DATA 624 Spring 2019: Homework-1

Ahmed Sajjad, Harpreet Shoker, Jagruti Solao, Chad Smith, Todd Weigel

March 12, 2019

library(mlbench)  
library(tidyr)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(ggplot2)  
library(forecast)  
library(fma)  
library(fpp2)

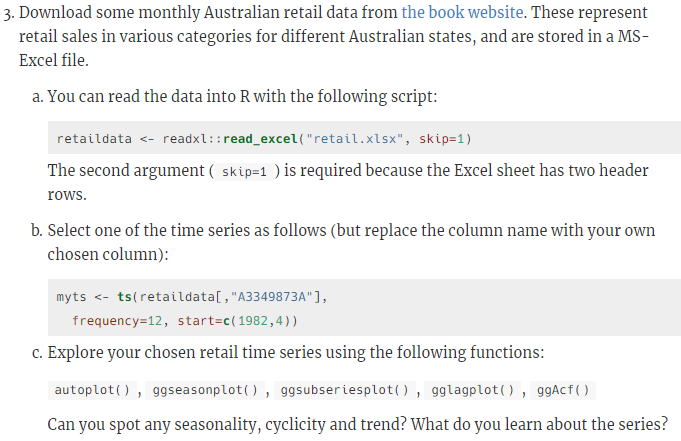
## Loading required package: expsmooth

library(gridExtra)

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':  
##   
## combine

##### HA# 2.3

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

retaildata <- readxl::read\_excel("retail.xlsx", skip=1)

The second argument (skip=1) is required because the Excel sheet has two header rows.

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

myts <- ts(retaildata[,"A3349873A"],frequency=12, start=c(1982,4))  
head(myts)

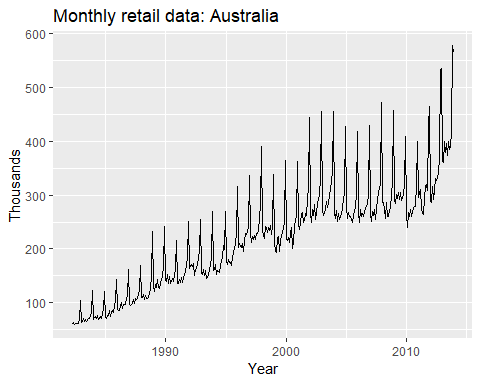
## Apr May Jun Jul Aug Sep  
## 1982 62.4 63.1 59.6 61.9 60.7 61.2

### c) Explore your chosen retail time series using the following functions:

autoplot(), ggseasonplot(), ggsubseriesplot(), gglagplot(), ggAcf()

Can you spot any seasonality, cyclicity and trend? What do you learn about the series?

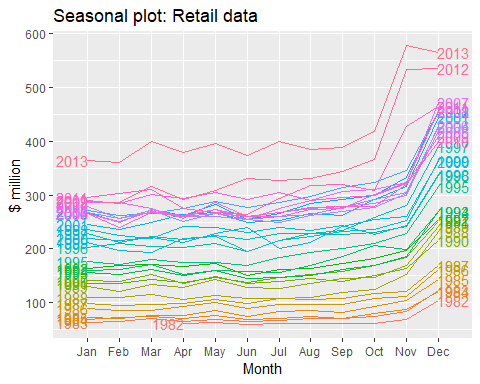
autoplot(myts) +  
 ggtitle("Monthly retail data: Australia") +  
 xlab("Year") +  
 ylab("Thousands")



Time series plot reveals following features:

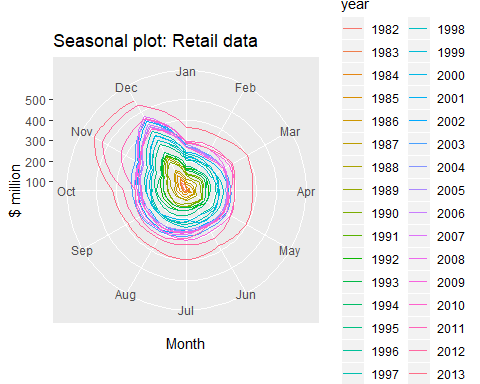
1. There is continuous increment for business in retail sector data from 1982 to2012.
2. Sudden dip was observed in year 2000 and 2010 may be due to resession and slowdown of market.
3. Retail market shows dips for start of year and gradual highs towards end of it.

ggseasonplot(myts, year.labels=TRUE, year.labels.left=TRUE) +  
 ylab("$ million") +  
 ggtitle("Seasonal plot: Retail data")

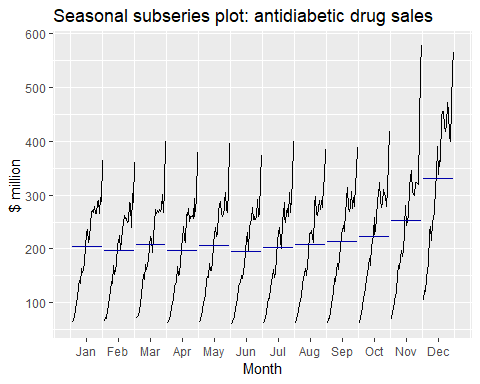


A seasonal plot allows the underlaying pattern to be seen more clearly and are useful in identifying years in which the pattern changes. In our case, above plot makes it clear that there is large jump in retail sector by end of year , mostly in Nov and Dec.patterns also make us notice that there are unsual dips in retail sector around Jun ,July.Seasonal plot is similar to time plot except the data are plotted against the individual "seasons" in which data were observed.There can also be variation made in plot using polar co-ordinates.

ggseasonplot(myts, polar = TRUE) +  
 ylab("$ million") +  
 ggtitle("Seasonal plot: Retail data")



ggsubseriesplot(myts) +  
 ylab("$ million") +  
 ggtitle("Seasonal subseries plot: antidiabetic drug sales")



Seasonality plot validates our clain that retail sector has maximum business in November and December each year. The horizontal line indicates means for each month.This form of plot enables the underlaying seasonal pattern to be seen clearly,also showing changes in seasonality over time.

gglagplot(myts)

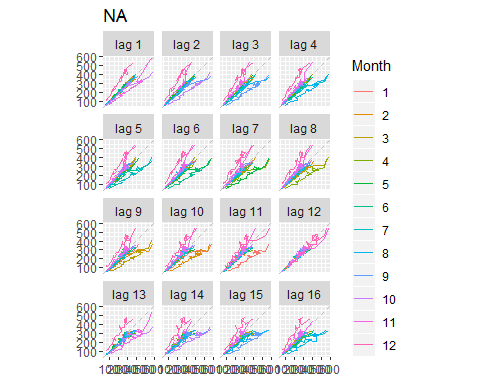


Figure shows Lagged scatterplots for monthly retail data. The relations are strongly positive for month 5,6,7 i.e May,June,July for lag 1,2,3,4 and strongly negative for month 5 in lag 5,6,7.

ggAcf(window(myts),start = 1982)

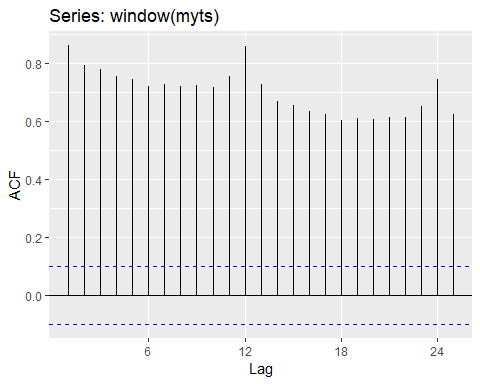
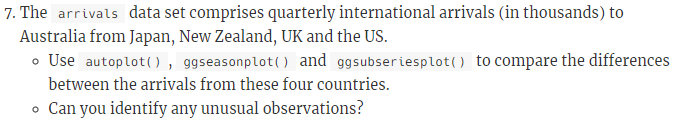
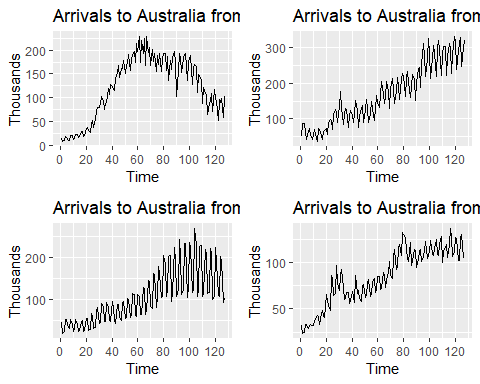


Figure 6 shows Autocorrelation function for monthly retail data for Australia. As we can observe there is no negative co-relation observed. The dashed blue lines indicate whether the correlations are significantly different from zero.

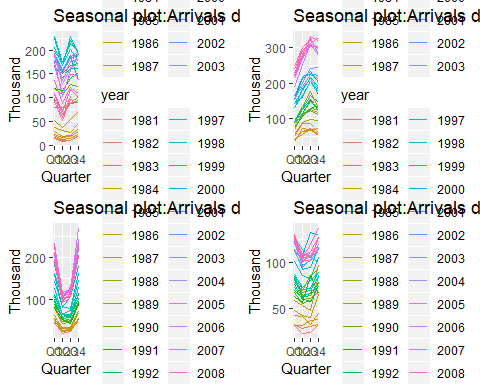
##### HA# 2.7



myts1 <- ts(arrivals[,"Japan"])  
a <- autoplot(myts1) + ggtitle("Arrivals to Australia from Japan") + ylab("Thousands")  
myts2 <- ts(arrivals[,"NZ"])  
b <- autoplot(myts2) + ggtitle("Arrivals to Australia from NZ") + ylab("Thousands")  
myts3 <- ts(arrivals[,"UK"])  
c <- autoplot(myts3) + ggtitle("Arrivals to Australia from UK") + ylab("Thousands")  
myts4 <- ts(arrivals[,"US"])  
d <- autoplot(myts4) + ggtitle("Arrivals to Australia from US") + ylab("Thousands")  
grid.arrange(a,b,c,d, nrow = 2)

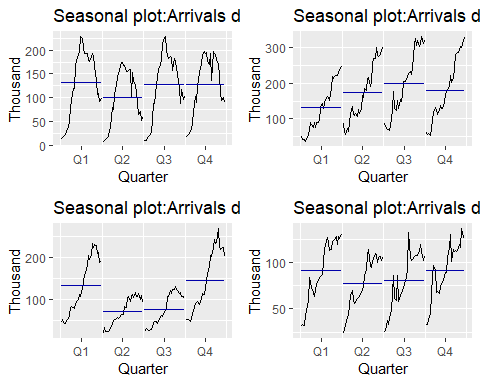


a1 <- ggseasonplot(arrivals[,"Japan"]) + ylab("Thousand") +ggtitle("Seasonal plot:Arrivals data for Japan")  
b1 <- ggseasonplot(arrivals[,"NZ"]) + ylab("Thousand") +ggtitle("Seasonal plot:Arrivals data for NZ")  
c1 <- ggseasonplot(arrivals[,"UK"]) + ylab("Thousand") +ggtitle("Seasonal plot:Arrivals data for UK")  
d1 <- ggseasonplot(arrivals[,"US"]) + ylab("Thousand") +ggtitle("Seasonal plot:Arrivals data for US")  
grid.arrange(a1,b1,c1,d1, nrow = 2)



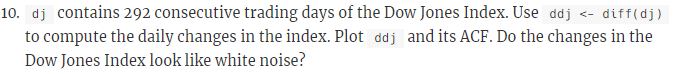
The most unusual observation is obsereved for arrival data in 'UK' for Q2 and Q3.The arrival numbers are drastically down but gradually increased in Q4.At the same time, New zealand shows highest arrival rates for Q3.

a2 <- ggsubseriesplot(arrivals[,"Japan"]) + ylab("Thousand") +ggtitle("Seasonal plot:Arrivals data for Japan")  
b2 <- ggsubseriesplot(arrivals[,"NZ"]) + ylab("Thousand") +ggtitle("Seasonal plot:Arrivals data for NZ")  
c2 <- ggsubseriesplot(arrivals[,"UK"]) + ylab("Thousand") +ggtitle("Seasonal plot:Arrivals data for UK")  
d2 <- ggsubseriesplot(arrivals[,"US"]) + ylab("Thousand") +ggtitle("Seasonal plot:Arrivals data for US")  
grid.arrange(a2,b2,c2,d2, nrow = 2)

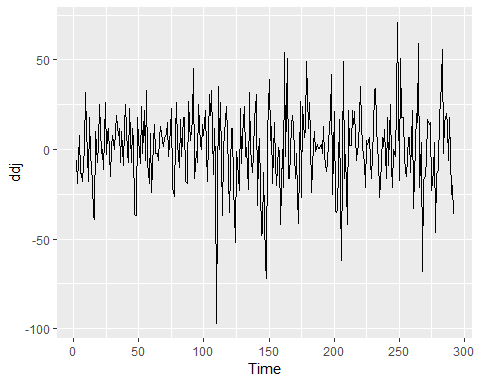


Seasonal subseries plot emphasises theseasonal patterns of data.The horizontal lines indicate means for each quarter.This form of plot enables the underlaying seasonal pattern to be seen clearly ,and also shows the changes in seasonality over time.If we observe seasonal plots for all four countries,New zealand has the highest rate of arrival data for all four quarters.UK seems to have lowest arrival rates for Q2 and Q3.

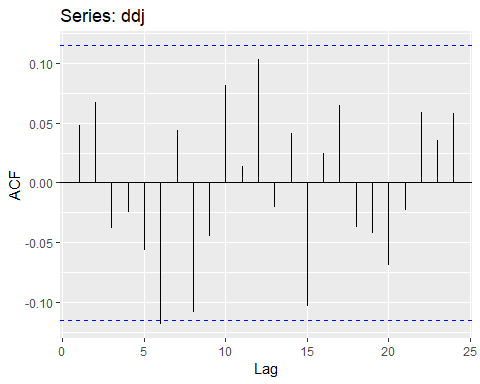
##### HA# 2.10



ddj <- diff(dj)  
autoplot(ddj)

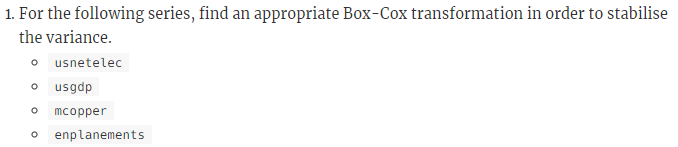


ggAcf(ddj)

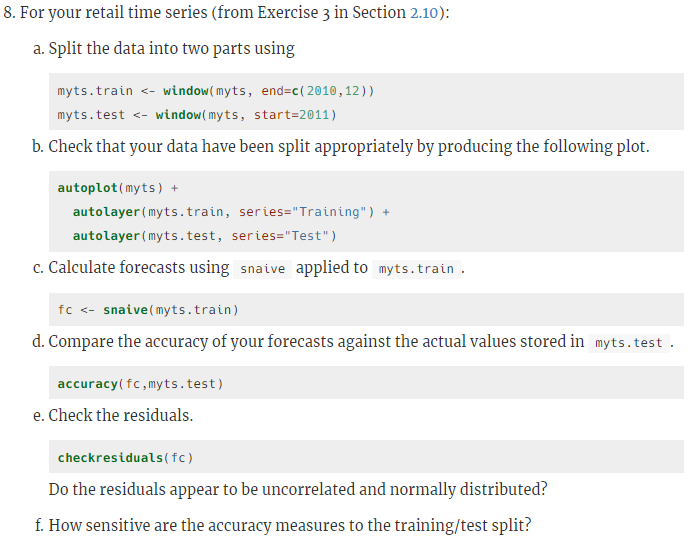


For white noise series, each autocorelation is expected to be close to zero.If one or more large spikes are outside these bounds of blue dotted lines, or if more than 5% of spikes are outside this bounds,then series is not a white noise. In our case,all the autocorelations lie within these limits,confirming that the data is white noise.

##### HA# 3.1



##### HA# 3.8



##### For your retail time series (from Exercise 3 in Section 2.10):

Loading the reatil data

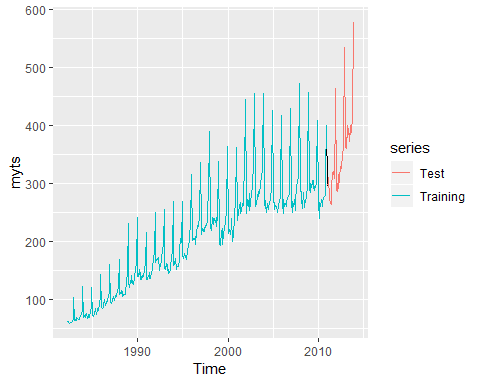
retaildata <- readxl::read\_excel("retail.xlsx", skip=1)  
myts <- ts(retaildata[,"A3349873A"],  
 frequency=12, start=c(1982,4))

##### a) Split the data into two parts using

myts.train <- window(myts, end=c(2010,12))  
myts.test <- window(myts, start=2011)

##### b) Check that your data have been split appropriately by producing the following plot.

autoplot(myts) + autolayer(myts.train, series="Training") +  
 autolayer(myts.test, series="Test")



##### c) Calculate forecasts using snaive applied to myts.train.

fc <- snaive(myts.train)

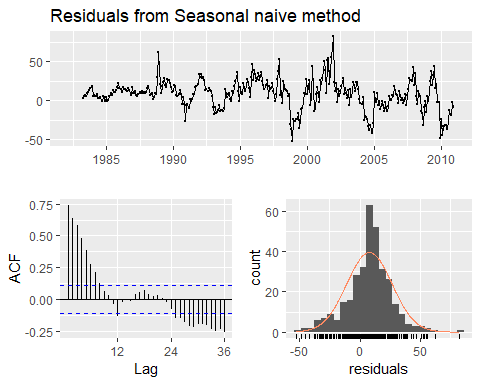
##### d) Compare the accuracy of your forecasts against the actual values stored in myts.test.

accuracy(fc,myts.test)

## ME RMSE MAE MPE MAPE MASE  
## Training set 7.772973 20.24576 15.95676 4.702754 8.109777 1.000000  
## Test set 55.300000 71.44309 55.78333 14.900996 15.082019 3.495907  
## ACF1 Theil's U  
## Training set 0.7385090 NA  
## Test set 0.5315239 1.297866

##### e) Check the residuals.Do the residuals appear to be uncorrelated and normally distributed?

checkresiduals(fc)



##   
## Ljung-Box test  
##   
## data: Residuals from Seasonal naive method  
## Q\* = 624.45, df = 24, p-value < 2.2e-16  
##   
## Model df: 0. Total lags used: 24

From the above it seems like residuals are correlated to each other. Residuals are not normally distributed.

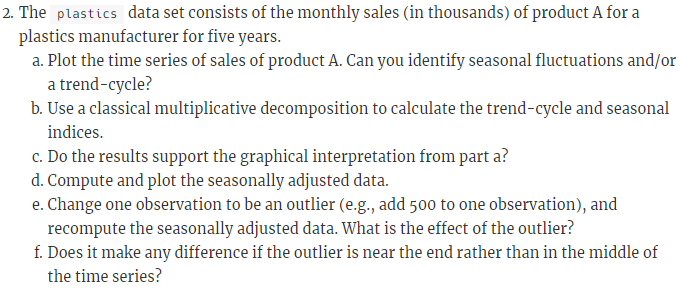
##### f) How sensitive are the accuracy measures to the training/test split?

myts2.train <- window(myts, end=c(2011,12))  
myts2.test <- window(myts, start=2012)  
fc2 <- snaive(myts2.train)  
accuracy(fc2,myts.test)

## ME RMSE MAE MPE MAPE MASE  
## Training set 8.828696 21.81237 16.76145 4.922591 8.223701 1.0000  
## Test set 65.429167 78.69376 67.34583 15.853615 16.520064 4.0179  
## ACF1 Theil's U  
## Training set 0.7198310 NA  
## Test set 0.7328968 1.356191

The accuracy measures are sensitive to the training/test split. Here we changed the train/test split percentage and run the accuracy check again and that reslts in low values in accuracy measure indicators. Comparing this to original matrix clearly indicates that the measures are sensitive to the split.

##### HA# 6.2

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

Loading the required libraries and plastics data

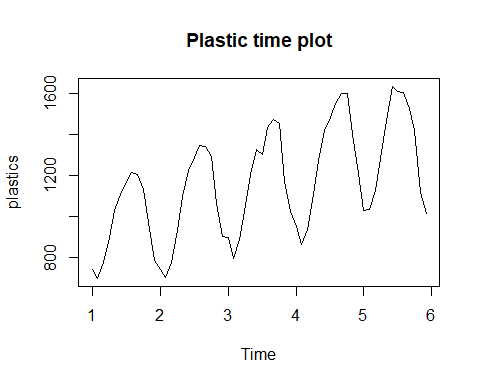
plastics

## 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

ts(plastics)

## Time Series:  
## Start = 1   
## End = 60   
## Frequency = 1   
## [1] 742 697 776 898 1030 1107 1165 1216 1208 1131 971 783 741 700  
## [15] 774 932 1099 1223 1290 1349 1341 1296 1066 901 896 793 885 1055  
## [29] 1204 1326 1303 1436 1473 1453 1170 1023 951 861 938 1109 1274 1422  
## [43] 1486 1555 1604 1600 1403 1209 1030 1032 1126 1285 1468 1637 1611 1608  
## [57] 1528 1420 1119 1013

plot(plastics,main = 'Plastic time plot')



From the above Time plot we can see there are seasonal fluctuations and upward trend. Seasonal sales are peaking in summer. Overall plot shows positive trnd with sales increasing yearly.

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

plastic\_model <- decompose(plastics, type="multiplicative")  
trend <- plastic\_model$trend #calculating trend  
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

seasonal <- plastic\_model$seasonal # calculating seasonal indices  
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

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

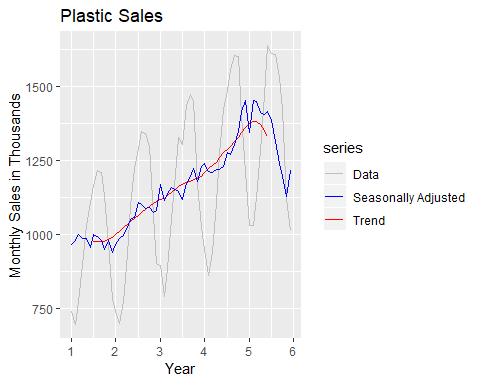
Yes, the results support the graphical interpretation.The graph indicates that the summer months have higher seasonal indices than the winter months

##### d) Compute and plot the seasonally adjusted data.

Here we are showing the trend-cycle component and the seasonally adjusted data, along with the original data.

autoplot(plastics, series="Data") +  
 autolayer(trendcycle(plastic\_model), series="Trend") +  
 autolayer(seasadj(plastic\_model), series="Seasonally Adjusted") +  
 xlab("Year") + ylab("Monthly Sales in Thousands") +  
 ggtitle("Plastic Sales") +  
 scale\_colour\_manual(values=c("gray","blue","red"), breaks=c("Data","Seasonally Adjusted","Trend"))

## Warning: Removed 12 rows containing missing values (geom\_path).



##### 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?

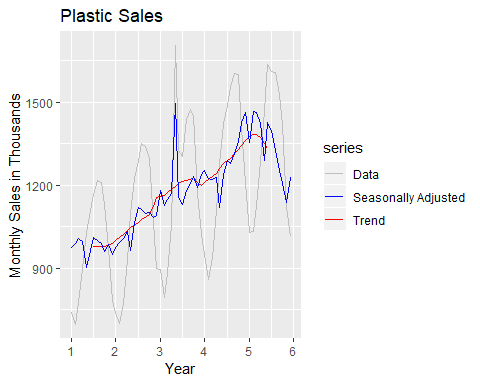
Here we are adding 500 to 29th observation

outlier\_plastic <- plastics  
outlier\_plastic[29] <- outlier\_plastic[29] + 500  
outlier\_model <- decompose(outlier\_plastic, type = "multiplicative")

Plot showing the trend-cycle component and the seasonally adjusted data, along with the original data with modified outlier data.

autoplot(outlier\_plastic, series = "Data") +  
 autolayer(trendcycle(outlier\_model), series = "Trend") +  
 autolayer(seasadj(outlier\_model), series = "Seasonally Adjusted") +  
 xlab("Year") + ylab("Monthly Sales in Thousands") +  
 ggtitle("Plastic Sales") +  
 scale\_color\_manual(values=c("gray", "blue", "red"), breaks=c("Data", "Seasonally Adjusted", "Trend"))

## Warning: Removed 12 rows containing missing values (geom\_path).



We can see from the above graph that outlier doesnot have much effects on Trend cycle but it is highly effecting the seasonal data

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

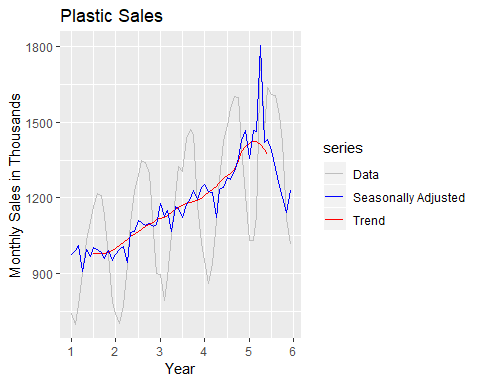
Here we are adding 500 to 52nd observation

outlierend\_plastic <- plastics  
outlierend\_plastic[52] <- outlierend\_plastic[52] + 500  
outlierend\_model <- decompose(outlierend\_plastic, type = "multiplicative")

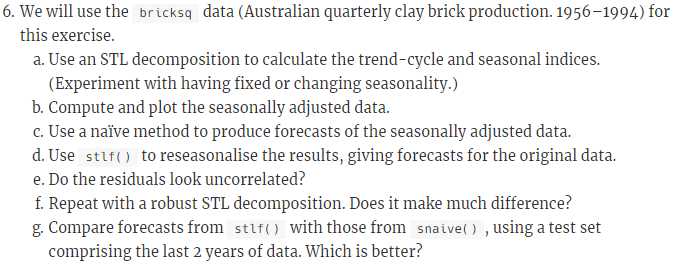
Plot showing the trend-cycle component and the seasonally adjusted data, along with the original data with modified outlier data

autoplot(outlierend\_plastic, series = "Data") +  
 autolayer(trendcycle(outlierend\_model), series = "Trend") +  
 autolayer(seasadj(outlierend\_model), series = "Seasonally Adjusted") +  
 xlab("Year") + ylab("Monthly Sales in Thousands") +  
 ggtitle("Plastic Sales") +  
 scale\_color\_manual(values=c("gray", "blue", "red"), breaks=c("Data", "Seasonally Adjusted", "Trend"))

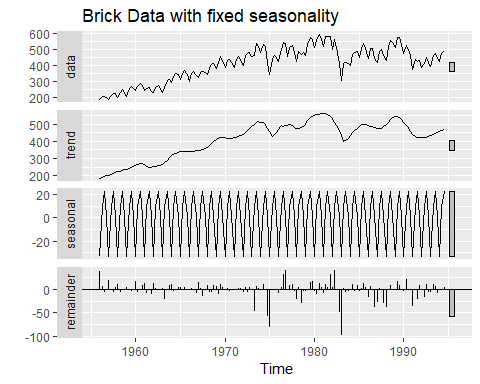
## Warning: Removed 12 rows containing missing values (geom\_path).

 We can conclude that an outlier causes a spike in the month it is present by increasing seasonality index of that month.

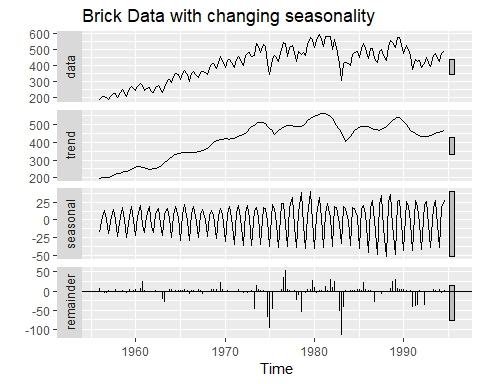
##### HA# 6.6

 ##### a) Use an STL decomposition to calculate the trend-cycle and seasonal indices. (Experiment with having fixed or changing seasonality.)

stl\_brickfixed <- stl(bricksq, s.window = "periodic",robust = TRUE)  
autoplot(stl\_brickfixed) +ggtitle("Brick Data with fixed seasonality")



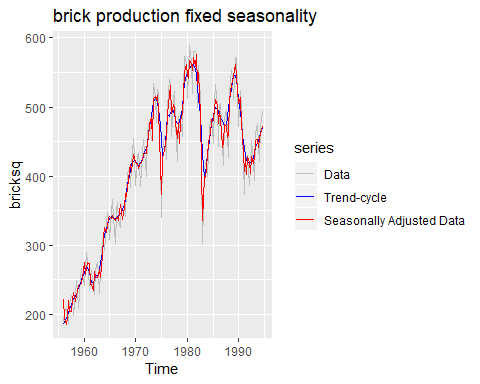
stl\_brickchange <- stl(bricksq,s.window = 5,robust = TRUE)  
autoplot(stl\_brickchange) +ggtitle("Brick Data with changing seasonality")



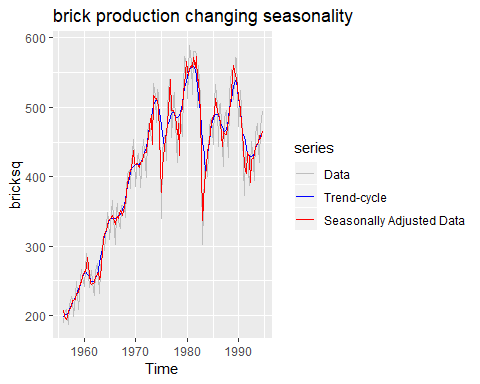
##### b) Compute and plot the seasonally adjusted data.

Here we are showing the trend-cycle component and the seasonally adjusted data, along with the original data.

# plot data which are decomposed by STL with fixed seasonality  
autoplot(bricksq, series = "Data") +  
 autolayer(trendcycle(stl\_brickfixed),  
 series = "Trend-cycle") +  
 autolayer(seasadj(stl\_brickfixed),  
 series = "Seasonally Adjusted Data") +  
 ggtitle("brick production fixed seasonality") +  
 scale\_color\_manual(values = c("gray", "red", "blue"),  
 breaks = c("Data", "Trend-cycle", "Seasonally Adjusted Data"))

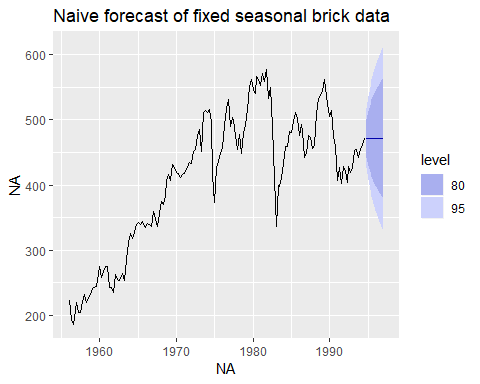


# plot data which are decomposed by STL with changing seasonality  
autoplot(bricksq, series = "Data") +  
 autolayer(trendcycle(stl\_brickchange),  
 series = "Trend-cycle") +  
 autolayer(seasadj(stl\_brickchange),  
 series = "Seasonally Adjusted Data") +  
 ggtitle("brick production changing seasonality") +  
 scale\_color\_manual(values = c("gray", "red", "blue"),  
 breaks = c("Data", "Trend-cycle", "Seasonally Adjusted Data"))

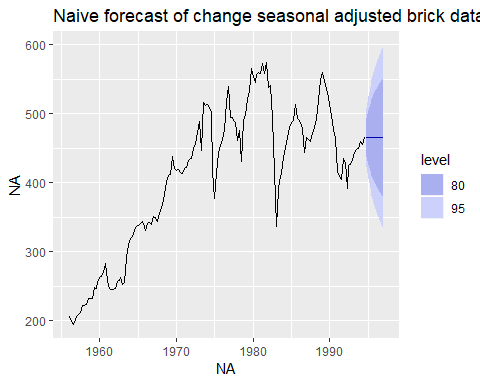


##### c) Use a naÃ¯ve method to produce forecasts of the seasonally adjusted data.

stl\_brickfixed %>% seasadj() %>% naive() %>% autoplot() +   
 ggtitle(label = "Naive forecast of fixed seasonal brick data")

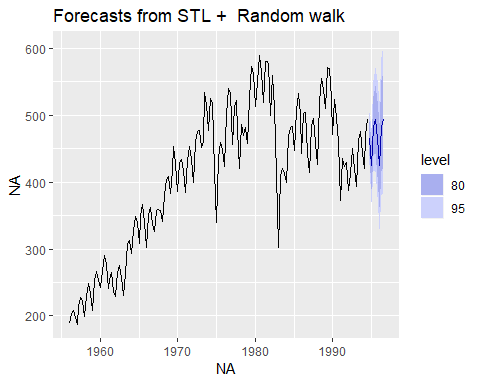


stl\_brickchange %>% seasadj() %>% naive() %>% autoplot() +   
 ggtitle(label = "Naive forecast of change seasonal adjusted brick data")

 From the above we can see that the prediction intervals of seasonally adjusted data decomposed by STL with changing seasonality have smaller range than the one with fixed seasonality. It happened because the variance of the remainder component decreased when the seasonality can be changed.

##### d) Use stlf() to reseasonalise the results, giving forecasts for the original data.

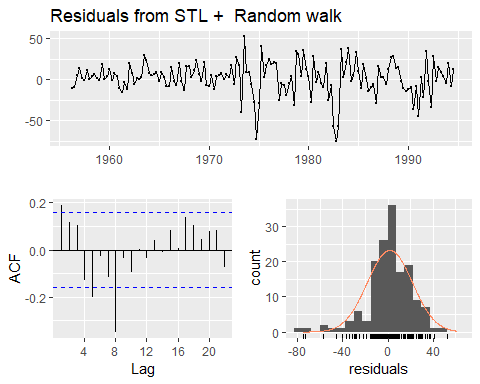
fcast <- stlf(bricksq, method='naive')  
autoplot(fcast)



##### e) Do the residuals look uncorrelated?

checkresiduals(fcast)

## Warning in checkresiduals(fcast): The fitted degrees of freedom is based on  
## the model used for the seasonally adjusted data.

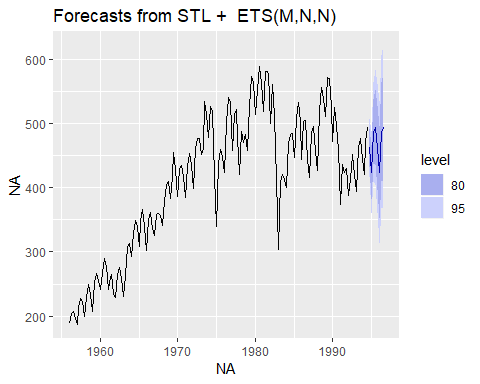


##   
## Ljung-Box test  
##   
## data: Residuals from STL + Random walk  
## Q\* = 40.829, df = 8, p-value = 2.244e-06  
##   
## Model df: 0. Total lags used: 8

The residuals are correlated with each other.

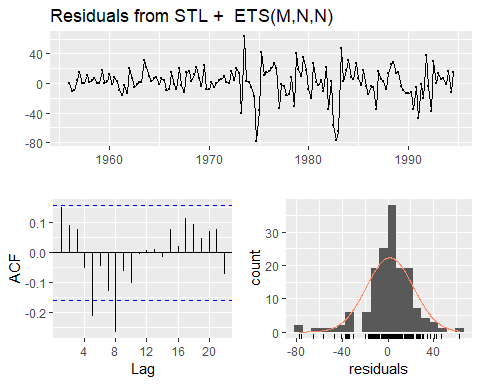
##### f) Repeat with a robust STL decomposition. Does it make much difference?

stlf\_brickrobust <- stlf(bricksq, robust = TRUE)  
autoplot(stlf\_brickrobust)



checkresiduals(stlf\_brickrobust)

## Warning in checkresiduals(stlf\_brickrobust): The fitted degrees of freedom  
## is based on the model used for the seasonally adjusted data.



##   
## Ljung-Box test  
##   
## data: Residuals from STL + ETS(M,N,N)  
## Q\* = 28.163, df = 6, p-value = 8.755e-05  
##   
## Model df: 2. Total lags used: 8

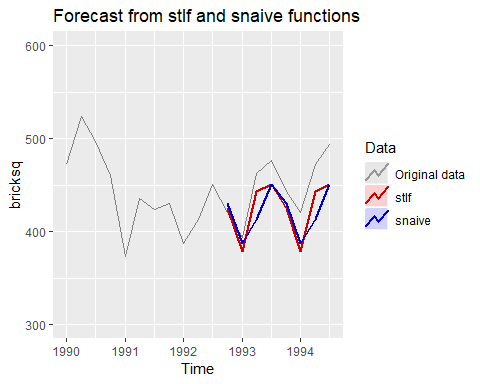
It looked like the autocorrelations became lower generally, but there are still some high values left.

##### g) Compare forecasts from stlf() with those from snaive(), using a test set comprising the last 2 years of data. Which is better?

Splitting data into test and train data set and then applying stlf and snaive on the train data.

#subsetting train data set leaving last 2 years  
train\_brick <- subset(bricksq,   
 end = length(bricksq) - 8)  
#subsetting test data set including only last 2 years data  
test\_brick <- subset(bricksq,  
 start = length(bricksq) - 7)  
  
snaive\_bricksq\_train <- snaive(train\_brick)  
  
stlf\_bricksq\_train <- stlf(train\_brick, robust = TRUE)  
# plot data and forecast results  
autoplot(bricksq, series = "Original data") +  
 autolayer(stlf\_bricksq\_train, PI = FALSE, size = 1,  
 series = "stlf") +  
 autolayer(snaive\_bricksq\_train, PI = FALSE, size = 1,  
 series = "snaive") +  
 scale\_color\_manual(values = c("gray50", "blue", "red"),  
 breaks = c("Original data", "stlf", "snaive")) +  
 scale\_x\_continuous(limits = c(1990, 1994.5)) +  
 scale\_y\_continuous(limits = c(300, 600)) +  
 guides(colour = guide\_legend(title = "Data")) +  
 ggtitle("Forecast from stlf and snaive functions")

## Scale for 'x' is already present. Adding another scale for 'x', which  
## will replace the existing scale.



accuracy(snaive\_bricksq\_train,test\_brick)

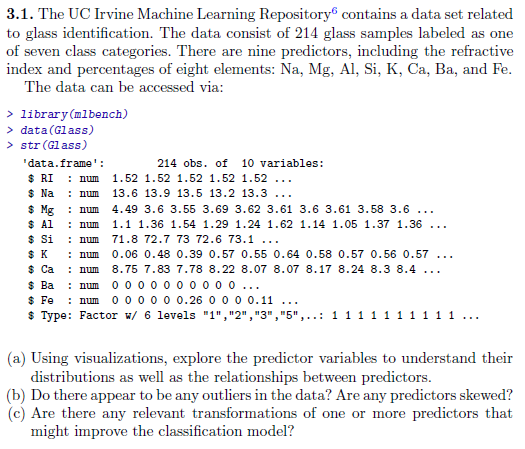
## ME RMSE MAE MPE MAPE MASE  
## Training set 6.174825 49.71281 36.41259 1.369661 8.903098 1.0000000  
## Test set 27.500000 35.05353 30.00000 5.933607 6.528845 0.8238909  
## ACF1 Theil's U  
## Training set 0.8105927 NA  
## Test set 0.2405423 0.9527794

accuracy(stlf\_bricksq\_train,test\_brick)

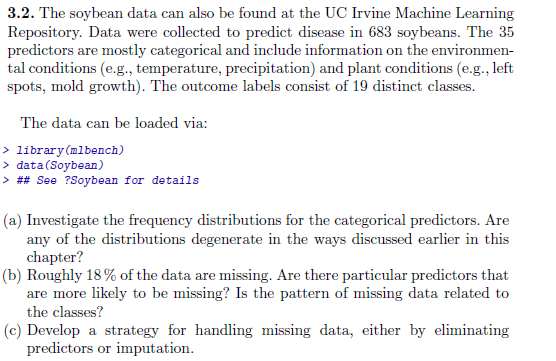
## ME RMSE MAE MPE MAPE MASE  
## Training set 1.452849 20.90158 14.60718 0.3303597 3.599758 0.4011574  
## Test set 23.261210 27.47526 24.55146 5.1141619 5.421365 0.6742575  
## ACF1 Theil's U  
## Training set 0.1652305 NA  
## Test set 0.2030710 0.7163537

From the above forescast plot we can see that the forecasts from stlf function are more similar to the original data than the forecasts from snaive function.stlf function can also use trend, and its seasonality can change over time. The test set have trend with seasonality.Sometimes, different accuracy measures will lead to different results as to which forecast method is best. However, in this case, all of the results point to the stlf method as the best method for this data set.

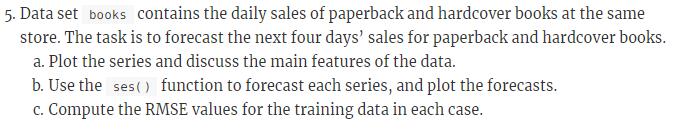
##### KJ# 3.1



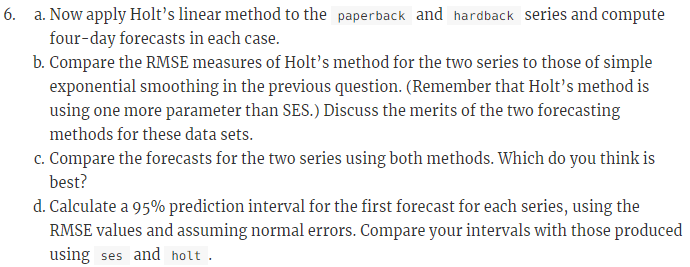
##### KJ# 3.2



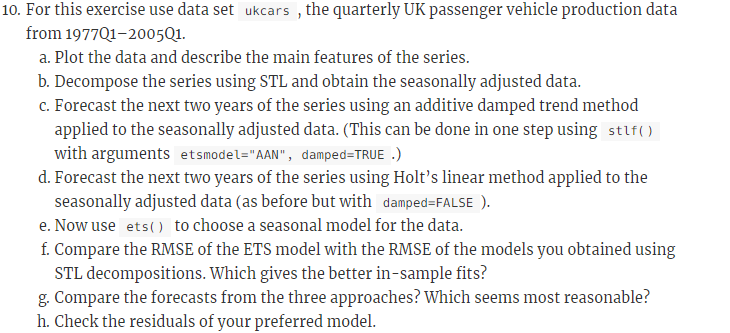
##### HA# 7.5



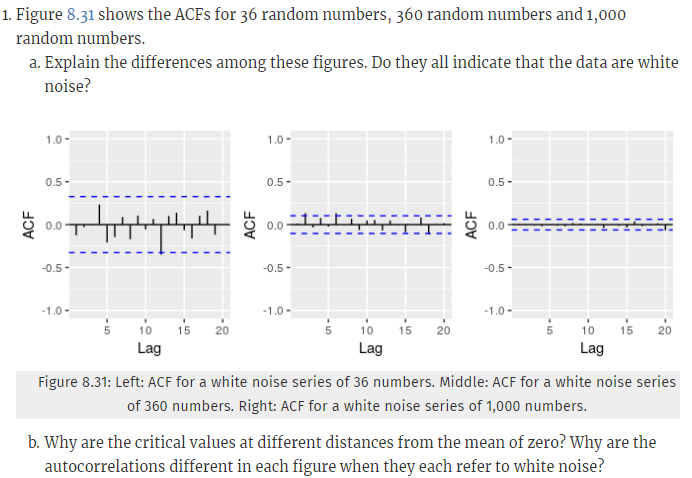
##### HA# 7.6



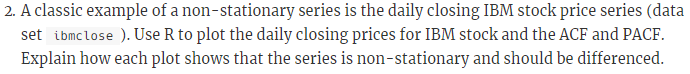
##### HA# 7.10



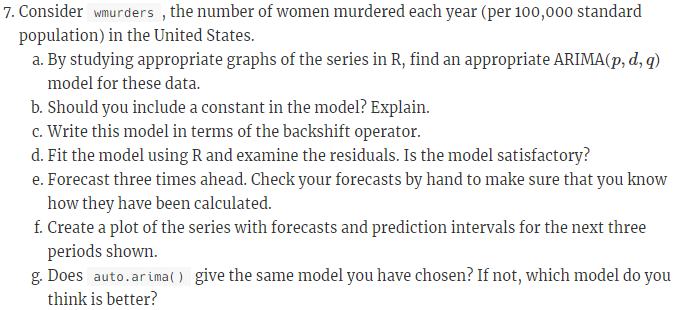
##### HA# 8.1



##### HA# 8.2



##### HA# 8.7



##### HA# 8.12

