HW1\_Assignment

Lwin Shwe

2023-06-04

### KJ - Applied Predictive Modeling (Kuhn, Johnson)

### HA - Forecasting: Principles and Practice, 2nd Ed. (Hyndman, Athanasopoulos)

library(fpp2)

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

## ── Attaching packages ────────────────────────────────────────────── fpp2 2.5 ──

## ✔ ggplot2 3.4.2 ✔ fma 2.5   
## ✔ forecast 8.21 ✔ expsmooth 2.3

##

library(ggplot2)  
library(ggfortify)

## Registered S3 methods overwritten by 'ggfortify':  
## method from   
## autoplot.Arima forecast  
## autoplot.acf forecast  
## autoplot.ar forecast  
## autoplot.bats forecast  
## autoplot.decomposed.ts forecast  
## autoplot.ets forecast  
## autoplot.forecast forecast  
## autoplot.stl forecast  
## autoplot.ts forecast  
## fitted.ar forecast  
## fortify.ts forecast  
## residuals.ar forecast

library(forecast)  
library(readxl)  
library(dplyr)

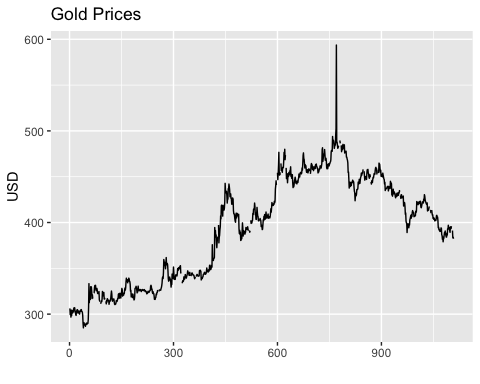
##   
## Attaching package: 'dplyr'

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

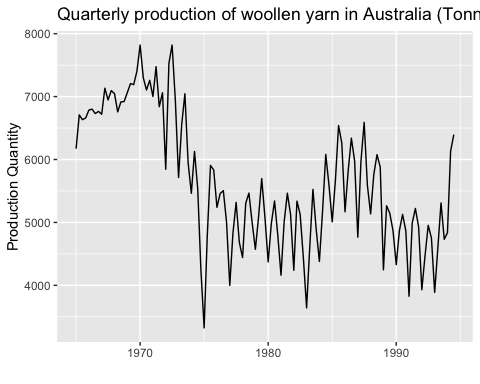
## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

### HA Exercise 2.1

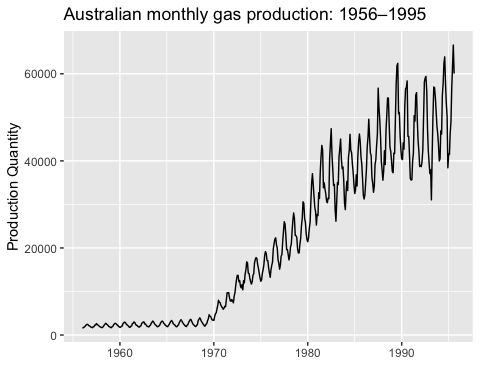
#Use the help function to explore what the series gold, woolyrnq and gas represent.  
help(gold)  
help(woolyrnq)  
help(gas)  
  
#Use autoplot() to plot each of these in separate plots.  
autoplot(gold) + labs(title = "Gold Prices", y = "USD")



autoplot(woolyrnq) + labs(title = "Quarterly production of woollen yarn in Australia (Tonnes)", y = "Production Quantity")



autoplot(gas) + labs(title = "Australian monthly gas production: 1956–1995 ", y = "Production Quantity")



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

## [1] 1

frequency(woolyrnq)

## [1] 4

frequency(gas)

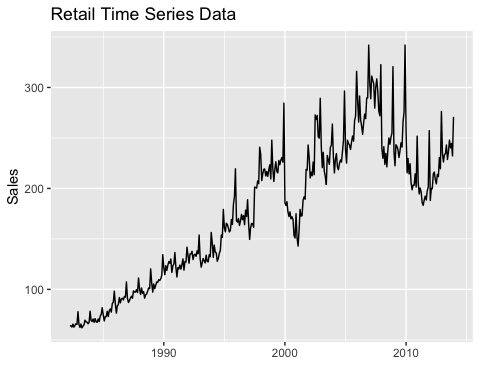
## [1] 12

#Use which.max() to spot the outlier in the gold series. Which observation was it?  
outlier\_observation <- gold[which.max(gold)]  
print(outlier\_observation)

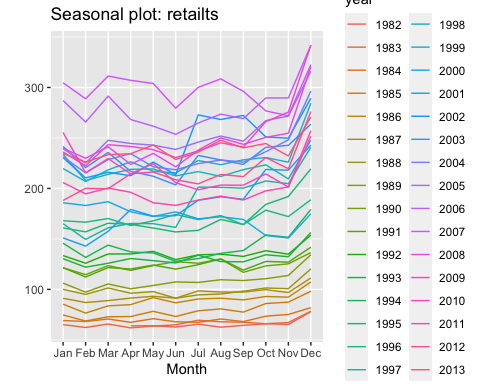
## [1] 593.7

### HA 2.3

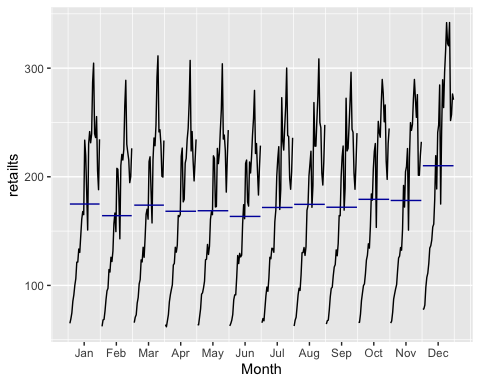
# (a). read the data into R  
retaildata <- read\_excel("~/DATA 624 assignment/Predictions/retail.xlsx", skip = 1)  
  
# (b). replace a column name  
myts <- ts(retaildata[,"A3349338X"], frequency = 12, start = c(1982,2))  
  
# (c). plot retail time series   
retailts <- ts(myts, frequency = 12, start = c(1982, 4))  
autoplot(retailts) + labs(title = "Retail Time Series Data", y = "Sales")



ggseasonplot(retailts) # Seasonality

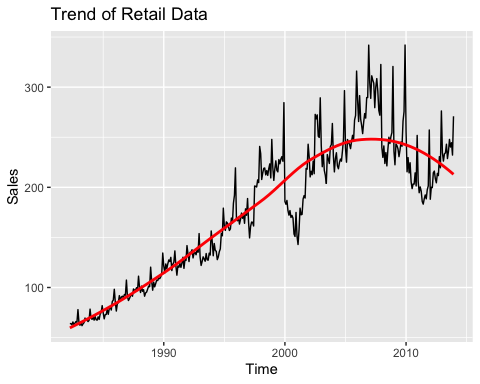


ggsubseriesplot(retailts) # Cyclicity



df <- data.frame(Time = time(retailts), Value = as.numeric(retailts)) # Trends  
ggplot(data = df, aes(x = Time, y = Value)) +  
 geom\_line() +  
 geom\_smooth(method = "loess", se = FALSE, color = "red") +  
 labs(title = "Trend of Retail Data", x = "Time", y = "Sales")

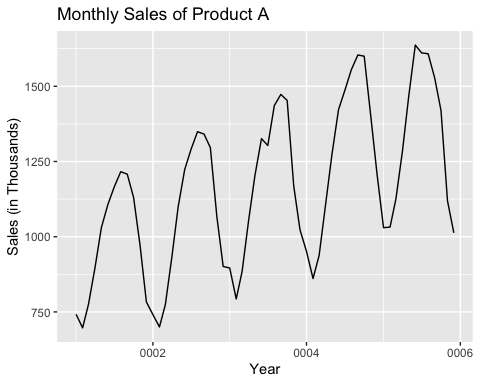
## Don't know how to automatically pick scale for object of type <ts>. Defaulting  
## to continuous.  
## `geom\_smooth()` using formula = 'y ~ x'



# I can learn the nature of time series observations with different graphic tools.

### Exercise HA 6.2

#Plot time series of sales of product A  
plastics\_ts <- ts(plastics, frequency = 12)  
autoplot(plastics\_ts) + ggtitle('Monthly Sales of Product A') + ylab('Sales (in Thousands)') + xlab('Year')

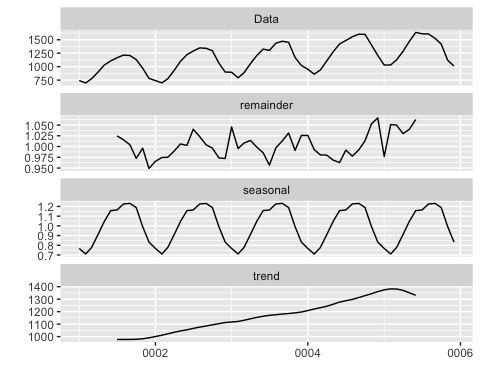


#Yes. I can identify seasonal fluctuations and/or a trend-cycle in a time series plot of the scale data.  
  
# Conduct decomposition  
plastics\_decomposed <- decompose(plastics\_ts, type = "multiplicative")  
  
# Print the decomposed components  
print(plastics\_decomposed)

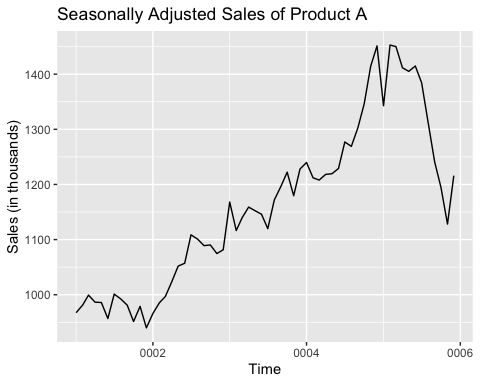
## $x  
## Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec  
## 1 742 697 776 898 1030 1107 1165 1216 1208 1131 971 783  
## 2 741 700 774 932 1099 1223 1290 1349 1341 1296 1066 901  
## 3 896 793 885 1055 1204 1326 1303 1436 1473 1453 1170 1023  
## 4 951 861 938 1109 1274 1422 1486 1555 1604 1600 1403 1209  
## 5 1030 1032 1126 1285 1468 1637 1611 1608 1528 1420 1119 1013  
##   
## $seasonal  
## Jan Feb Mar Apr May Jun Jul  
## 1 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317  
## 2 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317  
## 3 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317  
## 4 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317  
## 5 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317  
## Aug Sep Oct Nov Dec  
## 1 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834  
## 2 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834  
## 3 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834  
## 4 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834  
## 5 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834  
##   
## $trend  
## Jan Feb Mar Apr May Jun Jul  
## 1 NA NA NA NA NA NA 976.9583  
## 2 1000.4583 1011.2083 1022.2917 1034.7083 1045.5417 1054.4167 1065.7917  
## 3 1117.3750 1121.5417 1130.6667 1142.7083 1153.5833 1163.0000 1170.3750  
## 4 1208.7083 1221.2917 1231.7083 1243.2917 1259.1250 1276.5833 1287.6250  
## 5 1374.7917 1382.2083 1381.2500 1370.5833 1351.2500 1331.2500 NA  
## Aug Sep Oct Nov Dec  
## 1 977.0417 977.0833 978.4167 982.7083 990.4167  
## 2 1076.1250 1084.6250 1094.3750 1103.8750 1112.5417  
## 3 1175.5000 1180.5417 1185.0000 1190.1667 1197.0833  
## 4 1298.0417 1313.0000 1328.1667 1343.5833 1360.6250  
## 5 NA NA NA NA NA  
##   
## $random  
## Jan Feb Mar Apr May Jun Jul  
## 1 NA NA NA NA NA NA 1.0247887  
## 2 0.9656005 0.9745267 0.9750081 0.9894824 1.0061175 1.0024895 1.0401641  
## 3 1.0454117 0.9953920 1.0079773 1.0142083 0.9990100 0.9854384 0.9567618  
## 4 1.0257400 0.9924762 0.9807020 0.9798704 0.9684851 0.9627557 0.9917766  
## 5 0.9767392 1.0510964 1.0498039 1.0299302 1.0398787 1.0628077 NA  
## Aug Sep Oct Nov Dec  
## 1 1.0157335 1.0040354 0.9724119 0.9961368 0.9489762  
## 2 1.0230774 1.0040674 0.9962088 0.9735577 0.9721203  
## 3 0.9969907 1.0132932 1.0314752 0.9910657 1.0258002  
## 4 0.9776897 0.9920952 1.0133954 1.0527311 1.0665946  
## 5 NA NA NA NA NA  
##   
## $figure  
## [1] 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317  
## [8] 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834  
##   
## $type  
## [1] "multiplicative"  
##   
## attr(,"class")  
## [1] "decomposed.ts"

# Plot the decomposed time series  
autoplot(plastics\_decomposed)

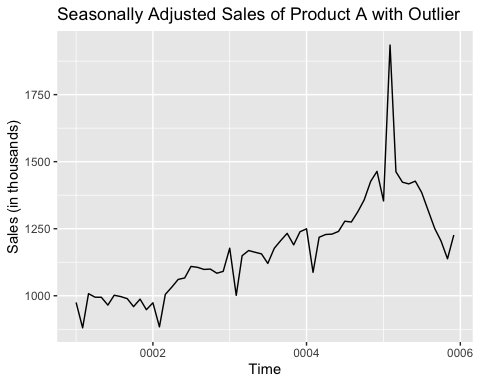
## Warning: Removed 24 rows containing missing values (`geom\_line()`).



#Yes, the result the trend over time consistently increases to the 5 years and the seasonal components show regular patterns that align with your identified seasonal fluctuations.  
  
# (d) Calculate the seasonally adjusted data  
plastics\_adjusted <- plastics\_decomposed$x / plastics\_decomposed$seasonal  
  
# Plot the seasonally adjusted data  
autoplot(plastics\_adjusted) +  
 ggtitle("Seasonally Adjusted Sales of Product A") +  
 xlab("Time") +  
 ylab("Sales (in thousands)")



# (e). Add 500 to one observation  
plastics\_ts\_outlier <- plastics\_ts  
plastics\_ts\_outlier[50] <- plastics\_ts\_outlier[50] + 500  
  
# Decompose the series with the outlier  
plastics\_decomposed\_outlier <- decompose(plastics\_ts\_outlier, type = "multiplicative")  
  
# Compute the seasonally adjusted data with the effect of outlier  
plastics\_adjusted\_outlier <- plastics\_decomposed\_outlier$x / plastics\_decomposed\_outlier$seasonal  
  
# Plot the seasonally adjusted data including outliers  
autoplot(plastics\_adjusted\_outlier) +  
 ggtitle("Seasonally Adjusted Sales of Product A with Outlier") +  
 xlab("Time") +  
 ylab("Sales (in thousands)")



# (f). The outlier might be different when it moves from the middle to the end of the time-series data.

### HA7.1

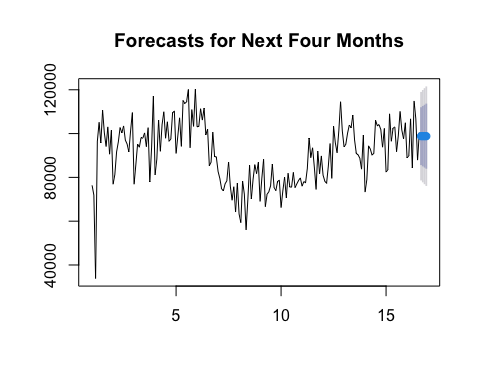
# (a).transform the pigs data set into time series object  
pigs\_ts <- ts(pigs, frequency = 12)  
fit <- ses(pigs\_ts)  
alpha <- fit$model$alpha  
l0 <- fit$model$states[1]  
forecast <- forecast(fit, h = 4)  
print(alpha)

## NULL

print(l0)

## [1] 77260.06

#Generate forecasts for the next four months  
plot(forecast, main = "Forecasts for Next Four Months")



# (b).Compute a 95 % prediction interval for the first forecast  
  
# find forecast-values and residuals  
forecast\_values <- forecast$mean   
residuals <- forecast$residuals   
  
# calculate standard deviation of residuals  
residual\_sd <- sd(residuals)   
  
# Construct the prediction intervals  
forecast\_intervals <- forecast\_values[1]  
lower\_interval <- forecast\_intervals - 1.96 \* residual\_sd  
upper\_interval <- forecast\_intervals + 1.96 \* residual\_sd  
  
# extract the 95% prediction interval for the first forecast  
prediction\_interval <- forecast$lower[1:2]

### 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()?

## Forecast of next observation by SES function:   
## 98816.4061115907

## Forecast of next observation by ses function: 98816.4061115907

### HA 7.3

# Define the sum\_squared\_errors function  
sum\_squared\_errors <- function(params, y) {  
 alpha <- params[1]  
 l0 <- params[2]  
 n <- length(y)  
 l <- numeric(n)  
 l[1] <- l0  
   
 # Compute the level estimate  
 for (i in 2:n) {  
 l[i] <- alpha \* y[i-1] + (1 - alpha) \* l[i-1]  
 }  
   
 # Compute the sum of squared errors  
 sum((y - l)^2)  
}  
  
pigs\_ts <- ts(pigs, frequency = 12)  
  
# Set the parameter bounds and initial values  
bounds <- c(0, 1) # Bounds for alpha and level  
initialvalue <- c(0.5, 5) # Initial values for alpha and level  
  
# Use the optim() function to find the optimal values of alpha and level  
optimum <- optim(initialvalue, sum\_squared\_errors, y = pigs\_ts, lower = bounds[2], upper =bounds[3])$par

## Warning in optim(initialvalue, sum\_squared\_errors, y = pigs\_ts, lower =  
## bounds[2], : bounds can only be used with method L-BFGS-B (or Brent)

optimum

## [1] 1 76378

### HA8.1

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

### Answers: Figure 8.31 for (Series: ×1)

# ACF for a white noise series of 36 numbers: Since the sample size is relatively small, it is expected that the ACF plot might have some significant random fluctuations with values decreases rapidly as the lag increases. This is because a small sample size may result in higher variability and less accurate estimation of the true autocorrelation.

### Answers: Figure 8.31 for (Series: ×2)

#ACF for a white noise series of 360 numbers: As the larger sample size, ACF plot shows reduced variability and more accurate estimation of true correlation. it is expected to decreases significant fluctuation values very quickly and indicates the absence of any siginificant correlation.

### Answers:Figure 8.31 for (Series: ×3)

#ACF for a white noise series of 1,000 numbers. With the largest sample sizes, the ACF plot depicts the highest accuracy estimation of the true autocorrelation.It is most likely to occur the significant values decrease more rapidly and a clearer pattern of randomness compared to the ACF for a white noices series of both 36 and 360 random numbers.

### HA8.1 (b).

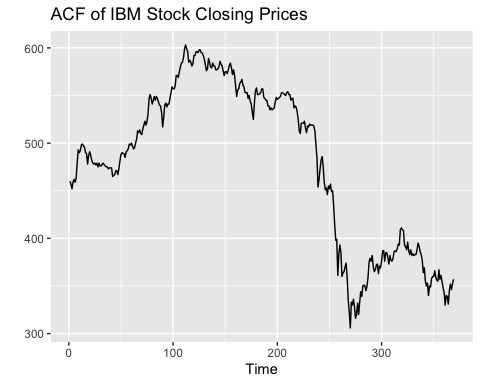
# Why are the critical values at different distances from the mean of zero? Why are the autocorrelations different in each figure when they each refer to white noise?

### Answer HA 8.1(b)

#With decreases in sample size, the critical values may be nearer to zero and have a wider range of acceptable values with lesser accuracy in forecasting the true population autocorrelation. #Conversely, as the sample size increases, the estimation of the true population autocorrelation becomes more precise. Consequently, the critical values may be farther from zero and have a narrower range of observed atutocorrelation values.

### HA 8.2

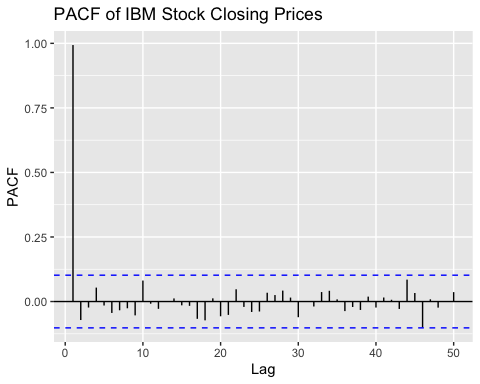
ibmclose <-ts(ibmclose)  
# plot the daily closing prices for IBM stock and the ACF and PACF  
autoplot(ibmclose) +  
 xlab("Time") +  
 ggtitle("ACF of IBM Stock Closing Prices")



ggAcf(ibmclose, lag.max = 50) +  
 xlab("Lag") +  
 ggtitle("ACF of IBM Stock Closing Prices")



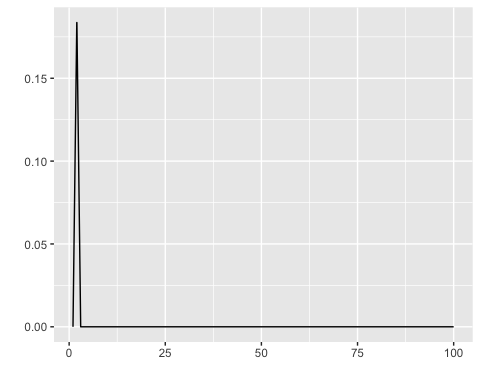
ggPacf(ibmclose, lag.max = 50) +  
 xlab("Lag") +  
 ggtitle("PACF of IBM Stock Closing Prices")

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

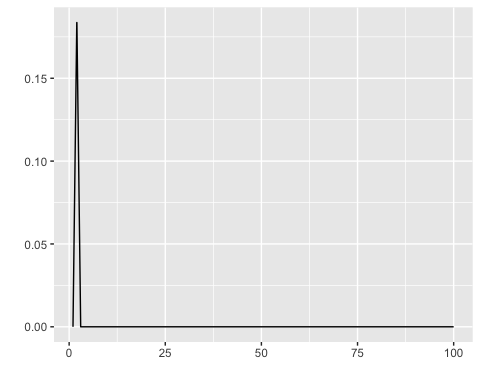
# (a).Use the following R code to generate data from an AR(1) model with ϕ1=0.6 and σ2 = 1   
y <- ts(numeric(100))  
e <- rnorm(100)  
ar1 = function(phi, y, e){  
for(i in 2:100)  
 y[i] <- 0.6\*y[i-1] + e[i]}  
  
  
#b.Produce a time plot for the series. How does the plot change as you change ϕ1  
?  
 autoplot(y)

## Help on topic 'autoplot' was found in the following packages:  
##   
## Package Library  
## forecast /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library  
## ggplot2 /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library  
##   
##   
## Using the first match ...

#When ϕ1 is closer to 1, the series will exhibit a stronger persistence, and the values will decay more slowly over time.  
#When ϕ1 is closer to 0, the series will show weaker persistence, and the values will decay more rapidly over time.  
#When ϕ1 is negative, the series may exhibit oscillatory behavior depending on the magnitude of ϕ1.  
  
# c.Write your own code to generate data from an MA(1) model with θ1 = 0.6 and σ2 = 1.  
ma2 = function(theta,seed.my){  
set.seed(seed.my)  
 y <- ts(numeric(100))  
 e <- rnorm(100)  
 for(i in 2:100){  
 y[i] <- theta\*y[i-1] + e[i]  
 return(autoplot(y))}  
}  
  
  
#d. Produce a time plot for the series. How does the plot change as you change θ1?  
ma2(.2,1)



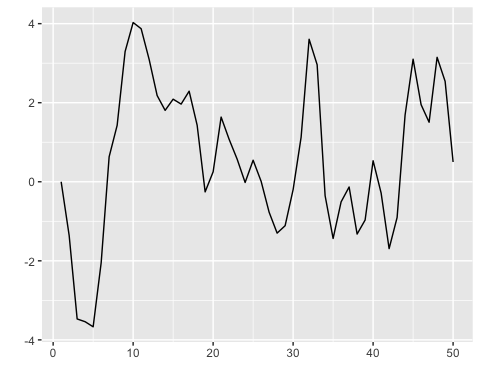
ma2(1,1)



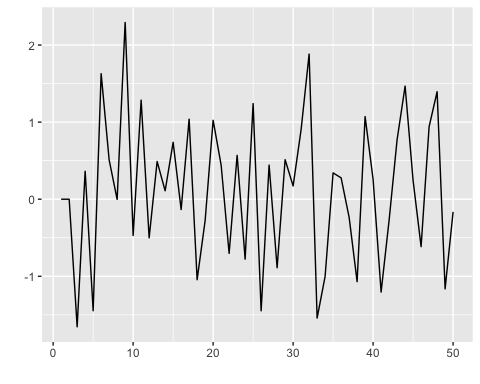
# e. Generate data from an ARMA(1,1) model with ϕ1=0.6,θ1=0.6,σ2=1  
  
ma.ar = ts(numeric(50))  
y2 = rnorm(50)  
for(i in 2:50){  
 ma.ar[i] = 0.6\*ma.ar[i-1] + 0.6\*e[i-1] + e[i]  
}  
  
#f. Generate data from an AR(2) model with ϕ1=−0.8,ϕ2=0.3,σ2=1 (Note that these parameters will give a non-stationary series.)  
ma.ar2 = ts(numeric(50))  
y2 = rnorm(50)  
for(i in 3:50){  
 ma.ar2[i] = -0.8\*ma.ar2[i-1] + 0.3\*ma.ar2[i-1] + e[i]  
}

# HA 8.6(g) Graph the latter two series and compare them.

autoplot(ma.ar)



autoplot(ma.ar2)

 ### HA 8.8

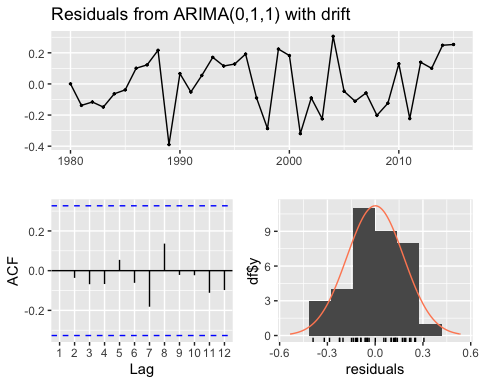
###Consider austa, the total international visitors to Australia (in millions) for the period 1980-2015.

#(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.

model <- auto.arima(austa)  
model

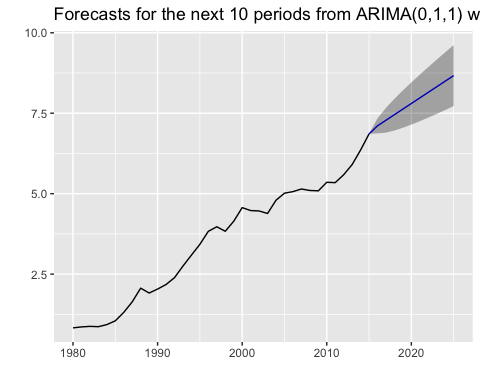
## Series: austa   
## ARIMA(0,1,1) with drift   
##   
## Coefficients:  
## ma1 drift  
## 0.3006 0.1735  
## s.e. 0.1647 0.0390  
##   
## sigma^2 = 0.03376: log likelihood = 10.62  
## AIC=-15.24 AICc=-14.46 BIC=-10.57

checkresiduals(model)

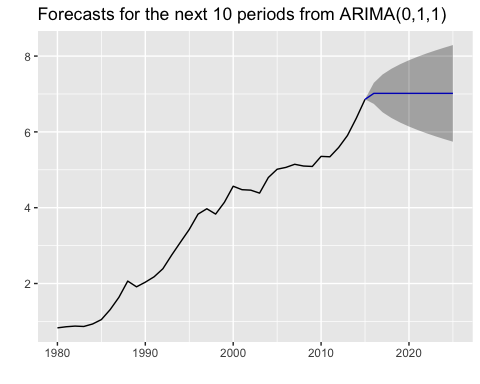


##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(0,1,1) with drift  
## Q\* = 2.297, df = 6, p-value = 0.8905  
##   
## Model df: 1. Total lags used: 7

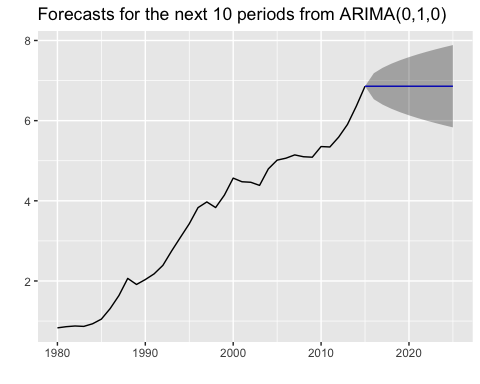
forecast <- forecast(model, h = 10)  
autoplot(forecast, main = "Forecasts for the next 10 periods from ARIMA(0,1,1) with drift")

 ### 8.8b.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.

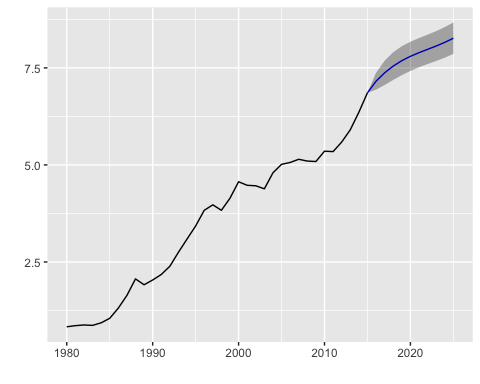
austa011 = forecast(arima(austa, c(0,1,1)), h=10)  
autoplot(austa011, main = "Forecasts for the next 10 periods from ARIMA(0,1,1)")



austa010 = forecast(arima(austa, c(0,1,0)), h=10)  
autoplot(austa010, main = "Forecasts for the next 10 periods from ARIMA(0,1,0)")

 ### HA 8.8(c) # Plot forecasts from an ARIMA(2,1,3) model with drift. Remove the constant and see what happens.

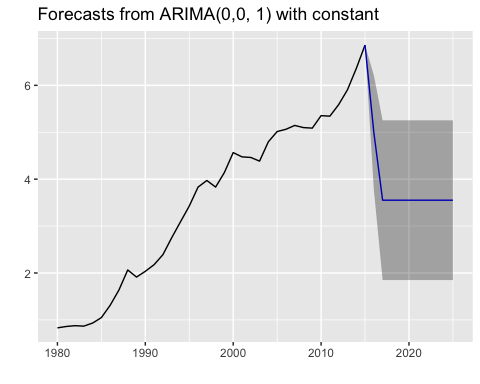
austa213 = forecast(Arima(austa, c(2,1,3), include.drift = TRUE), h=10)  
autoplot(austa213)

 ### HA8.8 (d) #Plot forecasts from an ARIMA(0,0,1) model with a constant. Remove the MA term and plot again.

austa001 = forecast(arima(austa, c(0,0,1)), include.constant=TRUE, h=10)

## Warning in forecast.Arima(arima(austa, c(0, 0, 1)), include.constant = TRUE, :  
## The non-existent include.constant arguments will be ignored.

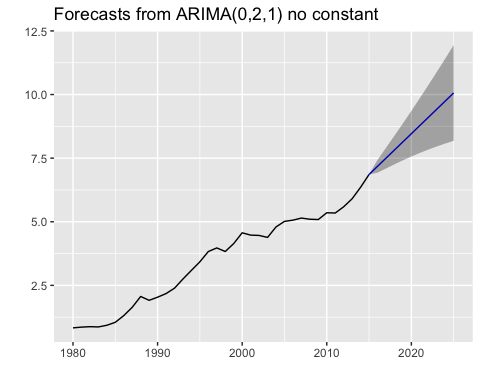
autoplot(austa001, main = "Forecasts from ARIMA(0,0, 1) with constant")

 ### HA 8.8(e) #Plot forecasts from an ARIMA(0,2,1) model with no constant.

austa021 = forecast(arima(austa, c(0,2,1)), include.constant=FALSE, h=10)

## Warning in forecast.Arima(arima(austa, c(0, 2, 1)), include.constant = FALSE, :  
## The non-existent include.constant arguments will be ignored.

autoplot(austa021, main = "Forecasts from ARIMA(0,2,1) no constant")

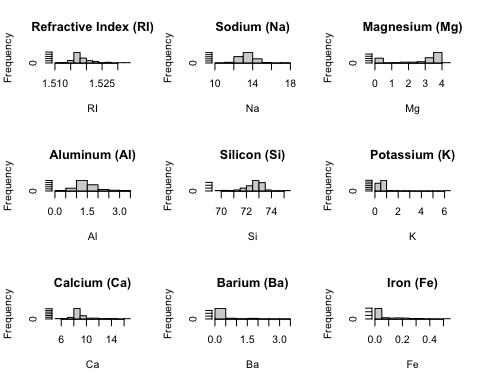
 ### KJ 3.1 # The UC Irvine Machine Learning Repository6 contains a data set related to glass identification. The data consist of 214 glass samples labeled as one of seven class categories. There are nine predictors, including the refractive index and percentages of eight elements: Na, Mg, Al, Si, K, Ca, Ba, and Fe.

# Load the Glass dataset from mlbench package  
library(mlbench)  
library(e1071)  
library(moments)

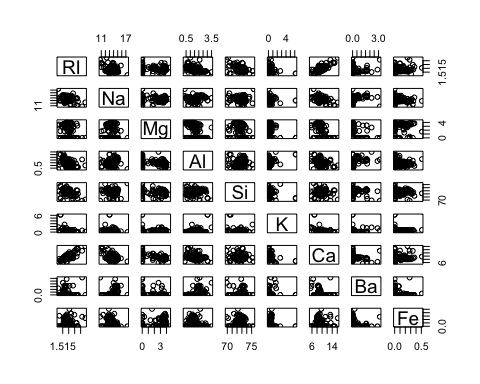
##   
## Attaching package: 'moments'

## The following objects are masked from 'package:e1071':  
##   
## kurtosis, moment, skewness

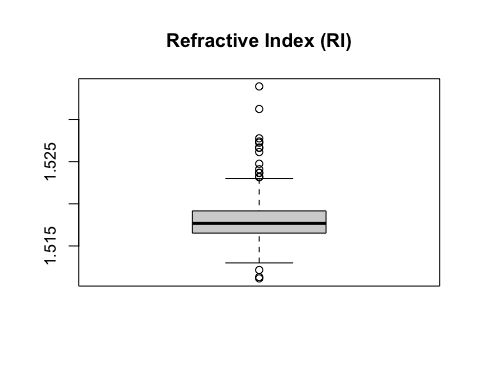
data(Glass)  
  
# Draw histograms for predictor variables  
par(mfrow = c(3, 3)) # Set the layout to 3 rows and 3 columns  
  
hist(Glass$RI, main = "Refractive Index (RI)", xlab = "RI")  
hist(Glass$Na, main = "Sodium (Na)", xlab = "Na")  
hist(Glass$Mg, main = "Magnesium (Mg)", xlab = "Mg")  
hist(Glass$Al, main = "Aluminum (Al)", xlab = "Al")  
hist(Glass$Si, main = "Silicon (Si)", xlab = "Si")  
hist(Glass$K, main = "Potassium (K)", xlab = "K")  
hist(Glass$Ca, main = "Calcium (Ca)", xlab = "Ca")  
hist(Glass$Ba, main = "Barium (Ba)", xlab = "Ba")  
hist(Glass$Fe, main = "Iron (Fe)", xlab = "Fe")



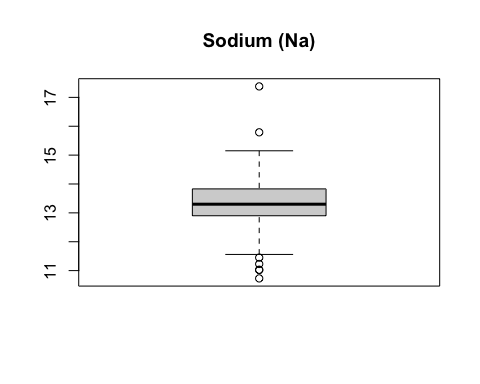
# Explore relationships between predictors using scatter plots  
pairs(Glass[, c("RI", "Na", "Mg", "Al", "Si", "K", "Ca", "Ba", "Fe")])



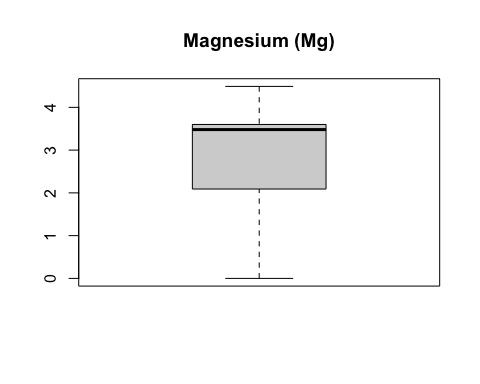
# Create boxplots for each predictor variable  
boxplot(Glass$RI, main = "Refractive Index (RI)")



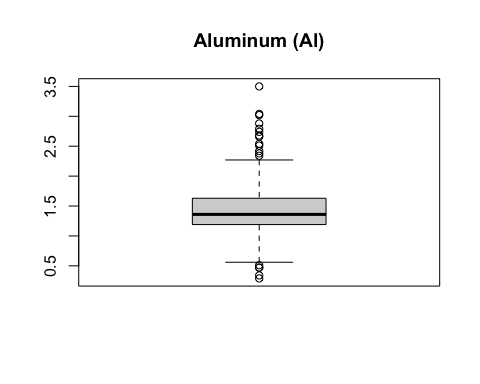
boxplot(Glass$Na, main = "Sodium (Na)")



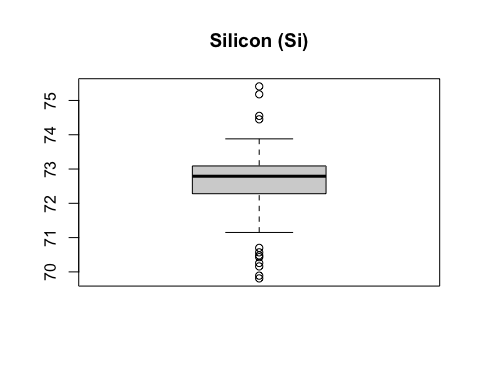
boxplot(Glass$Mg, main = "Magnesium (Mg)")



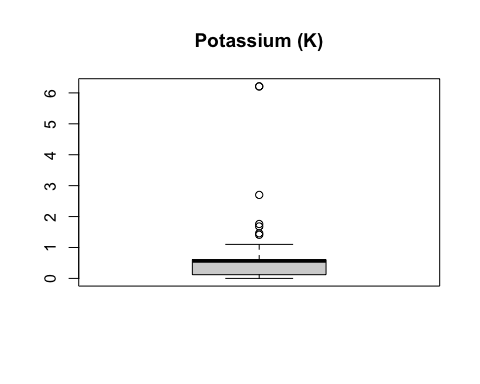
boxplot(Glass$Al, main = "Aluminum (Al)")



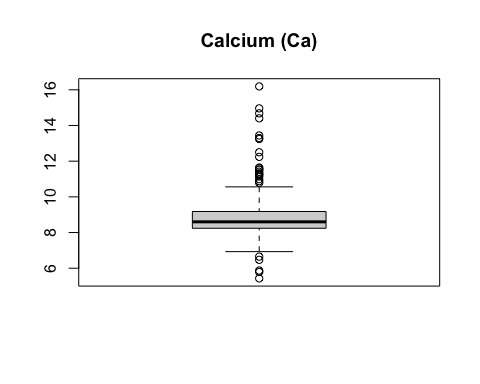
boxplot(Glass$Si, main = "Silicon (Si)")



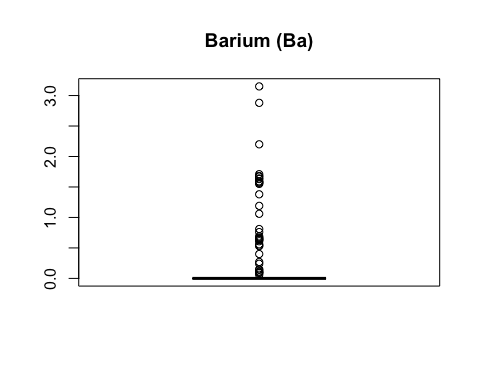
boxplot(Glass$K, main = "Potassium (K)")



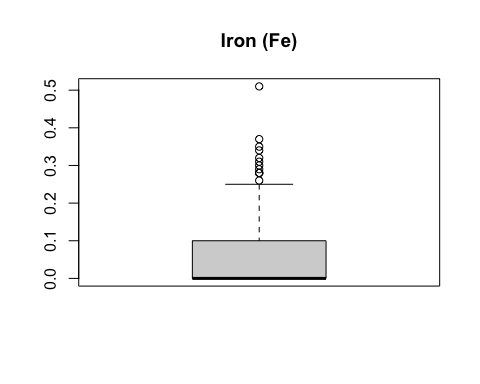
boxplot(Glass$Ca, main = "Calcium (Ca)")



boxplot(Glass$Ba, main = "Barium (Ba)")



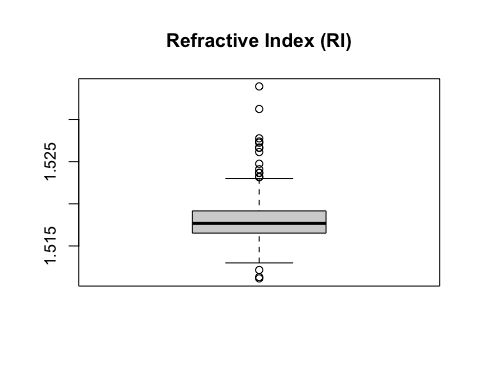
boxplot(Glass$Fe, main = "Iron (Fe)")



# Calculate skewness for each predictor variable  
skewness <- sapply(Glass[, -10], skewness)  
print(skewness)

## RI Na Mg Al Si K Ca   
## 1.6140150 0.4509917 -1.1444648 0.9009179 -0.7253173 6.5056358 2.0326774   
## Ba Fe   
## 3.3924309 1.7420068

# Create a boxplot showing outliers  
boxplot(Glass$RI, main = "Refractive Index (RI)", outline = TRUE)



# Identify outliers  
outliers <- boxplot(Glass$RI, plot = FALSE)$out  
  
# Print the outliers  
print(outliers)

## [1] 1.52667 1.52320 1.51215 1.52725 1.52410 1.52475 1.53125 1.53393 1.52664  
## [10] 1.52739 1.52777 1.52614 1.52369 1.51115 1.51131 1.52315 1.52365

### KJ 3.1(b)

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

The data distribution is left-skewed for negative skewness. Positive skewness indicates the data distribution is right-skewed. The skewness value is near zero, it means the median and the mean have the same value, hence, it’s a normal distribution.

###KJ 3.1(C) #Are there any relevant transformations of one or more predictors that might improve the classification model?

library(caret)

## Loading required package: lattice

# Transform data  
transdata <- preProcess(Glass, method = c("BoxCox", "center", "scale", "pca"))  
transdata

## Created from 214 samples and 10 variables  
##   
## Pre-processing:  
## - Box-Cox transformation (5)  
## - centered (9)  
## - ignored (1)  
## - principal component signal extraction (9)  
## - scaled (9)  
##   
## Lambda estimates for Box-Cox transformation:  
## -2, -0.1, 0.5, 2, -1.1  
## PCA needed 7 components to capture 95 percent of the variance

transformed <- predict(transdata, Glass)

### KJ3.2

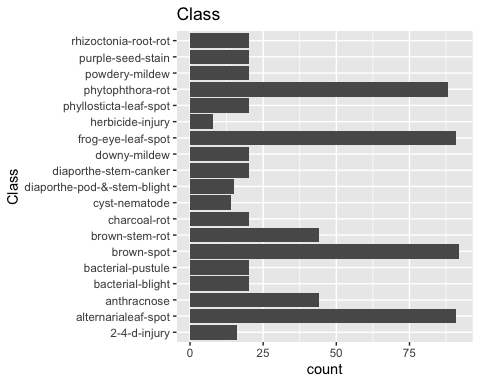
#The soybean data can also be found at the UC Irvine Machine Learning Repository. Data were collected to predict disease in 683 soybeans. The 35 predictors are mostly categorical and include information on the environmen-tal conditions (e.g., temperature, precipitation) and plant conditions (e.g., left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

###KJ 3.2(a) #(a) Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?

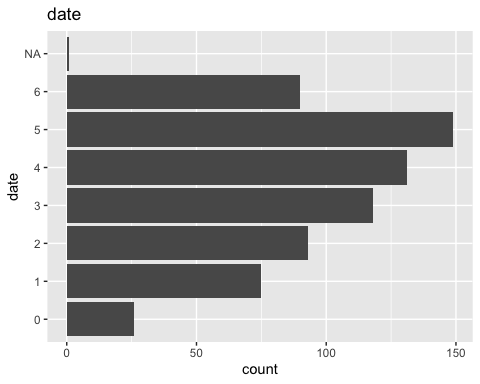
data("Soybean")  
  
# Identify the categorical predictors of interest  
columns <- colnames(Soybean)  
  
# Investigate frequency distributions and create bar plots  
lapply(columns,  
 function(col) {  
 ggplot(Soybean,   
 aes\_string(col)) + geom\_bar() + coord\_flip() + ggtitle(col)})

## Warning: `aes\_string()` was deprecated in ggplot2 3.0.0.  
## ℹ Please use tidy evaluation idioms with `aes()`.  
## ℹ See also `vignette("ggplot2-in-packages")` for more information.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

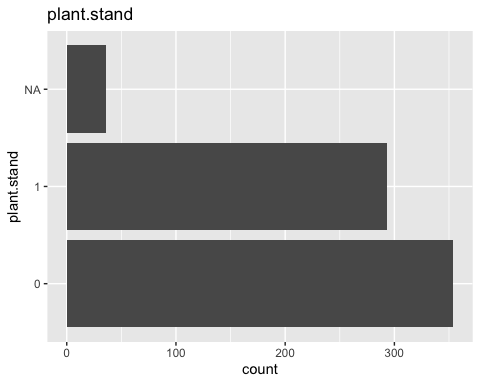
## [[1]]



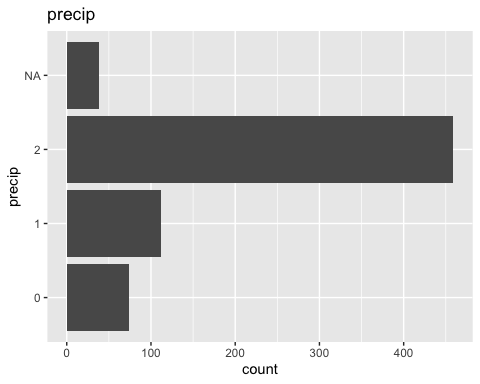
##   
## [[2]]



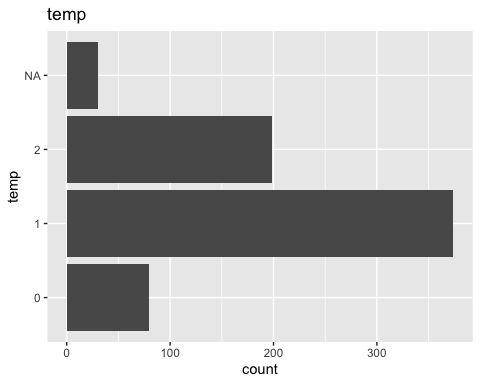
##   
## [[3]]



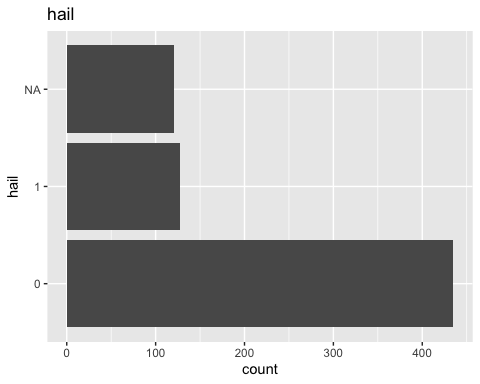
##   
## [[4]]



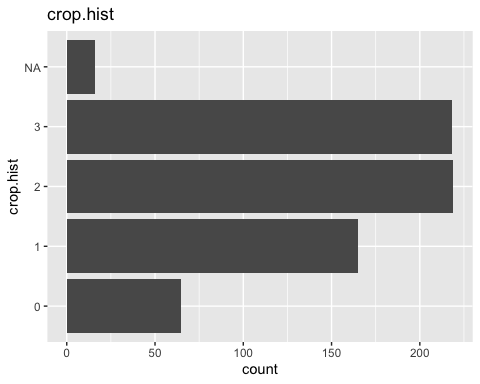
##   
## [[5]]



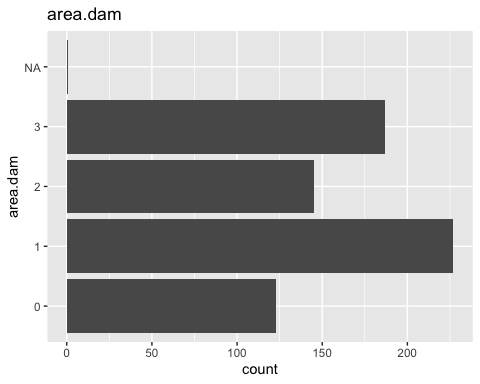
##   
## [[6]]



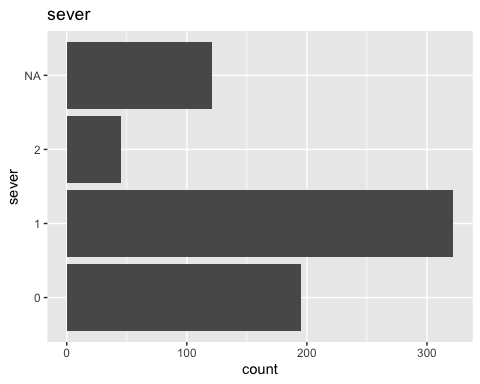
##   
## [[7]]



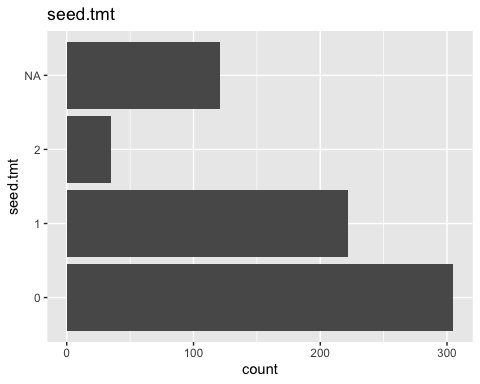
##   
## [[8]]



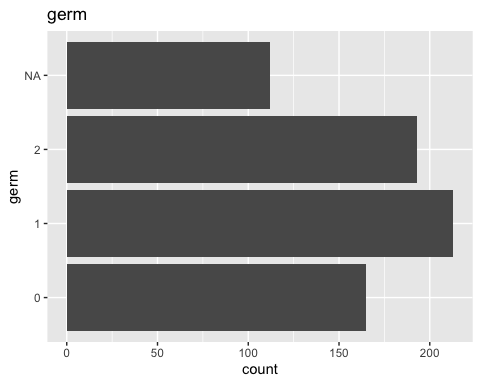
##   
## [[9]]



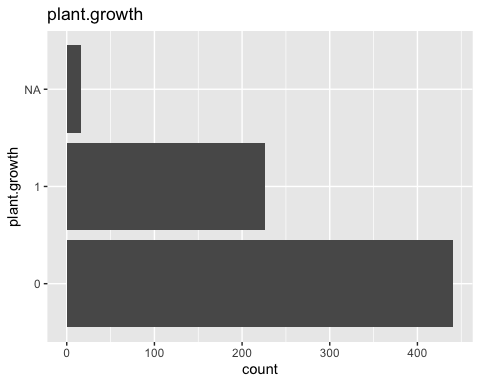
##   
## [[10]]



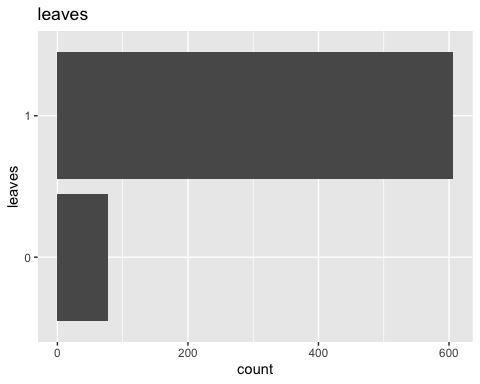
##   
## [[11]]



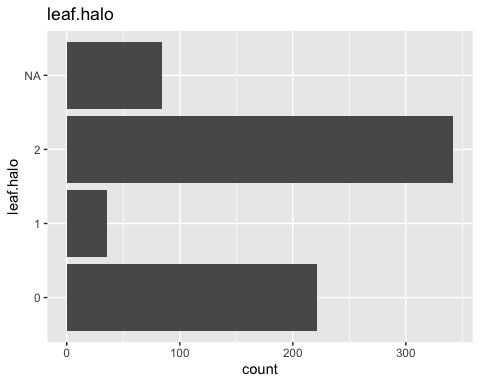
##   
## [[12]]



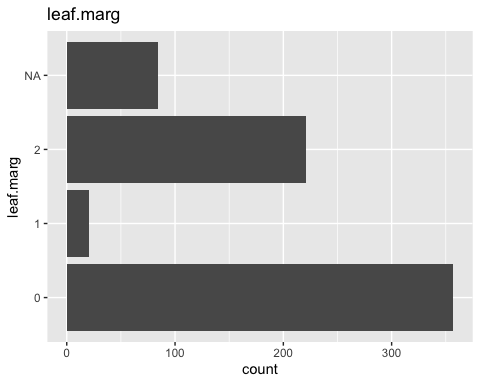
##   
## [[13]]



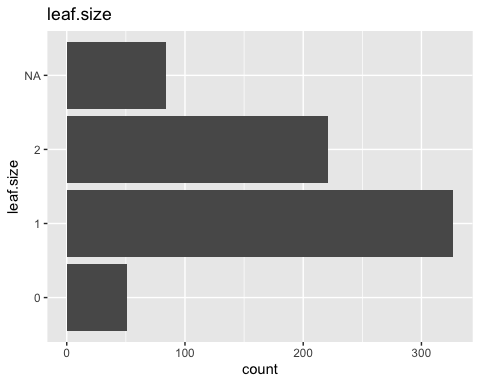
##   
## [[14]]



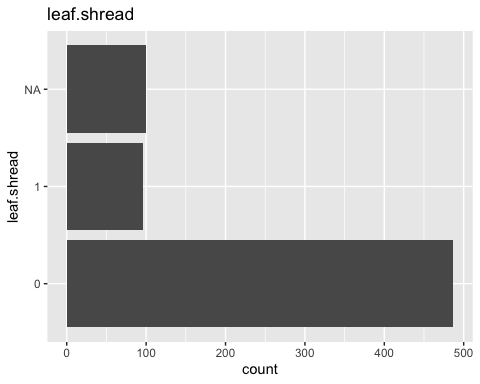
##   
## [[15]]



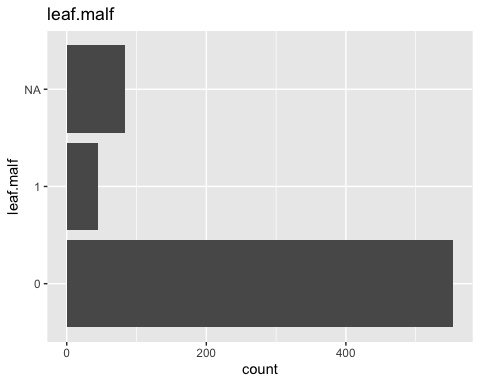
##   
## [[16]]



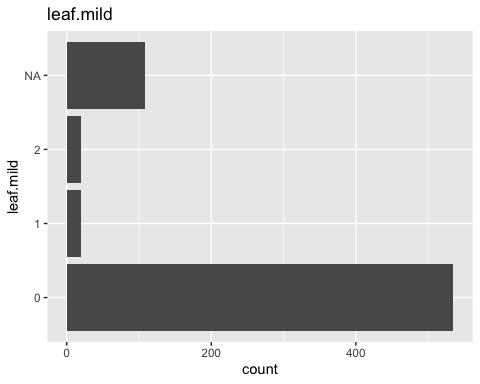
##   
## [[17]]



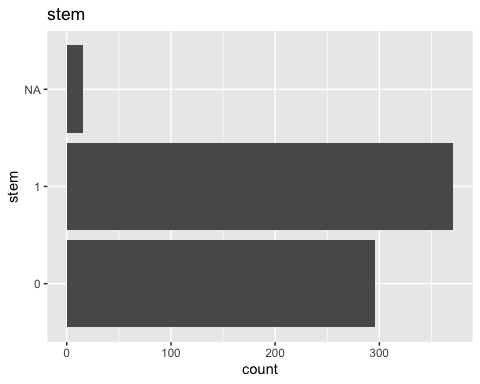
##   
## [[18]]



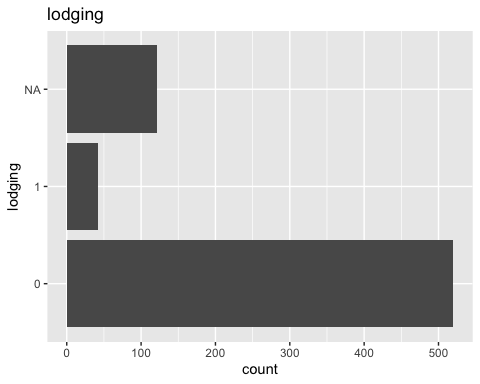
##   
## [[19]]



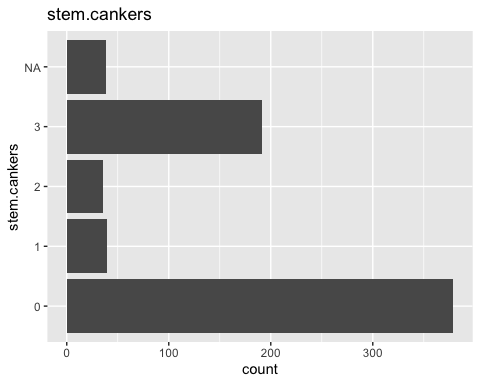
##   
## [[20]]



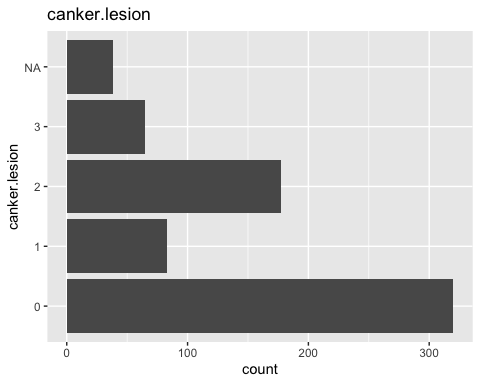
##   
## [[21]]



##   
## [[22]]



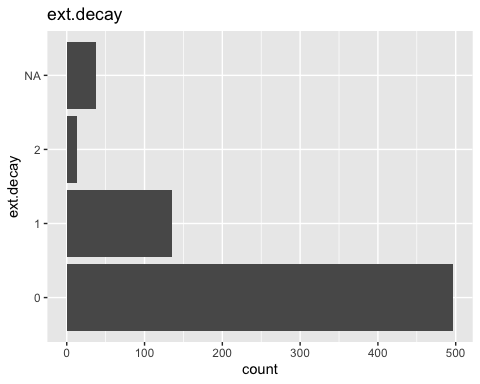
##   
## [[23]]



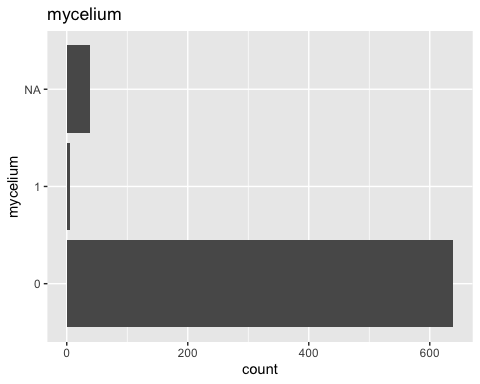
##   
## [[24]]



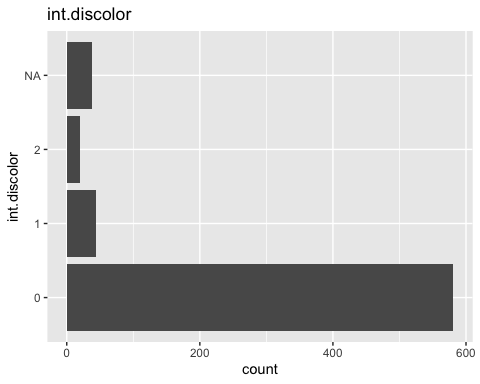
##   
## [[25]]



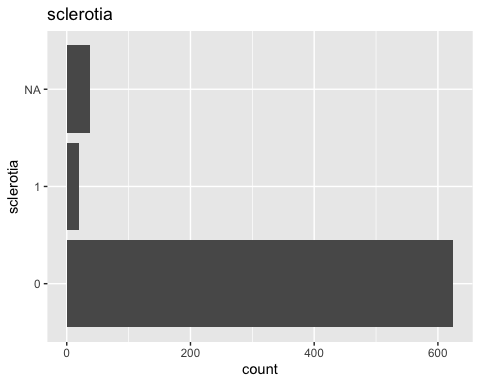
##   
## [[26]]



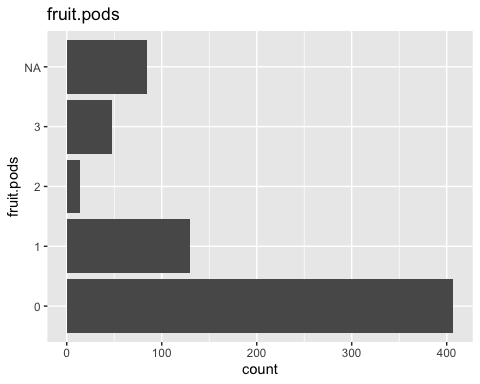
##   
## [[27]]



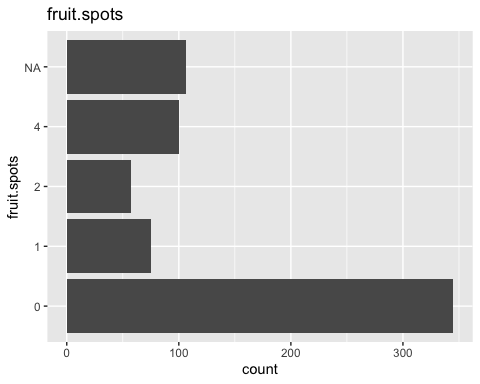
##   
## [[28]]



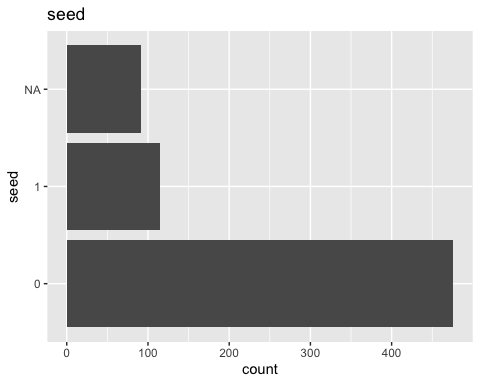
##   
## [[29]]



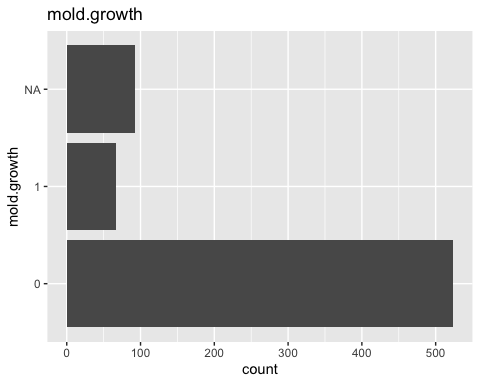
##   
## [[30]]



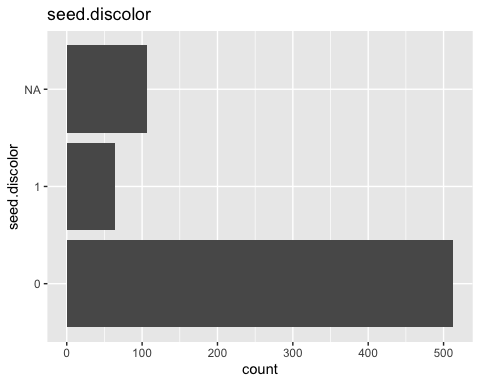
##   
## [[31]]



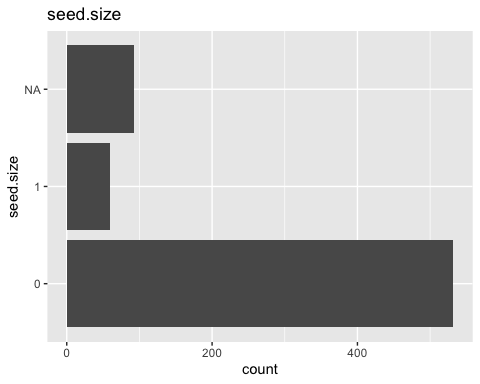
##   
## [[32]]



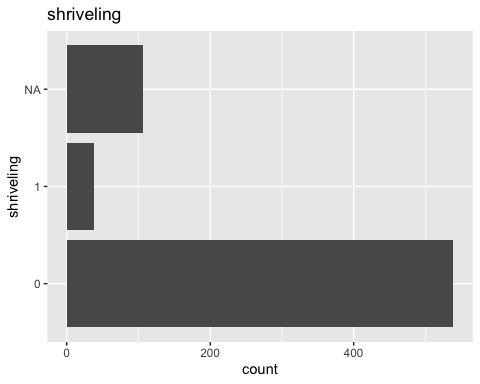
##   
## [[33]]



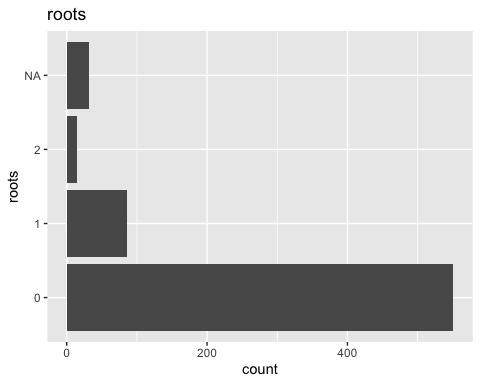
##   
## [[34]]



##   
## [[35]]



##   
## [[36]]



### KJ 3.2(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?

# Calculate proportion of missing values for each predictor  
missing\_values <- colMeans(is.na(Soybean))  
missing\_values

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

# Calculate missing values and the proportion of missing values  
Soybean %>%  
 group\_by(Class) %>%  
 summarize(Total = n(), Missing = sum(is.na(.)), Proportion = Missing / Total)

## # A tibble: 19 × 4  
## Class Total Missing Proportion  
## <fct> <int> <int> <dbl>  
## 1 2-4-d-injury 16 2337 146.   
## 2 alternarialeaf-spot 91 2337 25.7  
## 3 anthracnose 44 2337 53.1  
## 4 bacterial-blight 20 2337 117.   
## 5 bacterial-pustule 20 2337 117.   
## 6 brown-spot 92 2337 25.4  
## 7 brown-stem-rot 44 2337 53.1  
## 8 charcoal-rot 20 2337 117.   
## 9 cyst-nematode 14 2337 167.   
## 10 diaporthe-pod-&-stem-blight 15 2337 156.   
## 11 diaporthe-stem-canker 20 2337 117.   
## 12 downy-mildew 20 2337 117.   
## 13 frog-eye-leaf-spot 91 2337 25.7  
## 14 herbicide-injury 8 2337 292.   
## 15 phyllosticta-leaf-spot 20 2337 117.   
## 16 phytophthora-rot 88 2337 26.6  
## 17 powdery-mildew 20 2337 117.   
## 18 purple-seed-stain 20 2337 117.   
## 19 rhizoctonia-root-rot 20 2337 117.

### KJ 3.2 (c)

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

#Answer:Those 18% of missing data can be completely filled with KNN inputation methods that missing values are imputed by finding the nearest neighbors (observations with similar values) based on the available data.