Data 624 HW1

Group 3: Alain Kuiete Tchoupou, Jeff Littlejohn, Samriti Malhotra, Rajwant Mishra, Jimmy Ng

Due: 6/20/2020

First, we load the requisite packages.

## HA 2

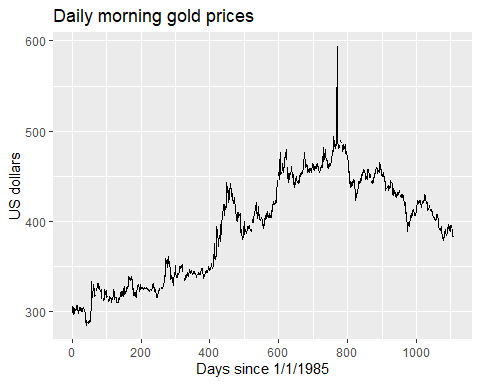
### 2.1

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

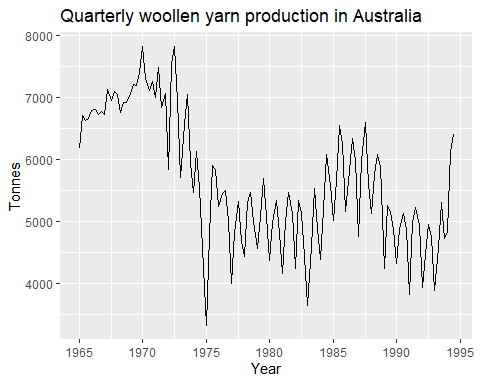
Each is a dataset contained within the forecast package. The gold dataset contains time series data of daily morning gold prices in US dollars from 1/1/1985 to 3/31/1989. The woolyrnq dataset includes quarterly woolen yarn production from Australia from 2Q 1965 to 3Q 1994. Finally, the gas dataset consists of time series data of Australian monthly gas production from 1956 to 1995.

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

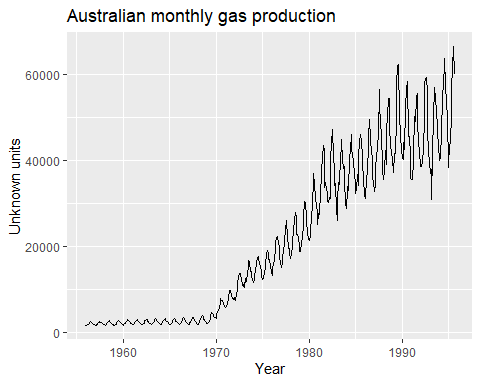
autoplot(gold) +  
 ggtitle("Daily morning gold prices") +  
 xlab("Days since 1/1/1985") +  
 ylab("US dollars")



autoplot(woolyrnq) +  
 ggtitle("Quarterly woollen yarn production in Australia") +  
 xlab("Year") +  
 ylab("Tonnes")



autoplot(gas) +  
 ggtitle("Australian monthly gas production") +  
 xlab("Year") +  
 ylab("Unknown units")



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

frequency(gold)

## [1] 1

Gold - daily

frequency(woolyrnq)

## [1] 4

Woolyrnq - quarterly

frequency(gas)

## [1] 12

Gas - monthly

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

which.max(gold)

## [1] 770

This function gives us 770, which corresponds with the single giant upward spike in prices. 770 signifies the number of days since the beginning of the time series, which would put the spike in Feb. 1987.

gold[770]

## [1] 593.7

A cursory google search revealed no details about such a spike. We want my want to review for the possibility of a data-entry error.

### 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.

[Manually downloaded to R working directory]

*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.*

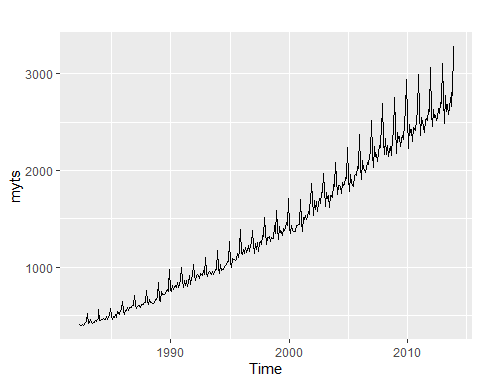
*Select one of the time series as follows (but replace the column name with your own chosen column):*

myts <- ts(retaildata[,"A3349398A"],  
 frequency=12, start=c(1982,4))

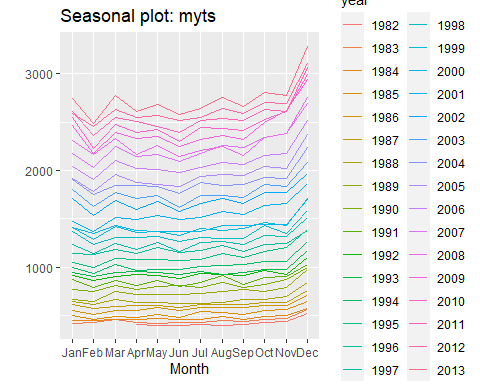
#### Explore your chosen retail time series using the following functions:

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

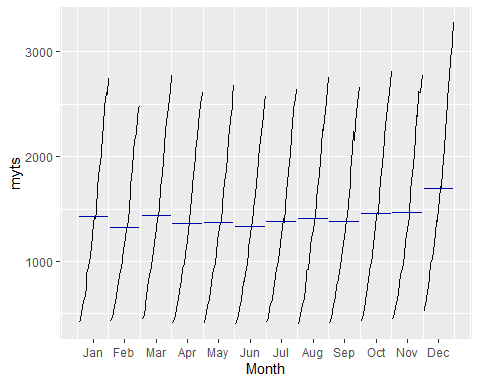
autoplot(myts)



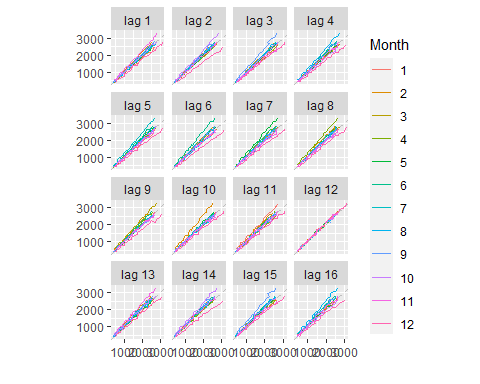
ggseasonplot(myts)



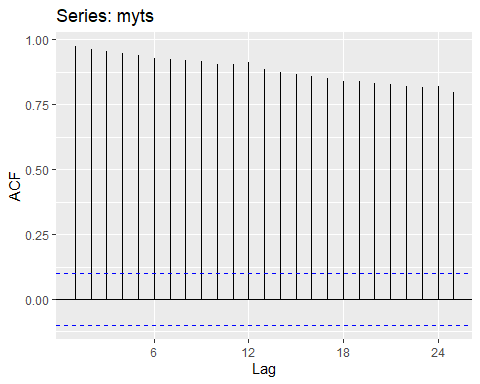
ggsubseriesplot(myts)



gglagplot(myts)



ggAcf(myts)



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

Starting with the autoplot, we see both an obvious and increasing trend along with a clear seasonal pattern that increases in nominal size over time. The seasonal plot shows highest monthly sales generally occur in December, with the biggest dip arriving in February, and the seasonal subseries visualization only confirms this.

Neither the lag nor the autocorrelation plots add significant value to our analysis that was not already conveyed in the earlier visualizations. They back up the trend of increasing sales over time with seasonal variation that’s likely tied to holiday shopping.

## HA 6

### Exercise 6.2

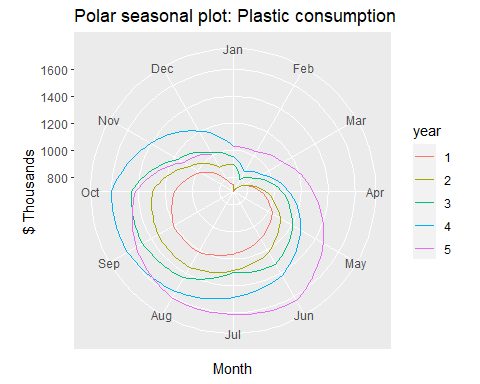
The plastics data set consists of the monthly sales (in thousands) of product A for a plastics manufacturer for five years.

plastics

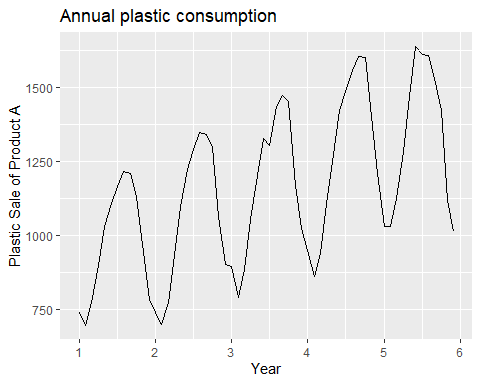
## Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec  
## 1 742 697 776 898 1030 1107 1165 1216 1208 1131 971 783  
## 2 741 700 774 932 1099 1223 1290 1349 1341 1296 1066 901  
## 3 896 793 885 1055 1204 1326 1303 1436 1473 1453 1170 1023  
## 4 951 861 938 1109 1274 1422 1486 1555 1604 1600 1403 1209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

ggseasonplot(plastics, polar=TRUE) +  
 ylab("$ Thousands") +  
 ggtitle("Polar seasonal plot: Plastic consumption")



autoplot(plastics) + ggtitle("Annual plastic consumption ") + xlab("Year") + ylab("Plastic Sale of Product A")

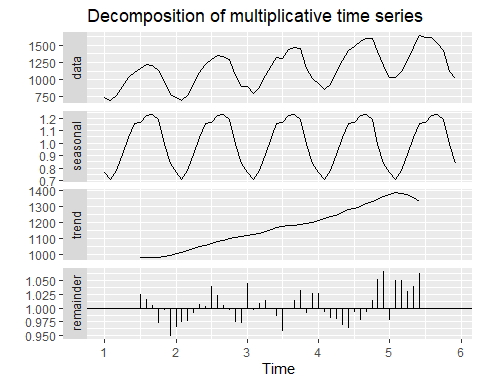


#  
# # With Moving average of 3 data point   
# autoplot(plastics, series="Data") +  
# autolayer(ma(plastics,3), series="3-MA") +  
# xlab("Year") + ylab("GWh") +  
# ggtitle("Annual electricity sales: South Australia") +  
# scale\_colour\_manual(values=c("Data"="grey50","3-MA"="red"),  
# breaks=c("Data","3-MA"))  
#   
# # Classical Decomposition : Deafult is additive  
# dec\_plastic <- decompose(plastics)  
# autoplot(dec\_plastic)  
# plastics %>% stl(s.window=6) %>% autoplot() +xlab("Year") + ylab("GWh") +  
# ggtitle("Annual electricity sales: South Australia")  
#   
# # X11 Decomposition  
# plastics %>% seas() %>% autoplot() + xlab("Year") + ylab("GWh") +  
# ggtitle("X11: Annual plastic consumptio")  
#   
#   
# elecequip  
#   
# as.ts(plastics,start=c(1959),frequency = 12)

Above plot of data very clearly shows that there is some pattern each year. This is the seasonality present in the data each year, which shows downward trend in the beginning of the year and peak in the mid of the year with decline again at the end of the year. Polar seasonal plot also suggest decline at its peak in the month of Jan and Feb and then picking up from there.

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

# Classical Decomposition : Deafult is additive  
dec\_plastic\_mul <- decompose(plastics,type = "multiplicative")  
autoplot(dec\_plastic\_mul)



# autoplot(stl(plastics,s.window = "periodic"))  
print("Trend Cycle:")

## [1] "Trend Cycle:"

trendcycle(dec\_plastic\_mul)

## 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

print("seasonal indices")

## [1] "seasonal indices"

print(dec\_plastic\_mul$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

Classical decomposition methods assume that the seasonal component repeats from year to year. For many series, this is a reasonable assumption, but for some longer series it is not. Here third plot shows the trend with m= 12, as it shows yearly increasing trend.As data suggest around the end we see some high ressidual suddden drop in demand, which our trend-cycle estimate is not able to capture.

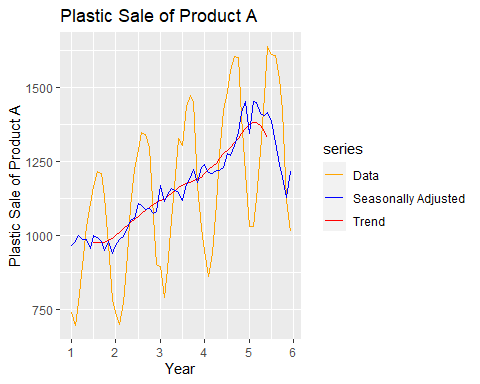
####C.Do the results support the graphical interpretation from part a?

Yes, yearly trend suggested from A. is very clearly seen in above plots. We can also see that some decline in number at the end of 5th year in last 6 months.

####D. Compute and plot the seasonally adjusted data.

# Decomposing time series data with multiplicative seasonal component  
dec\_plastic\_mul <- decompose(plastics,type = "multiplicative")  
autoplot(plastics, series="Data") +  
 autolayer(trendcycle(dec\_plastic\_mul), series="Trend") +  
 autolayer(seasadj(dec\_plastic\_mul), series="Seasonally Adjusted") +  
 xlab("Year") + ylab("Plastic Sale of Product A") +  
 ggtitle("Plastic Sale of Product A") +  
 scale\_colour\_manual(values=c("orange","blue","red"),  
 breaks=c("Data","Seasonally Adjusted","Trend"))

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

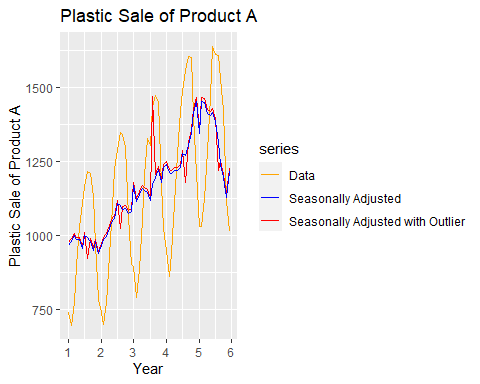


Blue line in the above plot shows Seasonally adjusted data. As we can see that there are not much seasonality fluctuation in the data any more after we have decomposed the data , only linear trend can be seen.

#### 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?

# Function to use for checking Outlier effect.  
plot\_outlier\_trend <- function(c\_index) {  
 c\_plastics <- plastics  
 print(paste("Ref Value:" ,plastics[c\_index], "at ", c\_index))  
 c\_plastics[c\_index] <- c\_plastics[c\_index] + 500  
 # Decomposing time series data with multiplicative seasonal component  
 c\_dec\_plastic\_mul <- decompose(c\_plastics,type = "multiplicative")  
 autoplot(plastics, series="Data") +  
 autolayer(c\_plastics, series="Data Outlier") +  
 autolayer(trendcycle(dec\_plastic\_mul), series="Trend") +  
 autolayer(trendcycle(c\_dec\_plastic\_mul), series="Trend with Outlier") +  
 xlab("Year") + ylab("Plastic Sale of Product A") +  
 ggtitle("Plastic Sale of Product A") +  
 scale\_colour\_manual(values=c("orange","gray","red","blue"),  
 breaks=c("Data","Data Outlier","Trend","Trend with Outlier"))  
}  
plot\_outlier\_seasadj <- function(c\_index) {  
 c\_plastics <- plastics  
 print(paste("Ref Value:" ,plastics[c\_index], "at ", c\_index))  
c\_plastics[c\_index] <- c\_plastics[c\_index] + 500  
# Decomposing time series data with multiplicative seasonal component  
c\_dec\_plastic\_mul <- decompose(c\_plastics,type = "multiplicative")  
autoplot(plastics, series="Data") +  
 autolayer(seasadj(c\_dec\_plastic\_mul), series="Seasonally Adjusted with Outlier") +  
 autolayer(seasadj(dec\_plastic\_mul), series="Seasonally Adjusted") +  
 xlab("Year") + ylab("Plastic Sale of Product A") +  
 ggtitle("Plastic Sale of Product A") +  
 scale\_colour\_manual(values=c("orange","blue","red"),  
 breaks=c("Data","Seasonally Adjusted","Seasonally Adjusted with Outlier"))  
   
}  
# Choose some index and update the value with +500  
set.seed(421)  
c\_index <- sample(1:59, 1)  
plot\_outlier\_seasadj(c\_index)

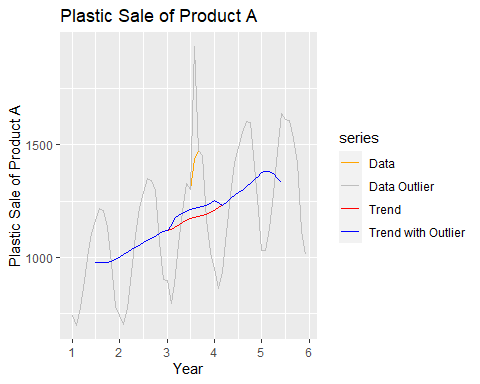
## [1] "Ref Value: 1436 at 32"



plot\_outlier\_trend(c\_index)

## [1] "Ref Value: 1436 at 32"

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

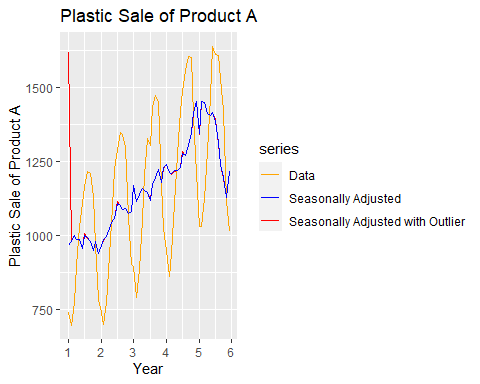


Addition of outliers shows some sharp variation in the data at the same season in other year, but trend line seem to have no effect except in the year outliers are noted. The spike in the seasonally adjusted data is due to th big ressidual at the ponit of outlier.

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

# Lets set the outlier in the begning and end and see how it impacts   
plot\_outlier\_seasadj(1)

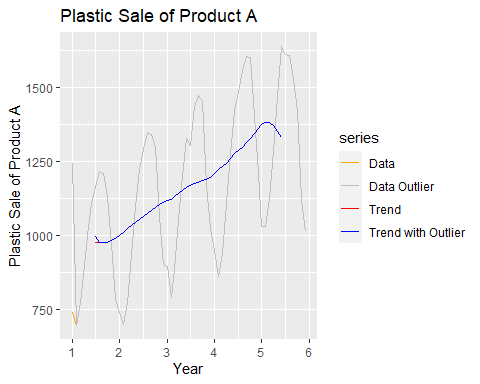
## [1] "Ref Value: 742 at 1"



plot\_outlier\_trend(1)

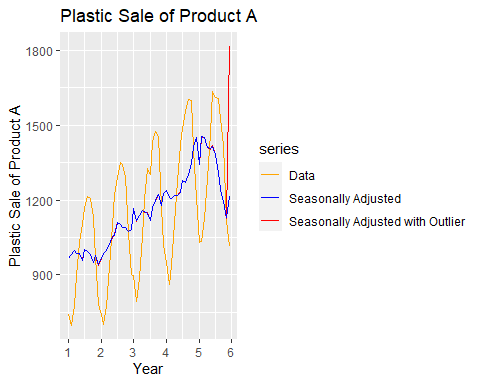
## [1] "Ref Value: 742 at 1"

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



plot\_outlier\_seasadj(60)

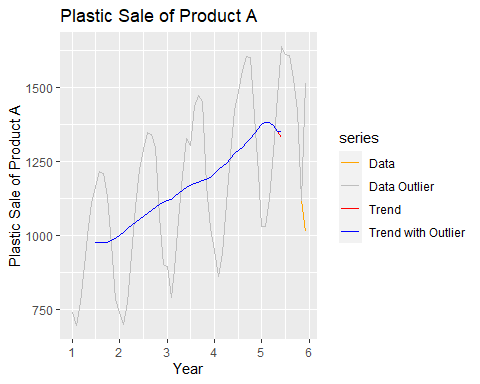
## [1] "Ref Value: 1013 at 60"



plot\_outlier\_trend(60)

## [1] "Ref Value: 1013 at 60"

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



Outliers at the beginning or at the end have very little effect in the same season’s data but its impact is negligible in other season. Possible reason could be not knowing the Trend Cycle for the first 6 and last 6 months of the data in the multiplicative seasonal decomposition.

## KJ 3

### 3.1

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

data(Glass)  
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 ...

The structure of the Glass data shows that all the predictors are numeric. The dependent variable is factor of 6 levels.

head(Glass)

## RI Na Mg Al Si K Ca Ba Fe Type  
## 1 1.52101 13.64 4.49 1.10 71.78 0.06 8.75 0 0.00 1  
## 2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83 0 0.00 1  
## 3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78 0 0.00 1  
## 4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22 0 0.00 1  
## 5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07 0 0.00 1  
## 6 1.51596 12.79 3.61 1.62 72.97 0.64 8.07 0 0.26 1

The head function displays the first values of each variables.

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

We remove the target variable

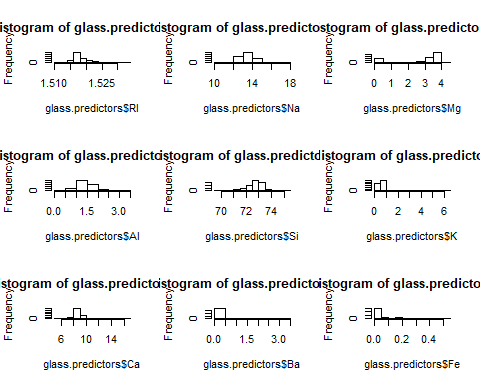
glass.predictors <- Glass[,-10]

Since all variables are numerical, we can use the skewness function of e1071 to estimated the predictors to represent.

skewValues <- apply(glass.predictors, 2, skewness)  
skewValues

## RI Na Mg Al Si K Ca   
## 1.6027151 0.4478343 -1.1364523 0.8946104 -0.7202392 6.4600889 2.0184463   
## Ba Fe   
## 3.3686800 1.7298107

par(mfrow = c(3,3))  
hist(x = glass.predictors$RI)  
hist(x = glass.predictors$Na)  
hist(x = glass.predictors$Mg)  
hist(x = glass.predictors$Al)  
hist(x = glass.predictors$Si)  
hist(x = glass.predictors$K)  
hist(x = glass.predictors$Ca)  
hist(x = glass.predictors$Ba)  
hist(x = glass.predictors$Fe)



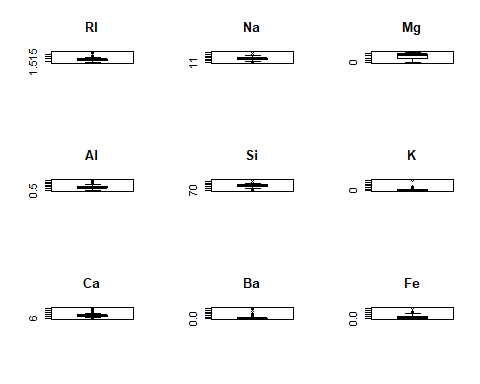
The predictors RI, Na, Al, Si and Ca are normally distributed. The predictors K, Ba, and Fe are rigtht skewed. we can aplied the log function on those variable to normalise or Boxcox to centralise, scale and transform.

The Mg predictor need to be centralise and scale. It is neither normal, nor skewed.

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

Looking for outliers:

par(mfrow = c(3,3))  
boxplot(x = glass.predictors$RI, main = "RI")  
boxplot(x = glass.predictors$Na, main = "Na")  
boxplot(x = glass.predictors$Mg, main = "Mg")  
boxplot(x = glass.predictors$Al, main = "Al")  
boxplot(x = glass.predictors$Si, main = "Si")  
boxplot(x = glass.predictors$K, main = "K")  
boxplot(x = glass.predictors$Ca, main = "Ca")  
boxplot(x = glass.predictors$Ba, main = "Ba")  
boxplot(x = glass.predictors$Fe, main = "Fe")



The boxplot graphs shows some outliers with the predictors RI, Na, Al, Si, K, Ca, Ba, and Fe. The outlier of Ba and K are extreme.

summary(glass.predictors)

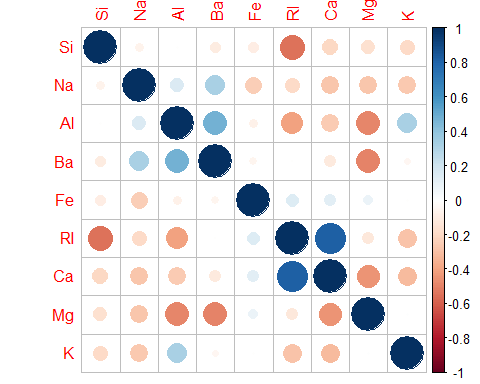
## RI Na Mg Al   
## Min. :1.511 Min. :10.73 Min. :0.000 Min. :0.290   
## 1st Qu.:1.517 1st Qu.:12.91 1st Qu.:2.115 1st Qu.:1.190   
## Median :1.518 Median :13.30 Median :3.480 Median :1.360   
## Mean :1.518 Mean :13.41 Mean :2.685 Mean :1.445   
## 3rd Qu.:1.519 3rd Qu.:13.82 3rd Qu.:3.600 3rd Qu.:1.630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

To visualize the correlation between predictors, we use the corrplot function in the package of the same name.

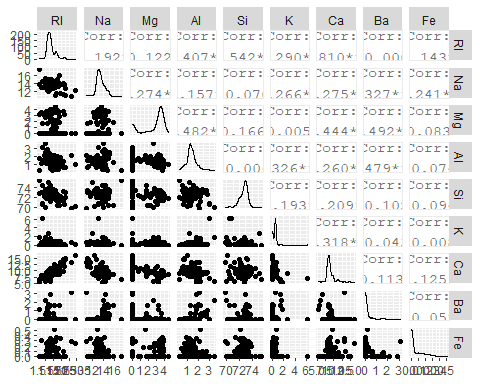
correlations <- cor(glass.predictors)  
correlations

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

corrplot(correlations, order = "hclust")



GGally::ggpairs(as.data.frame(glass.predictors))



The only notable correlation is between RI and Ca.

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

We use the powerTransform of the car package that calculates the Box-Cox transformation. The Box-Cox transformation uses the maximum likelihood approach and returns information on the estimated values along with convenient rounded values that are within 1.96 standard deviations of the maximum likelihood estimate.

summary(powerTransform(Glass[,1:9], family="yjPower"))$result[,1:2]

## Est Power Rounded Pwr  
## RI -25.0853114 -25.09  
## Na 1.3755562 1.00  
## Mg 1.7699080 2.00  
## Al 0.9773267 1.00  
## Si 10.9452696 10.95  
## K -0.1441078 0.00  
## Ca 0.6774333 0.50  
## Ba -6.8620464 -6.86  
## Fe -14.9245600 -14.92

The suggested transformations are:

No transformation for RI, Na, Si, and K since lambda=1. Log transformations for Mg, K, Ba, and Fe since lambda =0. Square root transformation for Ca since lambda = 0.5.

### 3.2

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

The data can be loaded via:

data(Soybean)  
## See ?Soybean for details  
str(Soybean)

## 'data.frame': 683 obs. of 36 variables:  
## $ Class : Factor w/ 19 levels "2-4-d-injury",..: 11 11 11 11 11 11 11 11 11 11 ...  
## $ date : Factor w/ 7 levels "0","1","2","3",..: 7 5 4 4 7 6 6 5 7 5 ...  
## $ plant.stand : Ord.factor w/ 2 levels "0"<"1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ precip : Ord.factor w/ 3 levels "0"<"1"<"2": 3 3 3 3 3 3 3 3 3 3 ...  
## $ temp : Ord.factor w/ 3 levels "0"<"1"<"2": 2 2 2 2 2 2 2 2 2 2 ...  
## $ hail : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...  
## $ crop.hist : Factor w/ 4 levels "0","1","2","3": 2 3 2 2 3 4 3 2 4 3 ...  
## $ area.dam : Factor w/ 4 levels "0","1","2","3": 2 1 1 1 1 1 1 1 1 1 ...  
## $ sever : Factor w/ 3 levels "0","1","2": 2 3 3 3 2 2 2 2 2 3 ...  
## $ seed.tmt : Factor w/ 3 levels "0","1","2": 1 2 2 1 1 1 2 1 2 1 ...  
## $ germ : Ord.factor w/ 3 levels "0"<"1"<"2": 1 2 3 2 3 2 1 3 2 3 ...  
## $ plant.growth : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ leaves : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ leaf.halo : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...  
## $ leaf.marg : Factor w/ 3 levels "0","1","2": 3 3 3 3 3 3 3 3 3 3 ...  
## $ leaf.size : Ord.factor w/ 3 levels "0"<"1"<"2": 3 3 3 3 3 3 3 3 3 3 ...  
## $ leaf.shread : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ leaf.malf : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ leaf.mild : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...  
## $ stem : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ lodging : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 2 1 1 1 ...  
## $ stem.cankers : Factor w/ 4 levels "0","1","2","3": 4 4 4 4 4 4 4 4 4 4 ...  
## $ canker.lesion : Factor w/ 4 levels "0","1","2","3": 2 2 1 1 2 1 2 2 2 2 ...  
## $ fruiting.bodies: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ ext.decay : Factor w/ 3 levels "0","1","2": 2 2 2 2 2 2 2 2 2 2 ...  
## $ mycelium : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ int.discolor : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...  
## $ sclerotia : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ fruit.pods : Factor w/ 4 levels "0","1","2","3": 1 1 1 1 1 1 1 1 1 1 ...  
## $ fruit.spots : Factor w/ 4 levels "0","1","2","4": 4 4 4 4 4 4 4 4 4 4 ...  
## $ seed : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ mold.growth : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ seed.discolor : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ seed.size : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ shriveling : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ roots : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...

All variables are factors or ordered factors.

head(Soybean)

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

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

nearZeroVar(Soybean,saveMetric=TRUE)

## freqRatio percentUnique zeroVar nzv  
## Class 1.010989 2.7818448 FALSE FALSE  
## 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

The predictors that correspond respectively to the position of variables 19, 26, 28 in the datafarme Soybean which are degenerate, are leaf.mild, mycelium and sclerotia.

summary(Soybean)

## Class date plant.stand precip temp   
## brown-spot : 92 5 :149 0 :354 0 : 74 0 : 80   
## alternarialeaf-spot: 91 4 :131 1 :293 1 :112 1 :374   
## frog-eye-leaf-spot : 91 3 :118 NA's: 36 2 :459 2 :199   
## phytophthora-rot : 88 2 : 93 NA's: 38 NA's: 30   
## anthracnose : 44 6 : 90   
## brown-stem-rot : 44 (Other):101   
## (Other) :233 NA's : 1   
## hail crop.hist area.dam sever seed.tmt germ plant.growth  
## 0 :435 0 : 65 0 :123 0 :195 0 :305 0 :165 0 :441   
## 1 :127 1 :165 1 :227 1 :322 1 :222 1 :213 1 :226   
## NA's:121 2 :219 2 :145 2 : 45 2 : 35 2 :193 NA's: 16   
## 3 :218 3 :187 NA's:121 NA's:121 NA's:112   
## NA's: 16 NA's: 1   
##   
##   
## leaves leaf.halo leaf.marg leaf.size leaf.shread leaf.malf leaf.mild   
## 0: 77 0 :221 0 :357 0 : 51 0 :487 0 :554 0 :535   
## 1:606 1 : 36 1 : 21 1 :327 1 : 96 1 : 45 1 : 20   
## 2 :342 2 :221 2 :221 NA's:100 NA's: 84 2 : 20   
## NA's: 84 NA's: 84 NA's: 84 NA's:108   
##   
##   
##   
## stem lodging stem.cankers canker.lesion fruiting.bodies ext.decay   
## 0 :296 0 :520 0 :379 0 :320 0 :473 0 :497   
## 1 :371 1 : 42 1 : 39 1 : 83 1 :104 1 :135   
## NA's: 16 NA's:121 2 : 36 2 :177 NA's:106 2 : 13   
## 3 :191 3 : 65 NA's: 38   
## NA's: 38 NA's: 38   
##   
##   
## mycelium int.discolor sclerotia fruit.pods fruit.spots seed   
## 0 :639 0 :581 0 :625 0 :407 0 :345 0 :476   
## 1 : 6 1 : 44 1 : 20 1 :130 1 : 75 1 :115   
## NA's: 38 2 : 20 NA's: 38 2 : 14 2 : 57 NA's: 92   
## NA's: 38 3 : 48 4 :100   
## NA's: 84 NA's:106   
##   
##   
## mold.growth seed.discolor seed.size shriveling roots   
## 0 :524 0 :513 0 :532 0 :539 0 :551   
## 1 : 67 1 : 64 1 : 59 1 : 38 1 : 86   
## NA's: 92 NA's:106 NA's: 92 NA's:106 2 : 15   
## NA's: 31   
##   
##   
##

Using the summary of Soybean, the fraction of unique values over the sample size of the predictors is low. There are 2, 3,0r 4 unique values over 683 observations.

The predictors leaf.mild, mycelium and sclerotia have the ratio of the frequency the most prevalent value to the frequency of the second most prevalent very large.

imbalance.leaf.mild = 535/20  
imbalance.leaf.mild

## [1] 26.75

imbalance.mycelium = 639/6  
imbalance.mycelium

## [1] 106.5

imbalance.sclerotia = 625/20  
imbalance.sclerotia

## [1] 31.25

The three predictors have a very strong imbalance. These are near-zero variance predictors

We can observe these large imbalance between uniques values in the plots below.

par(mfrow = c(3,3))  
plot(x = Soybean$leaves) + title(main = 'leaves')

## numeric(0)

plot(x = Soybean$leaf.malf) + title(main = 'leaf.malf')

## numeric(0)

plot(x = Soybean$leaf.mild) + title(main = 'leaf.mild')

## numeric(0)

plot(x = Soybean$lodging) + title(main = 'lodging')

## numeric(0)

plot(x = Soybean$mycelium) + title(main = 'mycelium')

## numeric(0)

plot(x = Soybean$int.discolor)+ title(main = 'int.discolor')

## numeric(0)

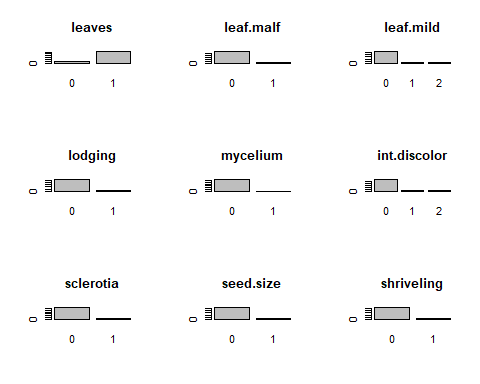
plot(x = Soybean$sclerotia) + title(main = 'sclerotia')

## numeric(0)

plot(x = Soybean$seed.size) + title(main = 'seed.size')

## numeric(0)

plot(x = Soybean$shriveling) + title(main = 'shriveling')



## numeric(0)

#### (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?

Observation of missing values

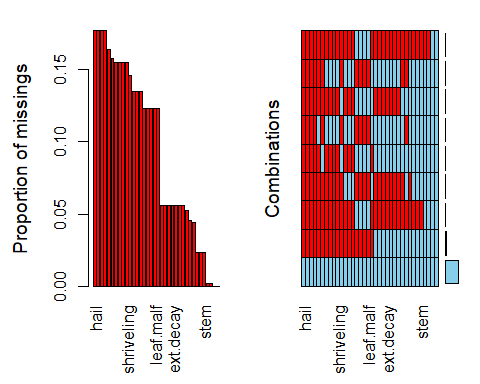
colSums(is.na(Soybean))

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

The aggr function in the VIM package plots and calculates the amount of missing values in each variable. The dply function is useful for wrangling data into aggregate summaries and is used to find the pattern of missing data related to the classes.

aggr(Soybean, prop = c(TRUE, TRUE), bars=TRUE, numbers=TRUE, sortVars=TRUE)

## Warning in plot.aggr(res, ...): not enough horizontal space to display  
## frequencies



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

The table above and the histograms show that the predictors hail, sever, seed.tmt, and lodging have around 18% of missing data. Other variables that are more likely to be missing are germ(16% of missing values), leaf.mild(16%),fruiting.bodies(15%), fruits.spots(15%), seed.discolor(15%), and shriveling(15%).

The grid shows the combination of all with 82% of data not missing in accordance with the problem description (18% missing). The remainder of the grid shows missing data for variable combinations with each row highlighting the missing values for the group of variables detailed in the x-axis. The non-graphical output of the function shows on top the exact proportion of missing values per variable.

Looking for pattern in missing data by classes

Soybean %>%  
 mutate(Total = n()) %>%   
 filter(!complete.cases(.)) %>%  
 group\_by(Class) %>%  
 mutate(Missing = n(), Proportion=Missing/Total) %>%  
 select(Class, Missing, Proportion) %>%  
 unique()

## # A tibble: 5 x 3  
## # Groups: Class [5]  
## Class Missing Proportion  
## <fct> <int> <dbl>  
## 1 phytophthora-rot 68 0.0996  
## 2 diaporthe-pod-&-stem-blight 15 0.0220  
## 3 cyst-nematode 14 0.0205  
## 4 2-4-d-injury 16 0.0234  
## 5 herbicide-injury 8 0.0117

Checking if a pattern of missing data related to the classes exists is done by checking if some classes hold most of the incomplete cases. This is accomplished by filtering, grouping, and mutating the data with dplyr. The majority of the missing values are in the phytophthora-rot class which has nearly 10% incomplete cases. The are only four more, out of the eighteen other, variables with incomplete cases. The pattern of missing data is related to the classes. Mostly the phytophthora-rot class however since the other four variables only have between 1% and 2% incomplete cases.

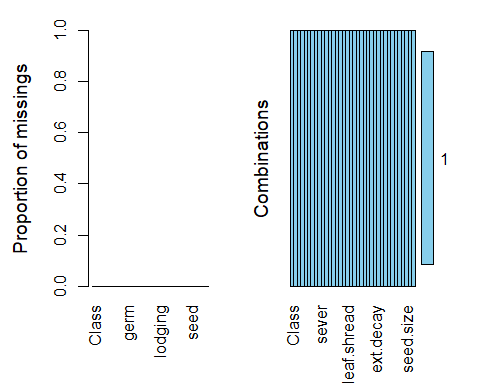
#### (c) Develop a strategy for handling missing data, either by eliminating predictors or imputation.

The strategy to handle missing data is by using the predictive mean matching method of the mice function to impute data. Next, we create a complete dataset with the function complete() We can previous the new dataset for missing values with aggr from VIM package

MICE <- mice(Soybean, method="pmm", printFlag=FALSE, seed=6)

## Warning: Number of logged events: 1665

aggr(complete(MICE), prop = c(TRUE, TRUE), bars=TRUE, numbers=TRUE, sortVars=TRUE)



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

The strategy we use to deal with missing data is the simple imputation method that uses predictive mean matching (pmm) and “imputes missing values by means of the nearest-neighbor donor with distance based on the expected values of the missing variables conditional on the observed covariates.”

After applying the mice function, we realise that there are no missing values in any variable.

#### KJ 3 References

<https://www.otexts.org/fpp/>

<https://rpubs.com/josezuniga/358605>

<https://rpubs.com/josezuniga/253955>

<https://rpubs.com/josezuniga/269297>

<http://appliedpredictivemodeling.com/>

## HA 7

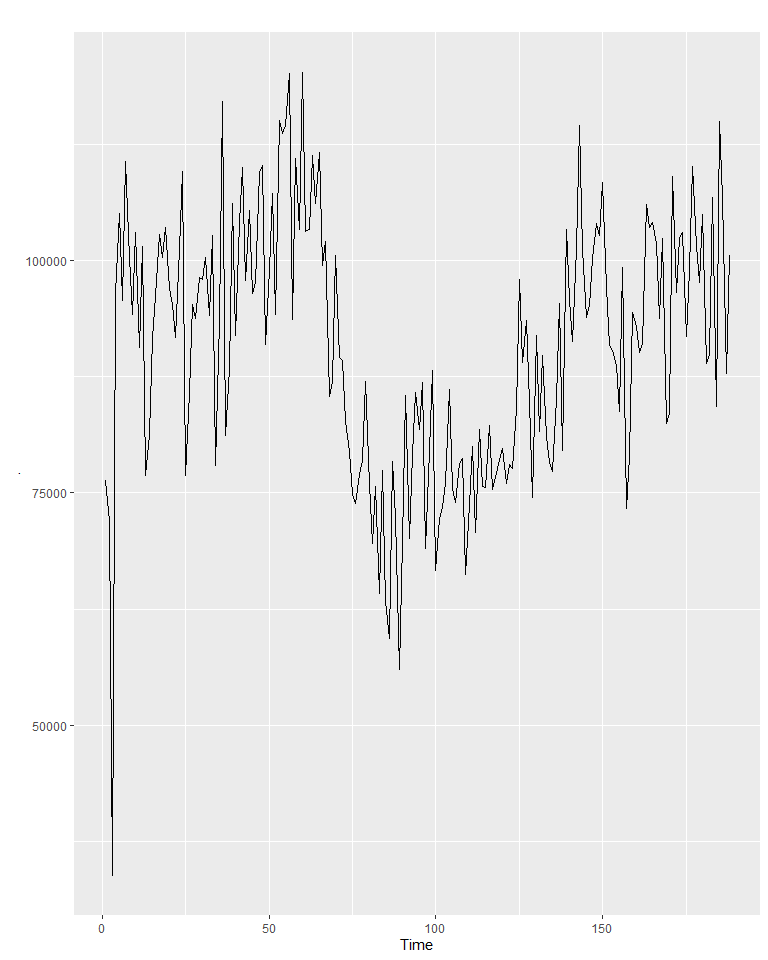
### 7.1

Consider the pigs series — the number of pigs slaughtered in Victoria each month.

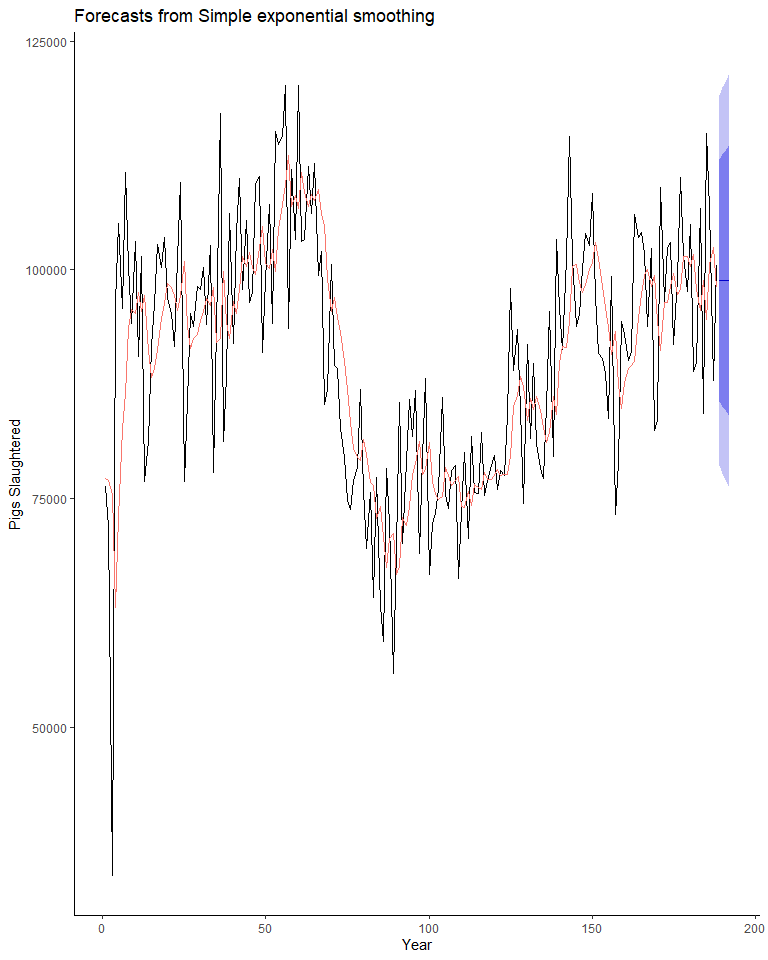
1. Use the ses() function in R to find the optimal values of and , and generate forecasts for the next four months.
2. Compute a 95% prediction interval for the first forecast using where is the standard deviation of the residuals. Compare your interval with the interval produced by R.

#### a. ses() function

# take a look of the ts object  
pigs <- read.csv("pigs.ts.csv") %>% ts()  
str(pigs)  
## Time-Series [1:188, 1] from 1 to 188: 76378 71947 33873 96428 105084 95741 110647 100331 94133 103055 ...  
## - attr(\*, "dimnames")=List of 2  
## ..$ : NULL  
## ..$ : chr "x"  
broom::glance(pigs %>% data.frame)  
## Warning: 'glance.data.frame' is deprecated.  
## See help("Deprecated")  
## # A tibble: 1 x 4  
## nrow ncol complete.obs na.fraction  
## <int> <int> <int> <dbl>  
## 1 188 1 188 0  
pigs %>% forecast::autoplot()



# forecast for the next 4 months  
pigs\_forecast4 <- forecast::ses(pigs, h = 4)  
# tidy forecast summary  
broom::tidy(pigs\_forecast4 %>% summary)  
##   
## Forecast method: Simple exponential smoothing  
##   
## Model Information:  
## Simple exponential smoothing   
##   
## Call:  
## forecast::ses(y = pigs, h = 4)   
##   
## Smoothing parameters:  
## alpha = 0.2971   
##   
## Initial states:  
## l = 77260.0561   
##   
## sigma: 10308.58  
##   
## AIC AICc BIC   
## 4462.955 4463.086 4472.665   
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set 385.8721 10253.6 7961.383 -0.922652 9.274016 0.8434758 0.01282239  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 189 98816.41 85605.43 112027.4 78611.97 119020.8  
## 190 98816.41 85034.52 112598.3 77738.83 119894.0  
## 191 98816.41 84486.34 113146.5 76900.46 120732.4  
## 192 98816.41 83958.37 113674.4 76092.99 121539.8  
## Warning: 'tidy.data.frame' is deprecated.  
## See help("Deprecated")  
## # A tibble: 5 x 13  
## column n mean sd median trimmed mad min max range skew  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 Point~ 4 9.88e4 0 9.88e4 98816. 0 9.88e4 9.88e4 0 NaN   
## 2 Lo 80 4 8.48e4 709. 8.48e4 84771. 538. 8.40e4 8.56e4 1647. 0.0420  
## 3 Hi 80 4 1.13e5 709. 1.13e5 112862. 538. 1.12e5 1.14e5 1647. -0.0420  
## 4 Lo 95 4 7.73e4 1084. 7.73e4 77336. 823. 7.61e4 7.86e4 2519. 0.0420  
## 5 Hi 95 4 1.20e5 1084. 1.20e5 120297. 823. 1.19e5 1.22e5 2519. -0.0420  
## # ... with 2 more variables: kurtosis <dbl>, se <dbl>  
# visualize output  
pigs\_forecast4\_fitted <- fitted(pigs\_forecast4)  
pigs\_forecast4 %>%  
 autoplot() +  
 autolayer(pigs\_forecast4\_fitted, series = "Fitted") +  
 theme\_classic() +  
 theme(legend.position = "none") +  
 xlab("Year") +  
 ylab("Pigs Slaughtered")



#### b. 95% prediction interval

# 95% prediction interval by hand  
lo.interval <- pigs\_forecast4$mean[1] - 1.96 \* sd(pigs\_forecast4$residuals)  
up.interval <- pigs\_forecast4$mean[1] + 1.96 \* sd(pigs\_forecast4$residuals)  
# R model output  
lo.interval.model <- pigs\_forecast4$lower[1, 2]  
up.interval.model <- pigs\_forecast4$upper[1, 2]  
# comparison   
type <- paste0(c("calculated by hand", "calculated by hand", "run by R", "run by R"))  
bound <- paste0(c("lower", "upper", "lower", "upper"))  
interval <- c(lo.interval, up.interval, lo.interval.model, up.interval.model)  
data.frame(type, bound, interval) %>%  
 tidyr::spread(., type, interval) %>%  
 dplyr::mutate(`diff (%)` = round((`calculated by hand` - `run by R`) / `calculated by hand` \* 100, 2)) %>%  
 kable() %>%  
 kable\_styling()

bound

calculated by hand

run by R

diff (%)

lower

78679.97

78611.97

0.09

upper

118952.84

119020.84

-0.06

### 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 and . Do you get the same values as the ses() function?

# custom function  
custom\_fun <- function(fn, ts){  
 a = fn[1]  
 b = fn[2]  
 n = length(ts)  
 yhat = c(b, 0 \* (2:n))  
 for (i in 1:(n-1)){  
 yhat[i +1] <- a \* ts[i] + (1-a) \* yhat[i]  
 }  
 res = ts - yhat  
 sse = sum(res^2) / (n-1)  
   
 return(sse)  
}  
# output parameters  
parameters <- optim(par = c(0.1, 10000), fn = custom\_fun, ts = pigs)  
a <- parameters$par[1]  
b <- parameters$par[2]  
# comparison  
model\_a <- pigs\_forecast4$model$fit$par[1]  
model\_b <- pigs\_forecast4$model$fit$par[2]  
# comparison   
type <- paste0(c("calculated by hand", "calculated by hand", "run by R", "run by R"))  
parameter <- paste0(c("alpha", "level", "alpha", "level"))  
value <- c(a, b, model\_a, model\_b)  
data.frame(type, parameter, value) %>%  
 dplyr::mutate(value = round(value, 3)) %>%  
 tidyr::spread(., type, value) %>%  
 kable() %>%  
 kable\_styling()

parameter

calculated by hand

run by R

alpha

0.297

0.297

level

77260.958

77260.056

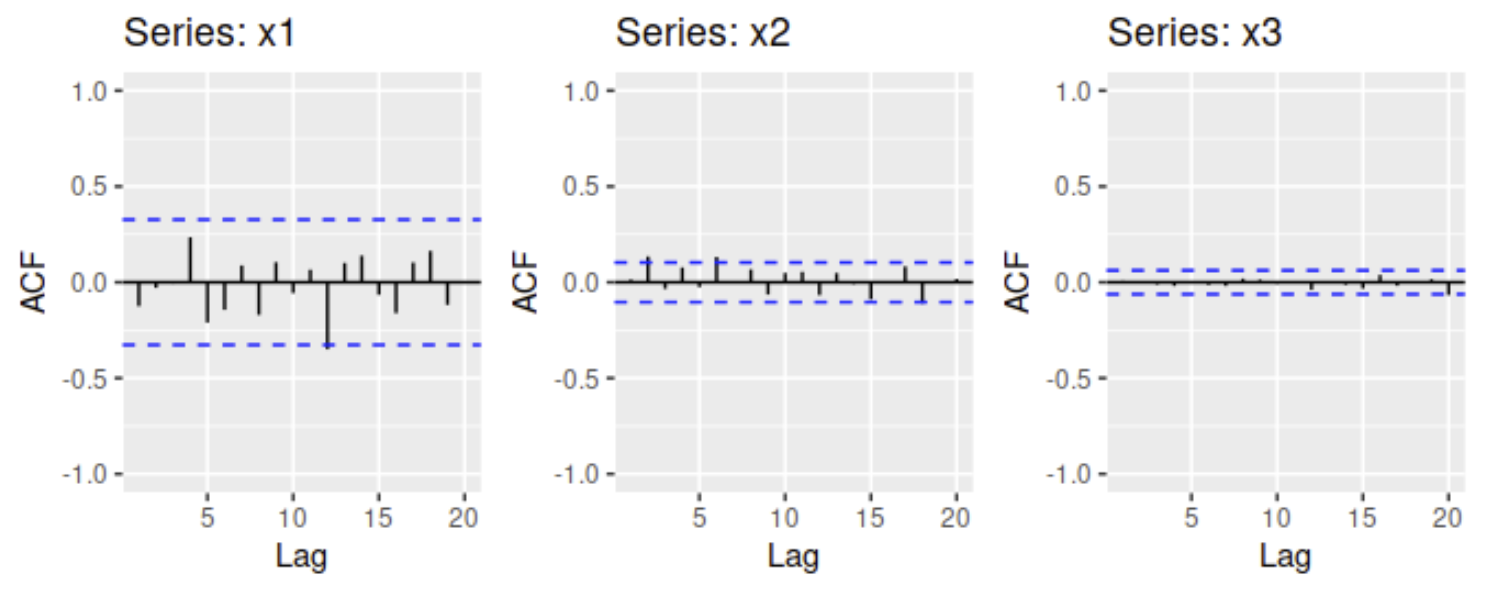
The result is almost identical.

## HA 8

### 8.1

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

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



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.

The Auto Correlation Function (ACF) conveys serial correlation between lagged values in time series data.(Along with the Partial Auto Correlation Functions (PACF) plots, they also help us determine good values for *p* and *q*.)

As white noise is truly randomly generated, there should be no correlation between lagging values. Our three examples show few if any correlation values that exceed the 95% confidence intervals, so we do probably do not detect significant correlation between values. This fits our expectation of white noise - that it’s stationary.

Based on our confidence intervals beings set at 95%, we would expect a small percentage of our white noise values to exceed the intervals. We might take a closer look at lag 12 series x1 with Ljung-Box test given a lag of 12 could indicate annual seasonality, but with a sample of 36 time series values, we would expect the occassional outlier.

#### 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?

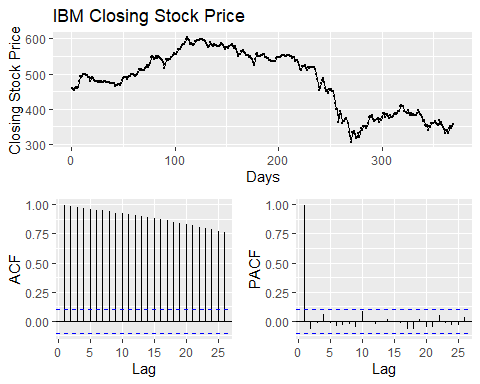
As sample size grows, the interval narrows. Note the x- and y-axes are the same. Our formula for the confidence interval here is:

It’s clear that as T grows, the interval shrinks. This seems intuitive, as smaller samples may erroneously show autocorrelation based on a few fluke outliers.

### 8.2

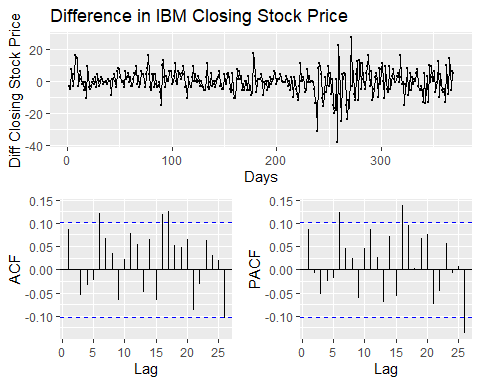
#### 2. A classic example of a non-stationary series is the daily closing IBM stock price series (data set ibmclose). Use R to plot the daily closing prices for IBM stock and the ACF and PACF. Explain how each plot shows that the series is non-stationary and should be differenced.

#https://www.rdocumentation.org/packages/forecast/versions/8.12/topics/ggtsdisplay  
ggtsdisplay(ibmclose,main="IBM Closing Stock Price",xlab="Days",ylab="Closing Stock Price")

 In the regular time series plot, we see changing, non-seasonal trends in the data over the year.The daily closing price is not a stationary time series. The ACF plot shows clear and significant autocorrelation to a lag of 25. The PACF plot shows a significant spike at one, which could indicate that all higher order autocorrelation is explained by that first lag autocorrelation. (We also know that the first partial autocorrelation coefficient will be identical to that of the first autocorrelation.)

Given the above, taking the differences of the closing stock price data could produce a stationary series.

ggtsdisplay(diff(ibmclose),main="Difference in IBM Closing Stock Price",xlab="Days",ylab="Diff Closing Stock Price")

 We might more correlation values that exceed the confidence interval than one would expect with a stationary dataset. Let’s conduct the Ljung-Box test to verify:

#https://stat.ethz.ch/R-manual/R-devel/library/stats/html/box.test.html  
Box.test(diff(ibmclose), type = c("Ljung-Box"))

##   
## Box-Ljung test  
##   
## data: diff(ibmclose)  
## X-squared = 2.717, df = 1, p-value = 0.09929

For the test, smaller p values means there’s more autocorrelation - not stationary. Taking one difference gets us over the hump, and the data can now be called stationary - a random walk.

### 8.6

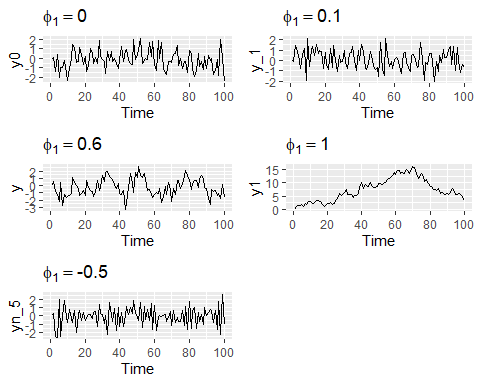
#### 6. Use R to simulate and plot some data from simple ARIMA models.

#### a. Use the following R code to generate data from an AR(1) model with and . The process starts with .

y <- ts(numeric(100))  
e <- rnorm(100)  
for(i in 2:100)  
y[i] <- 0.6\*y[i-1] + e[i]

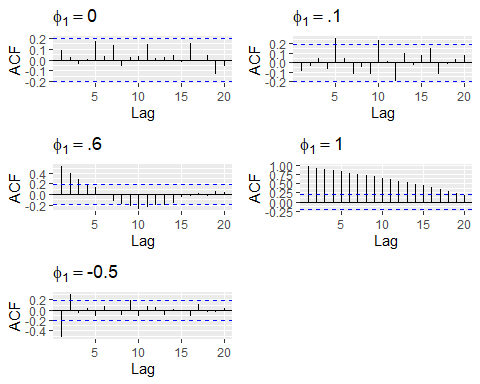
#### b. Produce a time plot for the series. How does the plot change as you change ?

#phi1 = 0  
y0 <- ts(numeric(100))  
e0 <- rnorm(100)  
for(i in 2:100)  
y0[i] <- 0\*y0[i-1] + e0[i]  
p0 = autoplot(y0) + ggtitle(TeX("$\\phi\_1 = 0$"))  
#phi1 = .1  
y\_1 <- ts(numeric(100))  
e\_1 <- rnorm(100)  
for(i in 2:100)  
y\_1[i] <- 0.1\*y\_1[i-1] + e\_1[i]  
p\_1 = autoplot(y\_1) + ggtitle(TeX("$\\phi\_1 = 0.1$"))  
#phi1 = .6  
p\_6 = autoplot(y) + ggtitle(TeX("$\\phi\_1 = 0.6$"))  
#phi1 = 1  
y1 <- ts(numeric(100))  
e1 <- rnorm(100)  
for(i in 2:100)  
y1[i] <- 1\*y1[i-1] + e1[i]  
p1 = autoplot(y1) + ggtitle(TeX("$\\phi\_1 = 1$"))  
#phi1 = -.5  
yn\_5 <- ts(numeric(100))  
en\_5 <- rnorm(100)  
for(i in 2:100)  
yn\_5[i] <- -.5\*yn\_5[i-1] + en\_5[i]  
pn\_5 = autoplot(yn\_5) + ggtitle(TeX("$\\phi\_1 = -0.5$"))  
grid.arrange(p0,p\_1,p\_6,p1,pn\_5)



When approaches 0, represents white noise. As approaches 1, we see greater variation and likely autocorrelation, as expected. Let’s verify by reviewing ACF plots.

acf0 = ggAcf(y0) + ggtitle(TeX("$\\phi\_1 = 0$"))  
acf\_1 = ggAcf(y\_1) + ggtitle(TeX("$\\phi\_1 = .1$"))  
acf\_6 = ggAcf(y) + ggtitle(TeX("$\\phi\_1 = .6$"))  
acf1 = ggAcf(y1) + ggtitle(TeX("$\\phi\_1 = 1$"))  
acfn\_5 = ggAcf(yn\_5) + ggtitle(TeX("$\\phi\_1 = -0.5$"))  
grid.arrange(acf0,acf\_1,acf\_6,acf1,acfn\_5)

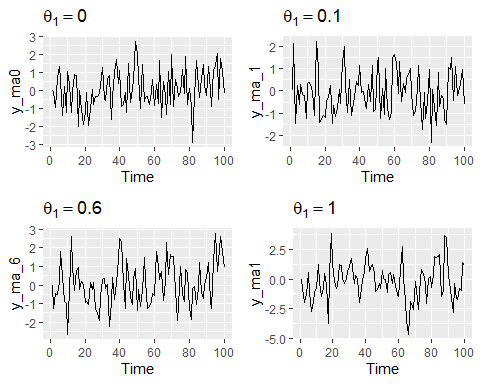
 We see that as increases, the correlations of more lag values exceed the 95% confidence interval, indicating that the data is/are autocorrelated. A negative value causes the correlations of the lags to jump back and forth around 30.

#### c. Write your own code to generate data from an MA(1) model with and .

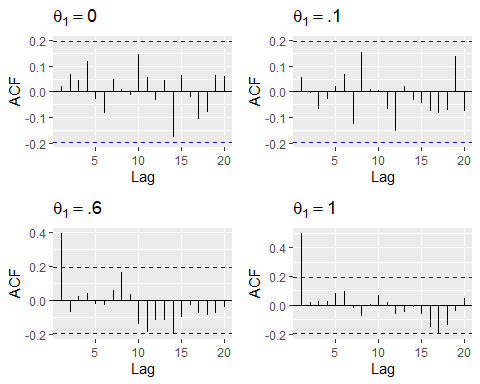
y\_ma\_6 <- ts(numeric(100))  
e\_ma\_6 <- rnorm(100,sd=1)  
for(i in 2:100)  
y\_ma\_6[i] <- 0.6\*e\_ma\_6[i-1] + e\_ma\_6[i]

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

#theta1 = 0  
y\_ma0 <- ts(numeric(100))  
e\_ma0 <- rnorm(100,sd=1)  
for(i in 2:100)  
y\_ma0[i] <- 0\*e\_ma0[i-1] + e\_ma0[i]  
t0 = autoplot(y\_ma0) + ggtitle(TeX("$\\theta\_{1} = 0$"))  
#theta1 = .1  
y\_ma\_1 <- ts(numeric(100))  
e\_ma\_1 <- rnorm(100,sd=1)  
for(i in 2:100)  
y\_ma\_1[i] <- .1\*e\_ma\_1[i-1] + e\_ma\_1[i]  
t\_1 = autoplot(y\_ma\_1) + ggtitle(TeX("$\\theta\_{1} = 0.1$"))  
#theta1 = .6  
t\_6 = autoplot(y\_ma\_6) + ggtitle(TeX("$\\theta\_{1} = 0.6$"))  
#theta1 = 1  
y\_ma1 <- ts(numeric(100))  
e\_ma1 <- rnorm(100,sd=1)  
for(i in 2:100)  
y\_ma1[i] <- 1\*e\_ma1[i-1] + e\_ma1[i]  
t1 = autoplot(y\_ma1) + ggtitle(TeX("$\\theta\_{1} = 1$"))  
grid.arrange(t0,t\_1,t\_6,t1)

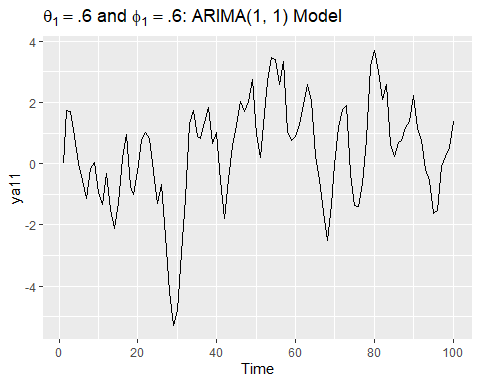


acft0 = ggAcf(y\_ma0) + ggtitle(TeX("$\\theta\_{1} = 0$"))  
acft\_1 = ggAcf(y\_ma\_1) + ggtitle(TeX("$\\theta\_{1} = .1$"))  
acft\_6 = ggAcf(y\_ma\_6) + ggtitle(TeX("$\\theta\_{1} = .6$"))  
acft1 = ggAcf(y\_ma1) + ggtitle(TeX("$\\theta\_{1} = 1$"))  
grid.arrange(acft0,acft\_1,acft\_6,acft1)

 Generally, the MA(1) model shows more variation as increases. However, the effect appears to be less pronounced than with the AR(1) model.

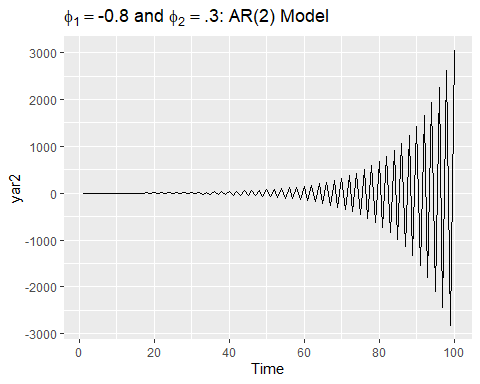
#### e. Generate data from an ARMA(1,1) model with , and .

ya11 <- ts(numeric(100))  
ea11 <- rnorm(100, sd=1)  
for(i in 2:100)  
 ya11[i] <- 0.6\*ya11[i-1] + 0.6\*ea11[i-1] + ea11[i]  
autoplot(ya11) +  
 ggtitle(TeX("$\\theta\_{1} = .6$ and $\\phi\_{1} = .6$: ARIMA(1,1) Model"))



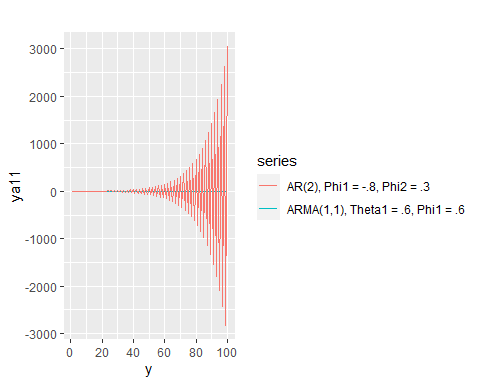
#### f. Generate data from an AR(2) model with , and . (Note that these parameters will give a non-stationary series.)

yar2 <- ts(numeric(100))  
ear2 <- rnorm(100, sd=1)  
for(i in 3:100)  
 yar2[i] <- -0.8\*yar2[i-1] + 0.3\*yar2[i-2] + ear2[i]  
autoplot(yar2) +  
 ggtitle(TeX("$\\phi\_{1} = -0.8$ and $\\phi\_{2} = .3$: AR(2) Model"))



#### g. Graph the latter two series and compare them.

autoplot(ya11, series = "ARMA(1,1), Theta1 = .6, Phi1 = .6") +  
 autolayer(yar2, series = "AR(2), Phi1 = -.8, Phi2 = .3") +  
 xlab("y")



Setting phi1 and ph2 at such values in the AR(2) model creates clear autocorrelation that will grow over time. It’s a non-stationary, sinusoidal pattern. The ARMA(1,1) model is stationary.

### 8.8

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

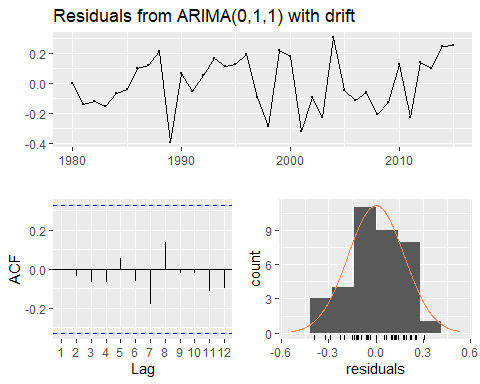
#### a. Use auto.arima() to find an appropriate ARIMA model. What model was selected. Check that the residuals look like white noise. Plot forecasts for the next 10 periods.

mausta <- auto.arima(austa)  
mausta

## 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

We see that ARIMA(0,1,1) with drift is selected as the best model to convey pattern in our Australian visitor data.

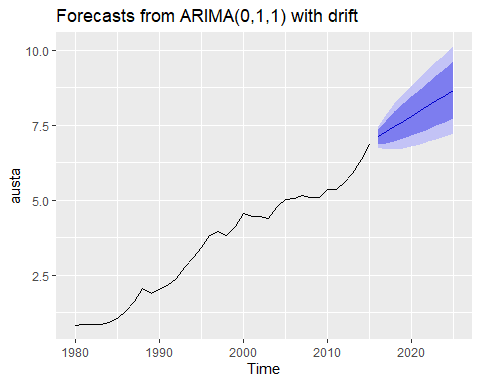
checkresiduals(mausta)



##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(0,1,1) with drift  
## Q\* = 2.297, df = 5, p-value = 0.8067  
##   
## Model df: 2. Total lags used: 7

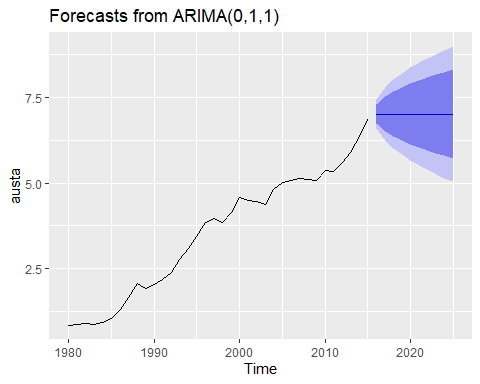
ACF plot shows stationary data, and residual distribution looks fine. Ljung-Box test values appear copacetic. Let’s plot those 10 additional years of visitors.

mausta %>% forecast(h=10) %>% autoplot()



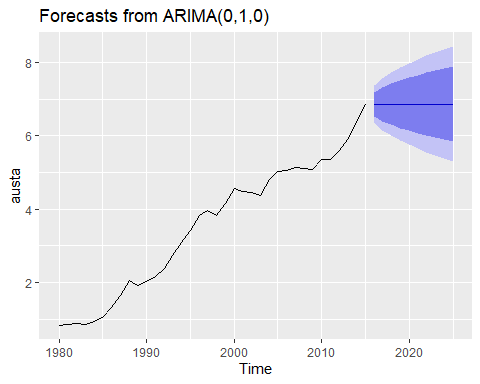
#### b. Plot forecasts from an ARIMA(0,1,1) model with no drift and compare these to part a. Remove the MA term and plot again.

mausta\_nodrift <- Arima(austa,c(0,1,1),include.drift=FALSE)  
mausta\_nodrift %>% forecast(h=10) %>% autoplot()



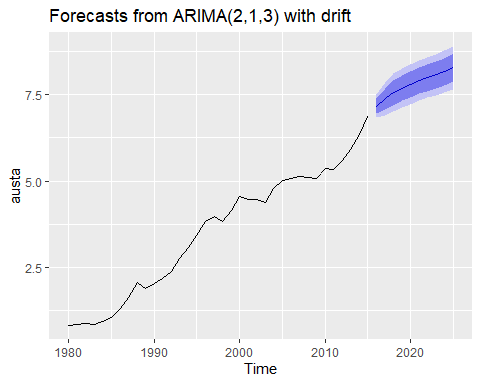
Without drift, forecast values are constant.

mausta\_nodrift\_noma <- Arima(austa,c(0,1,0),include.drift=FALSE)  
mausta\_nodrift\_noma %>% forecast(h=10) %>% autoplot()

 Without the MA term, the curve is slightly less smooth.

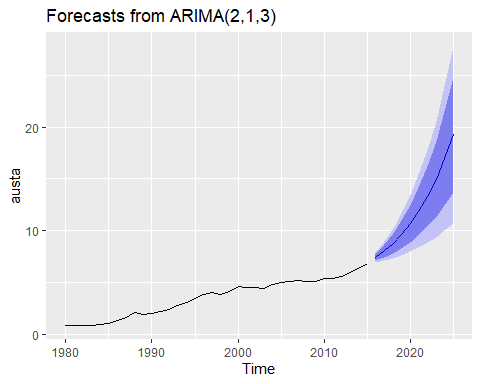
#### c. Plot forecasts from an ARIMA(2,1,3) model with drift. Remove the constant and see what happens.

mausta\_213\_drift <- Arima(austa,c(2,1,3),include.drift=TRUE)  
mausta\_213\_drift %>% forecast(h=10) %>% autoplot()



#get an error when trying to exclude constant without changing method  
mausta\_213\_drift\_noc <- Arima(austa,c(2,1,3),include.constant=FALSE, method="CSS")  
mausta\_213\_drift\_noc %>% forecast(h=10) %>% autoplot()

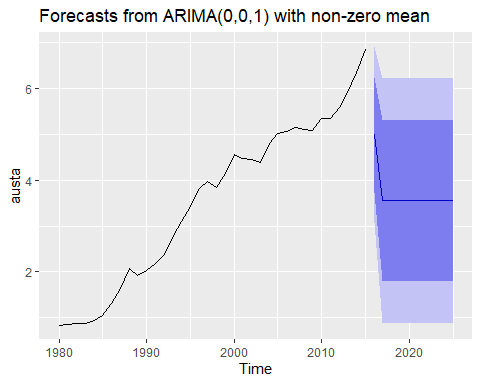
## Warning in predict.Arima(object, n.ahead = h): MA part of model is not  
## invertible



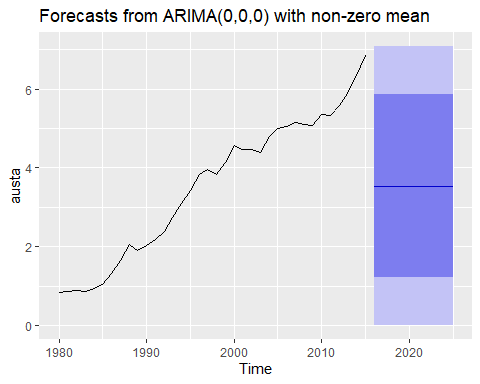
I don’t know what the method change does in order to get excluding the constant to work. Based on the plots, excluding the constant increases the confidence intervals of the forecast.

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

mausta\_001 <- Arima(austa,c(0,0,1))  
mausta\_001 %>% forecast(h=10) %>% autoplot()



mausta\_000 <- Arima(austa,c(0,0,0))  
mausta\_000 %>% forecast(h=10) %>% autoplot()



Without the MA term, the forecast is constant. With it, we see a drop early in the forecasted periods.

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

mausta\_021 <- Arima(austa,c(0,2,1),include.constant=FALSE)  
mausta\_021 %>% forecast(h=10) %>% autoplot()

