DATA 624 Spring 2019: Homework-1

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

March 12, 2019

library(mlbench)

## Warning: package 'mlbench' was built under R version 3.5.3

library(tidyr)

## Warning: package 'tidyr' was built under R version 3.5.3

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.5.3

##   
## 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

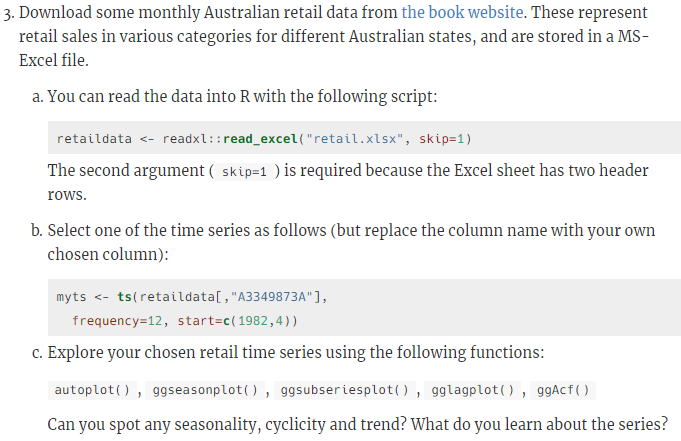
library(gridExtra)

## Warning: package 'gridExtra' was built under R version 3.5.3

##   
## 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)

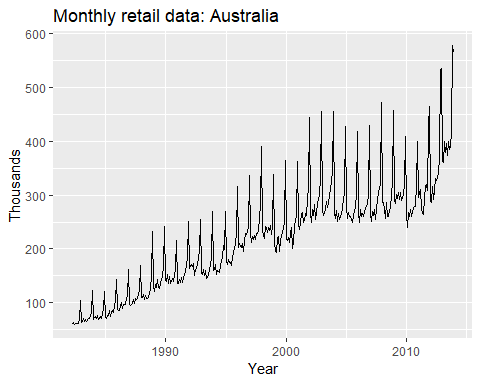
## Apr May Jun Jul Aug Sep  
## 1982 62.4 63.1 59.6 61.9 60.7 61.2

### 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?

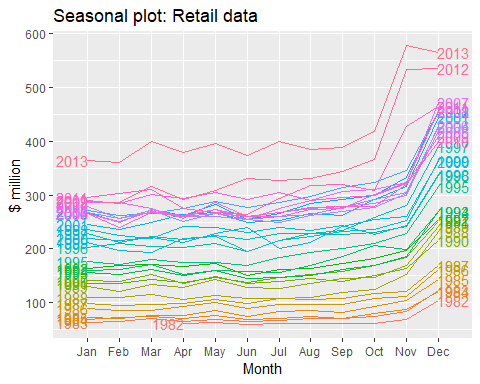
autoplot(myts) +  
 ggtitle("Monthly retail data: Australia") +  
 xlab("Year") +  
 ylab("Thousands")



Time series plot reveals following features:

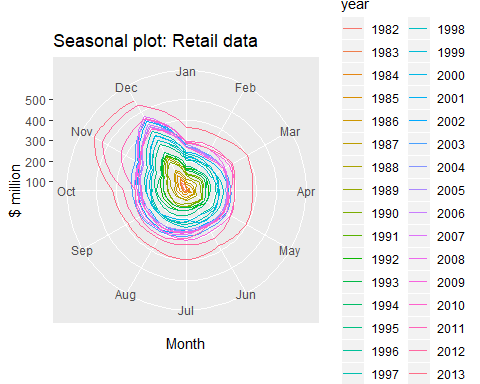
1. There is continuous increment for business in retail sector data from 1982 to2012.
2. Sudden dip was observed in year 2000 and 2010 may be due to resession and slowdown of market.
3. Retail market shows dips for start of year and gradual highs towards end of it.

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

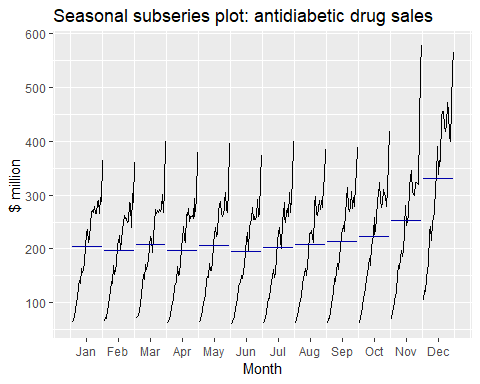


A seasonal plot allows the underlaying pattern to be seen more clearly and are useful in identifying years in which the pattern changes. In our case, above plot makes it clear that there is large jump in retail sector by end of year , mostly in Nov and Dec.patterns also make us notice that there are 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)

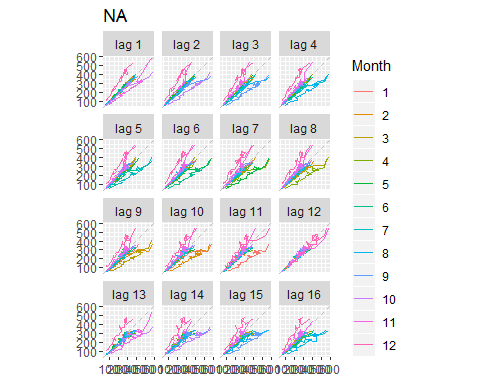


Figure shows Lagged scatterplots for monthly retail data. The relations are strongly positive for month 5,6,7 i.e May,June,July for lag 1,2,3,4 and strongly negative for month 5 in lag 5,6,7.

ggAcf(window(myts),start = 1982)

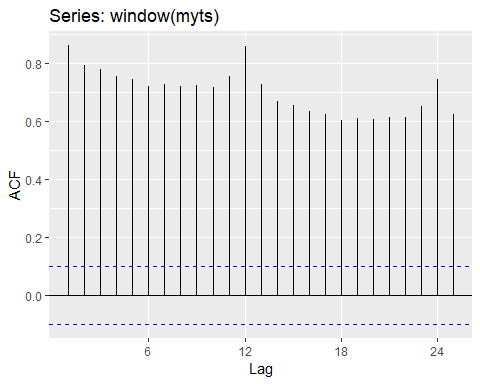
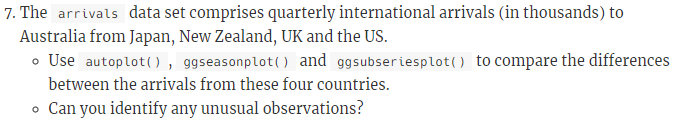
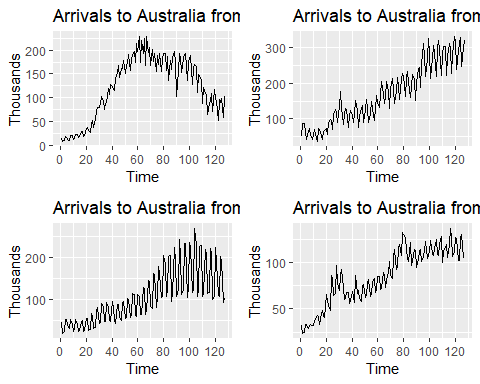


Figure 6 shows Autocorrelation function for monthly retail data for Australia. As we can observe there is no negative co-relation observed. The dashed blue lines indicate whether the correlations are significantly different from zero.

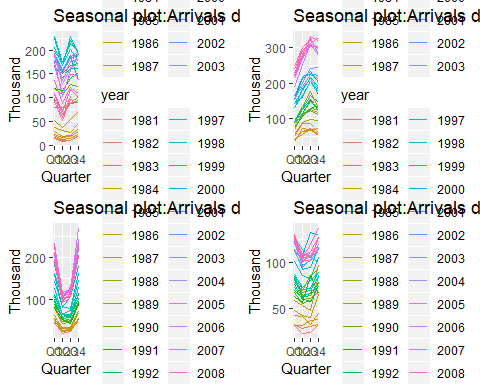
##### HA# 2.7



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)

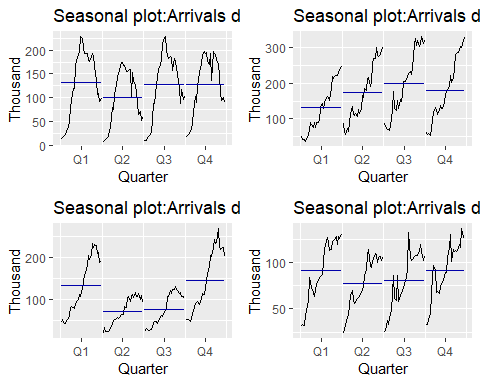


a1 <- ggseasonplot(arrivals[,"Japan"]) + ylab("Thousand") +ggtitle("Seasonal plot:Arrivals data for Japan")  
b1 <- ggseasonplot(arrivals[,"NZ"]) + ylab("Thousand") +ggtitle("Seasonal plot:Arrivals data for NZ")  
c1 <- ggseasonplot(arrivals[,"UK"]) + ylab("Thousand") +ggtitle("Seasonal plot:Arrivals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



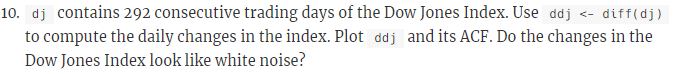
The most unusual observation is obsereved for arrival data in ‘UK’ for Q2 and Q3.The arrival numbers are drastically down but gradually increased in Q4.At the same time, New zealand shows highest arrival rates for Q3.

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)

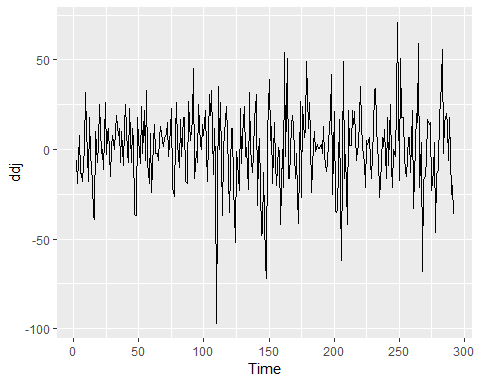


Seasonal subseries plot emphasises theseasonal patterns of data.The horizontal lines indicate means for each quarter.This form of plot enables the underlaying seasonal pattern to be seen clearly ,and also shows the changes in seasonality over time.If we observe seasonal plots for all four countries,New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

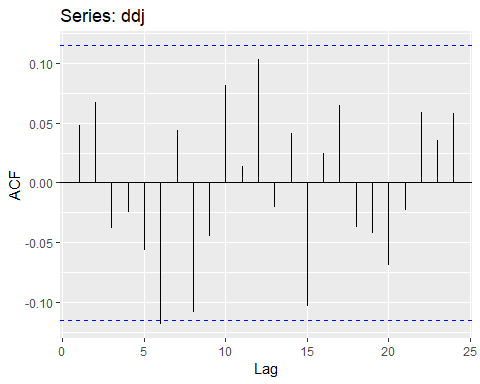
##### HA# 2.10



ddj <- diff(dj)  
autoplot(ddj)

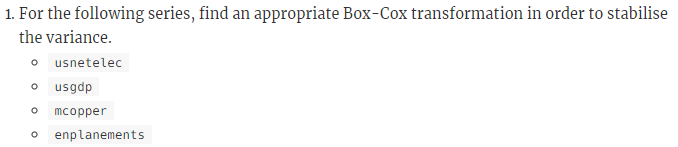


ggAcf(ddj)

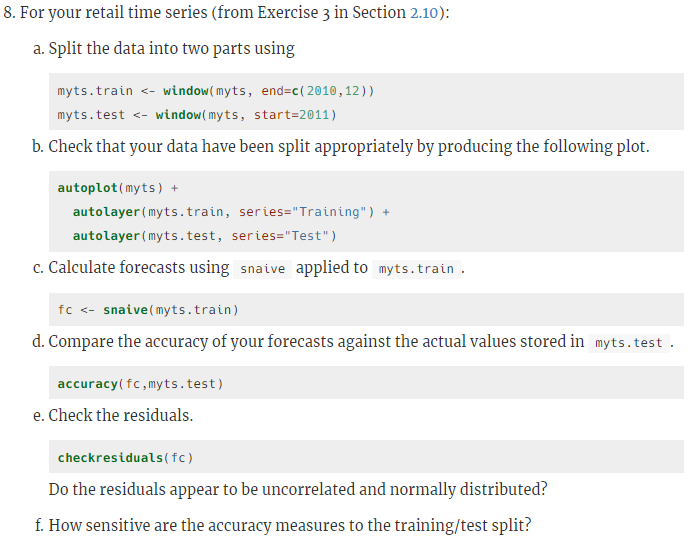


For white noise series, each autocorelation is expected to be close to zero.If one or more large spikes are outside these bounds of blue dotted lines, or if more than 5% of spikes are outside this bounds,then series is not a white noise. In our case,all the autocorelations lie within these limits,confirming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

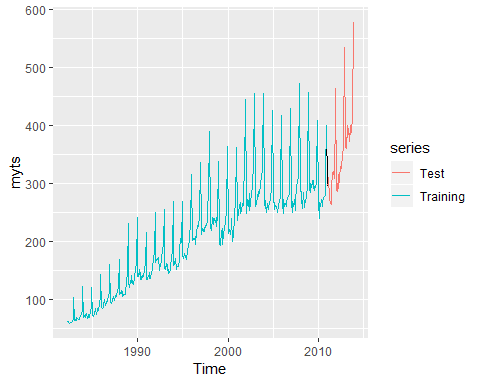
retaildata <- readxl::read\_excel("retail.xlsx", skip=1)  
myts <- ts(retaildata[,"A3349873A"],  
 frequency=12, start=c(1982,4))

##### 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)

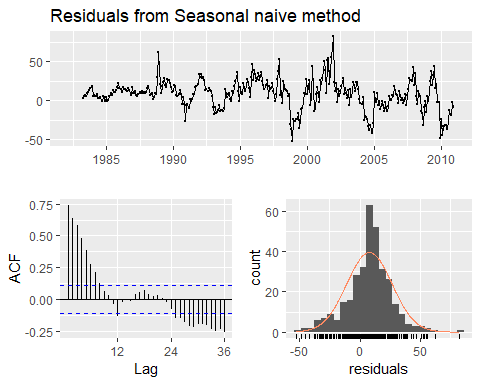
##### d) Compare the accuracy of your forecasts against the actual values stored in myts.test.

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.

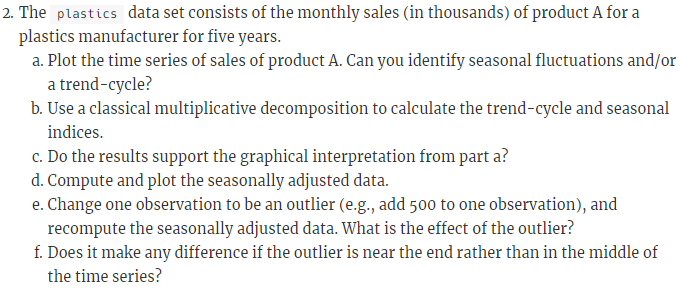
##### f) How sensitive are the accuracy measures to the training/test split?

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

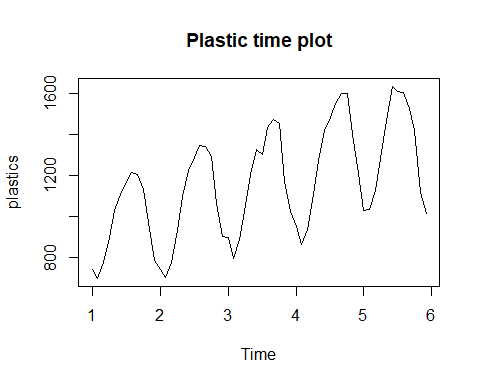
plastics

## 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?

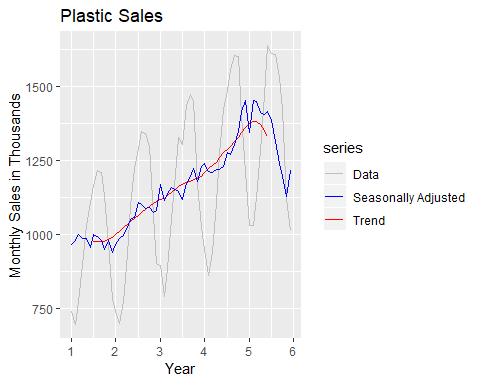
Yes, the results support the graphical interpretation.The graph indicates that the summer months have higher seasonal indices than the winter months

##### 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?

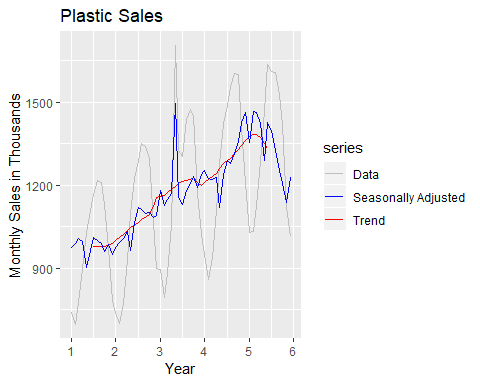
Here we are adding 500 to 29th observation

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?

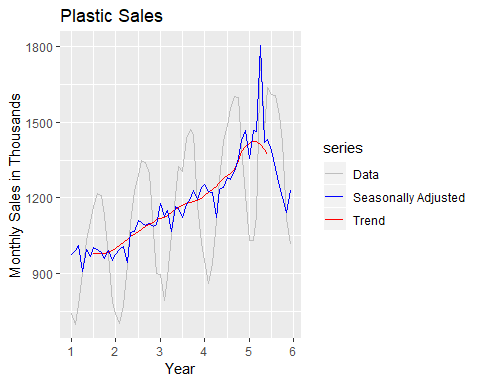
Here we are adding 500 to 52nd observation

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

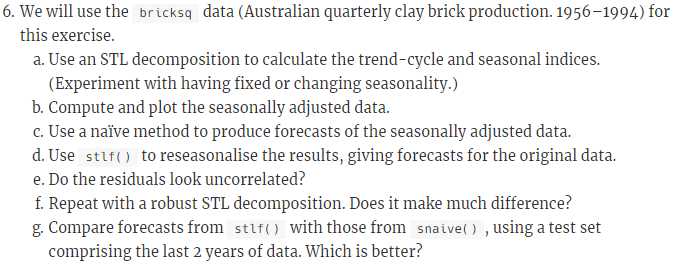
Plot showing the trend-cycle component and the seasonally adjusted data, along with the original data with modified outlier data

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"))

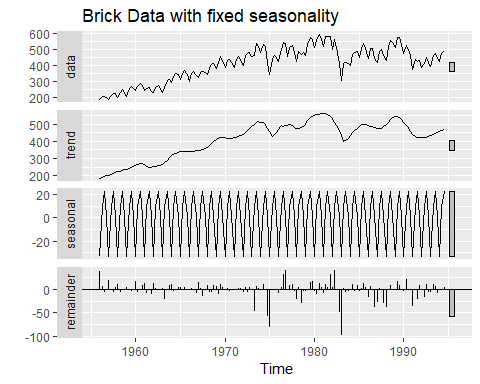
## Warning: Removed 12 rows containing missing values (geom\_path).

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

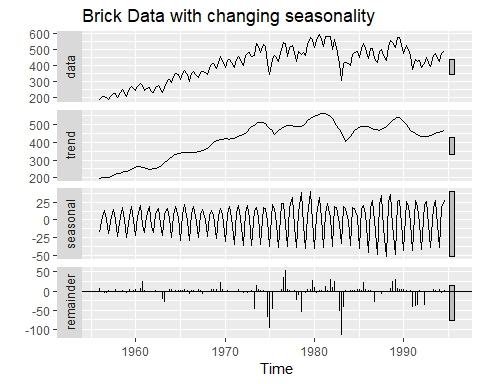
##### HA# 6.6

 ##### 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")



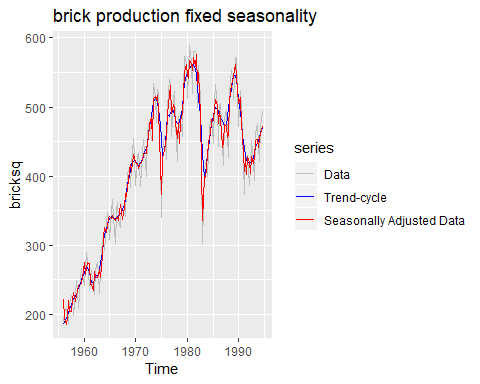
stl\_brickchange <- stl(bricksq,s.window = 5,robust = TRUE)  
autoplot(stl\_brickchange) +ggtitle("Brick Data with changing seasonality")



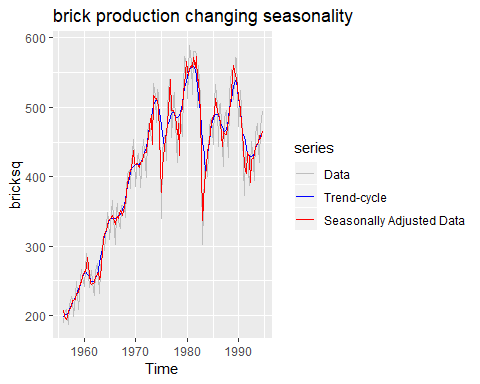
##### b) Compute and plot the seasonally adjusted data.

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"))

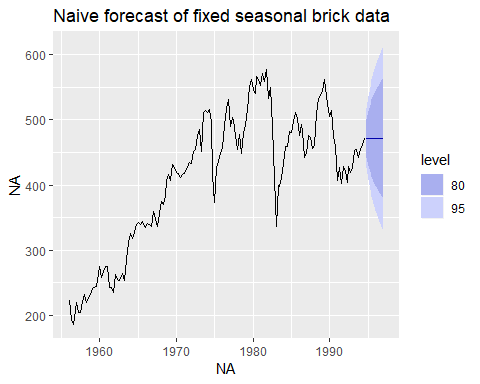


# plot data which are decomposed by STL with changing seasonality  
autoplot(bricksq, series = "Data") +  
 autolayer(trendcycle(stl\_brickchange),  
 series = "Trend-cycle") +  
 autolayer(seasadj(stl\_brickchange),  
 series = "Seasonally Adjusted Data") +  
 ggtitle("brick production changing seasonality")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

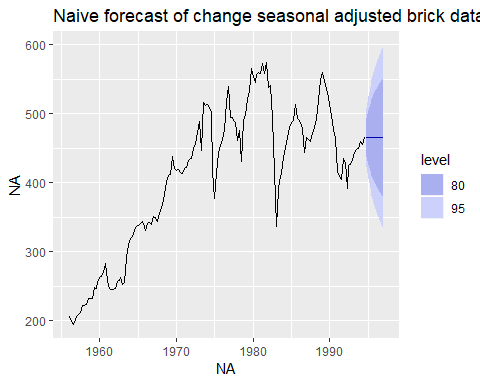


##### c) Use a naÃ¯ve method to produce forecasts of the seasonally adjusted data.

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

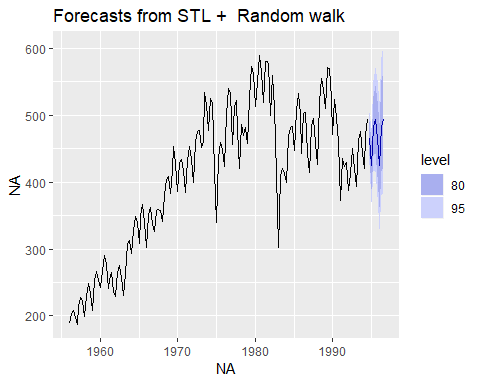


stl\_brickchange %>% seasadj() %>% naive() %>% autoplot() +   
 ggtitle(label = "Naive forecast of change seasonal adjusted brick data")

 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.

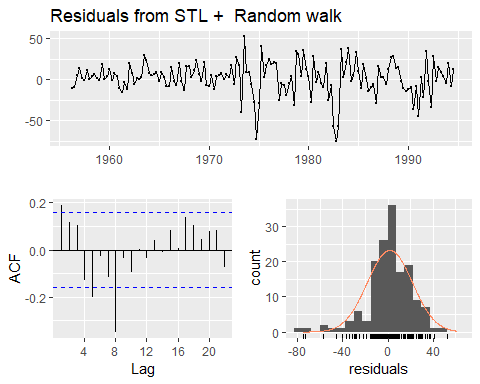
fcast <- stlf(bricksq, method='naive')  
autoplot(fcast)



##### 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.

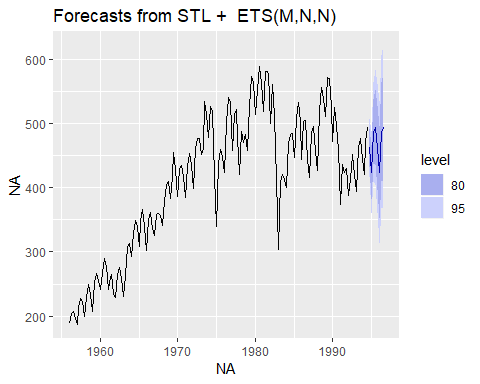


##   
## Ljung-Box test  
##   
## data: Residuals from STL + Random walk  
## Q\* = 40.829, df = 8, p-value = 2.244e-06  
##   
## Model df: 0. Total lags used: 8

The residuals are correlated with each other.

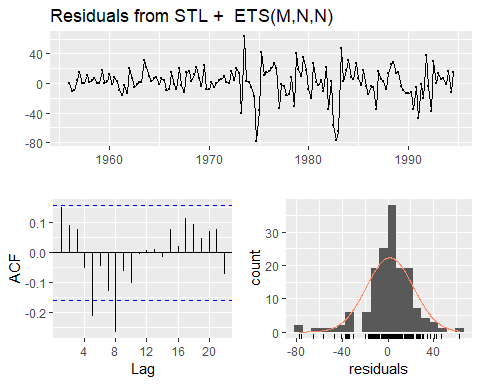
##### f) Repeat with a robust STL decomposition. Does it make much difference?

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

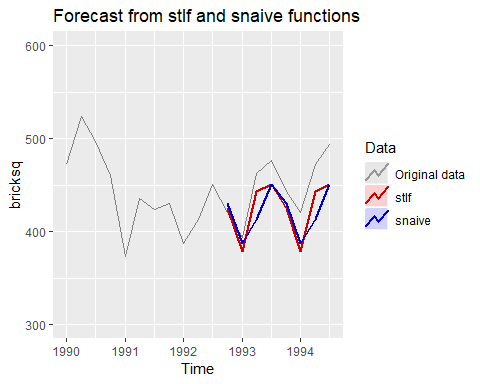
It looked like the autocorrelations became lower generally, but there are still some high values left.

##### 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)

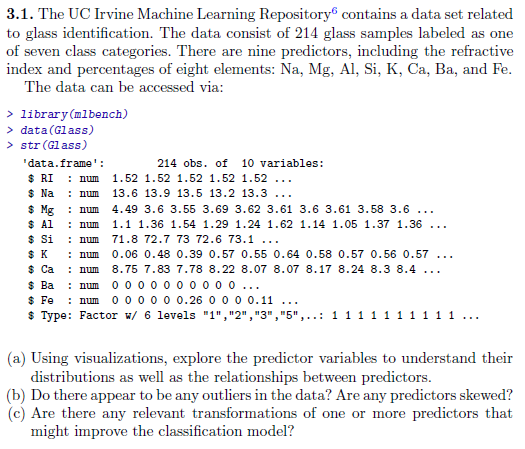
## ME RMSE MAE MPE MAPE MASE  
## Training set 6.174825 49.71281 36.41259 1.369661 8.903098 1.0000000  
## Test set 27.500000 35.05353 30.00000 5.933607 6.528845 0.8238909  
## ACF1 Theil's U  
## Training set 0.8105927 NA  
## Test set 0.2405423 0.9527794

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

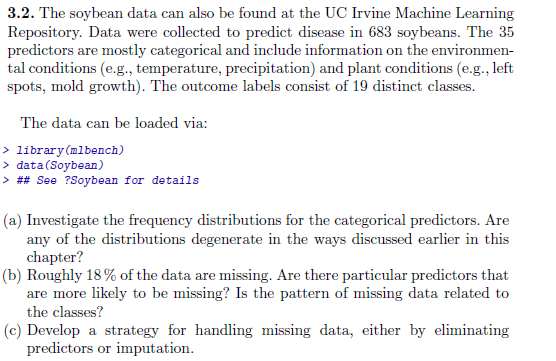
## ME RMSE MAE MPE MAPE MASE  
## Training set 1.452849 20.90158 14.60718 0.3303597 3.599758 0.4011574  
## Test set 23.261210 27.47526 24.55146 5.1141619 5.421365 0.6742575  
## ACF1 Theil's U  
## Training set 0.1652305 NA  
## Test set 0.2030710 0.7163537

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.

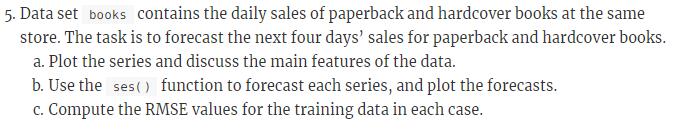
##### KJ# 3.1



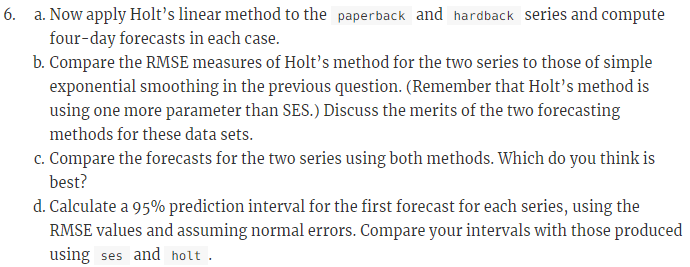
##### KJ# 3.2



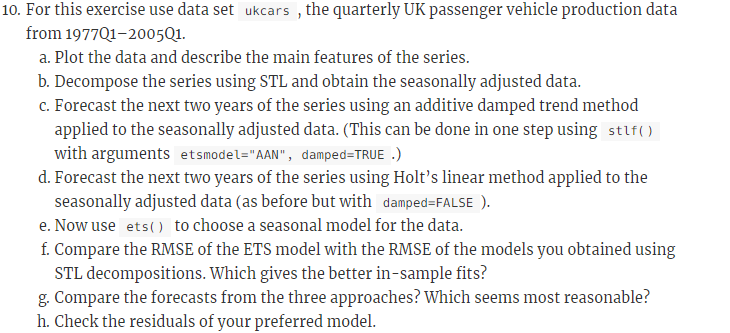
##### HA# 7.5



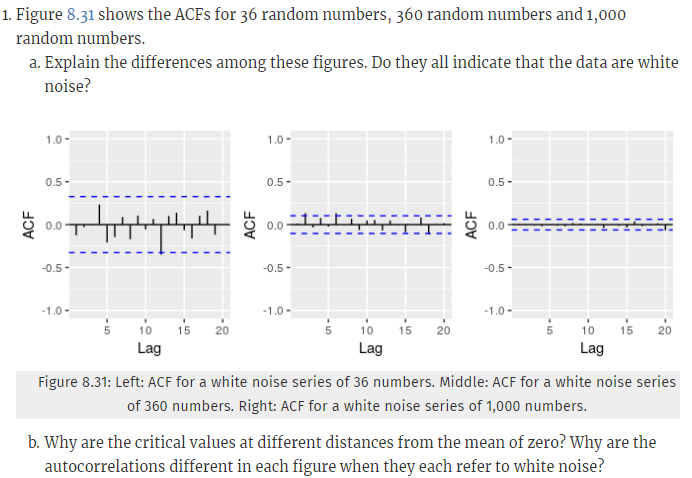
##### HA# 7.6



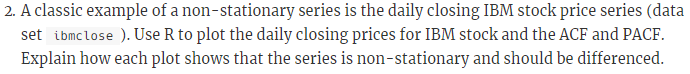
##### HA# 7.10



##### HA# 8.1



##### HA# 8.2

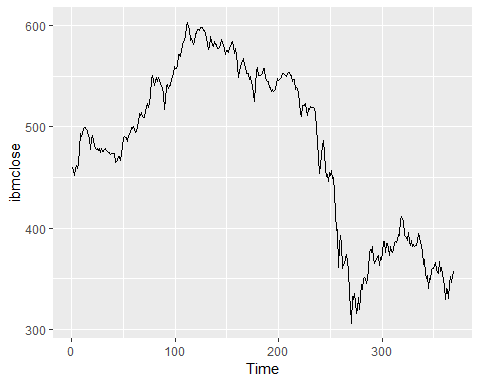


# HW8.11.2

This problem is to plot the IBM closing stock price of IBM over a period of time, and examine it’s lack of “stationariness”.

First lets just plot the raw data. From the plot, it is obvious (as expected) that the data is not stationary. There is an upward trend in the first third, then an accelerating downward trend in the second third, while the last third doesn’t exhibit a clear trend. The variability seems more pronounced in the first half of the graph, but that is perhaps only subjective. There doesn’t seem to be obvious seasonality, or cyclicity.

library(fpp2)  
autoplot(ibmclose)



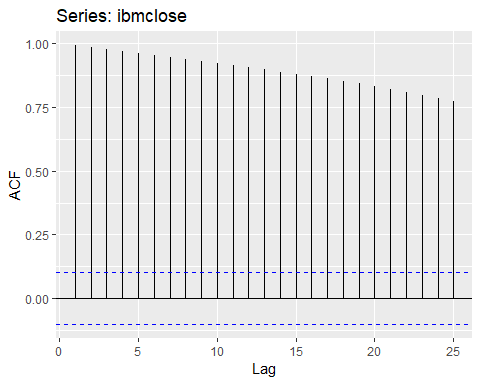
## ACF PACF Plots

Below we print the plots of the ACF (auto correlation function) and the PACF (partial ACF) to further examine the non-stationariness as well as to how we might need to difference the series to make it stationary.

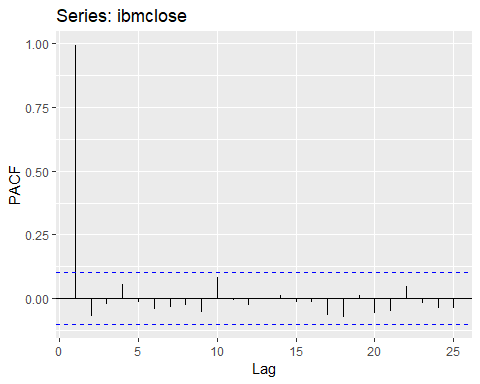
Looking at the ACF plot we see it is decaying slowly, a clear indication of non-stationary data. Also the value of R is quite large, further indicating the series isn’t stationary.

With the PACF model, we see 1 really large spike, this indicates differencing of 1 is probably all that is needed to make the data stationary.

ggAcf(ibmclose)

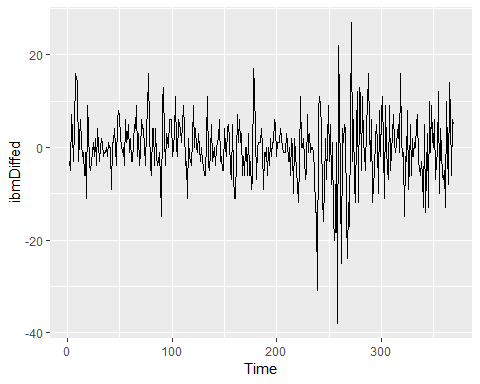


ggPacf(ibmclose)

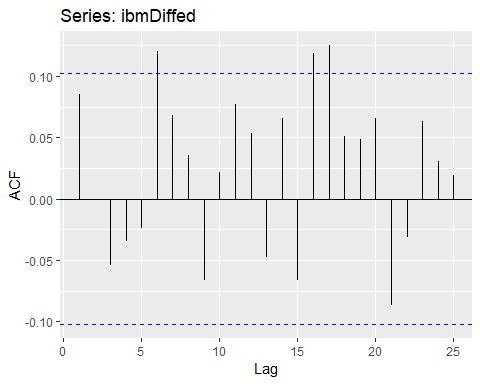
 ###Rerun After taking one difference.

Running the plots after we use the “diff” (difference) command, we see that data now doesn’t show upward/downward trend, but the variance looks like it might be slightly larger, towards the end of the sequence. In any case the ACF and PACF plots while still containing some spikes above the upper signifcance line, are only very slighly above, so the series is much closer to being a stationary series.

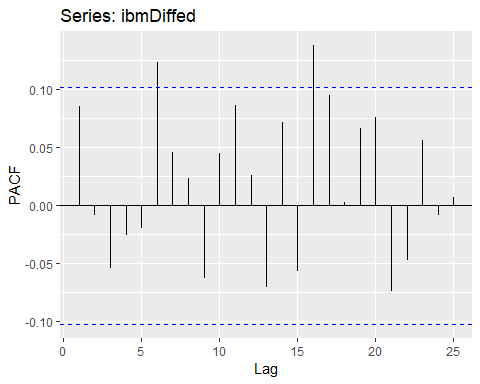
ibmDiffed <- diff(ibmclose)  
autoplot(ibmDiffed)



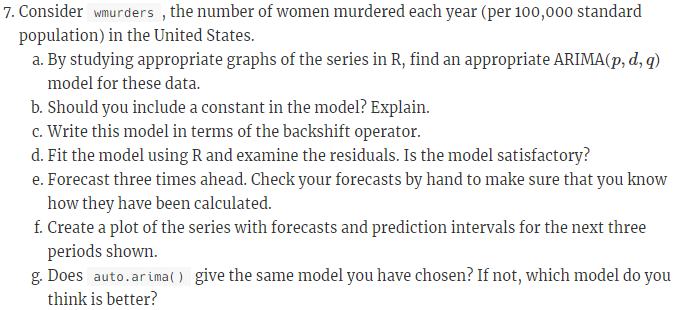
ggAcf(ibmDiffed)



ggPacf(ibmDiffed)



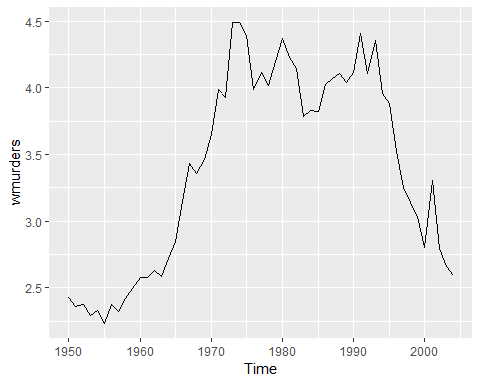
##### HA# 8.7



This problem deals with the dataset wmurders, which is a series containing the number of women murdered each year in the United States. First lets run some plots to examine the data.

From the raw plot of the data, we can definitely see obvious trend, with the murder rate climbing significantly through the 50’s and 60’s and then again with a strong decline from the early 90’s to the early 2000’s (until the end of the data). Seasonal differences would not apply as the data is yearly and it is unlikely there is seasonal cycles of multiple years.

autoplot(wmurders)



Unit Tests: Rather than just relying on subject “eyeballing” of the plotted data, let’s run some unit tests for more objectivity. We will run the KPSS (Kwiatkowski-Phillips-Schmidt-Shin) and the Augemented Dickey Fuller Tests. The results are below for both the intial raw data and the data differenced once.

library(urca)  
library(tseries)  
summary(ur.kpss(wmurders))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 3 lags.   
##   
## Value of test-statistic is: 0.6331   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

summary(ur.kpss(diff(wmurders)))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 3 lags.   
##   
## Value of test-statistic is: 0.4697   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

adf.test(wmurders)

##   
## Augmented Dickey-Fuller Test  
##   
## data: wmurders  
## Dickey-Fuller = -0.29243, Lag order = 3, p-value = 0.9878  
## alternative hypothesis: stationary

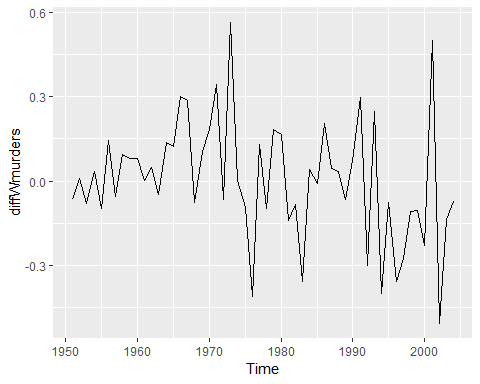
adf.test(diff(wmurders))

##   
## Augmented Dickey-Fuller Test  
##   
## data: diff(wmurders)  
## Dickey-Fuller = -3.7688, Lag order = 3, p-value = 0.02726  
## alternative hypothesis: stationary

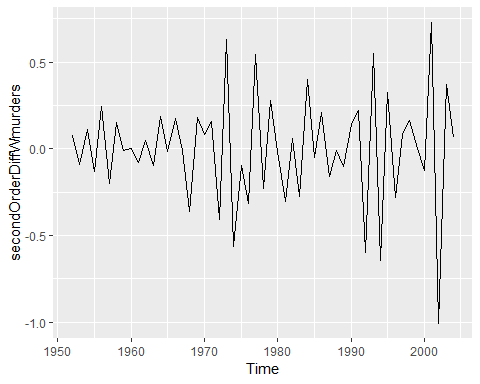
Interestingly, if we look at the KPSS test there doesn’t seem to be a huge need to difference at all, the test statistics is under 1; differencing once does make that number even smaller though. If we look at the ADF test though, the first result of 0.98 shows a strong need to difference, the result after differencing is much smaller at .0273, which indicates likely minimal trend, but we might wish to consider differencing twice to reduce further.

Anyhow, lets plot the differenced data (below). From the plot of the differenced data, we see that the trend has been removed, although there definitely still seems to be some increase in variation as time goes along, with small variation in the early stages and larger ones as we move forward in time. Plotting the 2nd order difference removes much of that variation, so perhaps 2 will be good for the “d” of the ARIMA model.

diffWmurders <- diff(wmurders)  
autoplot(diffWmurders)

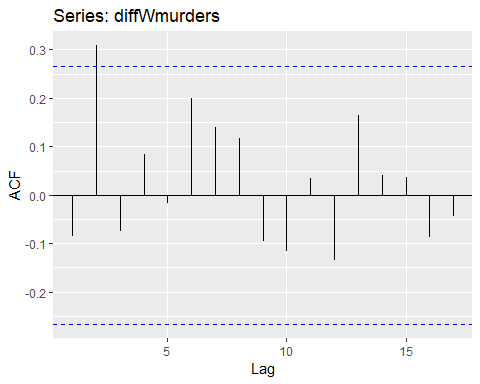


secondOrderDiffWmurders <- diff(diff(wmurders))  
autoplot(secondOrderDiffWmurders)

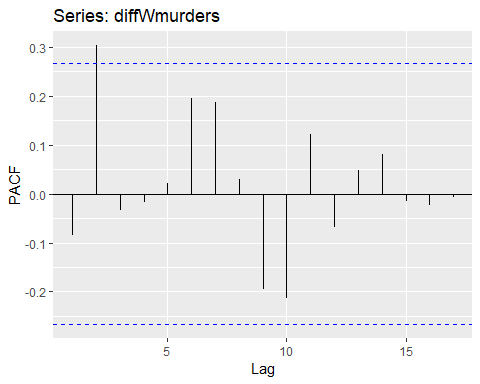


Now looking at the ACF and the PACF plots (below), we see one time containing a spike into the “significance” range. This was at time period 2 for both graphs. This suggests maybe having a “p” value of at least 1 for the ARIMA model.

ggAcf(diffWmurders)

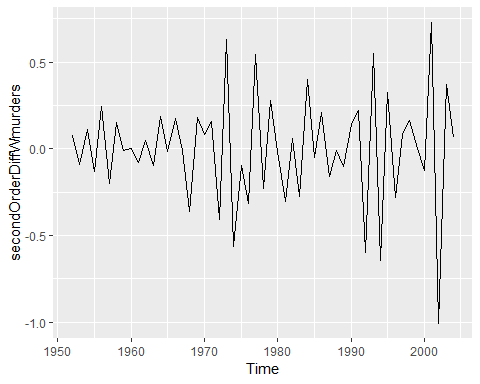


ggPacf(diffWmurders)

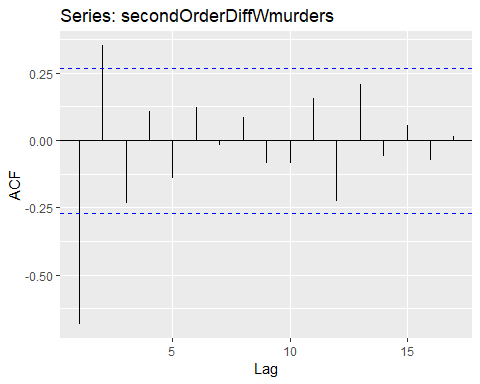


If we want to consider differencing a second time, let’s look at the two ACF plots. We see that the first lag in both plots is greater than negative 0.5, which may mean the plot is overdifferenced. But leaving that aside for now, we see with both plots that they are sinusoidal, indicating that both a p and q parameter would make sense. For the PACF, the first lag is significant so we should adding a parameter for the “q” or Moving Average (MA) part of the model. In the ACF the first two lags are significant, indicating when we difference twice we should add 2 to the “p” or Autoregressive ARIMA parameter, but since the second is so much smaller and guidance from <https://people.duke.edu/~rnau/arimrule.htm> suggests that if we have both MA and AR terms, we may want to reduce one or both of those by 1, since they may cancel each other out.

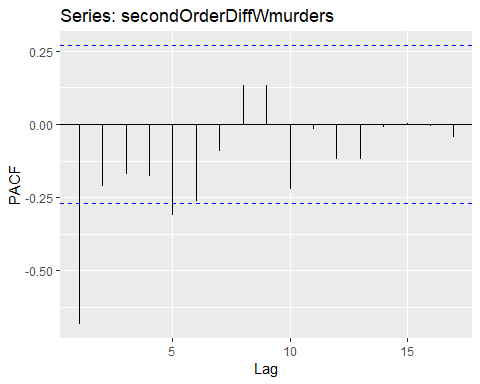
secondOrderDiffWmurders <- diff(diff(wmurders))  
autoplot(secondOrderDiffWmurders)



ggAcf(secondOrderDiffWmurders)



ggPacf(secondOrderDiffWmurders)



So, we are left with a few choices for models, if we go with 1 order of differencing as we saw potential of over differencing with 2 orders, our model is (1,1,1). If we choose 2 orders of differencing, we are left with either (1,2,2) as the likely choice, with the (1,2,1) also possible, if we lower the higher order MA term based on the comment from the Duke website. Let’s look at the 3 results:

Arima(wmurders, c(1,1,1))

## Series: wmurders   
## ARIMA(1,1,1)   
##   
## Coefficients:  
## ar1 ma1  
## -0.8255 0.6916  
## s.e. 0.1941 0.2333  
##   
## sigma^2 estimated as 0.04483: log likelihood=8.18  
## AIC=-10.36 AICc=-9.88 BIC=-4.39

Arima(wmurders, c(1,2,2))

## Series: wmurders   
## ARIMA(1,2,2)   
##   
## Coefficients:  
## ar1 ma1 ma2  
## -0.7677 -0.2812 -0.4977  
## s.e. 0.2350 0.2879 0.2762  
##   
## sigma^2 estimated as 0.04552: log likelihood=7.38  
## AIC=-6.75 AICc=-5.92 BIC=1.13

Arima(wmurders, c(1,2,1))

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

Looking at these results, the model with the lowest AIC values is actually (1,1,1). So, the fact that has the lowest AIC, the fewest params (simpler) and the lowest variation and the least differencing, then this perhaps is the best model. But (not shown), in checking the residuals for the (1,1,1) they were not as satisfactory. So that leaves the other models, of the two, the (1,2,2) has very marginally better results for AIC and the BIC is noticably worse, and since the it is slighly more complex, I will suggest the best model is ARIMA(1,2,1),

## 8.11.7.B

Should a constant be part of the model? Normally if the model is differenced (which in our case is yes) and especially differenced more than once, then a constant is not included, so we shall not include a constant in our model.

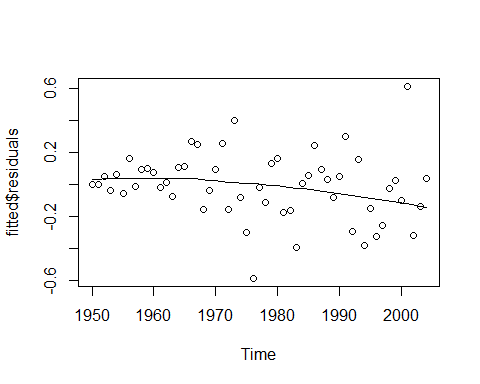
## 8.11.7.C

Using backshift notation, describe the differencing for our model. Since we are differencing twice, we need to compute . The full equation is:

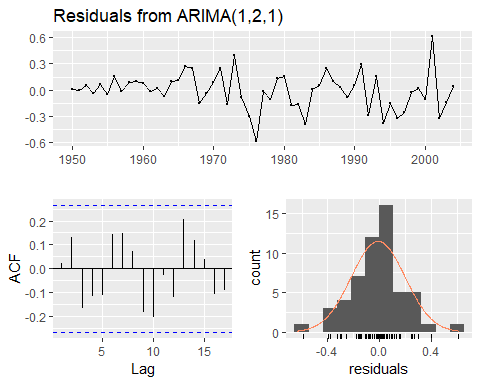
## 8.11.7.D

Fitting our ARIMA model in R and examing the residuals, see the plots below:

fitted <- Arima(wmurders, c(1,2,1))  
scatter.smooth(fitted$residuals)



checkresiduals(fitted)



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

Box.test(fitted$residuals, type="Ljung-Box")

##   
## Box-Ljung test  
##   
## data: fitted$residuals  
## X-squared = 0.027498, df = 1, p-value = 0.8683

The residuals look reasonable. They are evenly balanced around the mean. There are some outliers at 1977 and 2001, this is where the actual data had a sharp change in direction, i.e., there were a greatly fewer murders in 1977 and greatly more in 2001 than in the prior year. Not too suprisingly the model isn’t handling those well. The residuals do have a downward slope, so the model isn’t perfect. Also the ACF plot shows that the residuals are in tolerance, so basically these residuals are acceptable.

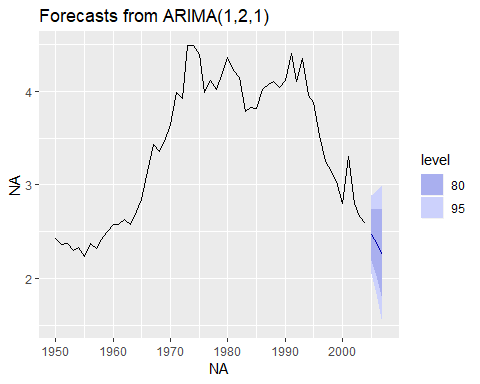
## 8.11.7.E and 8.11.7.F

Forecasting using our model.

forecast(fitted, h=3)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2005 2.470660 2.194836 2.746484 2.048824 2.892496  
## 2006 2.363106 1.986351 2.739862 1.786908 2.939304  
## 2007 2.252833 1.765391 2.740276 1.507354 2.998313

autoplot(forecast(fitted, h=3))



## 8.7.G

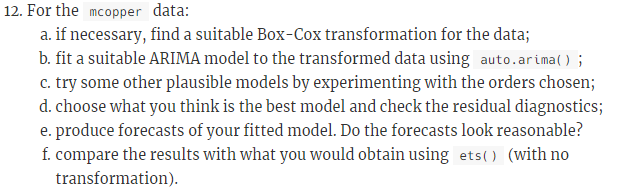
Comparing our model to what auto.ARIMA has generated.

Below we see that the auto.ARIMA function also chose a (1,2,1) model. Of course as noted way above, the (1,2,2) model could also be considered as an acceptable choice, and if one plots this (not done here for brevity, we see similar results to the above forecast plots).

auto.arima(wmurders)

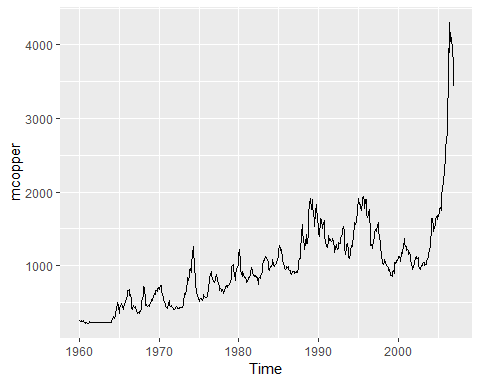
## Series: wmurders   
## ARIMA(1,2,1)   
##   
## Coefficients:  
## ar1 ma1  
## -0.2434 -0.8261  
## s.e. 0.1553 0.1143  
##   
## sigma^2 estimated as 0.04632: log likelihood=6.44  
## AIC=-6.88 AICc=-6.39 BIC=-0.97

##### HA# 8.12



This problem works on the mcopper dataset. First lets look at the data.

autoplot(mcopper)

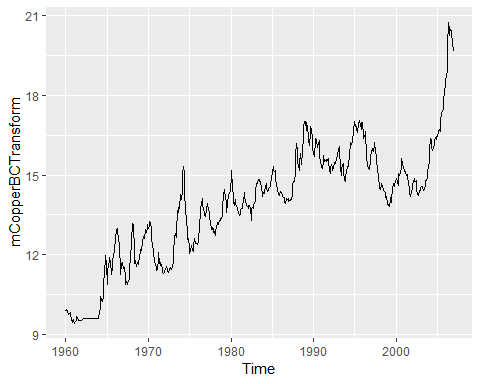


This looks like it needs a transformation, as the data shows variation over time. So let’s find an appropriate lambda value (0.191) for the transformation and plot that. We see doing this, that the variation is much less now from the early (left) portion of the time series to later entries.

lambda <- BoxCox.lambda((mcopper))  
lambda

## [1] 0.1919047

mCopperBCTransform <- BoxCox(mcopper, lambda)  
autoplot(mCopperBCTransform)



## 8.12.B

Now, let’s find an ARIMA function that will allow us to forecast. Using the R auto.Arima function we see that it predicts that an (0,1,1) model is appropriate.

auto.arima(mcopper, lambda = lambda)

## Series: mcopper   
## ARIMA(0,1,1)   
## Box Cox transformation: lambda= 0.1919047   
##   
## Coefficients:  
## ma1  
## 0.3720  
## s.e. 0.0388  
##   
## sigma^2 estimated as 0.04997: log likelihood=45.05  
## AIC=-86.1 AICc=-86.08 BIC=-77.43

## 8.12.C

Let’s experiment and see if other models might work better. First let’s adjust to the auto.Arima’s parameters to allow a deeper search of models.

auto.arima(mcopper, lambda = lambda, stepwise = FALSE, approximation = FALSE)

## Series: mcopper   
## ARIMA(0,1,1)   
## Box Cox transformation: lambda= 0.1919047   
##   
## Coefficients:  
## ma1  
## 0.3720  
## s.e. 0.0388  
##   
## sigma^2 estimated as 0.04997: log likelihood=45.05  
## AIC=-86.1 AICc=-86.08 BIC=-77.43

We see that the model returned was the same (0,1,1). So the autofunction even with deeper analysis, still feels (0,1,1) is appropriate.

We can do our own analysis (like in 8.11.7 above). Doing the unit tests, we see that both the KPSS and Augmented Dickey fuller show lots of trend without differencing as KPSS test statistic was well over 1 and the ADF results was substantially above 0.05 p-value, indicating trend. Applying an order of differencing of 1, removed any signficant trend from the models as shown by these two tests. So the having a “d” of one as indicated by auto.Arima seems correct.

summary(ur.kpss(mCopperBCTransform))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 6 lags.   
##   
## Value of test-statistic is: 6.2659   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

summary(ur.kpss(diff(mCopperBCTransform)))

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: mu with 6 lags.   
##   
## Value of test-statistic is: 0.0573   
##   
## Critical value for a significance level of:   
## 10pct 5pct 2.5pct 1pct  
## critical values 0.347 0.463 0.574 0.739

adf.test(mCopperBCTransform)

##   
## Augmented Dickey-Fuller Test  
##   
## data: mCopperBCTransform  
## Dickey-Fuller = -2.9549, Lag order = 8, p-value = 0.1741  
## alternative hypothesis: stationary

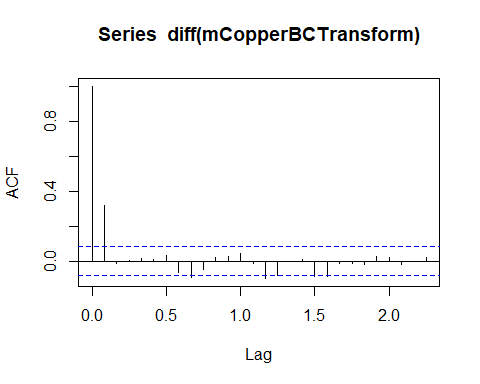
adf.test(diff(mCopperBCTransform))

## Warning in adf.test(diff(mCopperBCTransform)): p-value smaller than printed  
## p-value

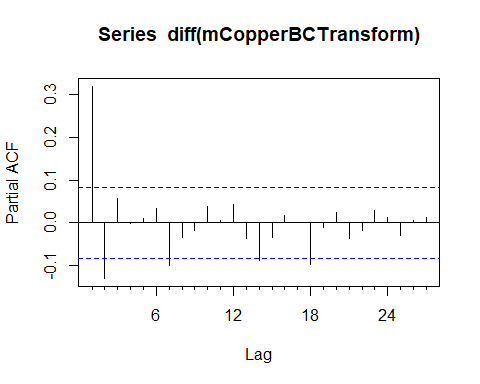
##   
## Augmented Dickey-Fuller Test  
##   
## data: diff(mCopperBCTransform)  
## Dickey-Fuller = -8.2555, Lag order = 8, p-value = 0.01  
## alternative hypothesis: stationary

Lets run ACF and PACF tests to see if the “p” and “q” values chosen by the auto.Arima seem accurate as well. Note due to substantial trend, we will only do the ACF tests on the data we already differenced.

acf(diff(mCopperBCTransform))



Pacf(diff(mCopperBCTransform))



We see on both the ACF and PACF two lags substantially above the significance threshold, which indicates trying a (2,1,2) model. Note there are a few other lags on each chart that are borderline significant, but making a model (5,1,5) is probably not wise, as it is likely we are overfitting at that point. So let’s try an ARIMA model of (2,1,2) and look at the results:

Arima(mcopper, c(2,1,2), lambda = lambda)

## Series: mcopper   
## ARIMA(2,1,2)   
## Box Cox transformation: lambda= 0.1919047   
##   
## Coefficients:  
## ar1 ar2 ma1 ma2  
## -0.9606 -0.1275 1.3337 0.4592  
## s.e. 0.1601 0.1194 0.1513 0.1113  
##   
## sigma^2 estimated as 0.05002: log likelihood=46.25  
## AIC=-82.5 AICc=-82.4 BIC=-60.84

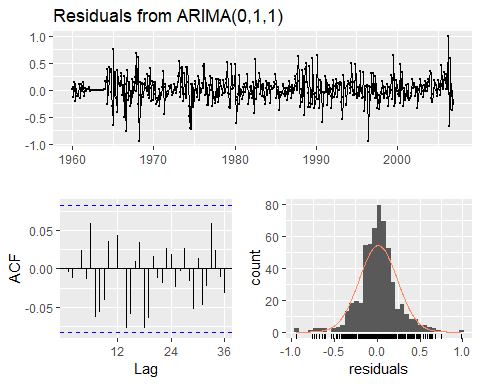
So the AIC is worse with this model. The log likelihood is pretty close, but once you add in the fact that we have 5 parameters vs 2 for the one the auto.arima found, then we see that AIC is noticeably lower for our choice. (Note AIC is a function of log likelihood and the paramters).

Therefore we will chose the (0,1,1) as the best model.

## 8.11.12.D

Using the (0,1,1) model let’s examine the residuals. The plots are below. We see that the residuals look like white noise. They are even around the 0 line with few outliers. The ACF plot shows no residuals in the “significant” category and the Ljung-Box test shows a very high p value, again indicating residuals are essentially “white noise”.

fitted <- Arima(mcopper, c(0,1,1), lambda = lambda)  
#fitted$residuals  
checkresiduals(fitted)



##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(0,1,1)  
## Q\* = 22.913, df = 23, p-value = 0.4659  
##   
## Model df: 1. Total lags used: 24

Box.test(fitted$residuals, type="Ljung-Box")

##   
## Box-Ljung test  
##   
## data: fitted$residuals  
## X-squared = 0.0099289, df = 1, p-value = 0.9206

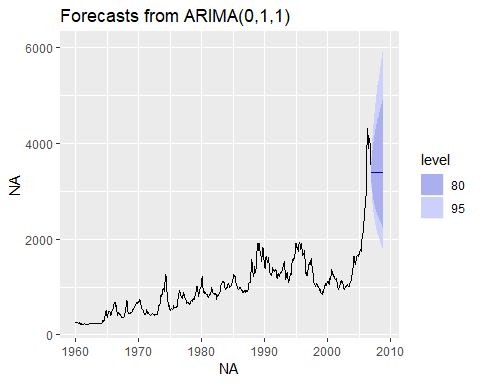
## 8.11.12.E

Forecasting the time series copper data. we see that the forecast is essentially a straight line. But the confidence levels are quite wide, meaning our predictions of a straight line is not that strong a prediction, the data could vary signifcantly.

forecast(fitted)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## Jan 2007 3387.143 3188.095 3596.115 3086.619 3710.882  
## Feb 2007 3387.143 3054.898 3747.998 2889.994 3951.229  
## Mar 2007 3387.143 2964.977 3856.608 2759.364 4125.610  
## Apr 2007 3387.143 2893.279 3947.003 2656.458 4272.299  
## May 2007 3387.143 2832.337 4026.647 2569.883 4402.686  
## Jun 2007 3387.143 2778.707 4098.983 2494.387 4522.024  
## Jul 2007 3387.143 2730.462 4165.940 2427.034 4633.250  
## Aug 2007 3387.143 2686.396 4228.725 2365.987 4738.201  
## Sep 2007 3387.143 2645.693 4288.152 2310.004 4838.118  
## Oct 2007 3387.143 2607.770 4344.804 2258.202 4933.884  
## Nov 2007 3387.143 2572.198 4399.111 2209.926 5026.155  
## Dec 2007 3387.143 2538.644 4451.405 2164.672 5115.434  
## Jan 2008 3387.143 2506.849 4501.947 2122.046 5202.116  
## Feb 2008 3387.143 2476.603 4550.944 2081.731 5286.517  
## Mar 2008 3387.143 2447.735 4598.569 2043.468 5368.896  
## Apr 2008 3387.143 2420.103 4644.964 2007.042 5449.469  
## May 2008 3387.143 2393.588 4690.247 1972.273 5528.415  
## Jun 2008 3387.143 2368.088 4734.520 1939.008 5605.888  
## Jul 2008 3387.143 2343.517 4777.871 1907.114 5682.019  
## Aug 2008 3387.143 2319.799 4820.374 1876.478 5756.923  
## Sep 2008 3387.143 2296.867 4862.096 1847.001 5830.699  
## Oct 2008 3387.143 2274.664 4903.095 1818.596 5903.434  
## Nov 2008 3387.143 2253.139 4943.421 1791.184 5975.204  
## Dec 2008 3387.143 2232.247 4983.121 1764.699 6046.080

autoplot(forecast(fitted))

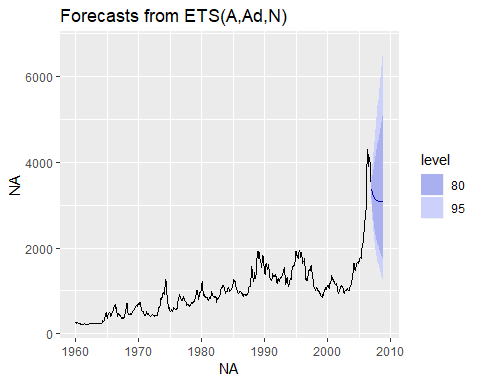


## 8.11.12.F

Compare the results of the ARIMA models to that achieved using ETS (error, trend, smoothing) model. From initial plotting, we see the forecast is quite noticeably lower ( 3100 vs 3400), and the 80% and 95% tolerance levels are even slightly wider that the ARIMA’s already wide intervals.

Comparing MSE’s from the two tests, we see that the ARIMA models errors are smaller, indicating that it is likely the ARIMA model is performing better.

autoplot(forecast(ets(mcopper, lambda = lambda)))



fets <- function(x,h, lambda)  
{  
 forecast(ets(x, lambda = lambda), h = h)  
}  
  
farima <- function(x,h, lambda)  
{  
 forecast(auto.arima(x, lambda = lambda), h = h)  
}  
  
etsCopper <- tsCV(mcopper, fets, h=1, lambda = lambda)  
arimaCopper <- tsCV(mcopper, farima, h=1, lambda = lambda)  
  
mean(etsCopper^2, na.rm=TRUE)

## [1] 6953.97

mean(arimaCopper^2, na.rm=TRUE)

## [1] 6198.188