total = n()) %>%
group_by(Class) %>%
mutate(Missing = n(), Proportion=Missing/total) %>%
```

```
dplyr::select(Class, Missing, Proportion) %>%
unique() %>%
  arrange(-Proportion)
```

```
## # A tibble: 19 x 3
## # Groups:   Class [19]
##    Class                       Missing Proportion
##    <fct>                         <int>      <dbl>
##  1 brown-spot                       92     0.135
##  2 alternarialeaf-spot              91     0.133
##  3 frog-eye-leaf-spot               91     0.133
##  4 phytophthora-rot                 88     0.129
##  5 brown-stem-rot                   44     0.0644
##  6 anthracnose                      44     0.0644
##  7 diaporthe-stem-canker            20     0.0293
##  8 charcoal-rot                     20     0.0293
##  9 rhizoctonia-root-rot             20     0.0293
## 10 powdery-mildew                   20     0.0293
## 11 downy-mildew                     20     0.0293
## 12 bacterial-blight                 20     0.0293
## 13 bacterial-pustule                20     0.0293
## 14 purple-seed-stain                20     0.0293
## 15 phyllosticta-leaf-spot           20     0.0293
## 16 2-4-d-injury                     16     0.0234
## 17 diaporthe-pod-&-stem-blight      15     0.0220
## 18 cyst-nematode                    14     0.0205
## 19 herbicide-injury                  8     0.0117
```

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

- Drop the rows having missing values. After dropping, 562 observations remain.

```
Soybean_complete <- na.omit(Soybean)
head(Soybean_complete)
```

```
##                  Class date plant.stand precip temp hail crop.hist
## 1 diaporthe-stem-canker    6           0      2    1    0         1
## 2 diaporthe-stem-canker    4           0      2    1    0         2
## 3 diaporthe-stem-canker    3           0      2    1    0         1
## 4 diaporthe-stem-canker    3           0      2    1    0         1
## 5 diaporthe-stem-canker    6           0      2    1    0         2
## 6 diaporthe-stem-canker    5           0      2    1    0         3
##   area.dam sever seed.tmt germ plant.growth leaves leaf.halo leaf.marg
## 1        1     1        0    0            1      1         0         2
## 2        0     2        1    1            1      1         0         2
## 3        0     2        1    2            1      1         0         2
## 4        0     2        0    1            1      1         0         2
## 5        0     1        0    2            1      1         0         2
## 6        0     1        0    1            1      1         0         2
##   leaf.size leaf.shread leaf.malf leaf.mild stem lodging stem.cankers
## 1         2           0         0         0    1       1            3
```

```
## 2         2           0         0         0    1       0            3
## 3         2           0         0         0    1       0            3
## 4         2           0         0         0    1       0            3
## 5         2           0         0         0    1       0            3
## 6         2           0         0         0    1       0            3
##   canker.lesion fruiting.bodies ext.decay mycelium int.discolor sclerotia
## 1             1               1         1        0            0         0
## 2             1               1         1        0            0         0
## 3             0               1         1        0            0         0
## 4             0               1         1        0            0         0
## 5             1               1         1        0            0         0
## 6             0               1         1        0            0         0
##   fruit.pods fruit.spots seed mold.growth seed.discolor seed.size
## 1          0           4    0           0             0         0
## 2          0           4    0           0             0         0
## 3          0           4    0           0             0         0
## 4          0           4    0           0             0         0
## 5          0           4    0           0             0         0
## 6          0           4    0           0             0         0
##   shriveling roots
## 1          0     0
## 2          0     0
## 3          0     0
## 4          0     0
## 5          0     0
## 6          0     0
```

```
dim(Soybean_complete)
```

```
## [1] 562  36
```

Data 624: Week 4 Homework

Angrand, Burke, Deboch, Groysman, Karr

October 15, 2019

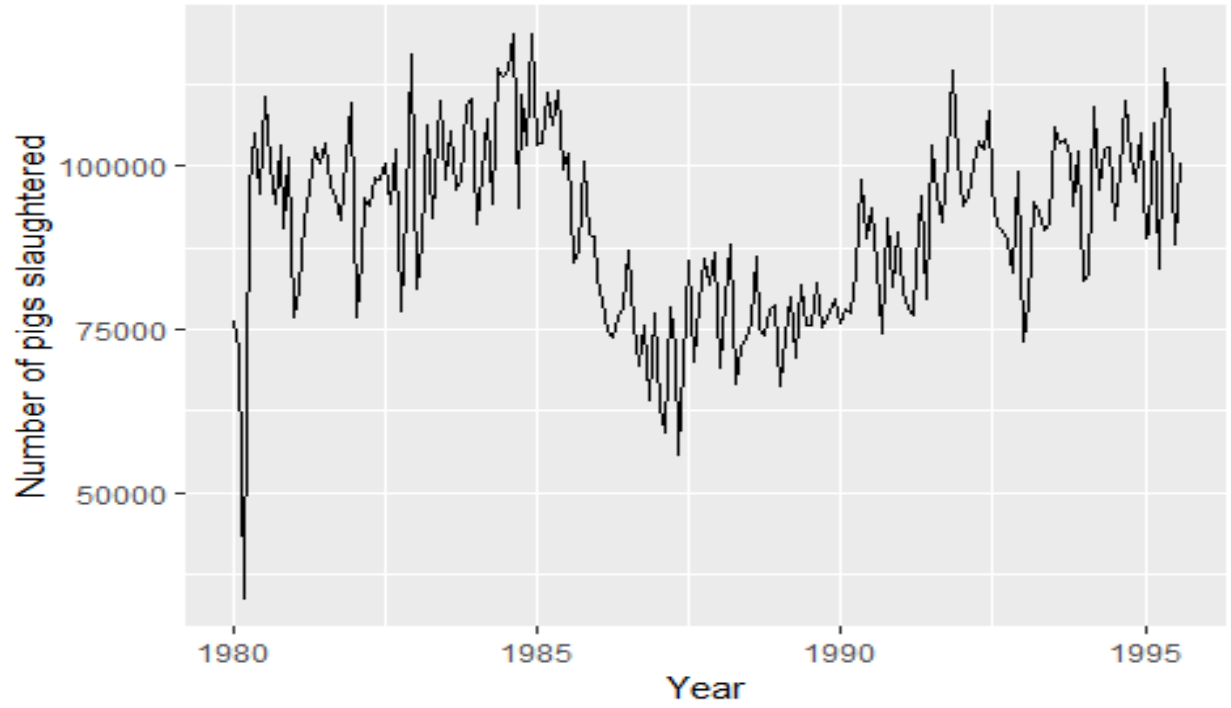
Week 4 Assignment

Chapter 7 HA 7.1, 7.3

*7.1 Consider the **pigs** series - the number of pigs slaughtered in Victoria each month.*

The pigs ts contains monthly total number of pigs slaughtered in Victoria, Australia (Jan 1980 to Aug 1995).

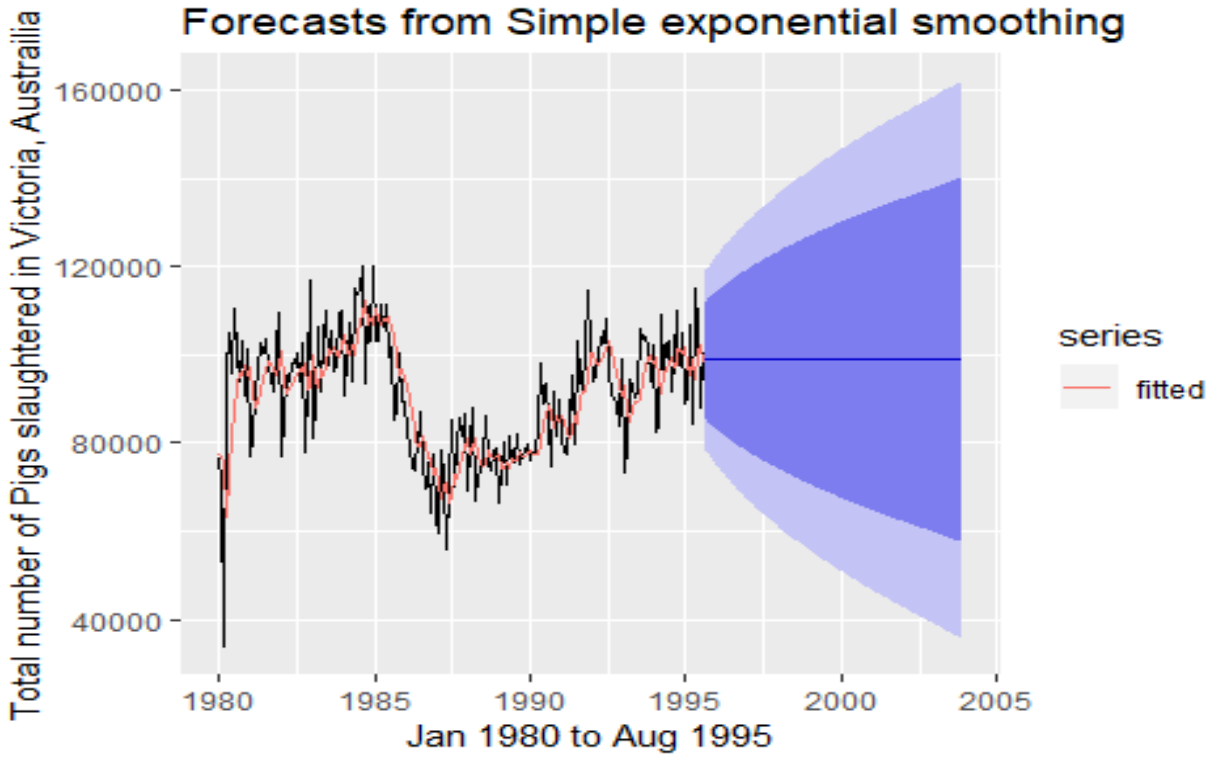
```
#sampling and shape of dataset, preliminary EDA
head(pigs)

##         Jan    Feb    Mar    Apr    May    Jun
## 1980  76378  71947  33873  96428 105084  95741

#autoplot the series
pigsdata<-window(pigs)
autoplot(pigsdata) +
  ylab("Number of pigs slaughtered") +xlab("Year")
```



a. Use the **ses()** function in R to find the optimal values of α and ℓ_0, and generate forecasts for the next four months.

```
#Estimate parameters
fc_pigs_ses<-ses(pigsdata, h=4)



#Get forecasted estimate parameters from model -(alpha and l)
round(fc_pigs_ses$model$par[1:2],4)

##      alpha          l
##     0.2971 77260.0561

#generate 4 months of forecasts
data.frame(fc_pigs_ses)

##          Point.Forecast    Lo.80    Hi.80    Lo.95    Hi.95
## Sep 1995       98816.41 85605.43 112027.4 78611.97 119020.8
## Oct 1995       98816.41 85034.52 112598.3 77738.83 119894.0
## Nov 1995       98816.41 84486.34 113146.5 76900.46 120732.4
## Dec 1995       98816.41 83958.37 113674.4 76092.99 121539.8

#Accuracy of one-step-ahead training errors
round(accuracy(fc_pigs_ses),2)

##                     ME    RMSE     MAE   MPE MAPE MASE ACF1
## Training set 385.87 10253.6 7961.38 -0.92 9.27  0.8 0.01

# see how SES model was fitted
fc_pigs_ses$model

## Simple exponential smoothing
##
## Call:
##  ses(y = pigsdata, h = 4)
##
##   Smoothing parameters:
##     alpha = 0.2971
##
##   Initial states:
##     l = 77260.0561
##
##   sigma:  10308.58
##
##      AIC     AICc      BIC
## 4462.955 4463.086 4472.665

# get 1st 4 months of forecasts
tsCV(pigs,ses,h=4)[1:4,]
```

```
##            h=1       h=2      h=3      h=4
## [1,]  -4431.00 -42505.00 20050.00 28706.00
## [2,] -42431.44  20123.56 28779.56 19436.56
## [3,]  62555.00  71211.00 61868.00 76774.00
## [4,]  28706.00  19363.00 34269.00 23953.00

#Plot (Note that forecast using ses doesn't have a trend component.)
fc_pigs_ses<-ses(pigs, h=100)
autoplot(fc_pigs_ses) +
    autolayer(fitted(fc_pigs_ses), series="fitted") +
    ylab("Total number of Pigs slaughtered in Victoria, Austrailia") + xlab("
Jan 1980 to Aug 1995")
```



b. Compute a 95% prediction interval for the first forecast using $\hat{y} \pm 1.96s$ is the standard deviation of the residuals. Compare your interval with the interval produced by R.

```
# 95% prediction interval for the first forecast

lower.upper <- data.frame( fc_pigs_ses$lower[1, "95%"], fc_pigs_ses$upper[1,
"95%"])
names(lower.upper)<- c("lower.limit", "upper.limit")
lower.upper

##     lower.limit upper.limit
## 95%    78611.97    119020.8
```

```r
# calculate standard deviation of residuals with and without model, s = 1027
3.69 vs s (estimated) 10308.58
s <- sd(fc_pigs_ses$residuals)
print(paste("Standard Deviation: ",round(s,2)))

## [1] "Standard Deviation:  10273.69"

# calculate 95% prediction interval with model
pred.interval.model<- data.frame(fc_pigs_ses$mean[1] - 1.96*s,fc_pigs_ses$mea
n[1] + 1.96*s)
names(pred.interval.model)<- c("Lower.Model", "Upper.Model")
pred.interval.model

##   Lower.Model Upper.Model
## 1    78679.97    118952.8

# calculate 95% prediction interval without model
pred.interval.womodel<- data.frame(mean(pigs)  - 1.96*s, mean(pigs) + 1.96*s)
names(pred.interval.womodel)<- c("Lower.NOModel", "Upper.NOModel")
pred.interval.womodel

##   Lower.NOModel Upper.NOModel
## 1         70504      110776.9
```

- R gives an interval of [78611.97, 119020.8] and by computing the standard deviation of the residuals we got [78679.97, 118952.8]. Both are really close.

7.3 Modify your function from the previous exercise to return the sum of squared errors rather than the forecast of the next observation. Then use the ***optim()*** *function to find the optimal values of* α *and* ℓ_0*. Do you get the same values as the* ***ses()*** *function?*

- my_ses_func returns an l value that is ~.01% different then the ses function.

```
##      alpha          l
##     0.2971 77269.3253

##      alpha          l
##     0.2971 77260.0561
```

Data 624: Week 5 Homework

Angrand, Burke, Deboch, Groysman, Karr

October 15, 2019

Week 5 Assignment

Chapter 7 HA 7.5, 7.6 and 7.10

7.5 Forecast the next four days of paperback and hardcover books using the data set books which contains the same store daily sales data for paperback and hardcover books.

The problem set uses the books timeseries which contains 30 observations of same-store hardback and paperback sales.

```
#sampling and shape of dataset, preliminary EDA
head(books)
```

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
##   Paperback Hardcover
## 1       199       139
## 2       172       128
## 3       111       172
## 4       209       139
## 5       161       191
## 6       119       168
```

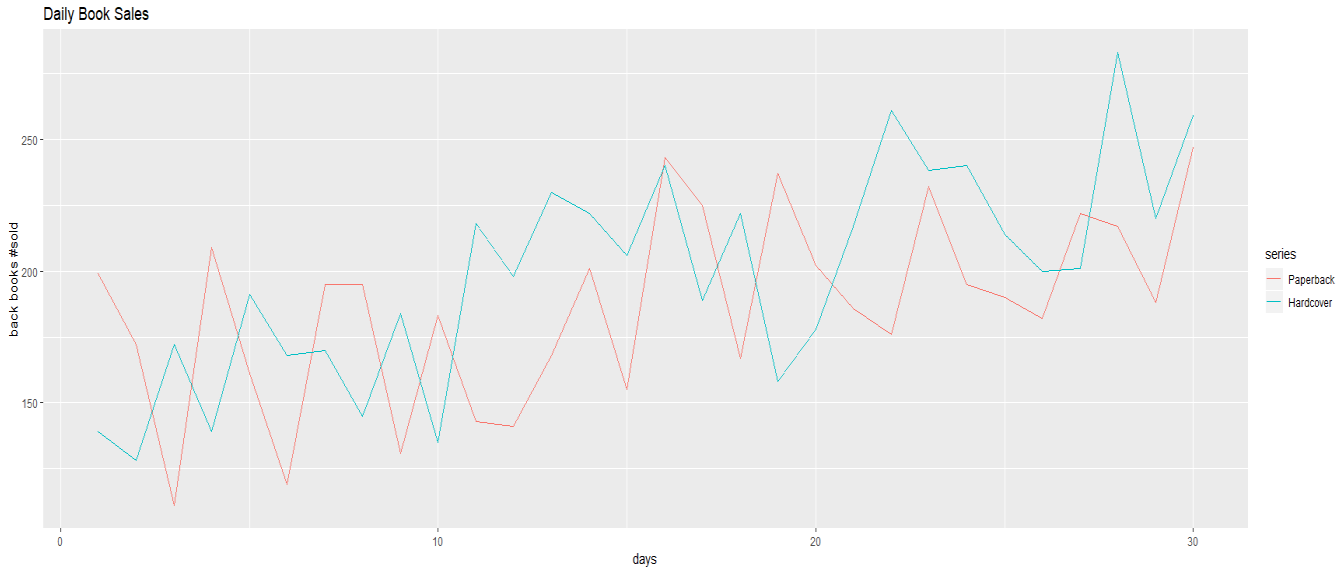
```
dim(books)
```

```
## [1] 30  2
```

a. Plot the series and discuss its main features.

- This shows the 30 day timeseries for both types of books by number sold. Visually, the main features of the data are the positive trend and a seasonality pattern of approximately every three days (up/down).

```
data(books)
autoplot(books) +
  ylab("back books #sold") + xlab("days")+ ggtitle("Daily Book Sales")
```

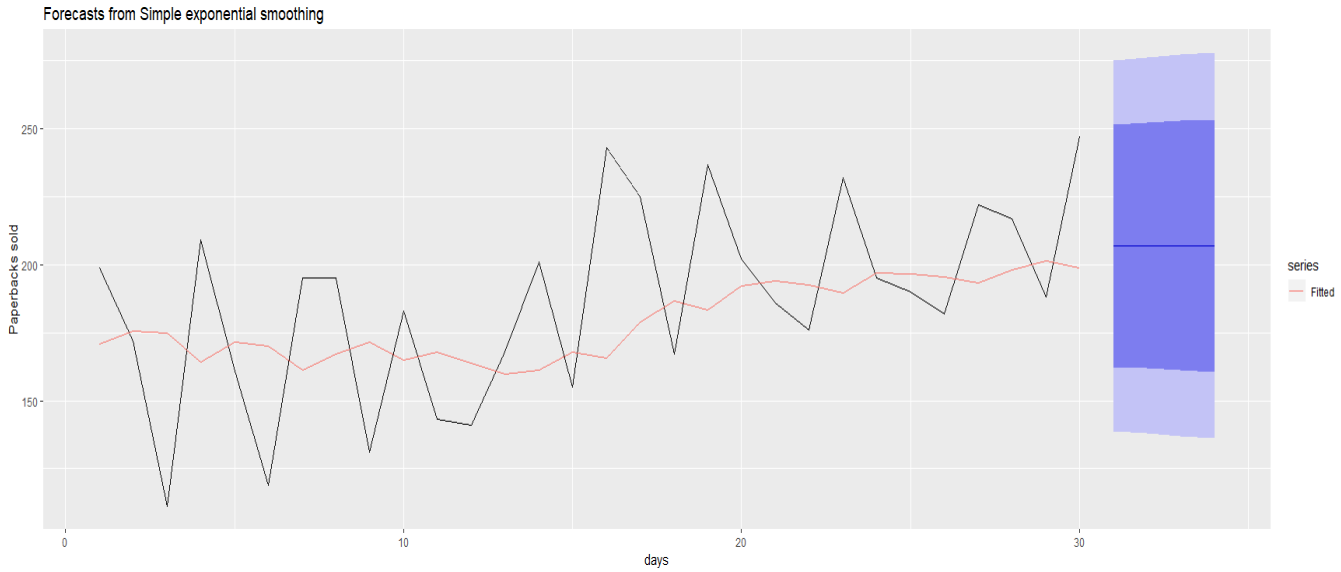


b. Uses the ses() function to forecast each series and plot the forecasts.

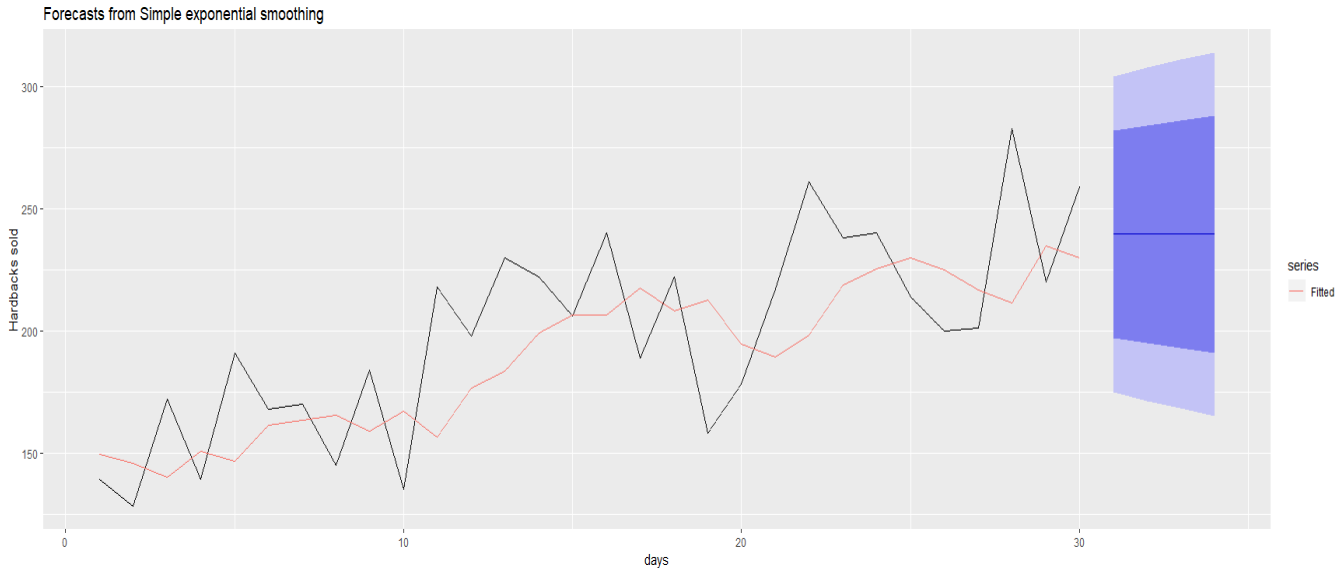
- Separate the paperback and hardback sales into distinct timeseries (books[,1], books[,2])
- Run the Simple Exponential Smoothing (SES) function against both hardbacks and paperbacks for h=4 or 4 days period of forecasting
  - The ses() function returns forecasts and metadata for exponential smoothing forecasts applied each timeseries
- Rounded the training errors and plotted the series

```
#1) create distinct timeseries
paperback_books_ts<-books[,1]
hardback_books_ts<-books[,2]

#2) Estimate parameters
fc_pb_ses<-ses(paperback_books_ts, h=4)
fc_hb_ses<-ses(hardback_books_ts, h=4)

data.frame(fc_pb_ses)

##    Point.Forecast    Lo.80    Hi.80    Lo.95    Hi.95
## 31       207.1097 162.4882 251.7311 138.8670 275.3523
## 32       207.1097 161.8589 252.3604 137.9046 276.3147
## 33       207.1097 161.2382 252.9811 136.9554 277.2639
## 34       207.1097 160.6259 253.5935 136.0188 278.2005

data.frame(fc_hb_ses)

##    Point.Forecast    Lo.80    Hi.80    Lo.95    Hi.95
## 31       239.5601 197.2026 281.9176 174.7799 304.3403
## 32       239.5601 194.9788 284.1414 171.3788 307.7414
## 33       239.5601 192.8607 286.2595 168.1396 310.9806
## 34       239.5601 190.8347 288.2855 165.0410 314.0792
```

```
autoplot(fc_pb_ses) +
  autolayer(fitted(fc_pb_ses),series="Fitted") +
  ylab("Paperbacks sold") + xlab("days")
```




```
autoplot(fc_hb_ses) +
  autolayer(fitted(fc_hb_ses),series="Fitted") +
  ylab("Hardbacks sold") + xlab("days")
```



c. Compute the RMSE values for the training data in each case.

- The accuracy function returns a range of summary measures of the forecast accuracy including Root Mean Square Error. for each training data timeseries including RMSE. For paperbacks books RMSE=33.64 and for hardback books RMSE=31.93.

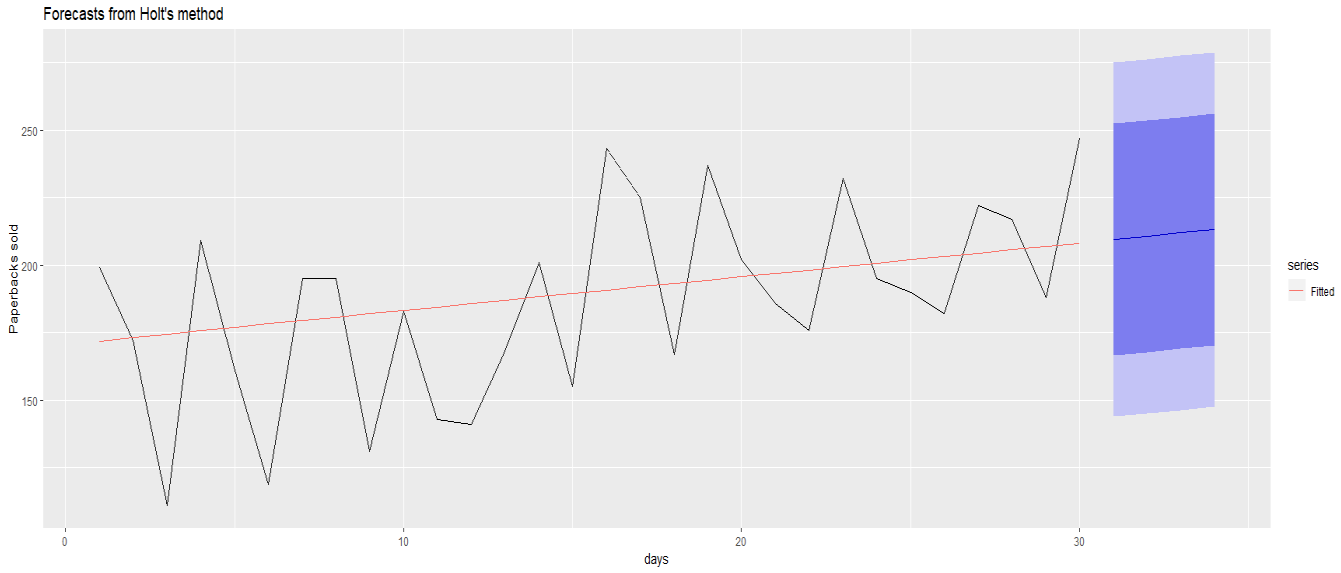
```
#3) Accuracy of one-step-ahead training errors paperback
round(accuracy(fc_pb_ses),2)

##                    ME  RMSE   MAE  MPE  MAPE MASE  ACF1
## Training set 7.18 33.64 27.84 0.47 15.58  0.7 -0.21

#4) Accuracy of one-step-ahead training errors hardback
round(accuracy(fc_hb_ses),2)

##                    ME  RMSE   MAE  MPE  MAPE MASE  ACF1
## Training set 9.17 31.93 26.77 2.64 13.39  0.8 -0.14
```

7.6 (a continuation of problem 7.5)

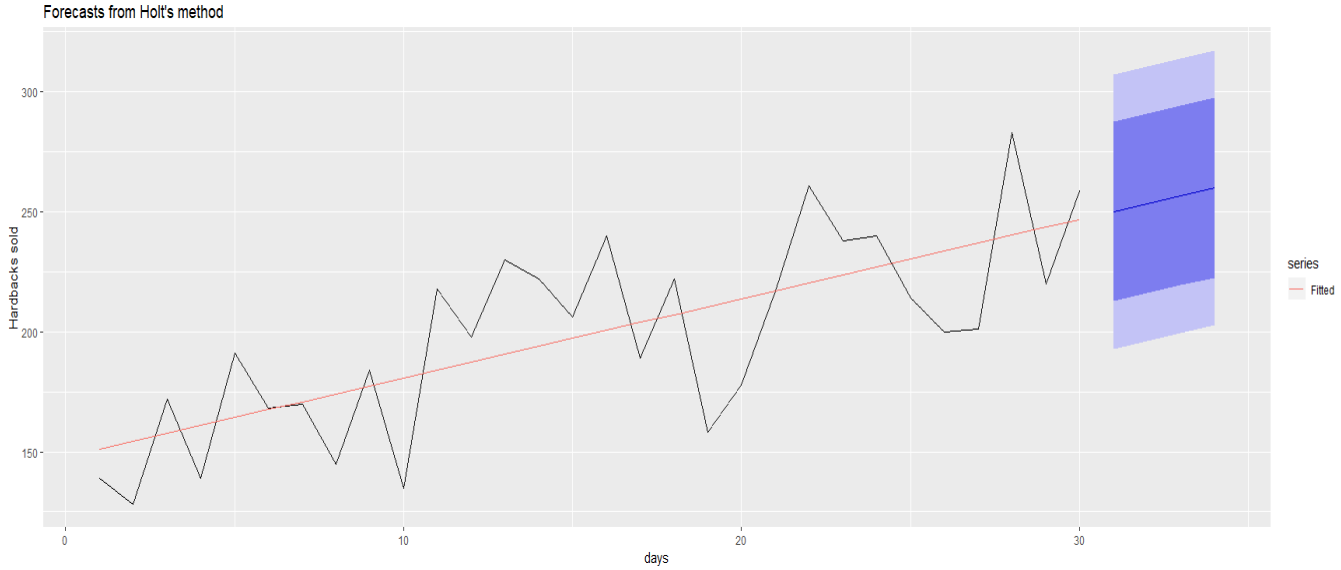
a. Now apply Holt's linear method to the paperback and hardback series and compute four-day forecasts in each case.

- The Holt method uses h=4 (4 days) as input parameters projecting a 4 day forecast.

```
# repeat with holt() functiobn, same params as ses()
fc_pb_holt<-holt(paperback_books_ts, h=4)
fc_hb_holt<-holt(hardback_books_ts, h=4)

data.frame(fc_pb_holt)

##    Point.Forecast    Lo.80    Hi.80    Lo.95    Hi.95
## 31       209.4668 166.6035 252.3301 143.9130 275.0205
## 32       210.7177 167.8544 253.5811 145.1640 276.2715
## 33       211.9687 169.1054 254.8320 146.4149 277.5225
## 34       213.2197 170.3564 256.0830 147.6659 278.7735

data.frame(fc_hb_holt)

##    Point.Forecast    Lo.80    Hi.80    Lo.95    Hi.95
## 31       250.1739 212.7390 287.6087 192.9222 307.4256
## 32       253.4765 216.0416 290.9113 196.2248 310.7282
## 33       256.7791 219.3442 294.2140 199.5274 314.0308
## 34       260.0817 222.6468 297.5166 202.8300 317.3334

autoplot(fc_pb_holt) +
  autolayer(fitted(fc_pb_holt),series="Fitted") +
  ylab("Paperbacks sold") + xlab("days")
```




```
autoplot(fc_hb_holt) +
  autolayer(fitted(fc_hb_holt),series="Fitted") +
  ylab("Hardbacks sold") + xlab("days")
```



b. Compare the RMSE measures of Holt's method for the two series to those of simple exponential smoothing in the previous question. (Remember that Holt's method is using one more parameter than SES.) Discuss the merits of the two forecasting methods for these data sets.

```
#HOLT Accuracy of one-step-ahead training errors paperback
round(accuracy(fc_pb_holt),2)

##                    ME   RMSE    MAE    MPE   MAPE MASE   ACF1
## Training set -3.72 31.14 26.18 -5.51 15.58 0.66 -0.18
```

```
#SES Accuracy of one-step-ahead training errors paperback
round(accuracy(fc_pb_ses),2)

##                ME  RMSE   MAE  MPE  MAPE MASE  ACF1
## Training set 7.18 33.64 27.84 0.47 15.58  0.7 -0.21

#HOLT Accuracy of one-step-ahead training errors hardback
round(accuracy(fc_hb_holt),2)

##                 ME  RMSE   MAE   MPE  MAPE MASE  ACF1
## Training set -0.14 27.19 23.16 -2.11 12.16 0.69 -0.03

#SES Accuracy of one-step-ahead training errors hardback
round(accuracy(fc_hb_ses),2)

##                ME  RMSE   MAE  MPE  MAPE MASE  ACF1
## Training set 9.17 31.93 26.77 2.64 13.39  0.8 -0.14
```

- The holt series yields an RMSE of 31.14 for paperback books and 27.19 for hardback books. The ses series yields an RMSE of 33.64 for paperback books and 31.93 for hardback books. The smaller value of RMSE for hardback books indicates a better fit for both series using the Holt method. Using both methods hardbacks yield a lower RMSE which indicates a better fit then paperbacks. It appears that by extending the SES method with a trend equation for forecasting, the overall fit of the Holt method is an improvement over the SES method, presumably where the data being examined has a clear trend.

c. Compare the forecasts for the two series using both methods. Which do you think is best?

- The RMSE is smaller for the paperback book data and therefore better fitted using the holt model. The trend line from the training data to the predictions does seem to extrapolate more accurately as well.

d. Calculate a 95% prediction interval for the first forecast for each series, using RMSE values and assuming normal errors. Compare your intervals with those produced using ses and holt.

- Note the 95% confidence intervals for the ses method have a greater range then that of the holt method for these predictions.The Holt 95% prediction interval for the 1st forecast of the paperback timeseries is between 149.69 and 265.45 and the Holt 95% prediction interval for the 1st forecast of the hardback timeseries is between 197.78 and 293.05. This compares with the ses 95% prediction interval for the 1st forecast of the paperback timeseries is between 135.96 and 277.16 and the ses 95% prediction interval for the 1st forecast of the hardback timeseries is between 197.78 and 293.05.

```
head(data.frame(holt(paperback_books_ts,bootstrap=TRUE)),1)

##    Point.Forecast    Lo.80   Hi.80    Lo.95    Hi.95
## 31       209.4668 162.1512 251.972 149.6907 265.4503

head(data.frame(ses(paperback_books_ts,bootstrap=TRUE)),1)
```

```
##    Point.Forecast    Lo.80    Hi.80   Lo.95    Hi.95
## 31       207.1097 159.1981 246.1439 135.961 277.1593
```

```
head(data.frame(holt(hardback_books_ts,bootstrap=TRUE)),1)
```

```
##    Point.Forecast    Lo.80    Hi.80    Lo.95    Hi.95
## 31       250.1739 214.3904 289.6273 197.7726 293.0467
```

```
head(data.frame(ses(hardback_books_ts,bootstrap=TRUE)),1)
```

```
##    Point.Forecast    Lo.80    Hi.80    Lo.95    Hi.95
## 31       239.5601 202.0118 291.8263 175.7541 301.8855
```

- The tsCV function computes the forecast errors obtained by applying forecast function to subsets of the time series paperback_books_ts and hardback_books_ts using a rolling forecast origin.

```
#get the tsCV for each model
e1<-tsCV(paperback_books_ts,ses,h=4)
e2<-tsCV(paperback_books_ts,holt,h=4)
e3<-tsCV(paperback_books_ts,holt,damped=TRUE,h=4)
e4<-tsCV(hardback_books_ts,ses,h=4)
e5<-tsCV(hardback_books_ts,holt,h=4)
e6<-tsCV(hardback_books_ts,holt, damped=TRUE,h=4)

#Compare MSE for paperbacks:
mse.pb <- data.frame(mean(e1^2,na.rm=TRUE),mean(e2^2,na.rm=TRUE),mean(e3^2,n
a.rm=TRUE))
names(mse.pb)<- c("mse.pb.ses", "mse.pb.holt", "mse.pb.holt.damped")
mse.pb
```

```
##   mse.pb.ses mse.pb.holt mse.pb.holt.damped
## 1   1475.265    2129.625           2152.224
```

```
#Compare MSE for hardbacks
mse.hb <- data.frame(mean(e4^2,na.rm=TRUE),mean(e5^2,na.rm=TRUE),mean(e6^2,n
a.rm=TRUE))
names(mse.hb)<- c("mse.hb.ses", "mse.hb.holt", "mse.hb.holt.damped")
mse.hb
```

```
##   mse.hb.ses mse.hb.holt mse.hb.holt.damped
## 1   1500.838    1554.578           1589.293
```

- display Holt model summary

```
fc_pb_holt<-holt(paperback_books_ts,h=4)
fc_hb_holt<-holt(hardback_books_ts,h=4)

fc_pb_holt[["model"]]
```

```
## Holt's method
##
## Call:
```

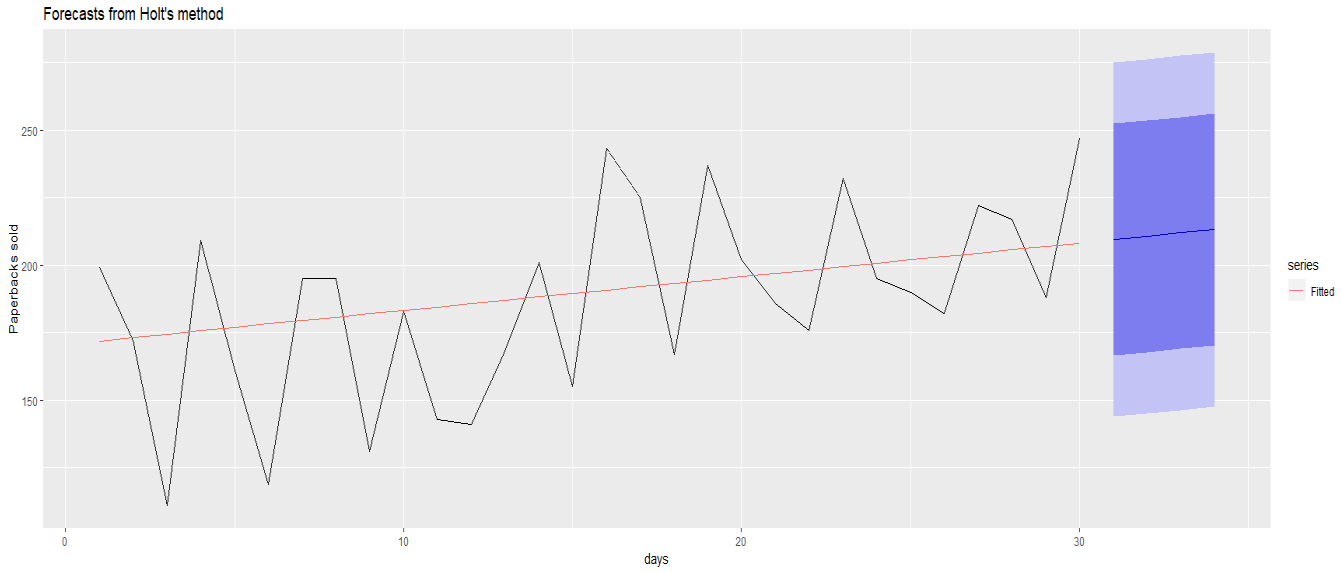
```
##  holt(y = paperback_books_ts, h = 4)
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##
##   Initial states:
##     l = 170.699
##     b = 1.2621
##
##   sigma:  33.4464
##
##      AIC     AICc      BIC
## 318.3396 320.8396 325.3456
```

```
fc_hb_holt[["model"]]
```

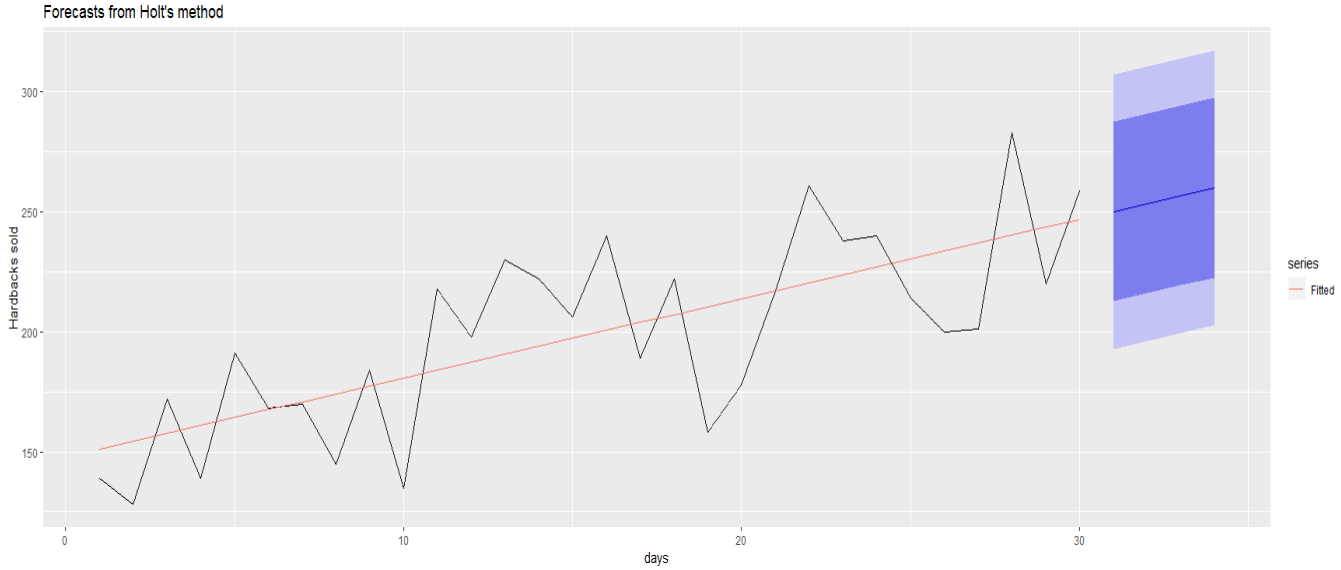
```
## Holt's method
##
## Call:
##  holt(y = hardback_books_ts, h = 4)
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##
##   Initial states:
##     l = 147.7935
##     b = 3.303
##
##   sigma:  29.2106
##
##      AIC     AICc      BIC
## 310.2148 312.7148 317.2208
```

- Plot Holt model summary forecasts

```
autoplot(fc_pb_holt) +
  autolayer(fitted(fc_pb_holt),series="Fitted") +
  ylab("Paperbacks sold") + xlab("days")
```




```
autoplot(fc_hb_holt) +
  autolayer(fitted(fc_hb_holt),series="Fitted") +
  ylab("Hardbacks sold") + xlab("days")
```




```
round(accuracy(fc_pb_holt),2)

##                    ME   RMSE    MAE    MPE   MAPE MASE   ACF1
## Training set -3.72 31.14 26.18 -5.51 15.58 0.66 -0.18

round(accuracy(fc_hb_holt),2)

##                    ME   RMSE    MAE    MPE   MAPE MASE   ACF1
## Training set -0.14 27.19 23.16 -2.11 12.16 0.69 -0.03

round(accuracy(fc_pb_holt,damped=TRUE),2)
```

```
##                    ME  RMSE   MAE   MPE  MAPE MASE  ACF1
## Training set -3.72 31.14 26.18 -5.51 15.58 0.66 -0.18
```

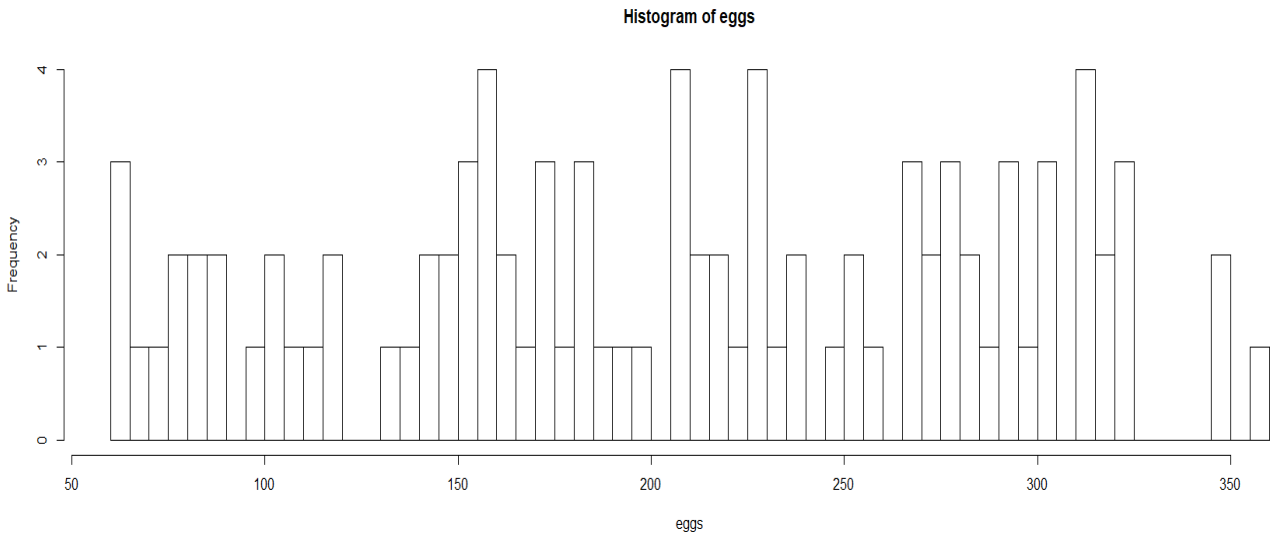
```
round(accuracy(fc_hb_holt,damped=TRUE),2)
```

```
##                    ME  RMSE   MAE   MPE  MAPE MASE  ACF1
## Training set -0.14 27.19 23.16 -2.11 12.16 0.69 -0.03
```

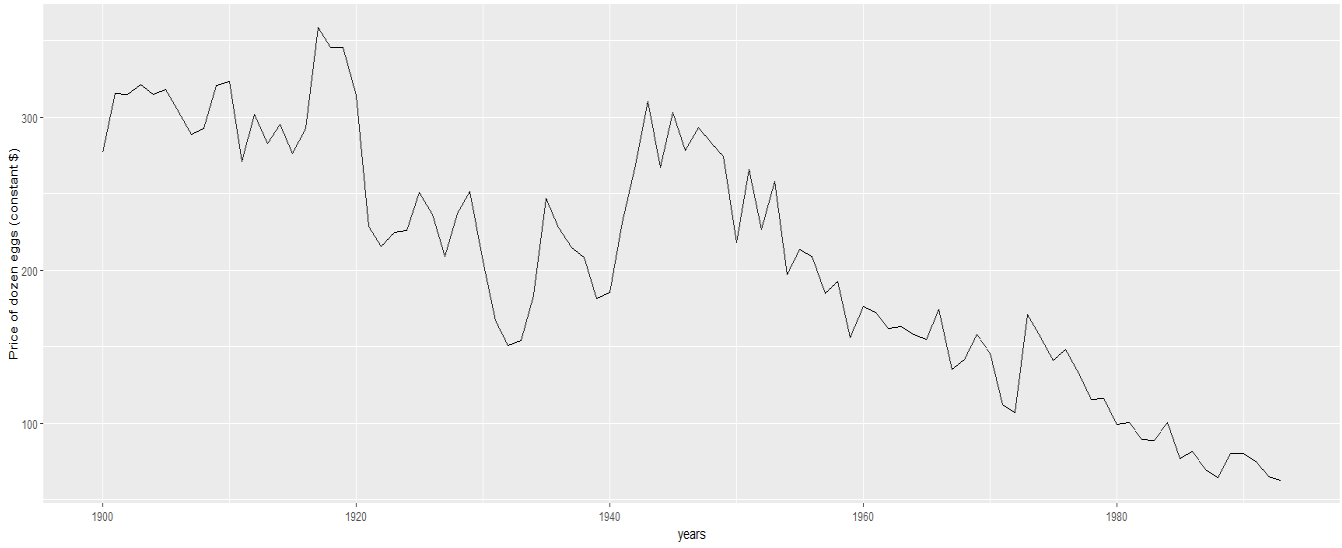
7.7 For this exercise use data set eggs, the price of a dozen eggs in the United States from 1900-1993.

a. Experiment with the various options in the holt() function to see how much the forecasts change with damped trend or Box-Cox transformation.

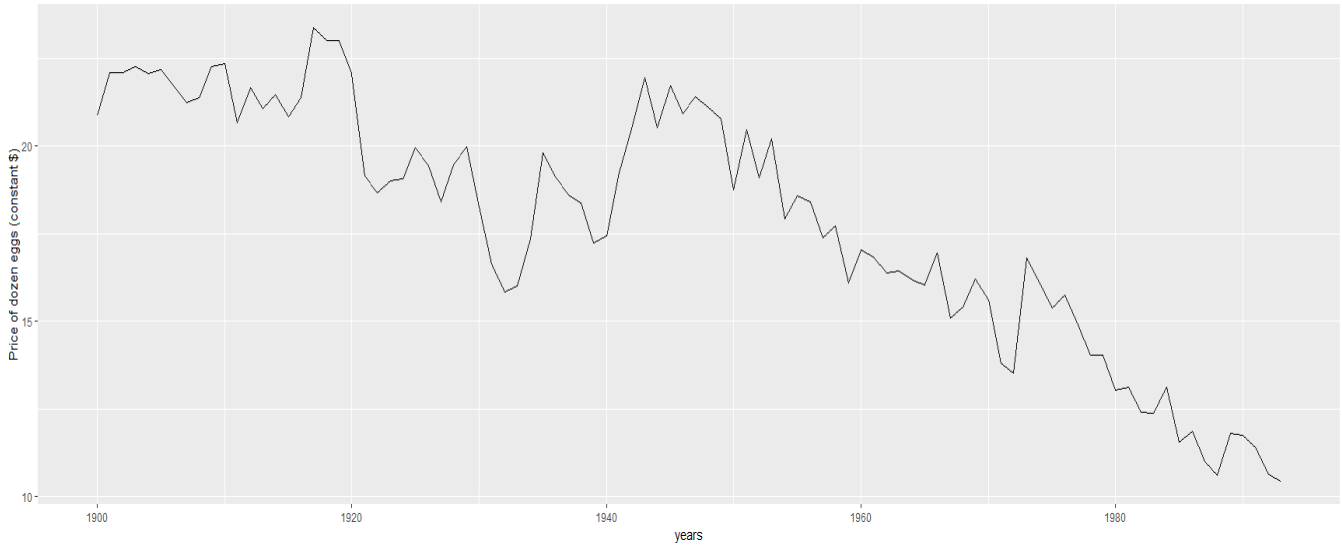
```
data(eggs)
hist(eggs,breaks=100)
```




```
#Price of dozen eggs in US, 1900 to 1993, in constant dollars.
autoplot(eggs) +
  ylab("Price of dozen eggs (constant $)") + xlab("years")
```




```
lambda<-BoxCox.lambda(eggs)
autoplot(BoxCox(eggs,lambda)) +
  ylab("Price of dozen eggs (constant $)") + xlab("years")
```



b. Try to develop an intuition of what each argument is doing to the forecasts. [Hint: use h=100 when calling holt() so you can clearly see the differences between the various options when plotting the forecasts.]

- MSE Evaluations

```
e1<-tsCV(eggs,ses,h=100)
e2<-tsCV(eggs,holt,h=100)
e3<-tsCV(eggs,holt,damped=TRUE,h=100)

#Compare MSE for eggs:
mse.eggs <- data.frame(mean(e1^2,na.rm=TRUE), mean(e2^2,na.rm=TRUE),mean(e3^
```

```
2,na.rm=TRUE))
names(mse.eggs)<- c("mse.ses", "mse.holt", "mse.holt.damped")
mse.eggs
```

```
##    mse.ses mse.holt mse.holt.damped
## 1 13200.06 77028.22        77608.15
```

- Model Components Summary

```
fc_eggs_ses<-ses(eggs,h=100)
fc_eggs_holt<-holt(eggs,h=100)
fc_eggs_holt_damped<-holt(eggs,damped=TRUE,h=100)

# Build Models
fc_eggs_ses[["model"]]
```

```
## Simple exponential smoothing
##
## Call:
##  ses(y = eggs, h = 100)
##
##   Smoothing parameters:
##     alpha = 0.8525
##
##   Initial states:
##     l = 282.4981
##
##   sigma:  26.8511
##
##      AIC     AICc      BIC
## 1049.626 1049.893 1057.256
```

```
fc_eggs_holt[["model"]]
```

```
## Holt's method
##
## Call:
##  holt(y = eggs, h = 100)
##
##   Smoothing parameters:
##     alpha = 0.8124
##     beta  = 1e-04
##
##   Initial states:
##     l = 314.7232
##     b = -2.7222
##
##   sigma:  27.1665
##
##      AIC     AICc      BIC
## 1053.755 1054.437 1066.472
```

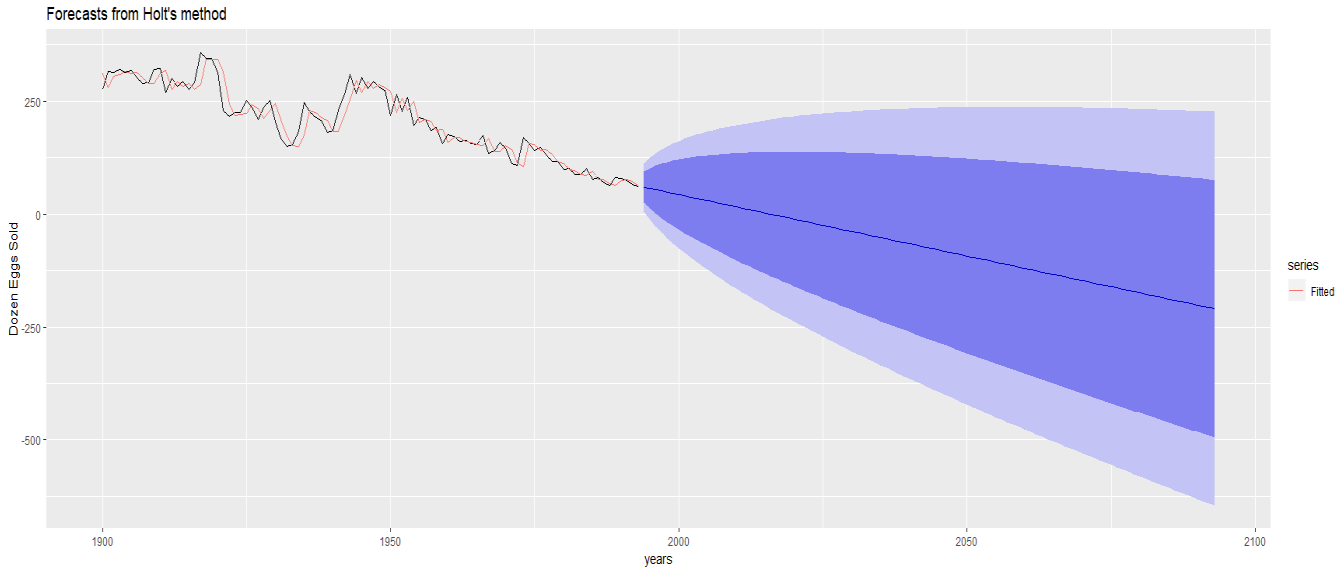
```
fc_eggs_holt_damped[["model"]]
```

```
## Damped Holt's method
##
## Call:
##  holt(y = eggs, h = 100, damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.8462
##     beta  = 0.004
##     phi   = 0.8
##
##   Initial states:
##     l = 276.9842
##     b = 4.9966
##
##   sigma:  27.2755
##
##      AIC     AICc      BIC
## 1055.458 1056.423 1070.718
```

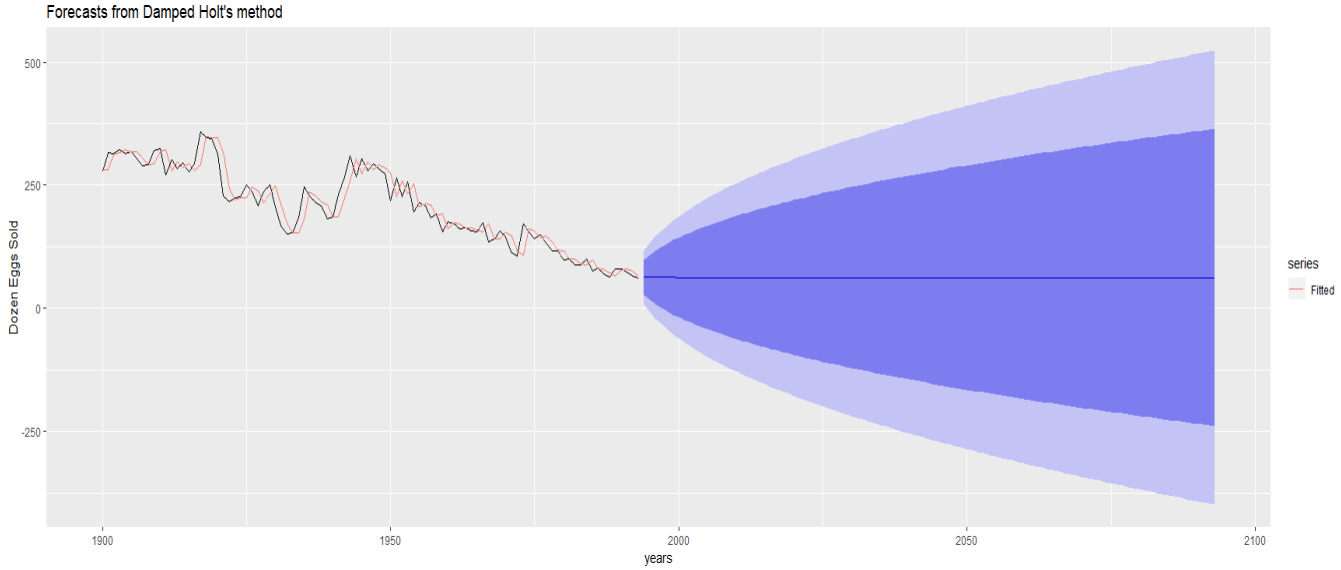
- Plot Forecasts

```
#plot forecasts predictions
autoplot(fc_eggs_ses) +
  autolayer(fitted(fc_eggs_ses),series="Fitted") +
  ylab("Dozen Eggs Sold") + xlab("years")
```

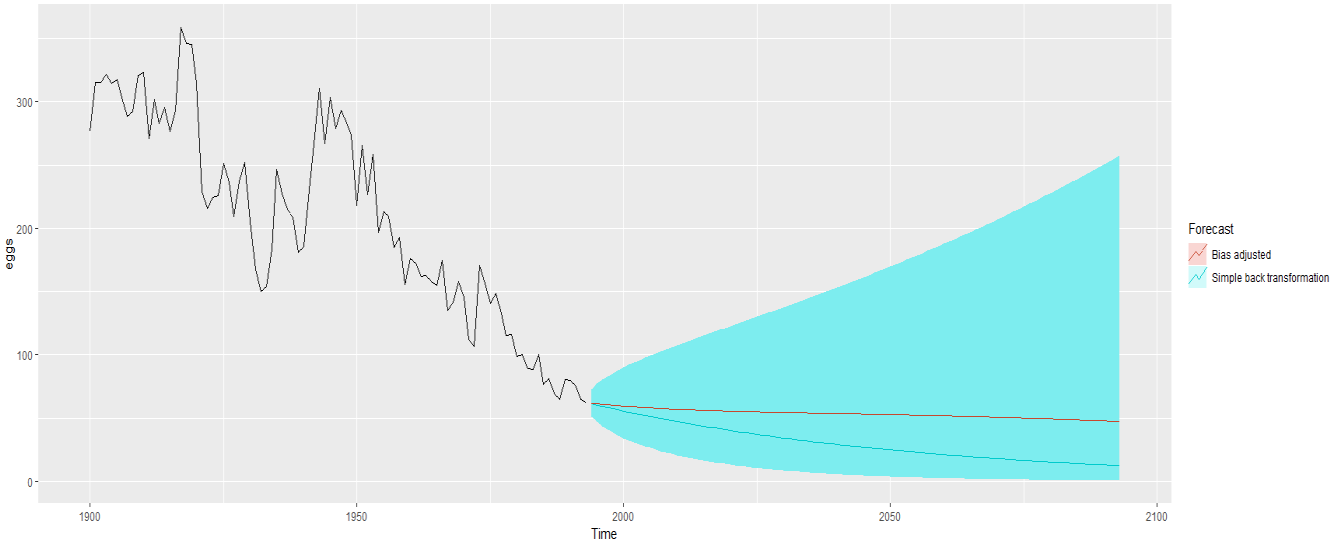

```
autoplot(fc_eggs_holt) +
  autolayer(fitted(fc_eggs_holt),series="Fitted") +
  ylab("Dozen Eggs Sold") + xlab("years")
```




```
autoplot(fc_eggs_holt_damped) +
  autolayer(fitted(fc_eggs_holt_damped),series="Fitted") +
  ylab("Dozen Eggs Sold") + xlab("years")
```




```
fc_eggs_BC<-rwf(eggs,drift=TRUE,lambda=0,h=100,level=80)
fc2_eggs_BC<-rwf(eggs,drift=TRUE,lambda=0,h=100,level=80,biasadj=TRUE)

autoplot(eggs) +
  autolayer(fc_eggs_BC,series="Simple back transformation") +
  autolayer(fc2_eggs_BC,series="Bias adjusted",PI=FALSE) +
  guides(colour=guide_legend(title="Forecast"))
```



c. Which model gives the best RMSE?

RMSE–the square root of a variance–can be interpreted as the standard deviation of the unexplained variance. The Holt damped RMSE is the lowest and best fit model in this case. Lower values indicate better fit. Based upon the diagrams, the Holt model and the simple back transformation of BoxCox appear to show the most accurate trendlines, but based on being the lowest RMSE values, the damped Holt model has the best fit.

Data 624: Week 6 Homework

Angrand, Burke, Deboch, Groysman, Karr

October 15, 2019

Week 6 Assignment

Chapter 8 HA 8.1, 8.2,8.6, 8.8

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

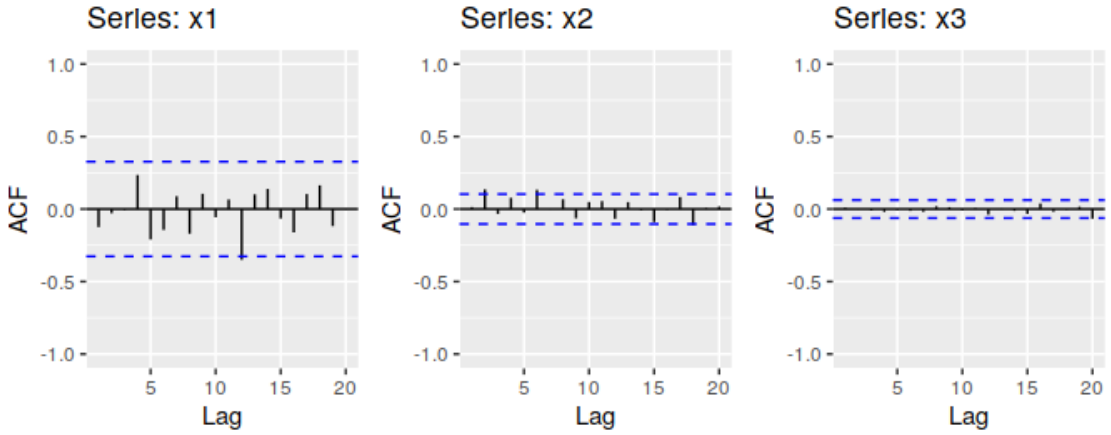


Fig. 8.31

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 that the data are white noise?

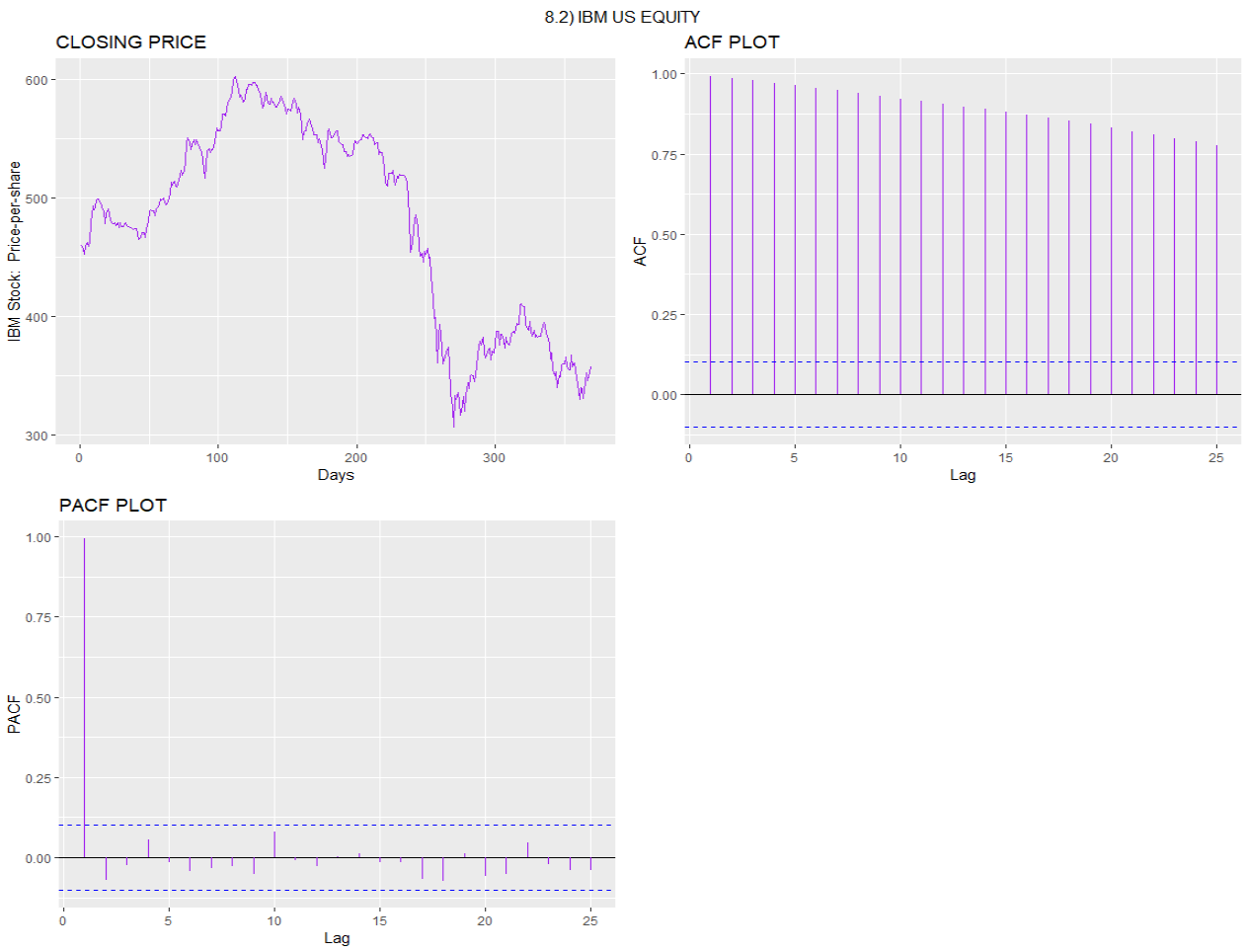
- Autocorrelation is the measurement of how correlated some lag in the time series is to the current lag. For ACF a previous point in time can either be directly or indirectly impacting the current value. For all three series objects, the autocorrelations, displayed with the ACF plots, are within the 95% error terms (essentially no different than zero). The confirms that all three series objects are essentially random and fit the definition of "White Noise" or ARIMA(0, 0, 0). The main difference between all three series objects is the number of samples. The smaller the number samples, the larger the ACF bars as the error terms are a function of the number of samples in a series ($\frac{\pm 1.96}{\sqrt{N}}$).

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 error terms are a function of the number of samples in a series, $\frac{\pm 1.96}{\sqrt{N}}$. The larger the number of samples, N, the closer the significance level/error terms are to zero. The error terms are larger for smaller data set, meaning higher series autocorrelation is needed to reject the null hypothesis that the series autocorrelation is zero. The larger the timeseries, the greater the denominator and the smaller the critical value.

*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.*

```
ibm.autoplot<- autoplot(ibmclose,col="purple") + ggtitle("CLOSING PRICE")+xla
b("Days") + ylab("IBM Stock:  Price-per-share")
ibm.acf <- ggAcf(ibmclose,col="purple") + ggtitle("ACF PLOT")
ibm.Pacf <- ggPacf(ibmclose,col="purple") + ggtitle("PACF PLOT")
grid.arrange(
  ibm.autoplot,
  ibm.acf,
  ibm.Pacf,
  nrow = 2,
  top = "8.2) IBM US EQUITY")
```



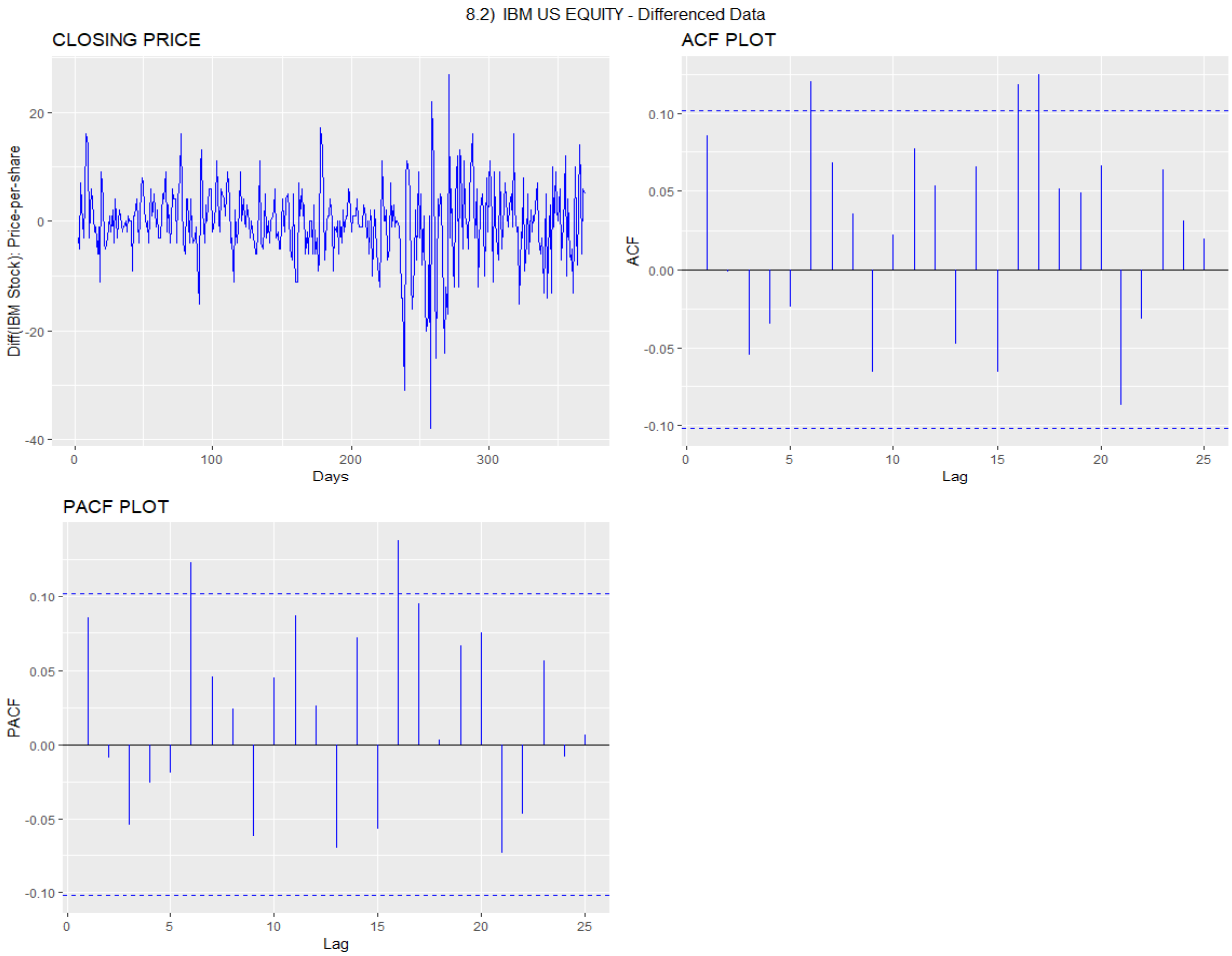
- **IBM Closing Price Plot** Non-stationary data has a mean that is either increasing or decreasing with time. Stationary data has a constant mean/variance the whole series. It does not matter when observed, it should look much the same at any point in time with no predictable patterns in the long-term. In the IBM US EQUITY closing price time series, there is clearly a decreasing trend over time, indicating that this time series needs to be differenced. Notice below in the "Differenced Data", the differenced data exhibits a roughly horizontal plot.

- **IBM Closing Price ACF Plot** ACF plots are great for testing of stationarity. For the IBM trended data, the lags are well outside the 95% "error terms" confidence interval meaning that they are meaningfully and serially correlated. The IBM ACF decreases slowly and its ACF lags are large and positive

- **IBM Closing Price PACF Plot** PACF plots are also great for testing of stationarity. For the IBM trended data, the first lag should be equal to the ACF first lag. The first PACF lag is close to one and all other PACF lags are close to zero, suggesting that the data is non-stationary.

```
ibm.autoplot<- autoplot(ibmclose%>% diff(1),col="blue" ) + ggtitle("CLOSING P
RICE")+xlab("Days") + ylab("Diff(IBM Stock): Price-per-share")
ibm.acf <- ggAcf(ibmclose%>% diff(1),col="blue" ) + ggtitle("ACF PLOT")
ibm.Pacf <- ggPacf(ibmclose%>% diff(1),col="blue" ) + ggtitle("PACF PLOT")
grid.arrange(
```

```
  ibm.autoplot,
  ibm.acf,
  ibm.Pacf,
  nrow = 2,
  top = "8.2)  IBM US EQUITY - Differenced Data")
```

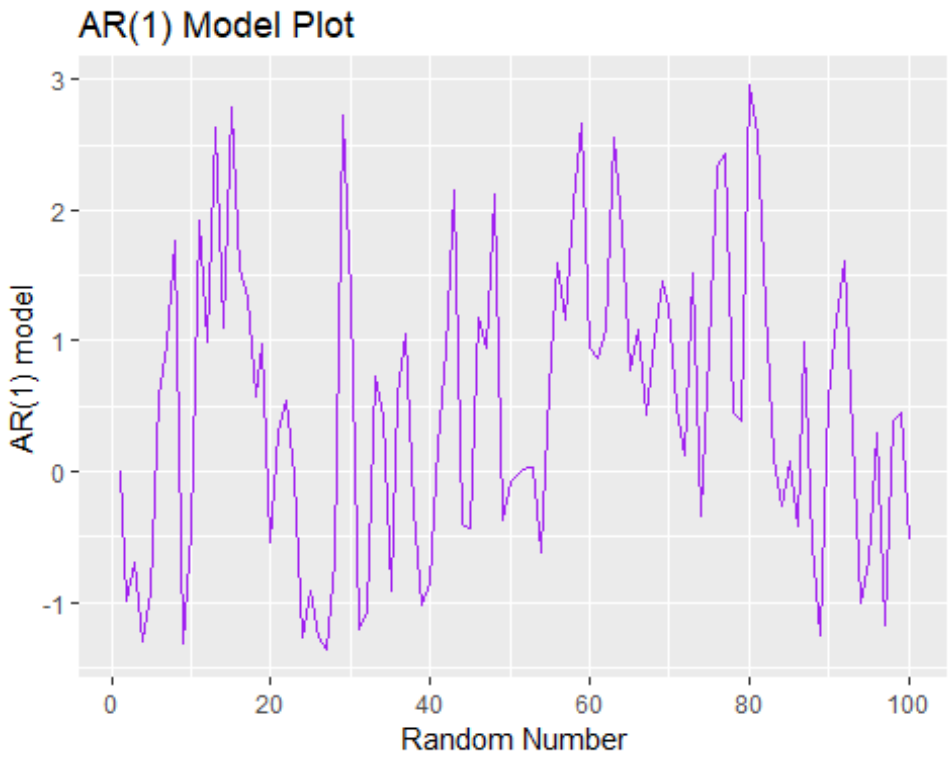


8.6 Use R to simulate and plot some data from simple ARIMA models.

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

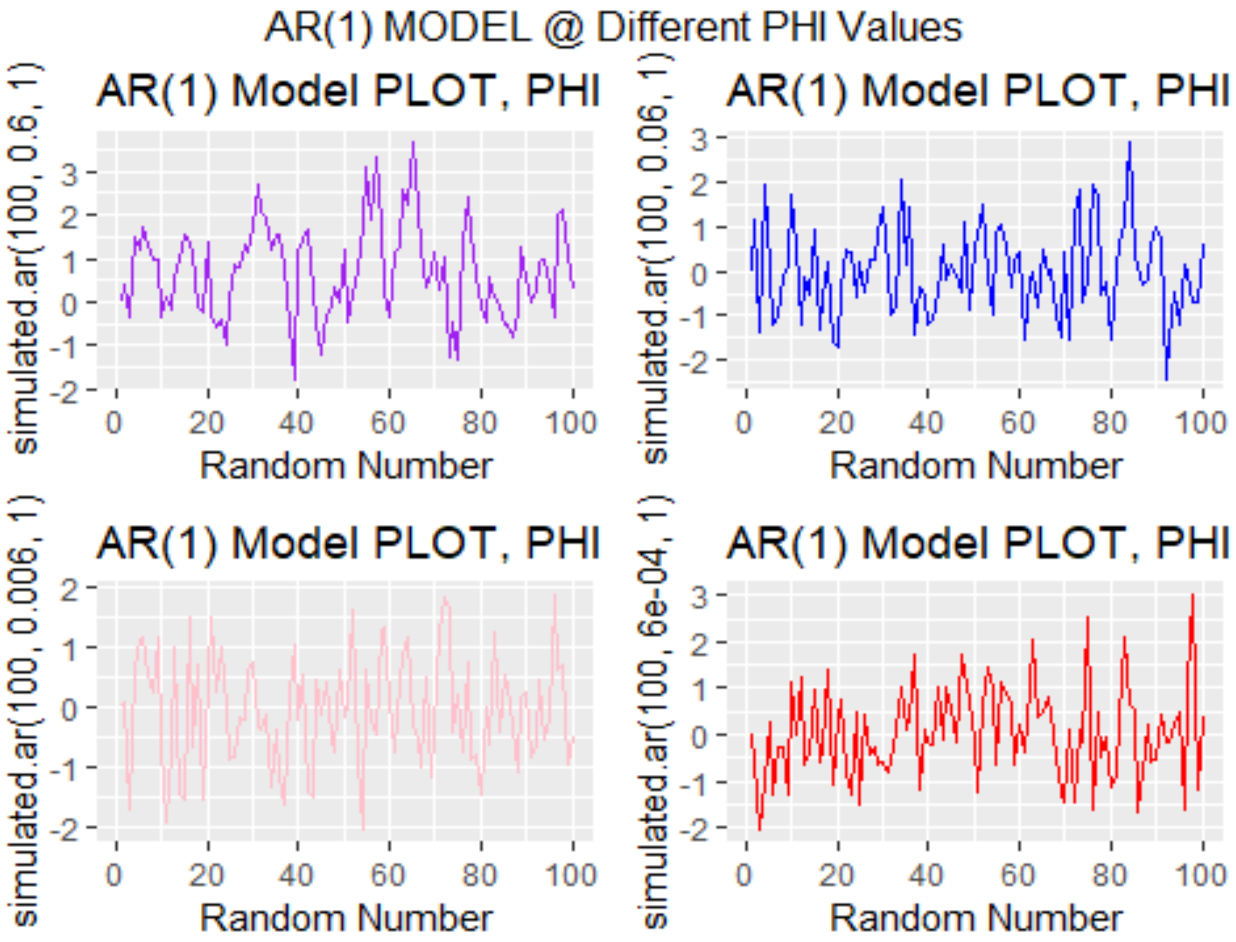
```
#created simulated.ar function from the given code so it can be reused in lat
er components of the exercise
simulated.ar <- function(N,phi, sd ){
  y <- ts(numeric(N))
  e <- rnorm(N, sd = sd)

  for( i in 2:N){
   y[i] <- phi*y[i-1] + e[i]

  }
  return(y)
```

```
}

initial.ar <- simulated.ar(100, 0.6, 1)
autoplot(initial.ar,col="purple")+ ggtitle("AR(1) Model Plot")+
  xlab("Random Number") +
  ylab("AR(1) model")
```



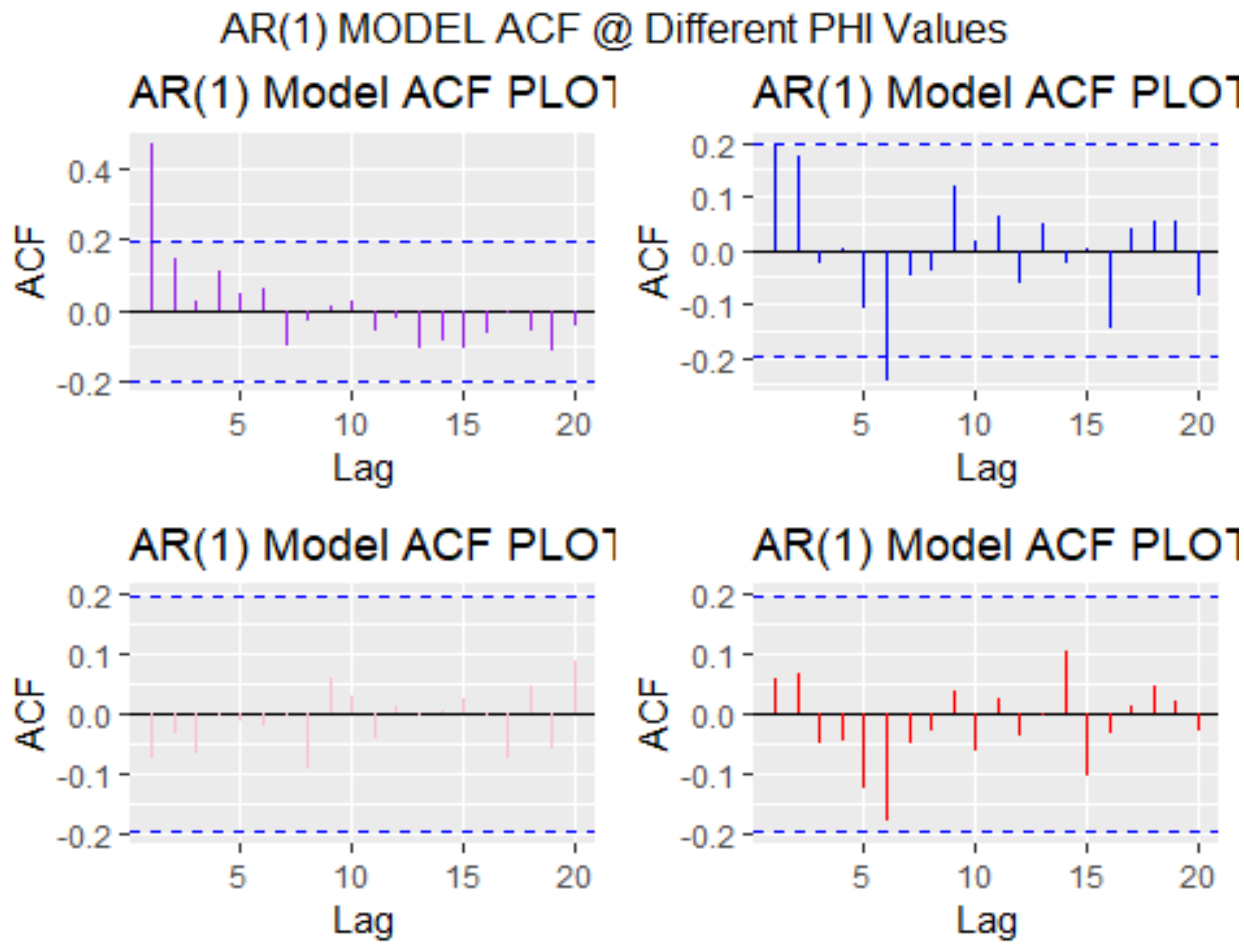
b. Produce a time plot for the series. How does the plot change as you change ϕ_1 ?

```
initial.ar <- autoplot(simulated.ar(100, 0.6, 1),col="purple")+ ggtitle("AR
(1) Model PLOT, PHI = 0.6")+ xlab("Random Number")
second.ar <- autoplot(simulated.ar(100,  0.06, 1), col="blue")+ ggtitle("AR
(1) Model PLOT, PHI = 0.06")+ xlab("Random Number")
third.ar <- autoplot(simulated.ar(100,  0.006, 1),col="pink")+ ggtitle("AR(1)
 Model PLOT, PHI = 0.006")+ xlab("Random Number")
fourth.ar <- autoplot(simulated.ar(100,  0.0006, 1),col="red")+ ggtitle("AR
(1) Model PLOT, PHI = 0.0006")+ xlab("Random Number")

grid.arrange(
  initial.ar,
  second.ar,
  third.ar,
  fourth.ar,
  nrow = 2,
  top = "AR(1) MODEL @ Different PHI Values")
```



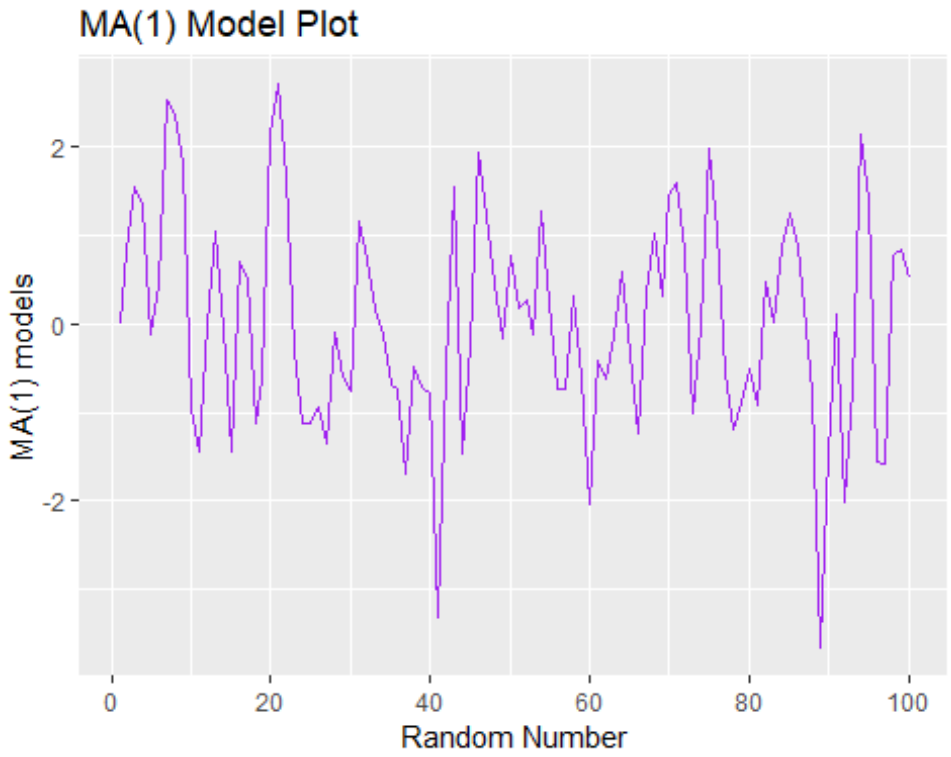
- Visually, it appears that with smaller ϕ_1 the more random the data. As per the above exercises, the auto correlation lags should be higher for the larger ϕ_1 values. The below ACF plots confirms the higher ACF plots.A change in ϕ_1 is directly proportional to variance in the timeseries.

```
initial.ar.acf <- ggAcf(simulated.ar(100, 0.6, 1),col="purple")+ ggtitle("AR
(1) Model ACF PLOT, PHI = 0.6")
second.ar.acf <- ggAcf(simulated.ar(100,  0.06, 1), col="blue")+ ggtitle("AR
(1) Model ACF PLOT, PHI = 0.06")
third.ar.acf <- ggAcf(simulated.ar(100,  0.006, 1),col="pink")+ ggtitle("AR
(1) Model ACF PLOT, PHI = 0.006")
fourth.ar.acf <- ggAcf(simulated.ar(100,  0.0006, 1),col="red")+ ggtitle("AR
(1) Model ACF PLOT, PHI = 0.0006")

grid.arrange(
  initial.ar.acf,
  second.ar.acf,
  third.ar.acf,
  fourth.ar.acf,
  nrow = 2,
  top = "AR(1) MODEL ACF @ Different PHI Values")
```



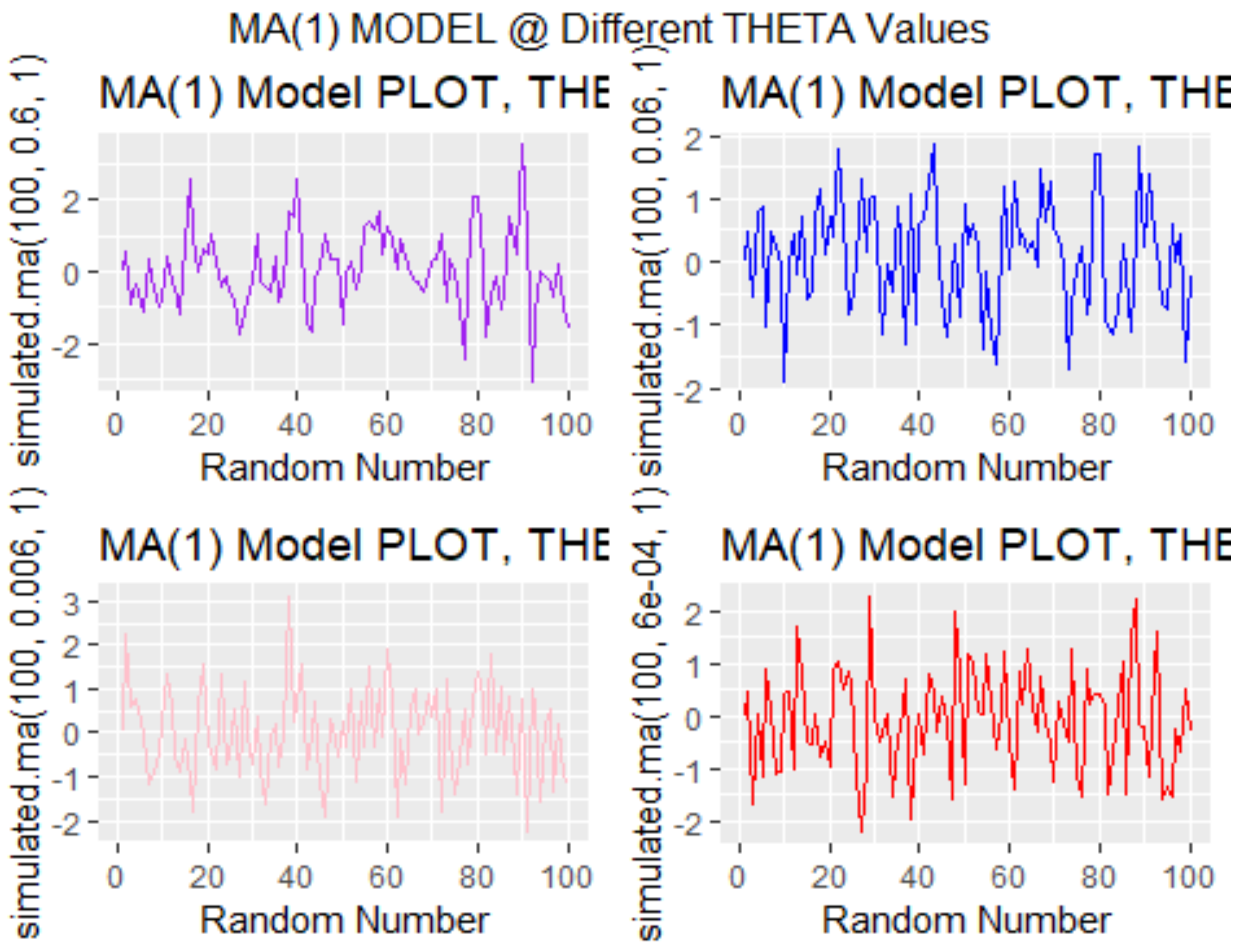
c. Write your own code to generate data from an MA(1) model with $\theta_1 = 0.6$ and $\sigma^2 = 1$

```
simulated.ma <- function(N,theta, sd ){
  y <- ts(numeric(N))
  e <- rnorm(N, sd = sd)
  e[1] <- 0
  for( i in 2:N){
   y[i] <- theta*e[i-1] + e[i]

  }
  return(y)

}

initial.ma <- simulated.ma(100, 0.6, 1)
autoplot(initial.ma,col="purple")+ ggtitle("MA(1) Model Plot")   +
  xlab("Random Number") +
  ylab("MA(1) models")
```



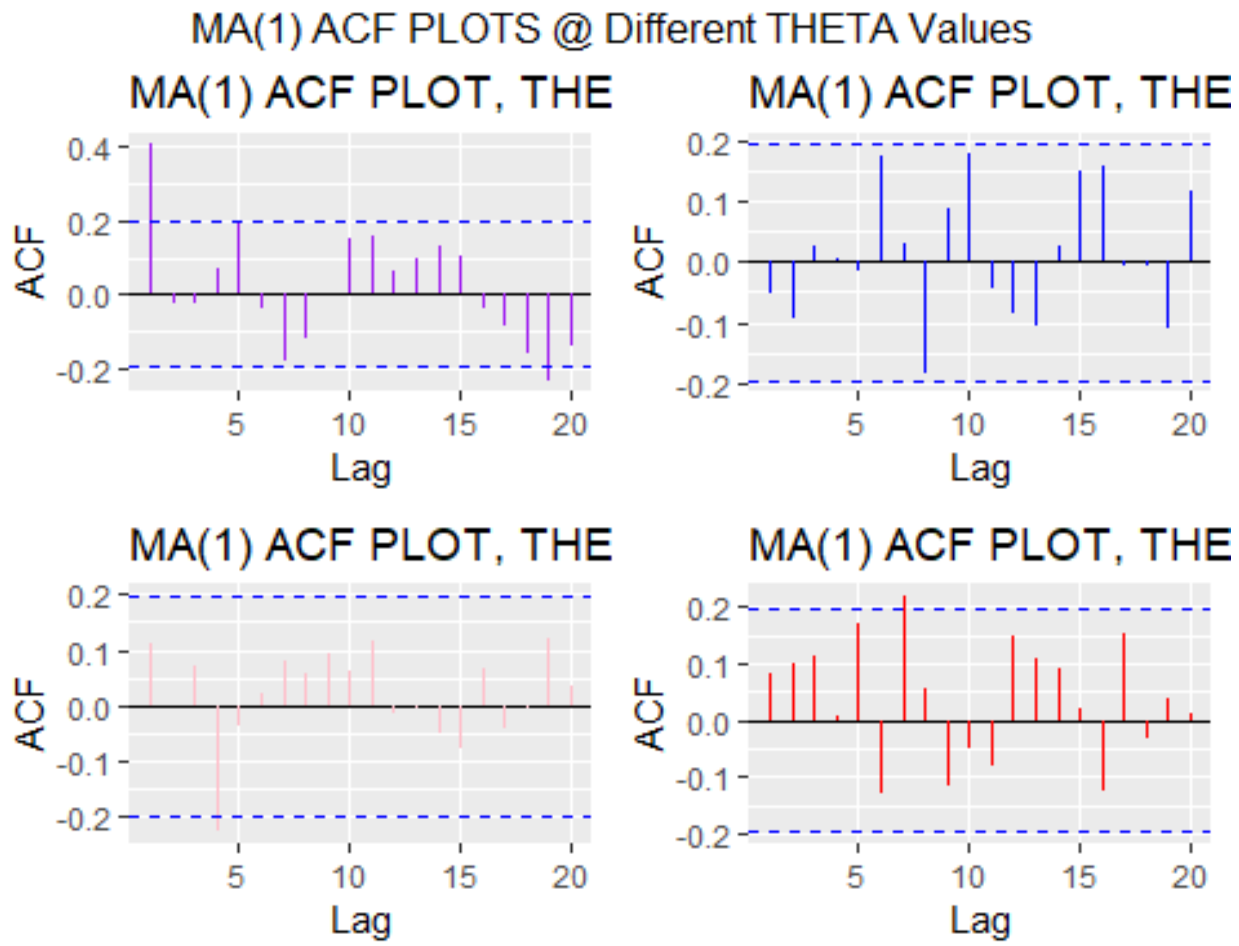
d. Produce a time plot for the series. How does the plot change as you change θ_1 ?

```
initial.ma <- autoplot(simulated.ma(100, 0.6, 1),col="purple")+ ggtitle("MA
(1) Model PLOT, THETA = 0.6")+ xlab("Random Number")
second.ma <- autoplot(simulated.ma(100,  0.06, 1), col="blue")+ ggtitle("MA
(1) Model PLOT, THETA = 0.06")+ xlab("Random Number")
third.ma <- autoplot(simulated.ma(100,  0.006, 1),col="pink")+ ggtitle("MA(1)
 Model PLOT, THETA = 0.006")+ xlab("Random Number")
fourth.ma <- autoplot(simulated.ma(100,  0.0006, 1),col="red")+ ggtitle("MA
(1) Model PLOT, THETA = 0.0006")+ xlab("Random Number")

grid.arrange(
  initial.ma,
  second.ma,
  third.ma,
  fourth.ma,
  nrow = 2,
  top = "MA(1) MODEL @ Different THETA Values")
```



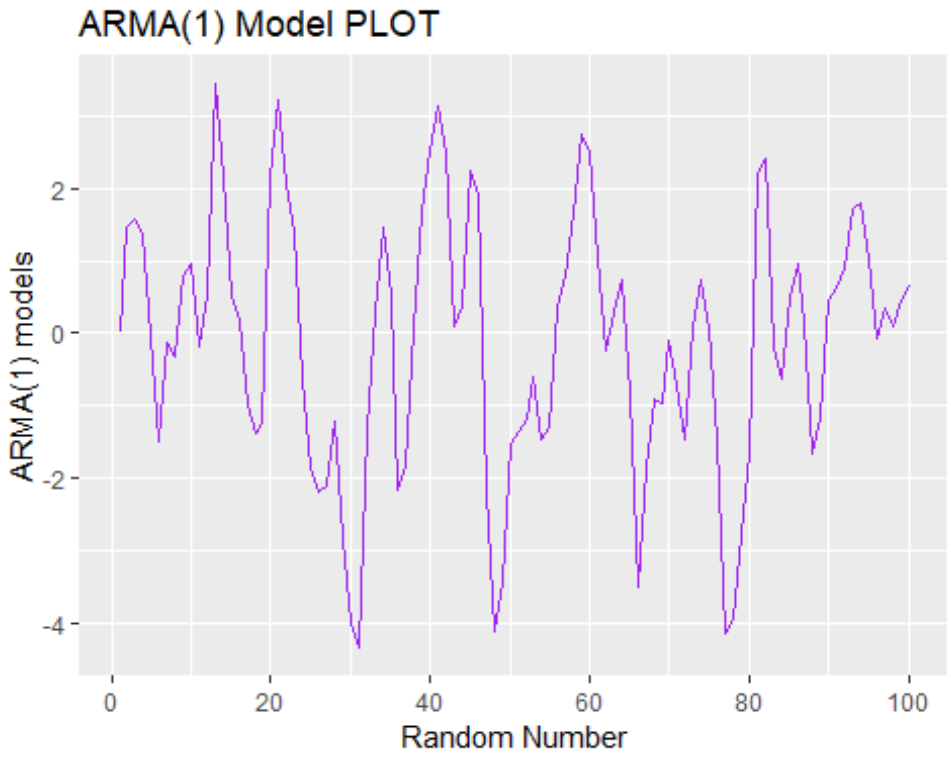
- Similar to AR(1), the smaller θ_1 in the MA(1), the more random the data appears. Below are ACF plots of the MA(1) plots to confirm.

```
initial.ma.acf <- ggAcf(simulated.ma(100, 0.6, 1),col="purple")+ ggtitle("MA
(1) ACF PLOT, THETA = 0.6")
second.ma.acf <- ggAcf(simulated.ma(100,  0.06, 1), col="blue")+ ggtitle("MA
(1) ACF PLOT, THETA = 0.06")
third.ma.acf <- ggAcf(simulated.ma(100,  0.006, 1),col="pink")+ ggtitle("MA
(1) ACF PLOT, THETA = 0.006")
fourth.ma.acf <- ggAcf(simulated.ma(100,  0.0006, 1),col="red")+ ggtitle("MA
(1) ACF PLOT, THETA = 0.0006")

grid.arrange(
  initial.ma.acf,
  second.ma.acf,
  third.ma.acf,
  fourth.ma.acf,
  nrow = 2,
  top = "MA(1) ACF PLOTS @ Different THETA Values")
```



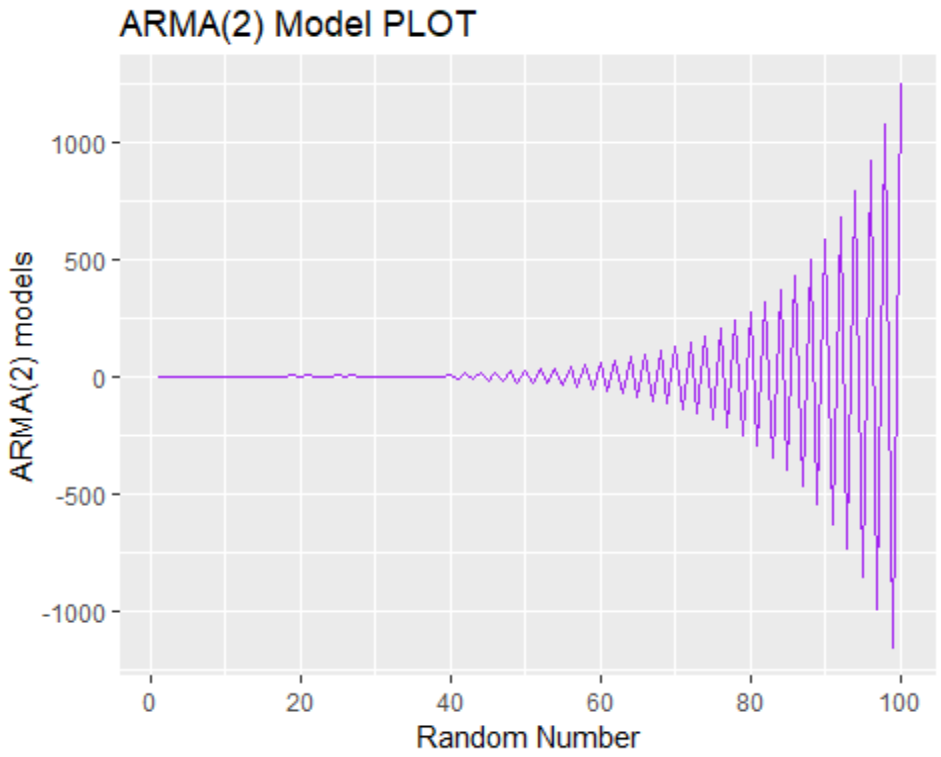
e. Generate data from an ARMA(1,1) model with $\phi_1 = 0.6$, $\theta_1 = 0.6$ and $\sigma^2 = 1$

```
simulated.arma <- function(N, phi, theta, sd ){
  y <- ts(numeric(N))
  e <- rnorm(N, sd = sd)

  for( i in 2:N){
   y[i] <- phi*y[i-1] + theta*e[i-1] +e[i]

  }
  return(y)

}

initial.arma <- simulated.arma(100, 0.6, 0.6, 1)
autoplot(initial.arma,col="purple")+ ggtitle("ARMA(1) Model PLOT")  +
  xlab("Random Number") +
  ylab("ARMA(1) models")
```



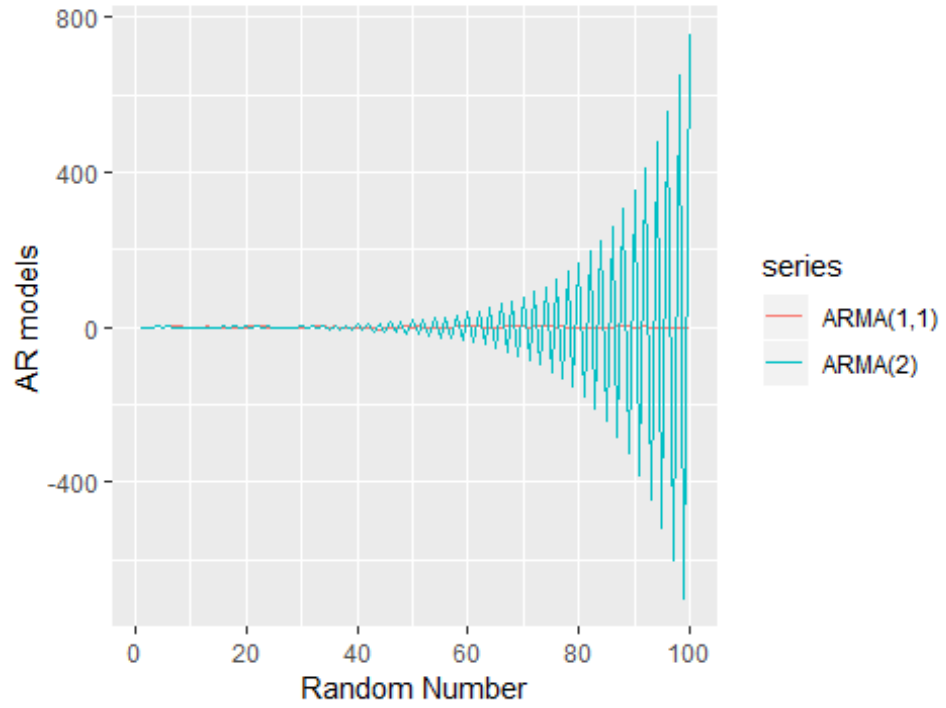
f. Generate data from an ARMA(2) model with $\phi_1 = -0.8$, $\phi_2 = 0.3$ and $\sigma^2 = 1$

```
simulated.arma.ns <- function(N, phi1, phi2, sd ){
  y <- ts(numeric(N))
  e <- rnorm(N, sd = sd)

  #2:N ran an error, changed to 3 and worked. Length zero error
  for( i in 3:N){
    y[i] <- phi1*y[i-1] + phi2*y[i-2] + e[i]


  }
  return(y)

}

non.standard.arma <- simulated.arma.ns(100, -0.8, 0.3, 1)
autoplot(non.standard.arma,col="purple")+ ggtitle("ARMA(2) Model PLOT")  +
  xlab("Random Number") +
  ylab("ARMA(2) models")
```



g. Graph the latter two series and compare them.

```
autoplot(
  simulated.arma(100, 0.6, 0.6, 1), series = "ARMA(1,1)") +
  autolayer(simulated.arma.ns(100, -0.8, 0.3, 1), series = "ARMA(2)")  +
  xlab("Random Number") +
  ylab("AR models")
```



- The AR(2) series exhibits non-stationary traits with a sinusoidal pattern amptitude curve that increases exponentially over time. The AR(2) series appears to have seasonality. The ARMA(1) series does not have this apparent seasonality and it appears to be random and much more stationary than the AR(2) series.

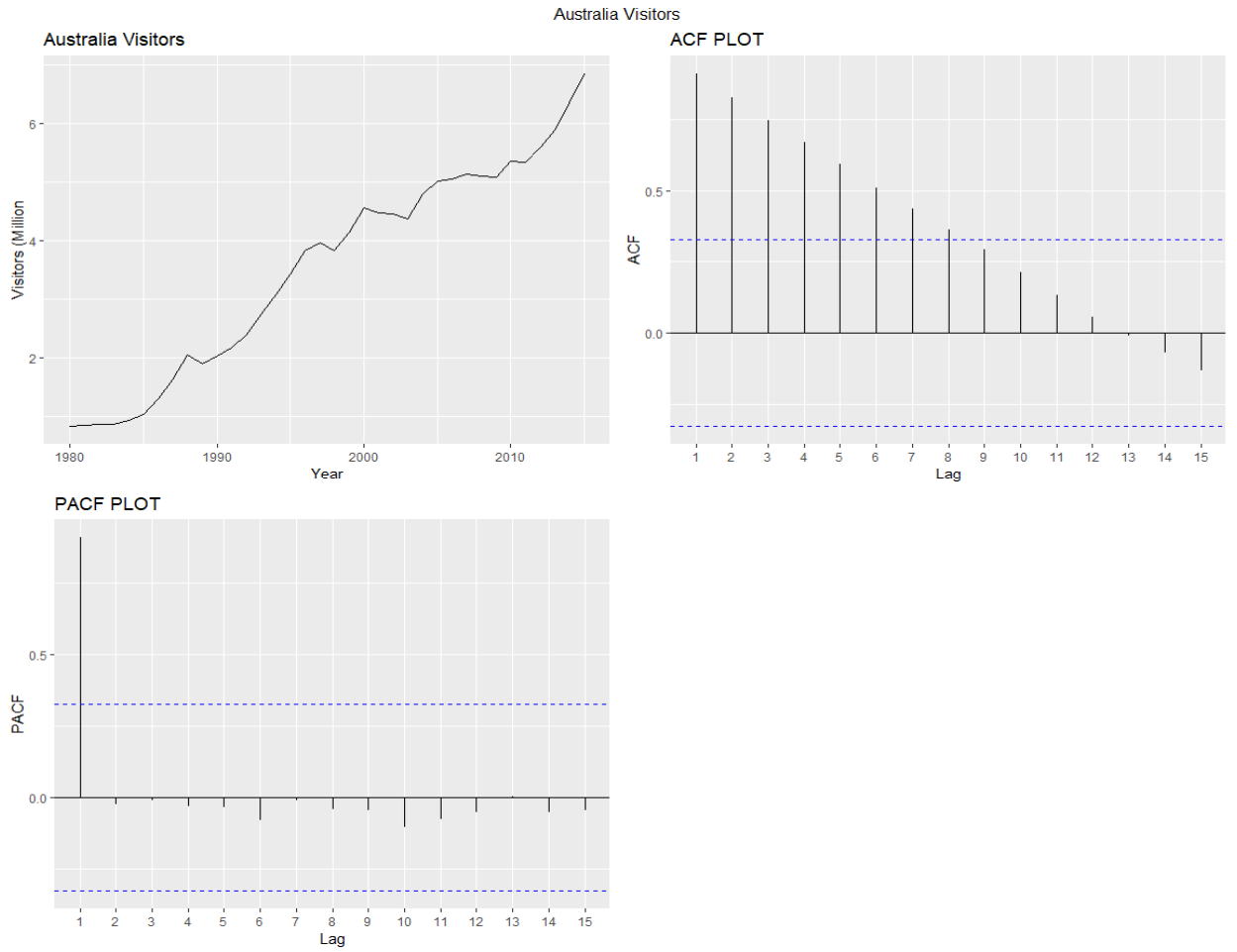
8.8 Consider austa, the total international visitors to Australia (in millions) for the period 1980-2015.

a. Use auto.arima() to find an appropriate ARIMA model. What model was selected. Check that the residuals look like white noise. Plot forecasts for the next 10 periods.

- ARIMA(0,1,1) was chosen for the auto-arima model. Residuals appear to show stationarity, confirming that the model is satisfactory. The residual plots appear to be random and the ACF & PACF indicate that the model is free of auto-correlation.

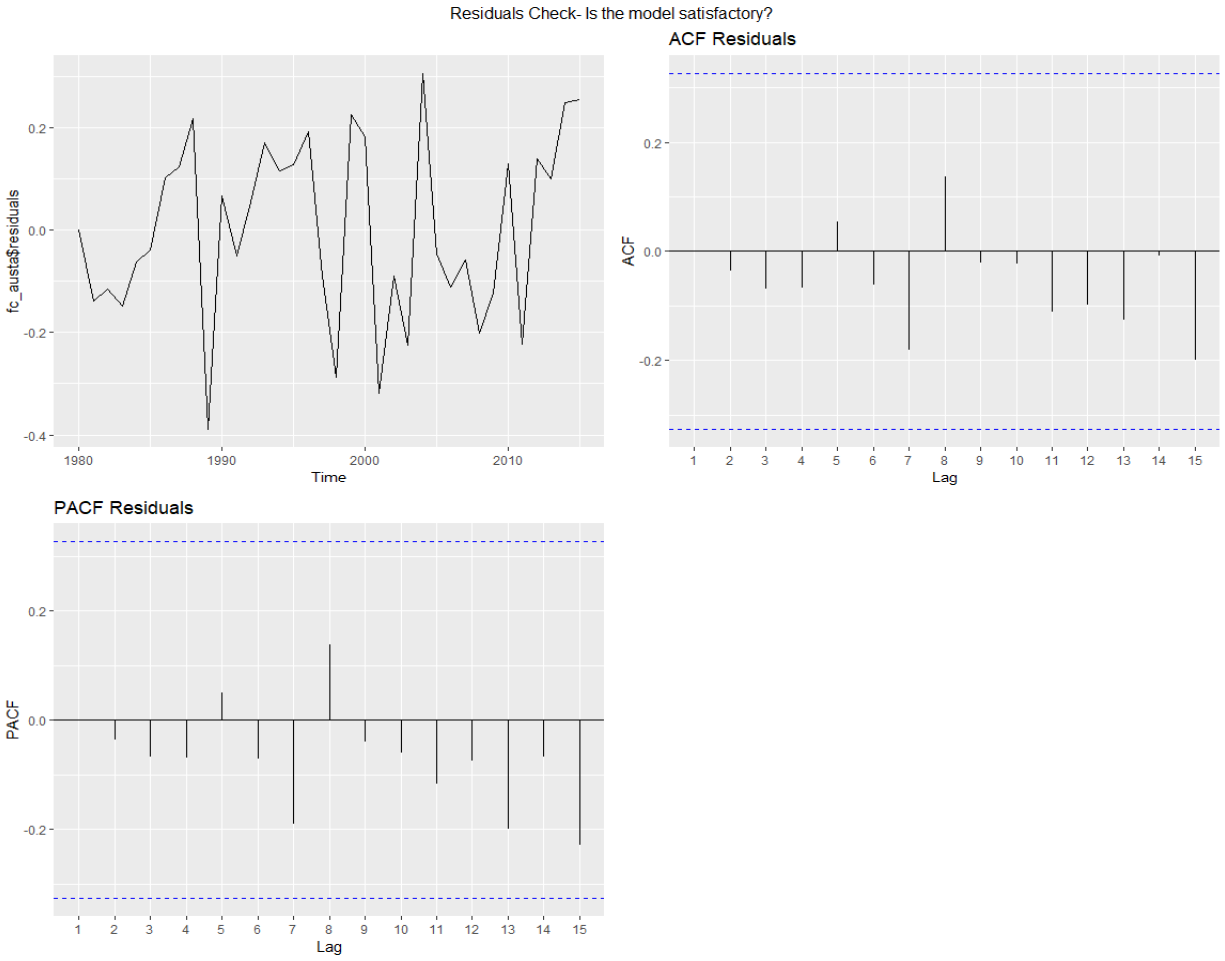
```
# plot

visitor.autoplot<- autoplot(austa) + ggtitle("Australia Visitors")+xlab("Year
") +
  ylab("Visitors (Million")
visitor.acf <- ggAcf(austa) + ggtitle("ACF PLOT")
visitor.Pacf <- ggPacf(austa) + ggtitle("PACF PLOT")
grid.arrange(
  visitor.autoplot,
  visitor.acf,
  visitor.Pacf,
```

```
  nrow = 2,
  top = "Australia Visitors")
```




```
#fit model with auto.arima

(fc_austa <- auto.arima(austa, seasonal=F))

## Series: austa
## ARIMA(0,1,1) with drift
##
## Coefficients:
##          ma1   drift
##       0.3006  0.1735
## s.e.  0.1647  0.0390
##
## sigma^2 estimated as 0.03376:  log likelihood=10.62
## AIC=-15.24   AICc=-14.46   BIC=-10.57

#check residuals to make sure the model is satisfactory

res <- autoplot(fc_austa$residuals)
acf.model <- ggAcf(ts(fc_austa$residuals))+ ggtitle("ACF Residuals")
```

```r
pacf.model <- ggPacf(ts(fc_austa$residuals))+ggtitle("PACF Residuals")

grid.arrange(
  res,
  acf.model,
  pacf.model,
  nrow = 2,
  top = "Residuals Check- Is the model satisfactory?")
```

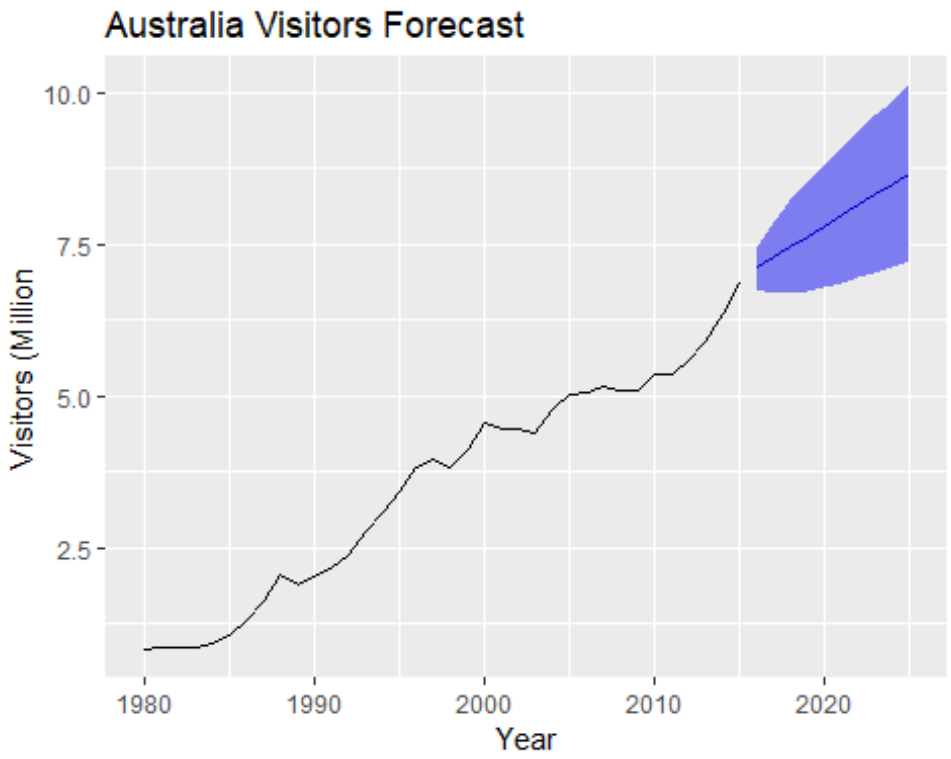



```r
# set the confidence level to 95% and the years out 10 years (h)
visitor.forecast <- forecast(fc_austa, level = c(95), h=10 )
data.frame(visitor.forecast)
```

```
##      Point.Forecast    Lo.95     Hi.95
## 2016       7.108647 6.748536  7.468758
## 2017       7.282129 6.691337  7.872922
## 2018       7.455612 6.701695  8.209529
## 2019       7.629094 6.741543  8.516644
## 2020       7.802576 6.799031  8.806120
## 2021       7.976058 6.868603  9.083513
## 2022       8.149540 6.947121  9.351960
## 2023       8.323023 7.032609  9.613436
```

```
## 2024       8.496505 7.123725  9.869284
## 2025       8.669987 7.219512 10.120462
```

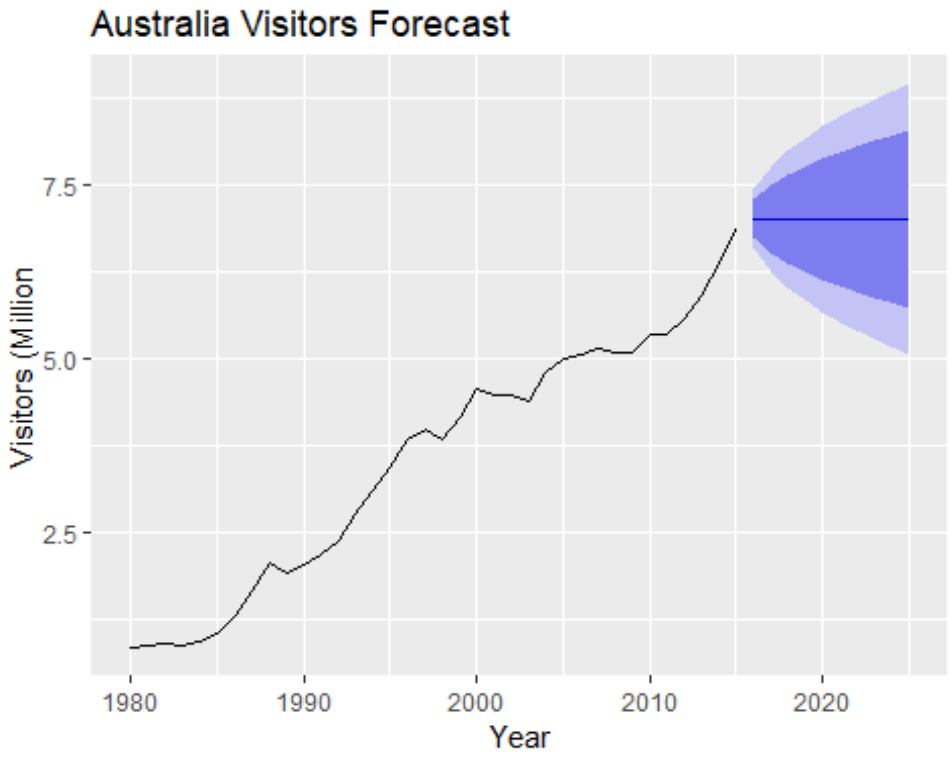
```
autoplot(visitor.forecast) + ggtitle("Australia Visitors Forecast")+xlab("Yea
r") +
  ylab("Visitors (Million")
```



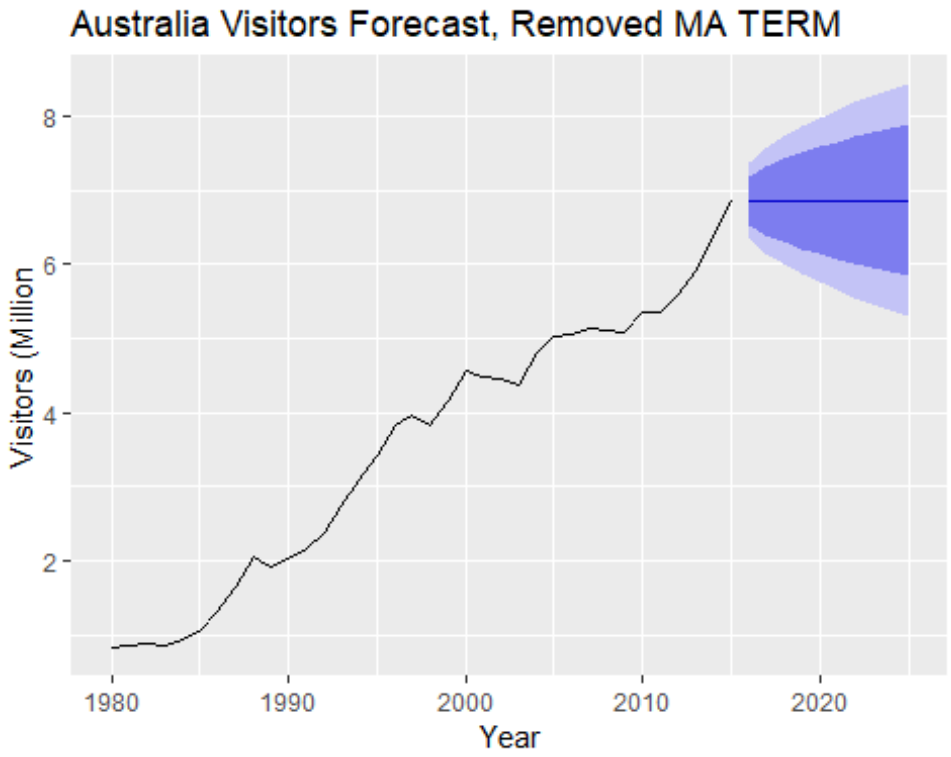
b. Plot forecasts from an ARIMA(0,1,1) model with no drift and compare these to part a. Remove the MA term and plot again.

- The for forecasts yield similar results but removing the MA parameter, slightly reduces the range of the forecast. Neither plot picks up trend.

```
fc_austa<-forecast(arima(austa,order =c(0, 1, 1)), h = 10)
autoplot(fc_austa)+ ggtitle("Australia Visitors Forecast")+xlab("Year") +
  ylab("Visitors (Million")
```



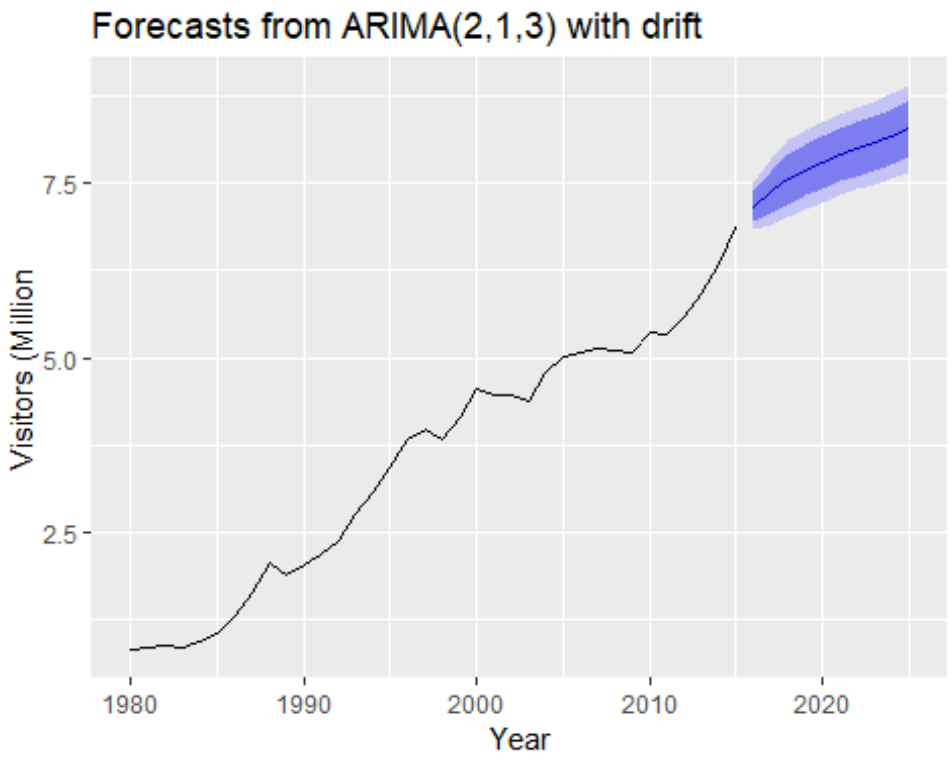

```
# remove moving average term.
fc_austa<-forecast(arima(austa,order=c(0, 1, 0)), h = 10)
autoplot(fc_austa)+ ggtitle("Australia Visitors Forecast, Removed MA TERM")+x
lab("Year") +
  ylab("Visitors (Million")
```



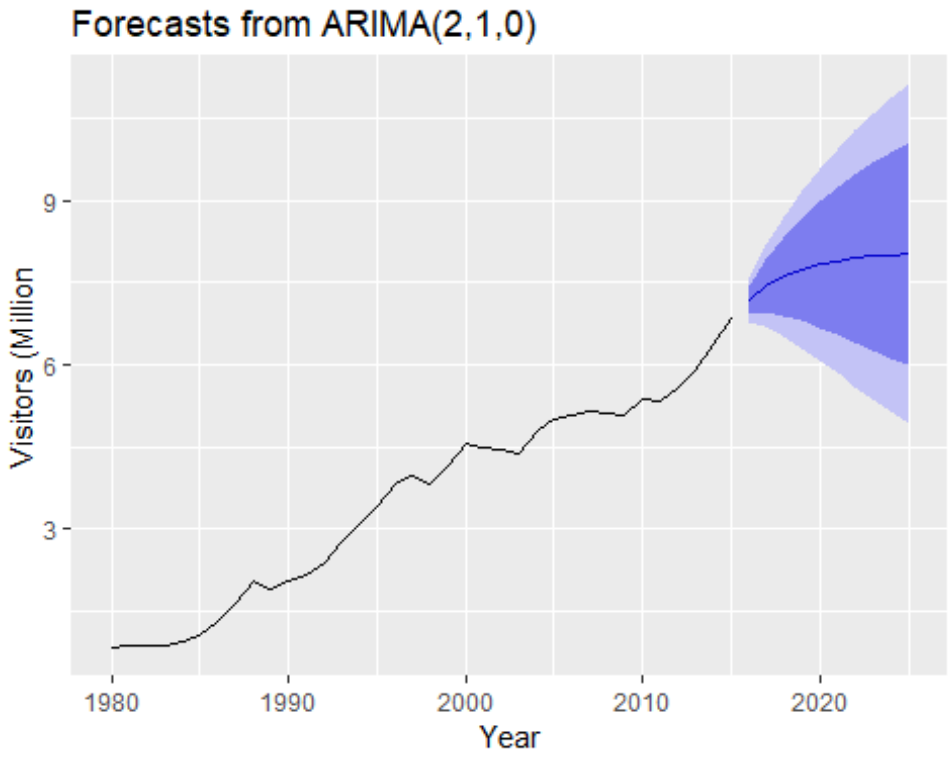
c. Plot forecasts from an ARIMA(2,1,3) model with drift. Remove the constant and see what happens.

- The forecast range significantly expands with the drift removed

```
fc_austa<-forecast(Arima(austa,order =c(2, 1, 3),include.drift=TRUE),h = 10)
autoplot(fc_austa)+xlab("Year") +
  ylab("Visitors (Million")
```



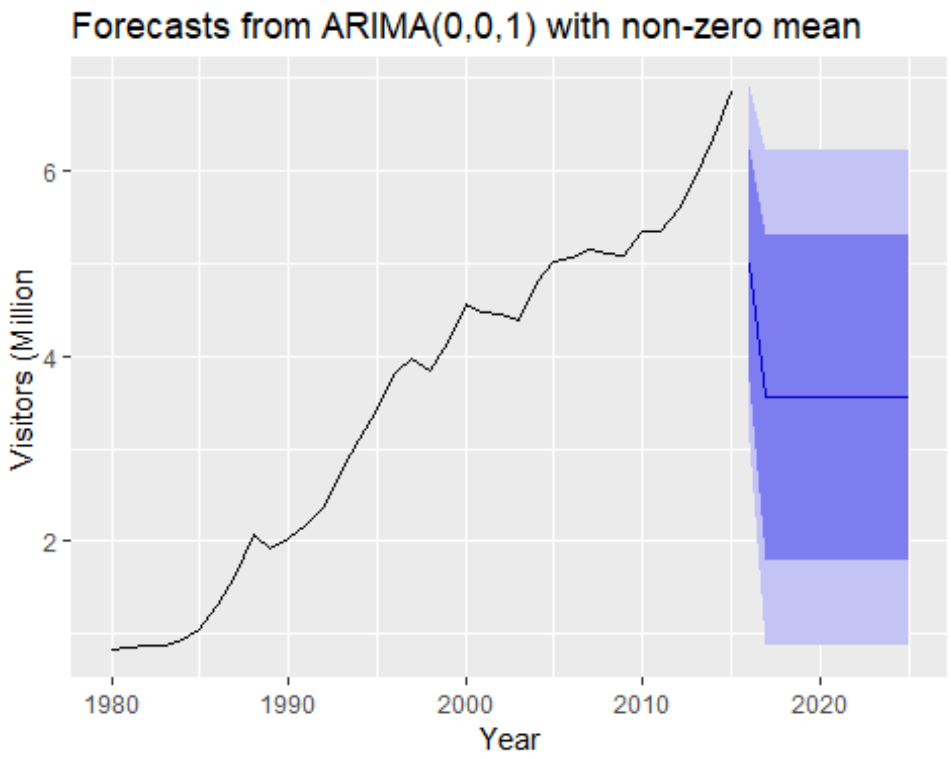

```
# remove moving average term.
fc_austa$model$coef[6] <- 0
fc_austa<-forecast(Arima(austa,order =c(2, 1, 0)),h = 10)
autoplot(fc_austa)+xlab("Year") +
  ylab("Visitors (Million")
```



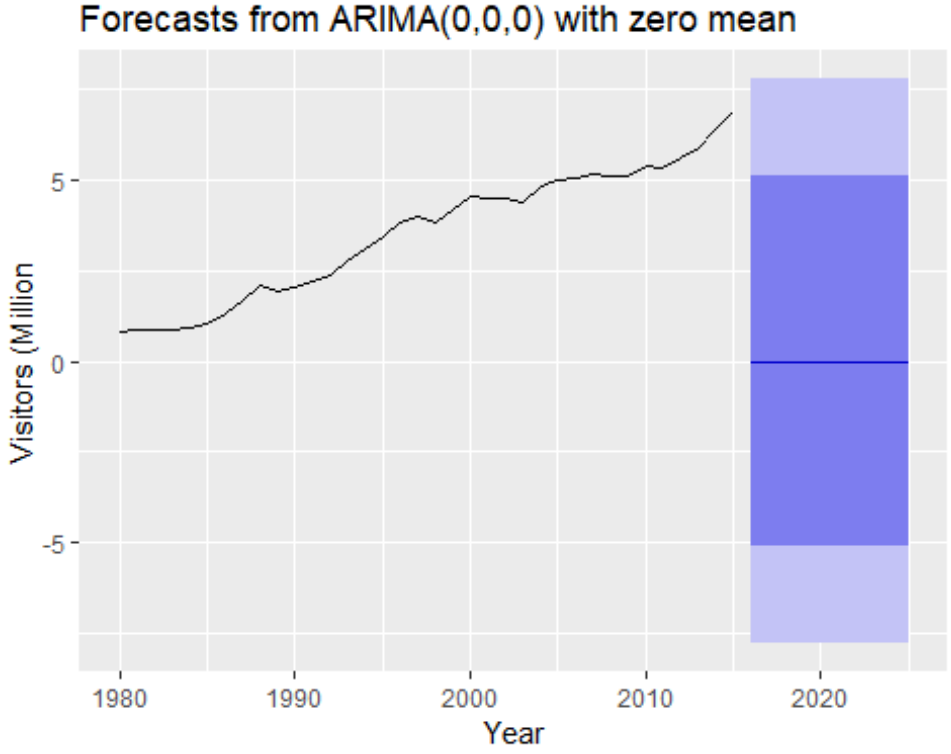
d. Plot forecasts from an ARIMA(0,0,1) model with a constant. Remove the MA term and plot again.

- The plot without constant shifts are below trend but with the MA term, the forecast shifts upward. The MA still doesn't pick up the trend line though.

```
fc_austa<-forecast(Arima(austa,order=c(0,0,1),include.constant=TRUE),h=10)
autoplot(fc_austa)+xlab("Year") +
  ylab("Visitors (Million")
```




```
fc_austa<-forecast(Arima(austa,order=c(0,0,0),include.constant=FALSE),h=10)
autoplot(fc_austa)+xlab("Year") +
  ylab("Visitors (Million")
```



e. Plot forecasts from an ARIMA(0,2,1) model with no constant.

- This plot picks up the trendline and is well fitted.

```
fc_austa<-forecast(Arima(austa,order=c(0,2,1),include.constant=FALSE),h=10)
autoplot(fc_austa)+xlab("Year") +
  ylab("Visitors (Million")
```