DATA624 Homework 1

Esteban Aramayo, Coffy Andrews-Guo, LeTicia Cancel, Joseph Connolly, Ian Costello

5/31/2022

Table of Contents

library(ggfortify)  
library(openxlsx)  
library(fpp2)  
library(fma)  
library(gridExtra)  
library(seasonal)  
library(ggplot2)  
library(patchwork)  
library(caret)  
library(grid)

# Week 1

## HA 2.1

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

The fpp2 forecast package for “Forecasting: Principles and Practice” (2nd Edition) was loaded to explore three time series data library: gold, woolyrnq, and gas. The RStudio IDE help function is a comprehensive built-in system providing the following:

| Data Set | Description | Format | Source |
| --- | --- | --- | --- |
| gold | Daily morning gold prices in US dollars. 1 January 1985 - 31 March 1989. | Time series data | Not Available |
| woolyrnq | Quarterly production of woollen yarn in Australia: tonnes. Mar 1965 - Sep 1994 | Time series data | Time Series Data Library |
| gas | Australian monthly gas production: 1956-1995 | Time series data | Australian Bureau of Statistics |

#help function for each series using question mark "??"  
??gold

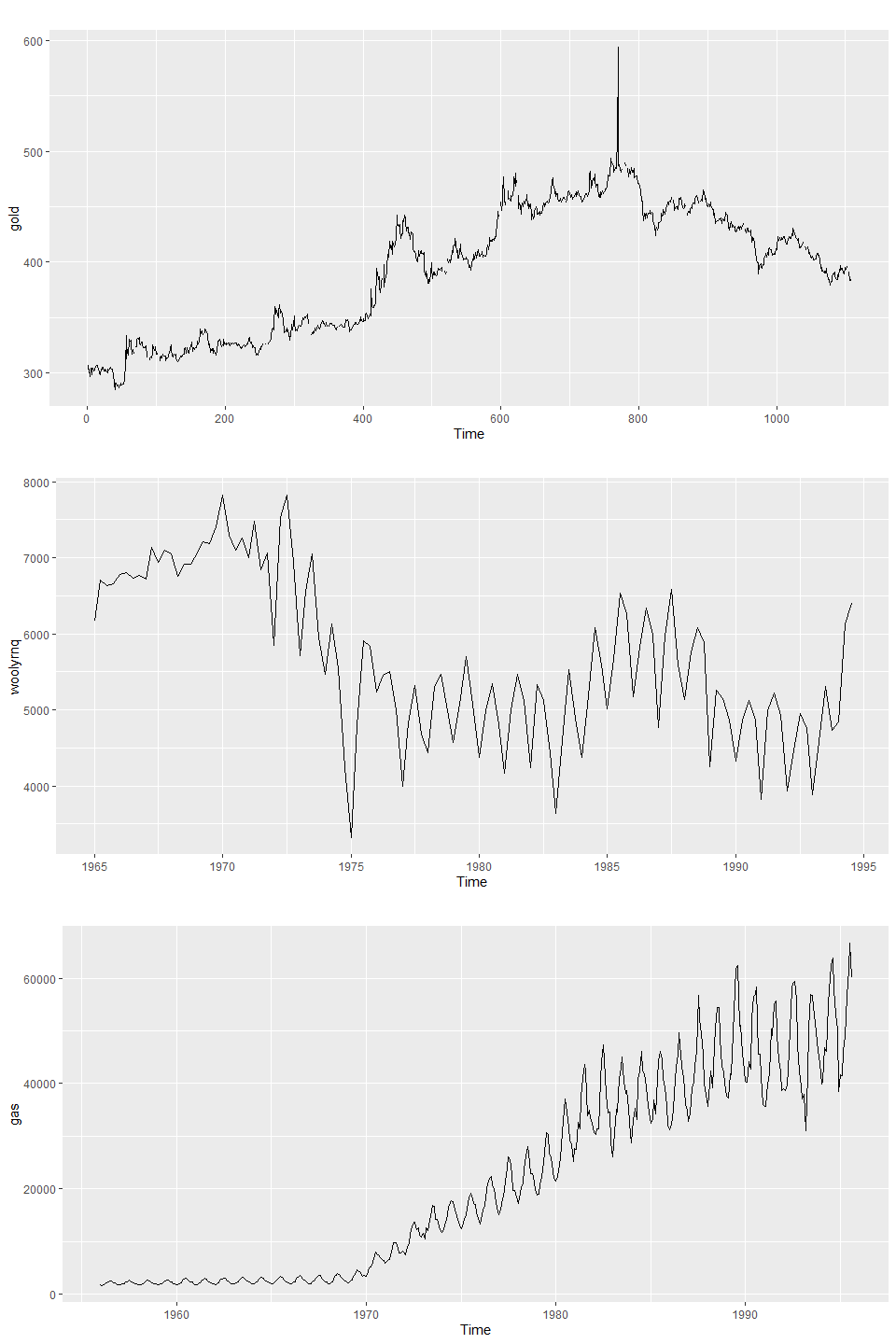
#help function for each series using question mark "??"  
??woolyrnq

#help function for each series using question mark "??"  
??gas

The help function may provide package code examples and hyperlink to additional research documentation.

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

#autoplots of each series  
grid.arrange(autoplot(gold),autoplot(woolyrnq),autoplot(gas))



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

* gold has a frequency of 1, meaning it is annual.
* Woolyrng has a frequency of 4, meaning it is quarterly.
* gas has a frequency of 12, meaning it is monthly.

#frequency of each series  
frequency(gold)

#> [1] 1

frequency(woolyrnq)

#> [1] 4

frequency(gas)

#> [1] 12

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

The outlier for the gold series is observation number 770, with the value of 593.7.

#calling which.max() function  
which.max(gold)

#> [1] 770

#printing maximum value  
gold[which.max(gold)]

#> [1] 593.7

## HA 2.3

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

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

retaildata <- read.xlsx("https://otexts.com/fpp2/extrafiles/retail.xlsx",startRow = 2)

### 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))  
#myts #code is silent because it represents an example - see "myts2"   
  
myts2 <- ts(retaildata[,"A3349791W"],  
 frequency=12, start=c(1982,4))

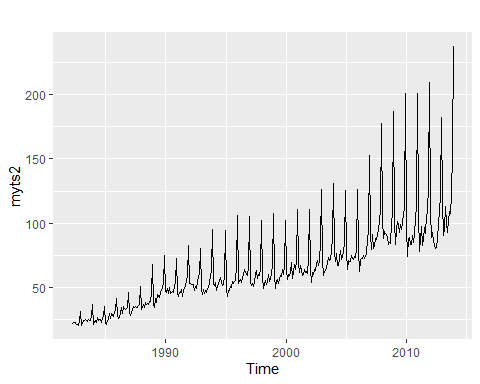
#### Explore your chosen retail time series using the following functions:

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

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

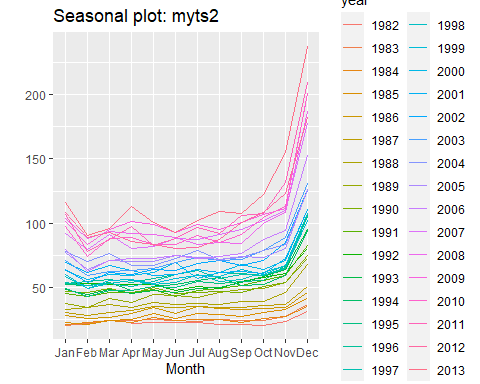
The column selected is the retail of “other recreational goods” in New South Wales. The first plot using autoplot() shows some seasonality for each year telling us that the price spikes at certain points. There is also an upward trend showing that the sales continues to increase overtime and the magnitude of the spikes grows steadily.

#time series qutoplot  
autoplot(myts2)



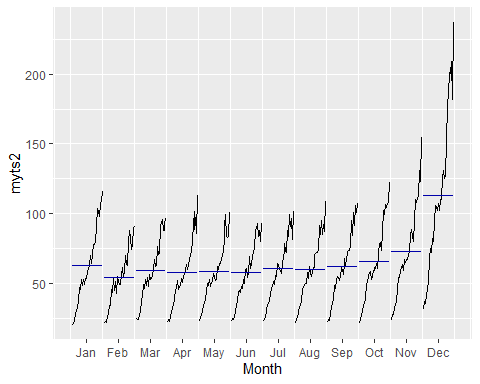
The seasonal plot below using ggseasonlot() gives us more clarity on when the seasonality takes place. Each year has a spike in December with the max prices increasing each year. We can take a guess as to why December is the month that spikes since this is when holiday shopping takes place as well as summer in the southern hemisphere, when more recreation may take place.

#time series seasonal plot  
ggseasonplot(myts2)



The subseries plot confirms the seasonality shown in the previous plots with December being the seasonal month. This plot creates a better visualization for a couple of details better than the previous plots. The difference between the minimum and maximum over the years is clearer. The minimum has a minor increase in December but the maximum has a dramatic increase when compared to the other months. The mean also has a big increase in December but then levels out during the Australian fall and winter.

#time series sub series plot  
ggsubseriesplot(myts2)



## HA 6.2

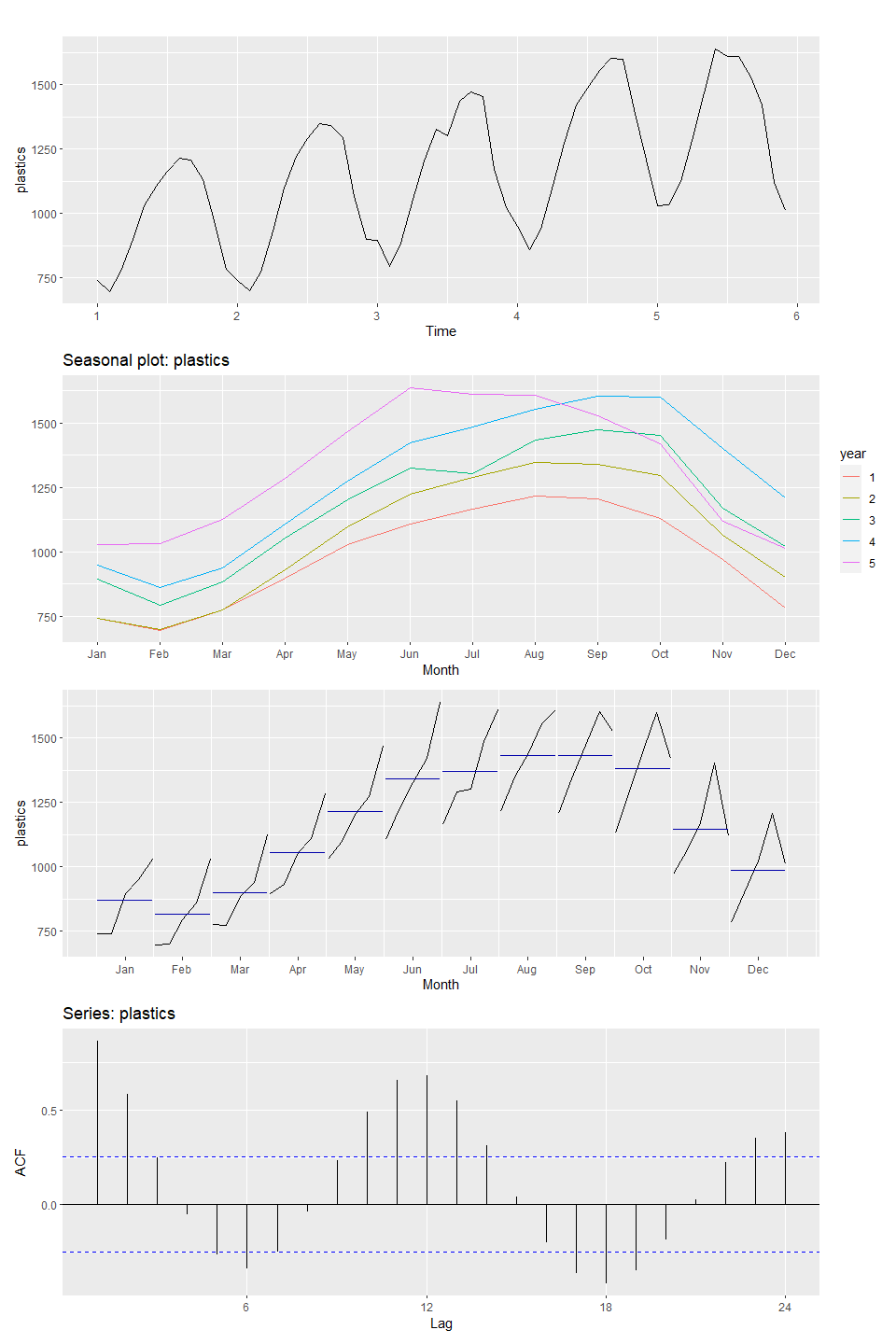
The plastics data set consists of the monthly sales (in thousands) of product A for a plastics manufacturer for five years.

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

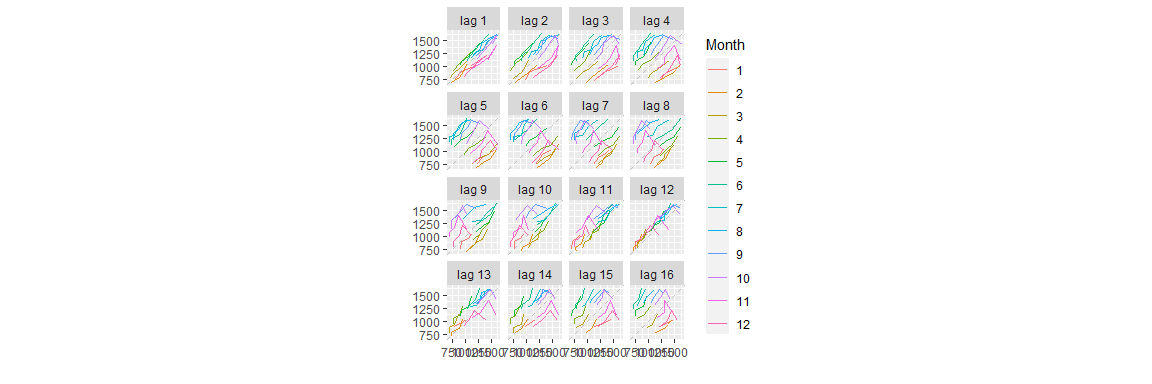
The time plot automatically shows the monthly sales (in thousands) of product A for a plastics manufacturer for five years and reveals: (1) there is an increasing trend; (2) there is a mild seasonal pattern that increases in size as the level of the series increases; and (3) the sudden drop at the year end/starting year is due to government subsidies for pollution control, such as [deposit-refund systems](https://media.rff.org/archive/files/sharepoint/WorkImages/Download/RFF-DP-11-47.pdf).

The season plot shows a seasonal pattern occurs from February to December for each year (five years) time series.

#autoplot(), ggseasonplot(), ggsubseriesplot(), gglagplot(), ggAcf()  
p1 <- autoplot(plastics)  
p2 <- ggseasonplot(plastics)  
p3 <- ggsubseriesplot(plastics)  
p4 <- gglagplot(plastics)  
p5 <- ggAcf(plastics)  
p1 / p2 / p3 / p5

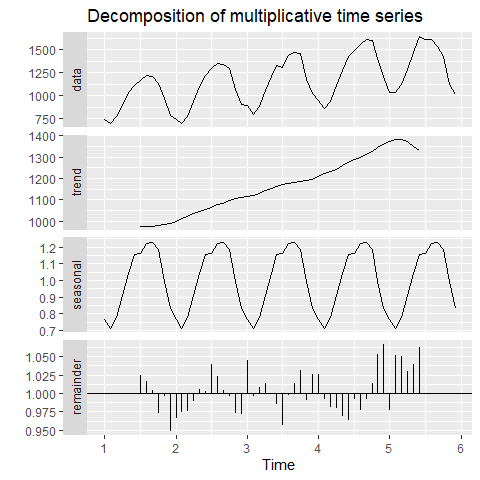


p4



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

#multiplicative decomposition  
decomp\_plastics <- plastics %>%  
 decompose(type = "multiplicative")  
  
decomp\_plastics %>% autoplot()

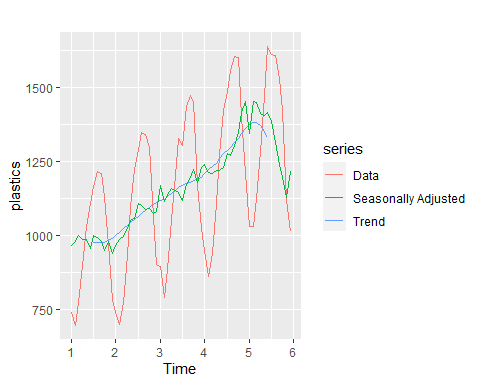


### c. Do the results support the graphical interpretation from part a?

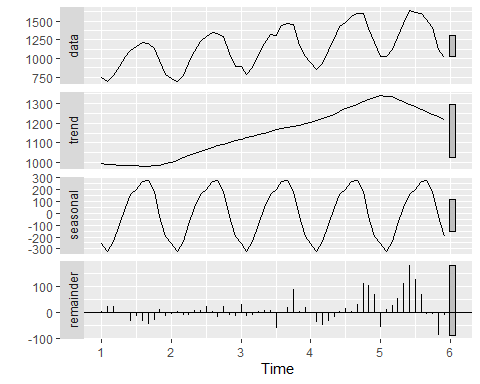
The “Data” and “Seasonal” sections of the Decomposition shows similar results to the autoplot() chart. The seasonal pattern is unchanging, the remainder component has a lot of large values, and has an increasing trend with some missing observations from the beginning and the end of the data set. There is a steady seasonal trend that has a similar duration for each seasonal cycle. The “Trend” section of the Decomposition also supports the visuals from part A and shows a steady upward trend from years 1-5.

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

plastics %>% decompose(type = "multiplicative") -> fit  
  
autoplot(plastics, series = "Data") +   
 autolayer(trendcycle(decomp\_plastics), series = "Trend") +  
 autolayer(seasadj(decomp\_plastics), series = "Seasonally Adjusted")



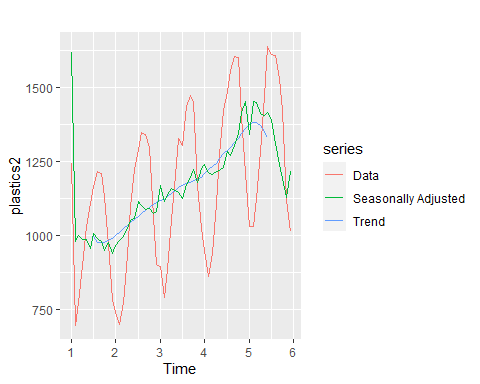
#fit <- seas(x = plastics, x11="")  
  
plastics %>%  
 stl(s.window = "periodic", robust = TRUE) %>%  
 autoplot()



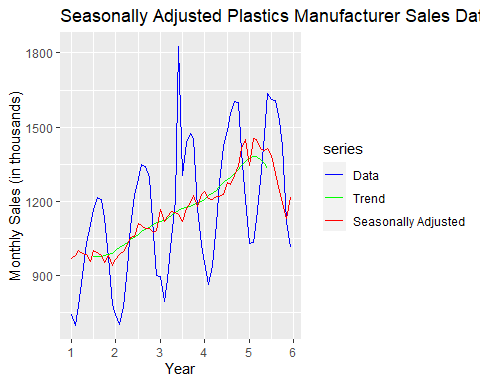
### e. Change one observation to be an outlier (e.g., add 500 to one observation), and recompute the seasonally adjusted data. What is the effect of the outlier?

The outlier was applied to the first observation in the plastics data so the chart changed only in year 1. The outlier being in the beginning of the data had no impact on the rest of the plot.

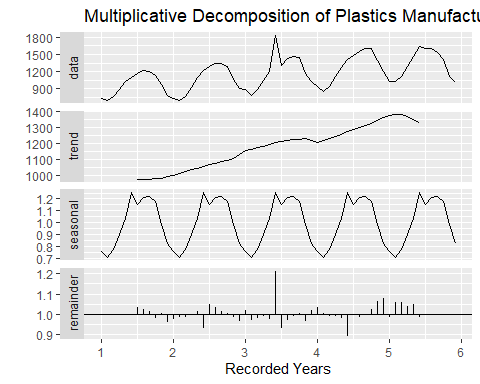
#creating a copy of plastics and adding 500 to one observation  
plastics2 <- plastics  
plastics2[1] <- plastics2[1] + 500  
  
decomp\_plastics2 <- plastics2 %>%  
 decompose(type = "multiplicative")  
  
autoplot(plastics2, series = "Data") +   
 autolayer(trendcycle(decomp\_plastics2), series = "Trend") +  
 autolayer(seasadj(decomp\_plastics2), series = "Seasonally Adjusted")



plasticsOutlier <- plastics  
plasticsOutlier[30] <- plasticsOutlier[30] + 500  
autoplot(plasticsOutlier, series = "Data") + autolayer(trendcycle(fit), series = "Trend") + autolayer(seasadj(fit), series = "Seasonally Adjusted") + xlab("Year") + ylab("Monthly Sales (in thousands)") + ggtitle("Seasonally Adjusted Plastics Manufacturer Sales Data") + scale\_color\_manual(values = c("blue", "green", "red"), breaks = c("Data", "Trend", "Seasonally Adjusted"))



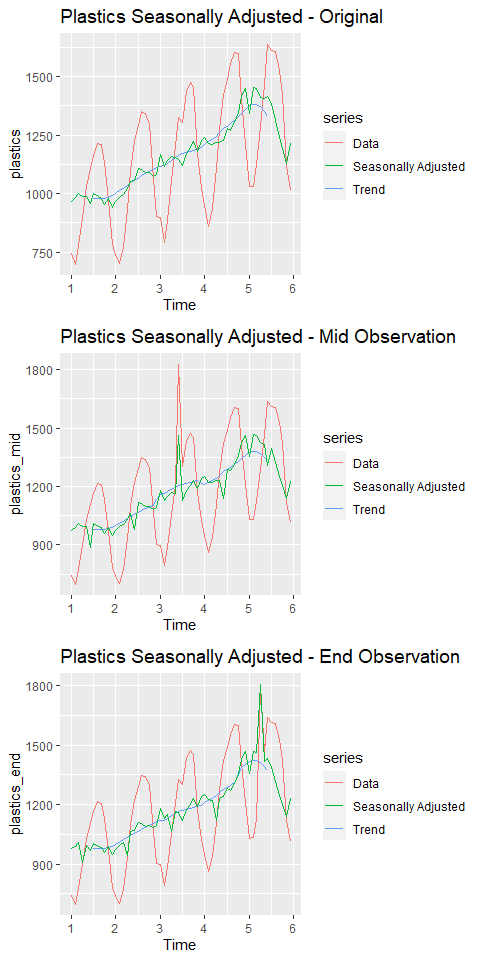
plasticsOutlier %>% decompose(type="multiplicative") %>%  
 autoplot() + xlab("Recorded Years") +  
 ggtitle("Multiplicative Decomposition of Plastics Manufacturer Sales Data")



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

The middle outlier has a greater impact on the seasonally adjusted line than the end outlier when compared to the original plot. When the last observation is the outlier the Data and Seasonally Adjusted lines trend upward and all other areas on the chart are not impacted. When a middle observation is the outlier we see a different seasonally adjusted line when compared to the original chart. The downward and upward spikes around the outlier are sharper on the middle outlier chart while the original chart has subtle changes with a fairly even upward trend.

plastics\_mid <- plastics  
plastics\_end <- plastics   
plastics\_mid[30] <- plastics\_mid[30] + 500 #outlier in the middle  
plastics\_end[52] <- plastics\_end[52] + 500 #outlier to reflect "near the end"  
  
decomp\_mid <- plastics\_mid %>%  
 decompose(type = "multiplicative")  
  
p <- autoplot(plastics, series = "Data") +   
 autolayer(trendcycle(decomp\_plastics), series = "Trend") +  
 autolayer(seasadj(decomp\_plastics), series = "Seasonally Adjusted") +  
 ggtitle("Plastics Seasonally Adjusted - Original") +  
 theme(plot.title = element\_text(size = 14))  
  
p1 <- autoplot(plastics\_mid, series = "Data") +   
 autolayer(trendcycle(decomp\_mid), series = "Trend") +  
 autolayer(seasadj(decomp\_mid), series = "Seasonally Adjusted") +  
 ggtitle("Plastics Seasonally Adjusted - Mid Observation")+  
 theme(plot.title = element\_text(size = 14))  
  
decomp\_end <- plastics\_end %>%  
 decompose(type = "multiplicative")  
  
p2 <- autoplot(plastics\_end, series = "Data") +   
 autolayer(trendcycle(decomp\_end), series = "Trend") +  
 autolayer(seasadj(decomp\_end), series = "Seasonally Adjusted") +  
 ggtitle("Plastics Seasonally Adjusted - End Observation")+  
 theme(plot.title = element\_text(size = 14))  
  
grid.arrange(p, p1,p2)



# Week 2

## KJ 3.1

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

library(mlbench)  
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 ...

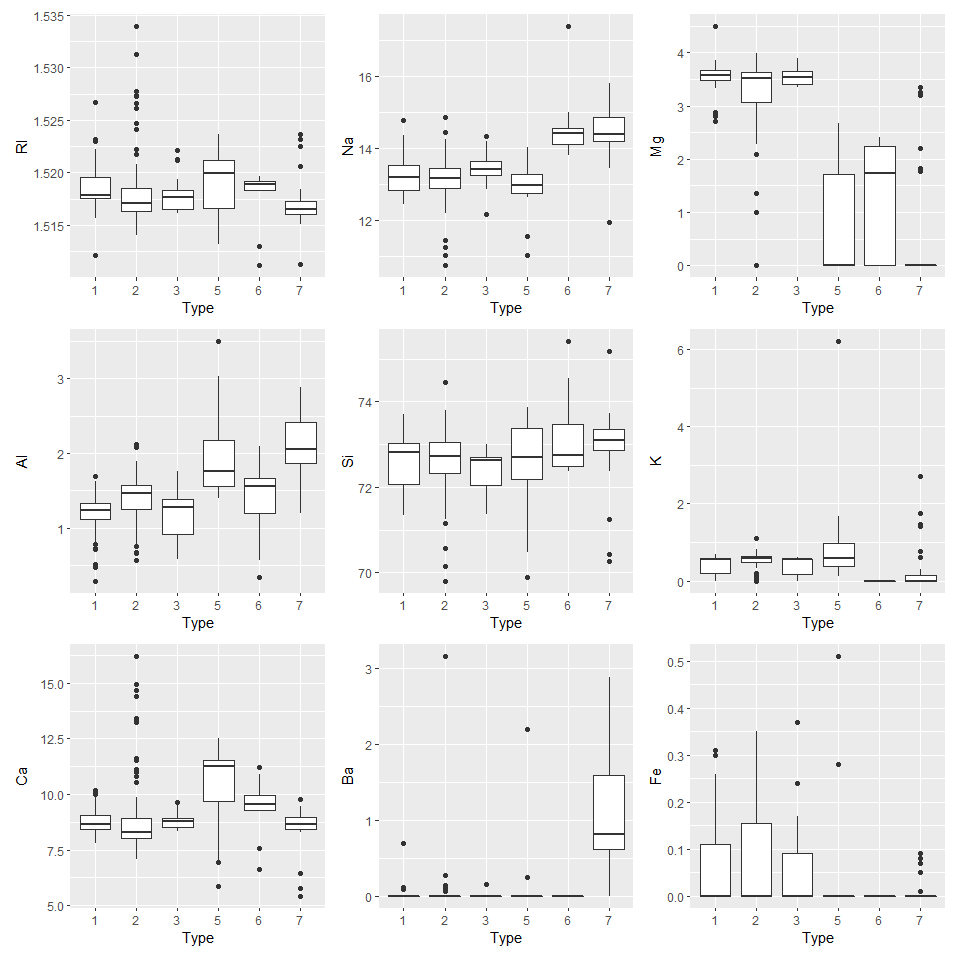
### a. Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors.

When we explore the Glass dataset using summary we see that there are no NA values for any of the predictors. There are also no negative values with the minimum value being 0.00 (zero) for some predictors.

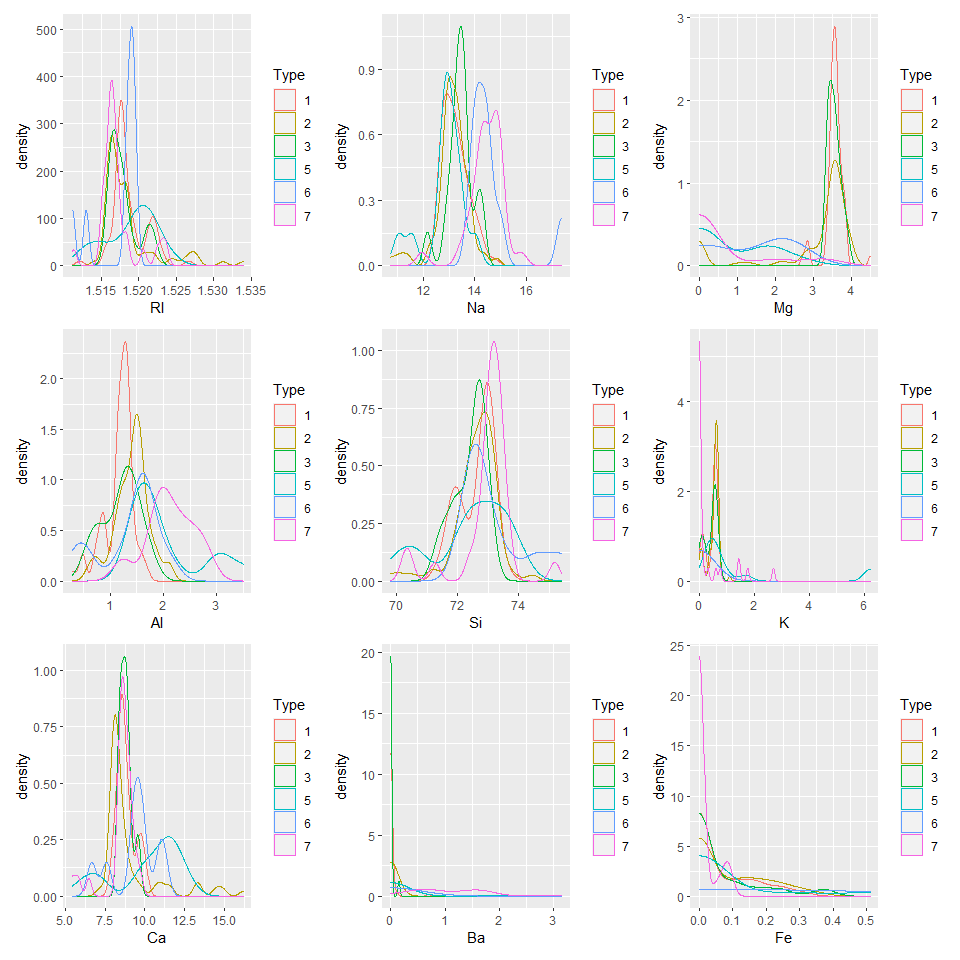
summary(Glass)

#> 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 Type   
#> Min. :0.00000 1:70   
#> 1st Qu.:0.00000 2:76   
#> Median :0.00000 3:17   
#> Mean :0.05701 5:13   
#> 3rd Qu.:0.10000 6: 9   
#> Max. :0.51000 7:29

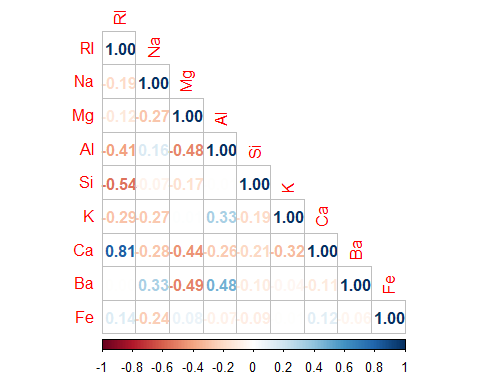
ri\_b <- ggplot(Glass, aes(x = Type, y = RI)) +  
 geom\_boxplot()  
  
na\_b <- ggplot(Glass, aes(x = Type, y = Na)) +  
 geom\_boxplot()  
  
mg\_b <- ggplot(Glass, aes(x = Type, y = Mg)) +  
 geom\_boxplot()  
  
al\_b <- ggplot(Glass, aes(x = Type, y = Al)) +  
 geom\_boxplot()  
  
si\_b <- ggplot(Glass, aes(x = Type, y = Si)) +  
 geom\_boxplot()  
  
k\_b <- ggplot(Glass, aes(x = Type, y = K)) +  
 geom\_boxplot()  
  
ca\_b <- ggplot(Glass, aes(x = Type, y = Ca)) +  
 geom\_boxplot()  
  
ba\_b <- ggplot(Glass, aes(x = Type, y = Ba)) +  
 geom\_boxplot()  
  
fe\_b <- ggplot(Glass, aes(x = Type, y = Fe)) +  
 geom\_boxplot()  
  
  
ri\_b+na\_b+mg\_b+al\_b+si\_b+k\_b+ca\_b+ba\_b+fe\_b+  
 plot\_layout(ncol=3)



ri\_d <- ggplot(Glass, aes(RI, color=Type)) + geom\_density()  
na\_d <- ggplot(Glass, aes(Na, color=Type)) + geom\_density()  
mg\_d <- ggplot(Glass, aes(Mg, color=Type)) + geom\_density()  
al\_d <- ggplot(Glass, aes(Al, color=Type)) + geom\_density()  
si\_d <- ggplot(Glass, aes(Si, color=Type)) + geom\_density()  
k\_d <- ggplot(Glass, aes(K, color=Type)) + geom\_density()  
ca\_d <- ggplot(Glass, aes(Ca, color=Type)) + geom\_density()  
ba\_d <- ggplot(Glass, aes(Ba, color=Type)) + geom\_density()  
fe\_d <- ggplot(Glass, aes(Fe, color=Type)) + geom\_density()  
  
ri\_d+na\_d+mg\_d+al\_d+si\_d+k\_d+ca\_d+ba\_d+fe\_d+  
 plot\_layout(ncol=3)



corrplot::corrplot(cor(Glass[c("RI", "Na", "Mg", "Al", "Si", "K", "Ca", "Ba", "Fe")]), method = 'number', type = 'lower')

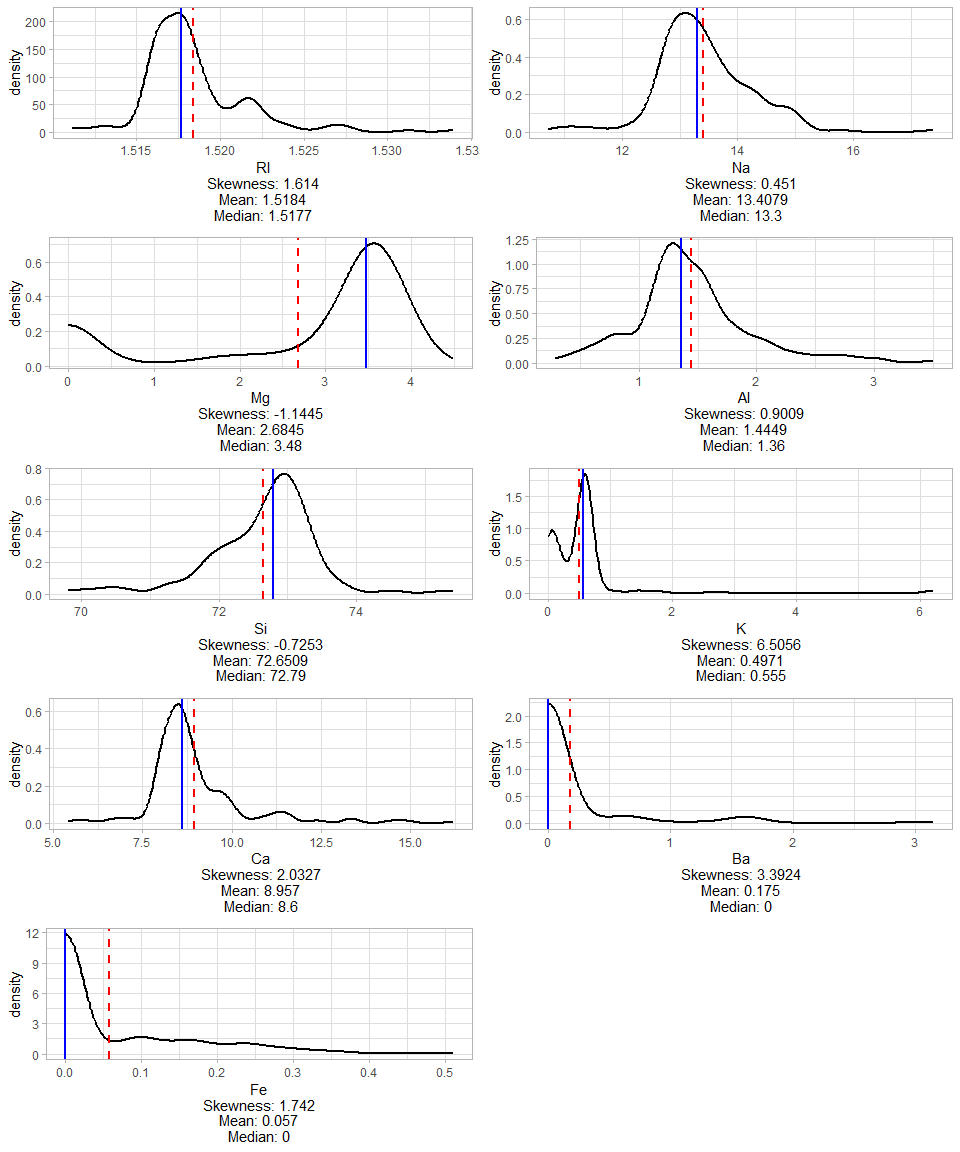


### b. Do there appear to be any outliers in the data? Are any predictors skewed?

The boxplots help us see the outliers for each type for each predictor. Type 2 has outliers for the Ri, Mg, and Ca predictors. The Ba predictor has more data for Type 7 with the rest of the types being scattered. The mean for Si is consistent between all types. When we explore the distribution plots we see that Si is closest to a normal distribution. K, Ba, and Fa are left skewed and Mg is right skewed. The other predictors are slightly skewed.

Looking at the density plots, it does appear that Mg, K, Ba, and Fe are especially more skewed than their counterparts. Mg follows a dramatic left skew, while Ba, K, and Fe follow a dramatic right skew. RI and Ca follow have similar distributions of right skews, and Na and Si seem to be the most normal, with Si following a more normal distribution. Al has distributions all over, but seems to follow a slight right skew.

library(moments)  
library(gridExtra)  
  
doPlots <- function(Glass, fun, ii, ncol=3) {  
pp <- list()  
for (i in ii) {  
p <- fun(Glass=Glass, i=i)  
pp <- c(pp, list(p))  
}  
do.call("grid.arrange", c(pp, ncol=ncol))  
}  
  
plotDen <- function(Glass, i){  
data <- data.frame(x=Glass[[i]], type = Glass$Type)  
df <- as.data.frame(mean(data$x))  
colnames(df) <- "mean"  
df$median <- median(data$x)  
p <- ggplot(data= data) +  
geom\_line(aes(x = x), stat = 'density', size = 1,alpha = 1.0) +  
xlab(paste0((colnames(Glass)[i]), '\n',  
'Skewness: ',round(skewness(Glass[[i]], na.rm = TRUE), 4), '\n',  
'Mean: ', round(df$mean, 4), '\n',  
'Median: ', round(df$median, 4))) +  
theme\_light() +  
geom\_vline(data=df, aes(xintercept=df$mean), linetype = "dashed", size=1, colour="red") +  
geom\_vline(data=df, aes(xintercept=df$median), size = 1, colour="blue")  
  
return(p)  
}  
  
  
doPlots(Glass, plotDen, ii=1:9, ncol=2)



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

Experimenting with BoxCox transformations for Ba and Fe since both predictors are heavily skewed to the left. The skewed variables could improve the classification model. This is because Box Cox transformation is an efficient way of an optimal transformation.

## KJ 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:

library(mlbench)  
data(Soybean)  
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

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   
#>   
#>   
#>

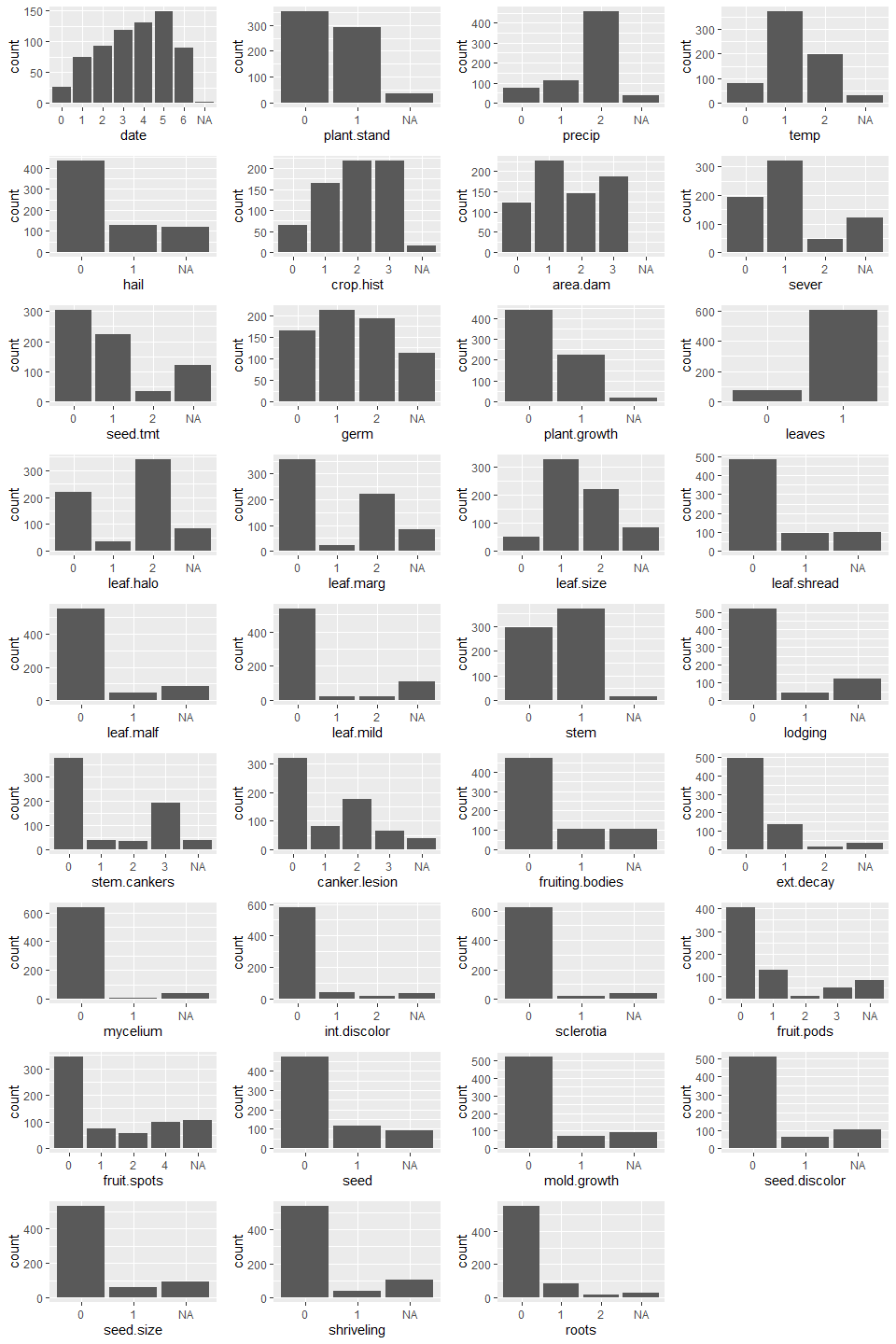
## See ?Soybean for details

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

All but one predictor contain NA values and can be evaluated to see if it is a degenerate distribution. There are no predictors where NA outnumber real values but there are some with a large number of NA. ‘hail’, ‘sever’, ‘seed.tmt’, and ‘germ’ have a large amount of NA values so we will explore how that impacts the data later.

\*\*\*\*\*\* Balance of the variables \*\*\*\*\*\*\*

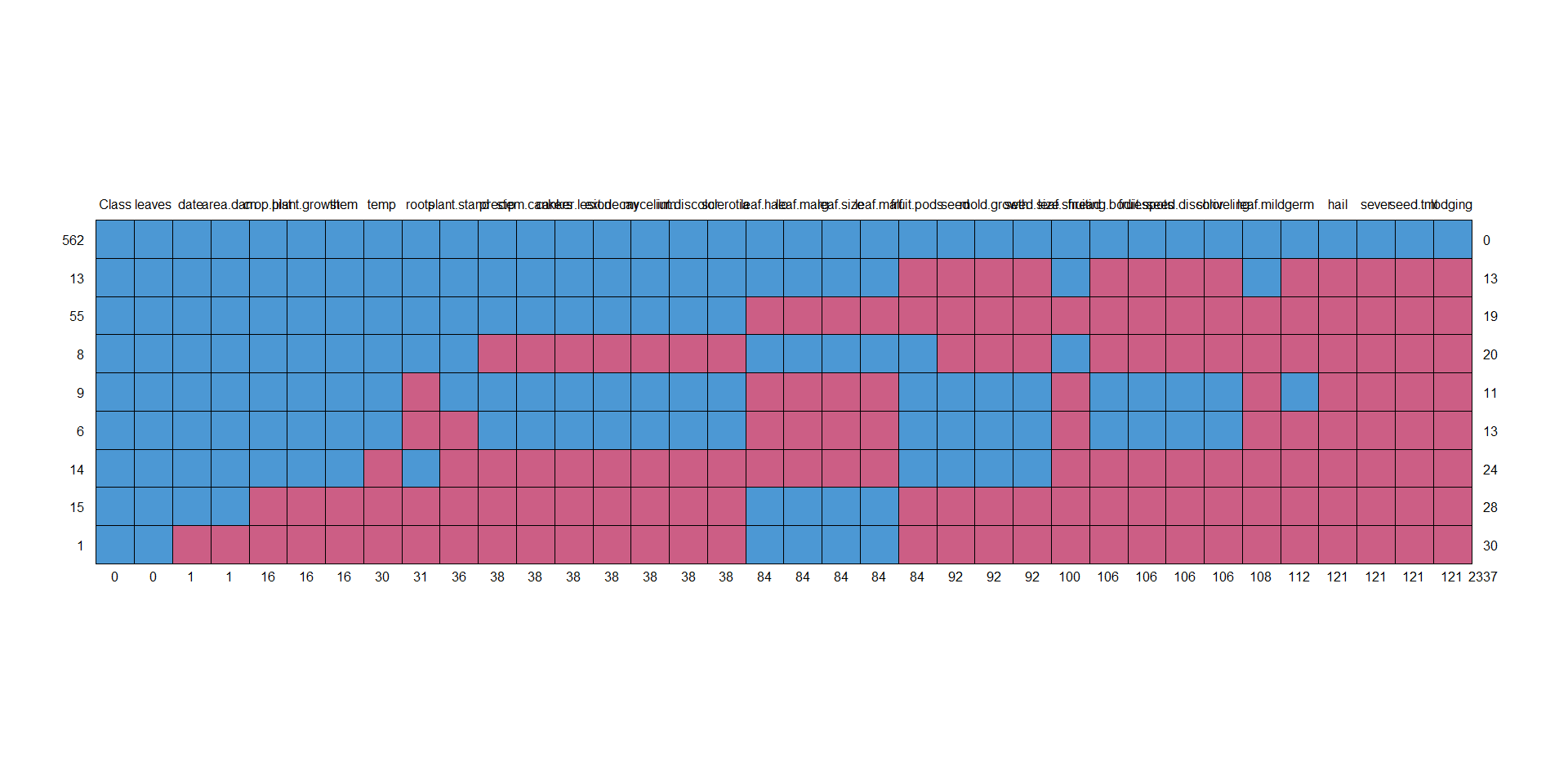
col\_names <- colnames(Soybean[-1])  
  
plot\_list <- list()  
  
for (i in col\_names){  
 plot <- ggplot(Soybean, aes\_string(Soybean[,i])) +  
 geom\_bar() +  
 xlab(colnames(Soybean[i]))   
 plot\_list[[i]] <- plot  
 #print(plot)  
}  
  
grid.arrange(grobs=plot\_list, ncol=4)



### 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?

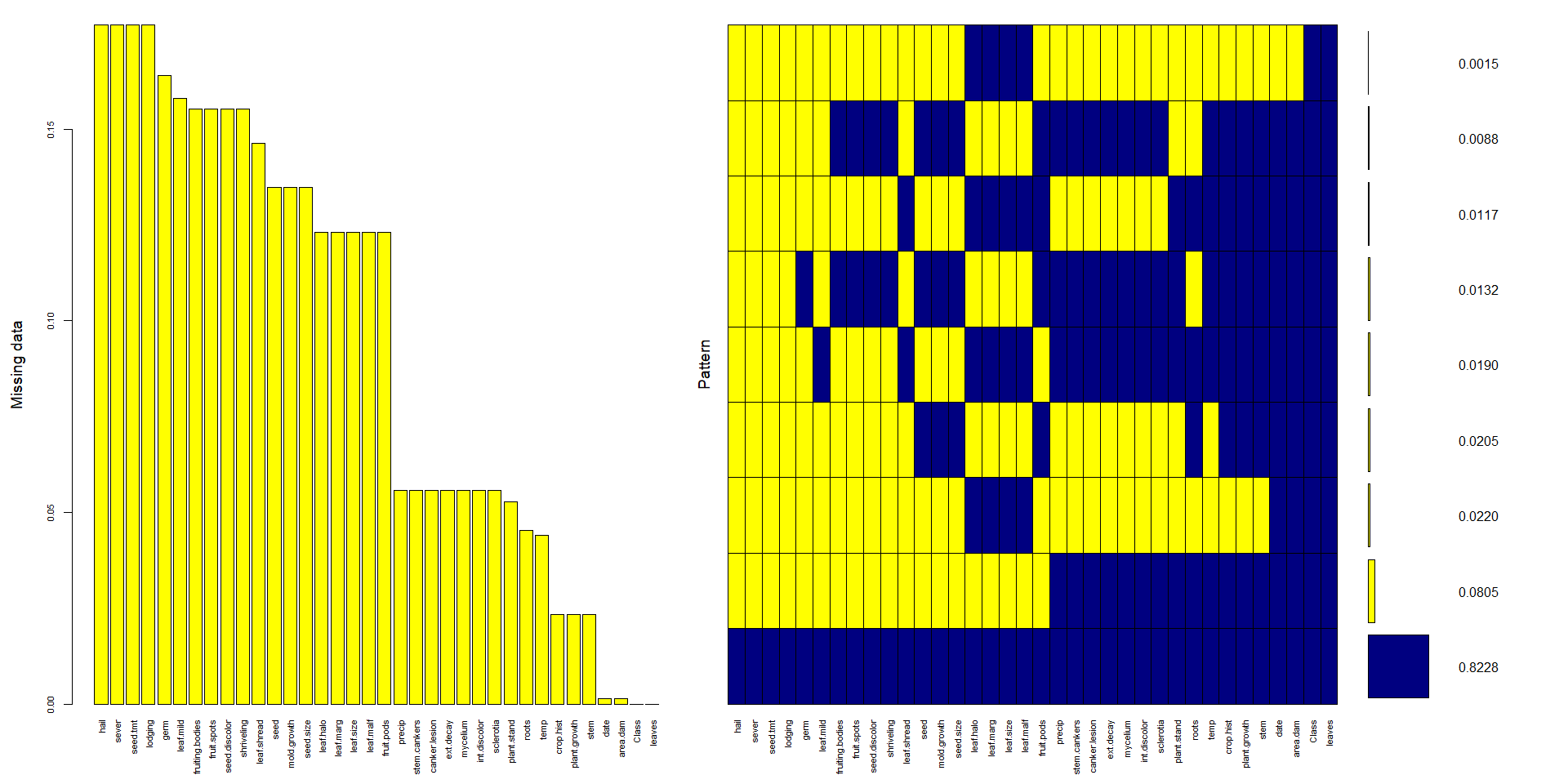
Of the 18% of missing data, 8% come from 19 out of 36 columns: sever, seed.tmt, germ, leaf.halo, leaf.marg, leaf.size, leaf.shread, leaf.mid, stem.cankers, canker.lesion, ext.decay, int.discolor, fruit.pods, fruit.spots, and roots all seem to have data that’s more likely to be missing (more than the smallest value).

library(mice)  
library(VIM)  
md.pattern(Soybean)



#> Class leaves date area.dam crop.hist plant.growth stem temp roots  
#> 562 1 1 1 1 1 1 1 1 1  
#> 13 1 1 1 1 1 1 1 1 1  
#> 55 1 1 1 1 1 1 1 1 1  
#> 8 1 1 1 1 1 1 1 1 1  
#> 9 1 1 1 1 1 1 1 1 0  
#> 6 1 1 1 1 1 1 1 1 0  
#> 14 1 1 1 1 1 1 1 0 1  
#> 15 1 1 1 1 0 0 0 0 0  
#> 1 1 1 0 0 0 0 0 0 0  
#> 0 0 1 1 16 16 16 30 31  
#> plant.stand precip stem.cankers canker.lesion ext.decay mycelium  
#> 562 1 1 1 1 1 1  
#> 13 1 1 1 1 1 1  
#> 55 1 1 1 1 1 1  
#> 8 1 0 0 0 0 0  
#> 9 1 1 1 1 1 1  
#> 6 0 1 1 1 1 1  
#> 14 0 0 0 0 0 0  
#> 15 0 0 0 0 0 0  
#> 1 0 0 0 0 0 0  
#> 36 38 38 38 38 38  
#> int.discolor sclerotia leaf.halo leaf.marg leaf.size leaf.malf fruit.pods  
#> 562 1 1 1 1 1 1 1  
#> 13 1 1 1 1 1 1 0  
#> 55 1 1 0 0 0 0 0  
#> 8 0 0 1 1 1 1 1  
#> 9 1 1 0 0 0 0 1  
#> 6 1 1 0 0 0 0 1  
#> 14 0 0 0 0 0 0 1  
#> 15 0 0 1 1 1 1 0  
#> 1 0 0 1 1 1 1 0  
#> 38 38 84 84 84 84 84  
#> seed mold.growth seed.size leaf.shread fruiting.bodies fruit.spots  
#> 562 1 1 1 1 1 1  
#> 13 0 0 0 1 0 0  
#> 55 0 0 0 0 0 0  
#> 8 0 0 0 1 0 0  
#> 9 1 1 1 0 1 1  
#> 6 1 1 1 0 1 1  
#> 14 1 1 1 0 0 0  
#> 15 0 0 0 0 0 0  
#> 1 0 0 0 0 0 0  
#> 92 92 92 100 106 106  
#> seed.discolor shriveling leaf.mild germ hail sever seed.tmt lodging   
#> 562 1 1 1 1 1 1 1 1 0  
#> 13 0 0 1 0 0 0 0 0 13  
#> 55 0 0 0 0 0 0 0 0 19  
#> 8 0 0 0 0 0 0 0 0 20  
#> 9 1 1 0 1 0 0 0 0 11  
#> 6 1 1 0 0 0 0 0 0 13  
#> 14 0 0 0 0 0 0 0 0 24  
#> 15 0 0 0 0 0 0 0 0 28  
#> 1 0 0 0 0 0 0 0 0 30  
#> 106 106 108 112 121 121 121 121 2337

aggr(Soybean, col=c('navyblue','yellow'),  
 numbers=TRUE, sortVars=TRUE,  
 labels=names(Soybean), cex.axis=.7,  
 gap=3, ylab=c("Missing data","Pattern"))



#>   
#> 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

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

We chose to impute the data instead of eliminating predictors altogether. Predictors with NA values still had enough data where we didn’t feel that complete removal was necessary. Below is a comparison of the summary for the original dataset and the imputed dataset. After the imputation, the numbers do not differ too much but we would need to continue this analysis to see how much the imputation impacted the dataset.

impute\_soybean <- parlmice(Soybean, maxit = 5, m = 1, printFlag = FALSE, seed = 500, cluster.seed = 500)  
Soybean\_2 <- complete(impute\_soybean,1)

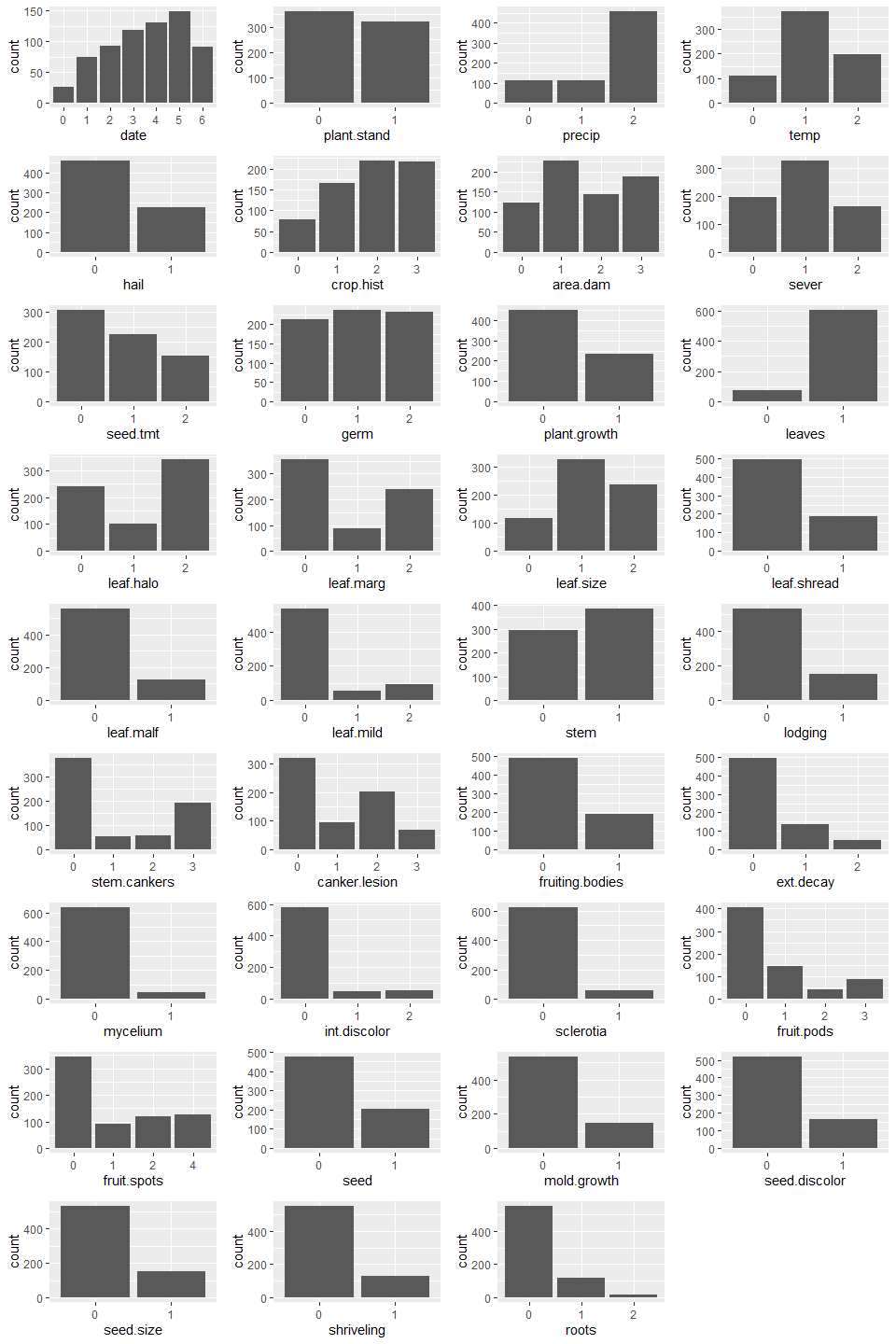
#original Soybean data  
summary(Soybean)

#> 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   
#>   
#>   
#>

#imputed soybean data  
summary(Soybean\_2)

#> Class date plant.stand precip temp hail   
#> brown-spot : 92 0: 26 0:362 0:112 0:110 0:460   
#> alternarialeaf-spot: 91 1: 75 1:321 1:112 1:374 1:223   
#> frog-eye-leaf-spot : 91 2: 93 2:459 2:199   
#> phytophthora-rot : 88 3:118   
#> anthracnose : 44 4:131   
#> brown-stem-rot : 44 5:149   
#> (Other) :233 6: 91   
#> crop.hist area.dam sever seed.tmt germ plant.growth leaves leaf.halo  
#> 0: 79 0:123 0:195 0:306 0:212 0:451 0: 77 0:239   
#> 1:166 1:228 1:326 1:224 1:238 1:232 1:606 1:102   
#> 2:220 2:145 2:162 2:153 2:233 2:342   
#> 3:218 3:187   
#>   
#>   
#>   
#> leaf.marg leaf.size leaf.shread leaf.malf leaf.mild stem lodging  
#> 0:357 0:118 0:497 0:559 0:536 0:296 0:533   
#> 1: 88 1:328 1:186 1:124 1: 54 1:387 1:150   
#> 2:238 2:237 2: 93   
#>   
#>   
#>   
#>   
#> stem.cankers canker.lesion fruiting.bodies ext.decay mycelium int.discolor  
#> 0:379 0:320 0:492 0:497 0:639 0:581   
#> 1: 53 1: 94 1:191 1:136 1: 44 1: 46   
#> 2: 58 2:202 2: 50 2: 56   
#> 3:193 3: 67   
#>   
#>   
#>   
#> sclerotia fruit.pods fruit.spots seed mold.growth seed.discolor seed.size  
#> 0:625 0:407 0:345 0:477 0:537 0:520 0:532   
#> 1: 58 1:146 1: 92 1:206 1:146 1:163 1:151   
#> 2: 42 2:121   
#> 3: 88 4:125   
#>   
#>   
#>   
#> shriveling roots   
#> 0:552 0:551   
#> 1:131 1:116   
#> 2: 16   
#>   
#>   
#>   
#>

col\_names <- colnames(Soybean\_2[-1])  
  
plot\_list <- list()  
  
for (i in col\_names){  
 plot <- ggplot(Soybean\_2, aes\_string(Soybean\_2[,i])) +  
 geom\_bar() +  
 xlab(colnames(Soybean\_2[i]))   
 plot\_list[[i]] <- plot  
 #print(plot)  
}  
  
grid.arrange(grobs=plot\_list, ncol=4)

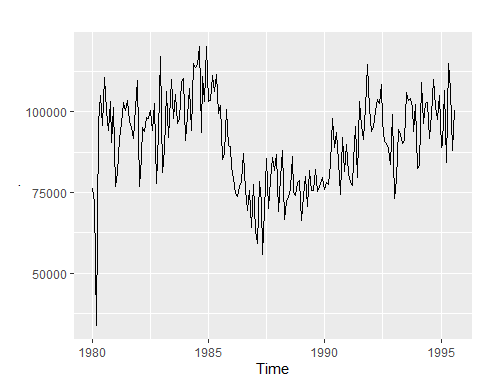


## HA 7.1

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

The Simple Exponential Method (“SES”) method is suitable to forecast time series with no clear trend or seasonality. The Pigs data does not have a clear pattern and therefore we will use SES.

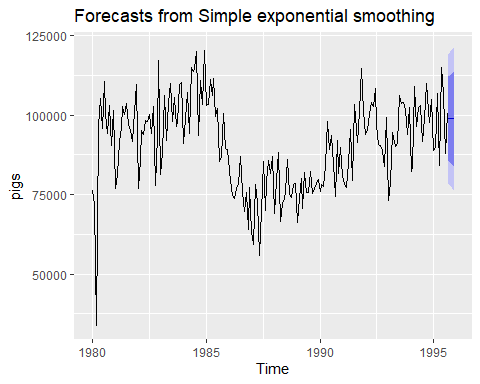
help(pigs)  
pigs %>% autoplot()



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

Using the summary() function and .

#estimate parameter - the next four months  
fc<- ses(pigs, h=4)  
  
#timeseries plot with autoplot() function  
fc %>%  
 autoplot()



summary(fc)

#>   
#> Forecast method: Simple exponential smoothing  
#>   
#> Model Information:  
#> Simple exponential smoothing   
#>   
#> Call:  
#> 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 0.7966249 0.01282239  
#>   
#> Forecasts:  
#> Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
#> Sep 1995 98816.41 85605.43 112027.4 78611.97 119020.8  
#> Oct 1995 98816.41 85034.52 112598.3 77738.83 119894.0  
#> Nov 1995 98816.41 84486.34 113146.5 76900.46 120732.4  
#> Dec 1995 98816.41 83958.37 113674.4 76092.99 121539.8

### b. 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.

From chapter 3.5. prediction intervals are calculated as . The multiplier for 95% interval is 1.96 and the residuals are also equal to the RMSE. Using the SES model formula from part a, the $ values are stored in a vector as are the residuals. Then, the values are subbed in for the formula.

The computed intervals vs. the predicted intervals are quite close. The September 1995 low interval at 95% is 78611.97, compared to the computed 78679.97 only a difference of 68. The high values for the predicted interval at 95% is 119020.80, compared to 118952.84 again differing by 68.

y\_hat <- c(1.96, -1.96)  
s <- sd(residuals(fc))  
  
ses(pigs, h=4)$mean[1]+(y\_hat\*s)

#> [1] 118952.84 78679.97

## HA 7.2

#### Write your own function to implement simple exponential smoothing. The function should take arguments y (the time series), alpha (the smoothing parameter α) and level (the initial level ℓ0). It should return the forecast of the next observation in the series. Does it give the same forecast as ses()?

Let’s create our own version of the () function using the formula for weighted average form

myses <- function(y, alpha, level) {  
   
 # set initial estimated y with level  
 y\_hat <- level  
   
 # traverse elements of series  
 for(i in 1:length(y)) {  
   
 # calculate the next estimated y  
 y\_hat <- alpha \* y[i] + (1 - alpha) \* y\_hat  
   
 }  
   
 return(y\_hat)  
   
}

Now let’s see if the forecast of the next observation in the series returned by our () function matches the value returned by the () function.

optimal\_alpha <- fc$model$par[1]  
optimal\_l0 <- fc$model$par[2]  
  
print(optimal\_alpha)

#> alpha   
#> 0.2971488

print(optimal\_l0)

#> l   
#> 77260.06

fc\_myses <- myses(y = pigs, alpha = optimal\_alpha, level = optimal\_l0)  
  
fc\_ses <- ses(y = pigs, h = 4)  
  
print(fc\_myses)

#> alpha   
#> 98816.41

print(fc\_ses$mean[1])

#> [1] 98816.41

When comparing the forecast results for the next observation we can see that the two calculation methods yield very close results.

| Calculation Method | Forecast value |
| --- | --- |
| myses() | 98816.41 |
| ses() | 98816.41 |

## HA 7.3

### Modify your function from the previous exercise to return the sum of squared errors rather than the forecast of the next observation. Then use the optim() function to find the optimal values of α and ℓ0. Do you get the same values as the ses() function?

Based on the () function, create a function to return the Sum of Squared Errors (SSE).

mySSE <- function( pars = c(alpha, level), y ) {  
   
 # unpack pars array to get alpha and level values  
 alpha <- pars[1]  
 level <- pars[2]  
   
 # set initial estimated y with level  
 y\_hat <- level  
   
 err <- 0  
   
 SSE <- 0  
  
 # traverse elements of series  
 for(i in 1:length(y)) {  
   
 # calculate error by subtracting estimated y from actual y  
 err <- y[i] - y\_hat  
   
 # sum up and accumulate squared errors  
 SSE <- SSE + err ^ 2  
   
 # calculate the next estimated y  
 y\_hat <- alpha \* y[i] + (1 - alpha) \* y\_hat  
  
 }  
   
 return(SSE)  
   
}

Let’s use our () function to calculate the optimum values and compare them to the values from R’s () function.

result\_mySSE <- optim( par = c(0.5, pigs[1]), y = pigs, fn = mySSE )  
  
  
mySSE\_optimal\_alpha <- result\_mySSE$par[1]  
mySSE\_optimal\_l0 <- result\_mySSE$par[2]  
  
ses\_optimal\_alpha <- fc\_ses$model$par[1]  
ses\_optimal\_l0 <- fc\_ses$model$par[2]

When comparing the results of both methods, we see that the values are very close. The Optimal for the () function is slightly bigger than that of the () function. While the Optimal for the () function is slightly smaller than that of the () function.

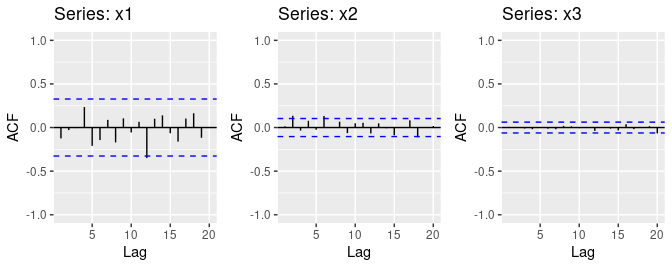
| Calculation Method | Optimal | Optimal |
| --- | --- | --- |
| () function | 0.2990081 | 76379.27 |
| R () function | 0.2971488 | 77260.06 |

# Week 3

## HA 8.1

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

url\_img <- "https://otexts.com/fpp2/fpp\_files/figure-html/wnacfplus-1.png"



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

All the figures do appear to indicate the data are white noise. This is because the autocorrelation is close to zero and mostly within the 95% bounds indicated by the blue dotted lines. In series x2, there are a few spikes that exceed the bounds. Because they are not extensive, do not excessively exceed the limit, and there is no obvious pattern they can still be considered all white noise.

### b. Why are the critical values at different distances from the mean of zero? Why are the autocorrelations different in each figure when they each refer to white noise?

The critical values and the 95% bounds are dependent on the length of the time series . As increases, the boundaries are narrowed. Likewise, from chapter 2.8 and 2.9, the calculations for autocorrelation coefficients () are dependent on the length of the time series, and as the time series grows the coefficients will decrease.

#### Series x1 Bounds

c(2, -2) / sqrt(36)

#> [1] 0.3333333 -0.3333333

#### Series x2 Bounds

c(2, -2) / sqrt(360)

#> [1] 0.1054093 -0.1054093

#### Series x3 Bounds

c(2, -2) / sqrt(1000)

#> [1] 0.06324555 -0.06324555

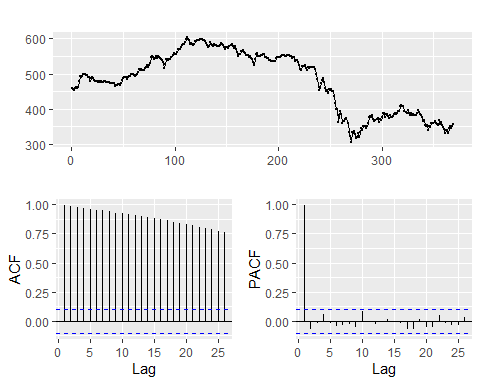
## HA 8.2

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

* Stationary data will not show changes for seasonality, levels, or changes in variance. In the trend plot for the raw data, there are clear changes in levels across the time series, clearly ruling out these data as stationary.
* The ACF plot confirms this by the spikes gradually reducing to zero. Stationary data will quickly drop to zero.
* The PACF plot, while appears that most of the values are within the bounds the first value is extremely high. In fact, it is identical to the first value in the ACF. This is because of how the PACF is calculated the first value is always the same as the first value of the ACF.

### Not Differenced Plots

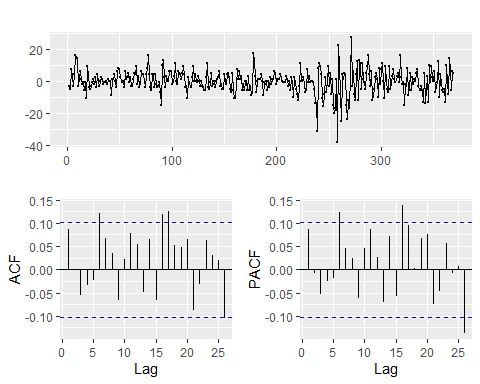
library(fma)  
  
ibmclose %>% ggtsdisplay(main="")



Comparatively, with these data now differenced. The concerns mentioned above are rectified for the most part. Additional analysis should be done to verify that the values in the ACF and PACF exeeding the bounds are not excessive.

### Differenced Plots

ibmclose %>% diff() %>% ggtsdisplay(main="")



## HA 8.6

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

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

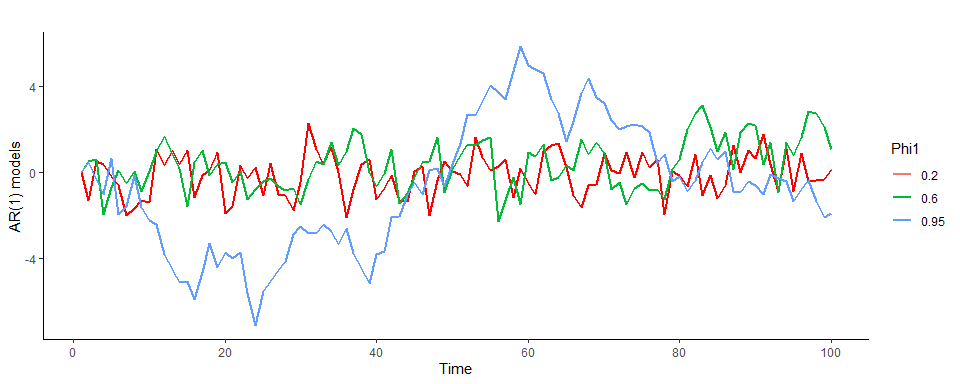
y <- ts(numeric(100))  
e <- rnorm(100)  
for(i in 2:100)  
 y[i] <- 0.6\*y[i-1] + e[i]  
  
head(y)

#> Time Series:  
#> Start = 1   
#> End = 6   
#> Frequency = 1   
#> [1] 0.0000000 0.4948018 0.6113392 -1.9770629 -0.6887980 0.1022994

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

It appears that as the increases to 1.0, the variation and “randomness” increases. Even with the green () and red () lines in the figure below have similar variation though random points. The blue line () has rather extreme spikes and level changes in both directions, particularly compared to the others.

set.seed(999)  
  
ar1 <- function(phi1){  
   
 y <- ts(numeric(100)) # generate 100 data points from an AR(1) model with input phi1.  
  
 e <- rnorm(100) # error 'e's have variation sigma^2 as 1.  
   
 for(i in 2:100){  
 y[i] <- phi1\*y[i-1] + e[i]  
 }  
 return(y)  
}  
  
  
# produce plots changing phi1 value.  
autoplot(ar1(0.20), series = "0.2") +  
 geom\_line(size = 1, colour = "red") +  
 autolayer(y, series = "0.6", size = 1) +  
 autolayer(ar1(0.95), size = 1, series = "0.95") +  
 ylab("AR(1) models") +  
 guides(colour = guide\_legend(title = "Phi1")) + theme\_classic()



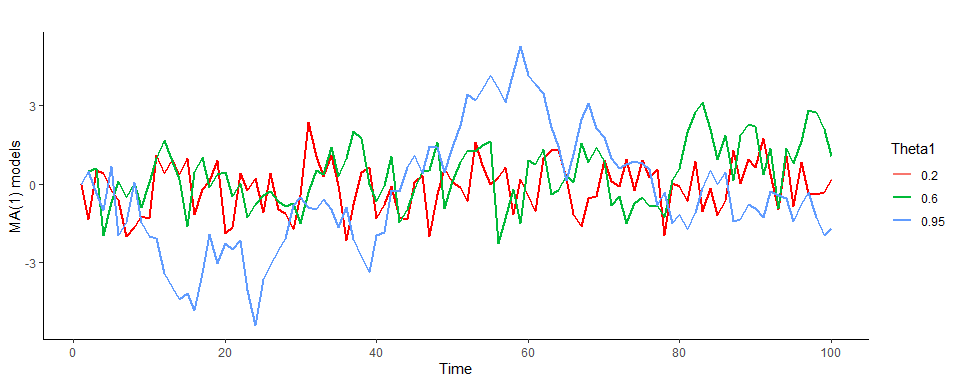
### c. Write your own code to generate data from an MA(1) model with and

ma1 <- function(theta1){  
 # generate 100 data points from an MA(1) model with input theta1.  
 y <- ts(numeric(100))  
 # error 'e's have variation sigma^2 as 1.  
 e <- rnorm(100)  
 for(i in 2:100){  
 y[i] <- theta1\*e[i-1] + e[i]  
 }  
 return(y)  
}

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

Similar to the changes in part b, the changes in the blue line are certainly more pronounced compared to the other two. It is interesting to not the difference between 0.2, 0.6, 0.95 are relatively close and yet the effects in the time plot are more extreme as it approaches 1.0.

set.seed(999)  
  
autoplot(ma1(0.2), series = "0.2") +  
 geom\_line(size = 1, colour = "red") +  
 autolayer(y, series = "0.6", size = 1) +  
 autolayer(ar1(0.9), size = 1, series = "0.95") +  
 ylab("MA(1) models") +  
 guides(colour = guide\_legend(title = "Theta1")) + theme\_classic()



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

y\_arima.1.0.1 <- ts(numeric(50))  
e <- rnorm(50)  
for(i in 2:50){  
 y\_arima.1.0.1[i] <- 0.6\*y\_arima.1.0.1[i-1] + 0.6\*e[i-1] + e[i]  
}

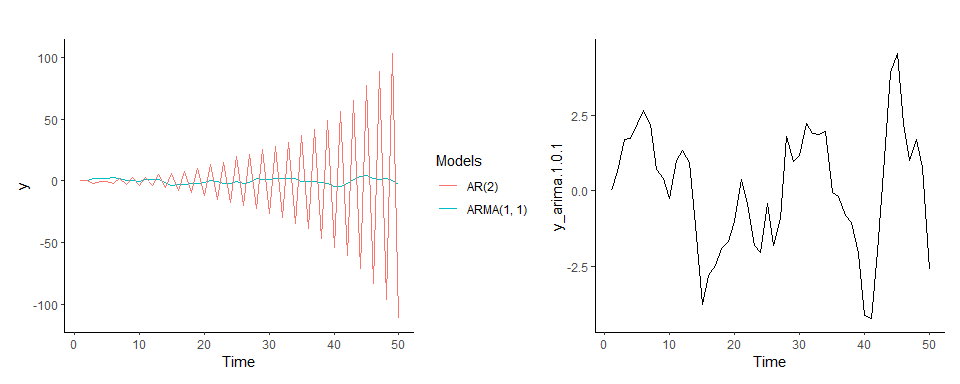
### f. Generate data from an AR(2) model with , and . (Note that these parameters will give a non-stationary series.)

y\_arima.2.0.0 <- ts(numeric(50))  
e <- rnorm(50)  
for(i in 3:50){  
 y\_arima.2.0.0[i] <- -0.8\*y\_arima.2.0.0[i-1] + 0.3\*y\_arima.2.0.0[i-2] + e[i]  
}

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

The AR(2) values appear to be exponentially increasing over time, while the ARMA(1,1) model appears to vary little.

autoplot(y\_arima.1.0.1, series = "ARMA(1, 1)") +  
 autolayer(y\_arima.2.0.0, series = "AR(2)") +  
 ylab("y") +  
 guides(colour = guide\_legend(title = "Models")) + theme\_classic() +   
autoplot(y\_arima.1.0.1) + theme\_classic()



## HA 8.8

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

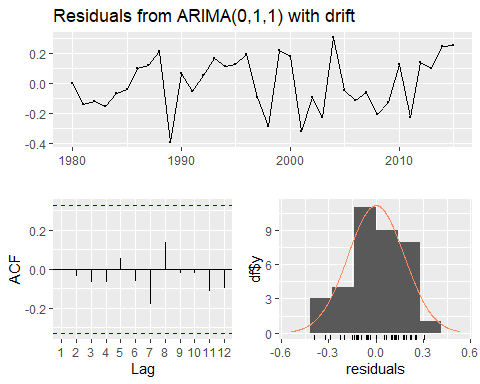
### a. Use auto.arima() to find an appropriate ARIMA model. What model was selected. Check that the residuals look like white noise. Plot forecasts for the next 10 periods.

* The auto.arima() function selected the **ARIMA(0,1,1) with drift** model.
* The residuals from the selected model look like white noise because appear to be normal and not obviously significant.
* The forecasts plot for the next 10 periods shows a steady upwards positive trend.

# Fit best ARIMA model to the austa time series  
auto\_arima\_austa <-forecast::auto.arima(austa)  
  
# Show the results summary of the fitted model   
summary(auto\_arima\_austa)

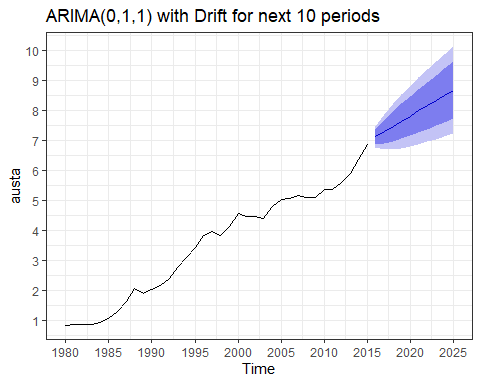
#> Series: austa   
#> ARIMA(0,1,1) with drift   
#>   
#> Coefficients:  
#> ma1 drift  
#> 0.3006 0.1735  
#> s.e. 0.1647 0.0390  
#>   
#> sigma^2 = 0.03376: log likelihood = 10.62  
#> AIC=-15.24 AICc=-14.46 BIC=-10.57  
#>   
#> Training set error measures:  
#> ME RMSE MAE MPE MAPE MASE  
#> Training set 0.0008313383 0.1759116 0.1520309 -1.069983 5.513269 0.7461559  
#> ACF1  
#> Training set -0.000571993

# Check that residuals from the time series model look like white noise  
# and produce a time plot of the residuals  
checkresiduals(auto\_arima\_austa, plot = TRUE)



#>   
#> 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

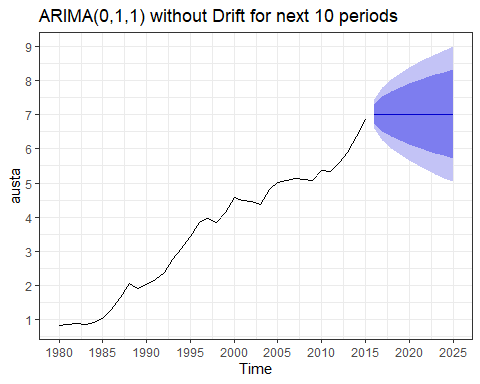
# Forecasts for the next 10 periods  
fc\_auto\_arima\_austa <- forecast(auto\_arima\_austa, h = 10)  
  
# Plot forecasts for the next 10 periods  
autoplot(fc\_auto\_arima\_austa) +  
 ggtitle("ARIMA(0,1,1) with Drift for next 10 periods") +  
 theme\_bw() +  
 scale\_x\_continuous(breaks = seq(from = 1980, to = 2030, by = 5)) +  
 scale\_y\_continuous(breaks = seq(from = 0, to = 11, by = 1))



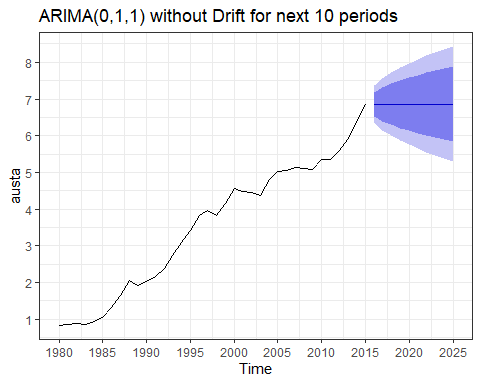
### b. Plot forecasts from an ARIMA(0,1,1) model with no drift and compare these to part a. Remove the MA term and plot again.

* Compared to the forecast plot in part (a), the forecasts plot from an ARIMA(0,1,1) model with no drift, no longer longer shows an upward trend. Instead, it shows a constant flat line at 7.0.
* Compared to the forecast plot in part (a), the forecasts plot from an ARIMA(0,1,0) model with no drift and with the MA term removed, also shows a constant flat line but shifted 0.1735 units below 7.0. Additionally, the confidence intervals (80 and 95) seem to have narrowed slightly.

# Fit an ARIMA(0,1,1) model to the austa time series  
# by default the include.drift parameter is set to FALSE  
arima\_011\_nd\_austa <- forecast::Arima(austa, order = c(0, 1, 1), include.drift = FALSE)  
  
# Forecasts for the next 10 periods  
fc\_arima\_011\_nd\_austa <- forecast(arima\_011\_nd\_austa, h = 10)  
  
# Plot forecasts for the next 10 periods  
autoplot(fc\_arima\_011\_nd\_austa) +  
 ggtitle("ARIMA(0,1,1) without Drift for next 10 periods") +  
 theme\_bw() +  
 scale\_x\_continuous(breaks = seq(from = 1980, to = 2030, by = 5)) +  
 scale\_y\_continuous(breaks = seq(from = 0, to = 11, by = 1))



# Fit an ARIMA(0,1,0) model (MA term removed) to the austa time series  
# by default the include.drift parameter is set to FALSE  
arima\_010\_nd\_austa <- forecast::Arima(austa, order = c(0, 1, 0), include.drift = FALSE)  
  
# Forecasts for the next 10 periods  
fc\_arima\_010\_nd\_austa <- forecast(arima\_010\_nd\_austa, h = 10)  
  
# Plot forecasts for the next 10 periods  
autoplot(fc\_arima\_010\_nd\_austa) +  
 ggtitle("ARIMA(0,1,1) without Drift for next 10 periods") +  
 theme\_bw() +  
 scale\_x\_continuous(breaks = seq(from = 1980, to = 2030, by = 5)) +  
 scale\_y\_continuous(breaks = seq(from = 0, to = 11, by = 1))

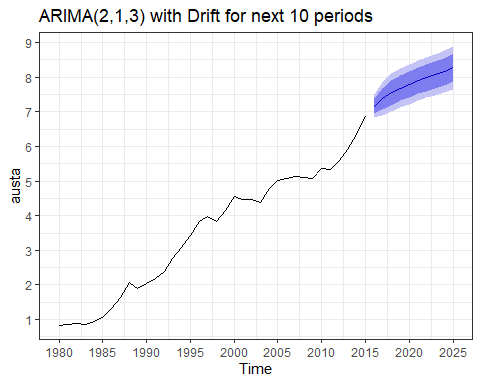


### c. Plot forecasts from an ARIMA(2,1,3) model with drift. Remove the constant and see what happens.

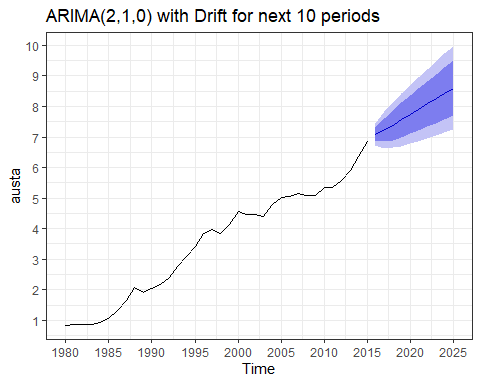
When comparing the ARIMA(2,1,3) with drift model vs the same model but with the constant removed, we see that

* The first forecast plot shows a slowly upward trend (almost concave down shape)
* The second forecast plot shows a faster upward trend (almost linear shape)
* The confidence intervals for the second one are wider.

# Fit an ARIMA(2,1,3) model with drift to the austa time series  
arima\_213\_wd\_austa <- forecast::Arima(austa, order = c(2,1,3), include.drift = TRUE)  
  
# Forecasts for the next 10 periods  
fc\_arima\_213\_wd\_austa <- forecast(arima\_213\_wd\_austa, h = 10)  
  
# Plot forecasts for the next 10 periods  
autoplot(fc\_arima\_213\_wd\_austa) +  
 ggtitle("ARIMA(2,1,3) with Drift for next 10 periods") +  
 theme\_bw() +  
 scale\_x\_continuous(breaks = seq(from = 1980, to = 2030, by = 5)) +  
 scale\_y\_continuous(breaks = seq(from = 0, to = 11, by = 1))



# Fit an ARIMA(2,1,0) model (constant removed) with drift to the austa time series  
arima\_210\_wd\_austa <- forecast::Arima(austa, order = c(2,1,0), include.drift = TRUE)  
  
# Forecasts for the next 10 periods  
fc\_arima\_210\_wd\_austa <- forecast(arima\_210\_wd\_austa, h = 10)  
  
# Plot forecasts for the next 10 periods  
autoplot(fc\_arima\_210\_wd\_austa) +  
 ggtitle("ARIMA(2,1,0) with Drift for next 10 periods") +  
 theme\_bw() +  
 scale\_x\_continuous(breaks = seq(from = 1980, to = 2030, by = 5)) +  
 scale\_y\_continuous(breaks = seq(from = 0, to = 11, by = 1))

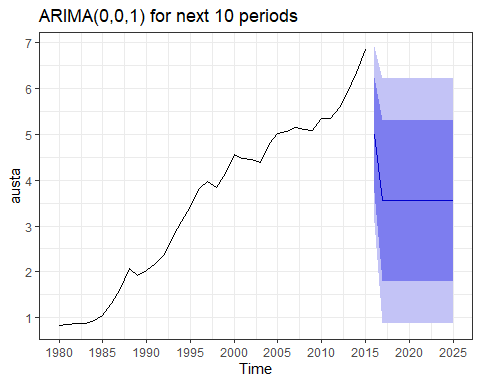


### d. Plot forecasts from an ARIMA(0,0,1) model with a constant. Remove the MA term and plot again.

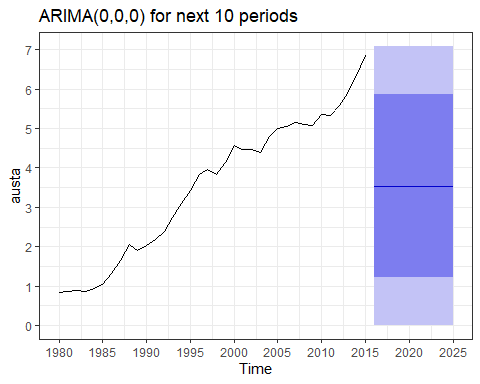
When comparing the forecast plots of ARIMA(0,0,1) model with that of the ARIMA(0,0,0) model, we see that

* The first one shows an immediate trend drop to 5.0 followed by asharp linear downwards trend for the first period, then followed by a flat line trend.
* The second one shows an immediate trend drop to 3.5 followed by a flat line trend.
* The confidence intervals on the second one are wider.

# Fit an ARIMA(0,0,1) model to the austa time series  
arima\_001\_nd\_austa <- forecast::Arima(austa, order = c(0,0,1), include.drift = FALSE)  
  
# Forecasts for the next 10 periods  
fc\_arima\_001\_nd\_austa <- forecast(arima\_001\_nd\_austa, h = 10)  
  
# Plot forecasts for the next 10 periods  
autoplot(fc\_arima\_001\_nd\_austa) +  
 ggtitle("ARIMA(0,0,1) for next 10 periods") +  
 theme\_bw() +  
 scale\_x\_continuous(breaks = seq(from = 1980, to = 2030, by = 5)) +  
 scale\_y\_continuous(breaks = seq(from = 0, to = 11, by = 1))



# Fit an ARIMA(0,0,0) model (MA term removed) to the austa time series  
arima\_000\_nd\_austa <- forecast::Arima(austa, order = c(0,0,0), include.drift = FALSE)  
  
# Forecasts for the next 10 periods  
fc\_arima\_000\_nd\_austa <- forecast(arima\_000\_nd\_austa, h = 10)  
  
# Plot forecasts for the next 10 periods  
autoplot(fc\_arima\_000\_nd\_austa) +  
 ggtitle("ARIMA(0,0,0) for next 10 periods") +  
 theme\_bw() +  
 scale\_x\_continuous(breaks = seq(from = 1980, to = 2030, by = 5)) +  
 scale\_y\_continuous(breaks = seq(from = 0, to = 11, by = 1))



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

# Fit an ARIMA(0,2,1) model to the austa time series  
arima\_021\_nd\_austa <- forecast::Arima(austa, order = c(0,2,1), include.drift = FALSE)  
  
# Forecasts for the next 10 periods  
fc\_arima\_021\_nd\_austa <- forecast(arima\_021\_nd\_austa, h = 10)  
  
# Plot forecasts for the next 10 periods  
autoplot(fc\_arima\_021\_nd\_austa) +  
 ggtitle("ARIMA(0,2,1) for next 10 periods") +  
 theme\_bw() +  
 scale\_x\_continuous(breaks = seq(from = 1980, to = 2030, by = 5)) +  
 scale\_y\_continuous(breaks = seq(from = 0, to = 11, by = 1))

