**Data 624 Homework Batch 1**

**CUNY SPS MSDS**

**Dr. Scott Burk**

**Summary 2022**

**Group 4:**

Melvin Matanos Claire Meyer Chinedu Onyeka

Euclid Zhang Jie Zou

**6/1/2022**

**Table of Contents**

HA 2.1 2

HA 2.3 5

HA 6.2 9

KJ 3.1 14

KJ 3.2 18

HA 7.1 24

HA 7.2 26

HA 7.3 27

HA 8.1 29

HA 8.2 31

HA 8.6 34

HA 8.8 38

# HA 2.1

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

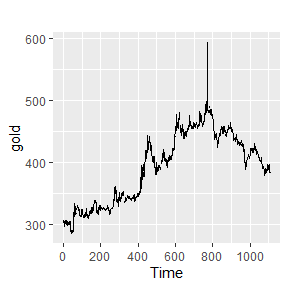
# ??forecast::gold  
# ??forecast::woolyrnq  
# ??forecast::gas

Using the help function in R, we retrieve the following information of the three time series:

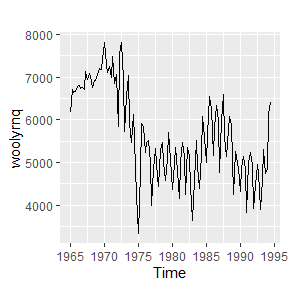
* gold: Daily morning gold prices in US dollars. 1 January 1985 – 31 March 1989.
* woolyrnq: Quarterly production of woollen yarn in Australia: tonnes. Mar 1965 – Sep 1994.
* gas: Australian monthly gas production: 1956–1995.

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

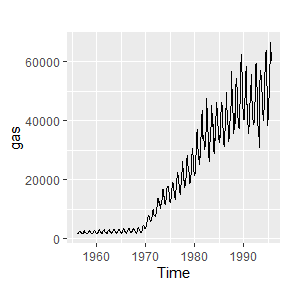
autoplot(gold)



autoplot(woolyrnq)



autoplot(gas)



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

frequency(gold)

## [1] 1

The frequency of the gold series is 1. This may be an error since the data is daily gold price. It should be more appropriate to use frequency 365.25 for an annual cycle.

frequency(woolyrnq)

## [1] 4

The frequency of the woolyrnq series is 4, which is correct as a quarterly time series.

frequency(gas)

## [1] 12

The frequency of the gas series is 12, which is correct as a monthly time series.

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

which.max(gold)

## [1] 770

The 770th observation is an outlier, the gold price is 462.75

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

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

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

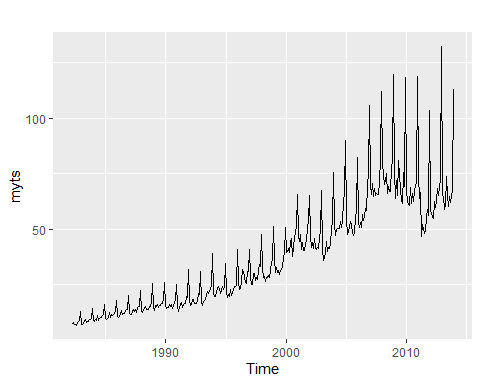
Here we pick the column at the middle of the data frame

myts <- ts(retaildata[,floor(ncol(retaildata)/2)], frequency=12, start=c(1982,4))

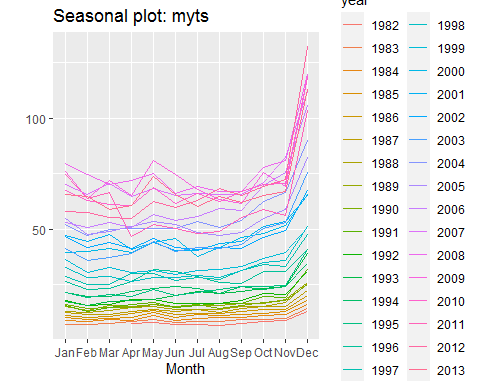
## c. Explore your chosen retail time series using the following functions:

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

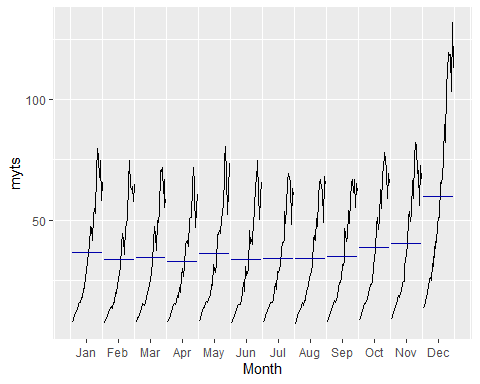
autoplot(myts)



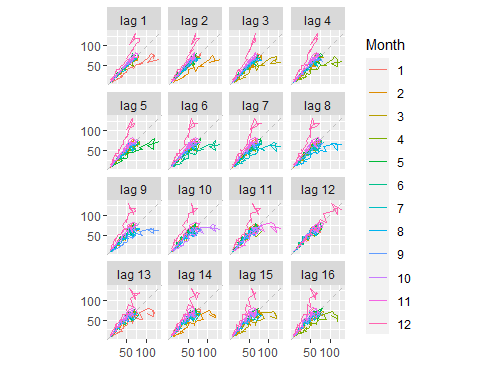
ggseasonplot(myts)



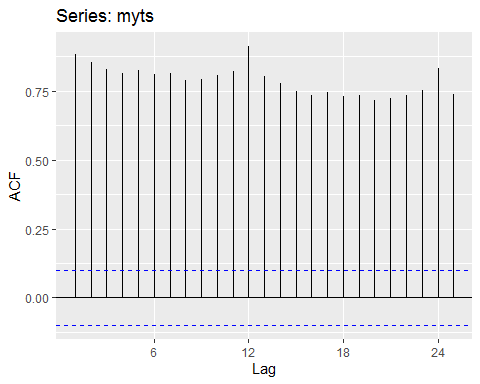
ggsubseriesplot(myts)



gglagplot(myts)



ggAcf(myts)



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

seasonality: looking at the time plot and seasonal subseries plot, there is a large jump in sales in December each year and the sales resume normal in January. From the seasonal plot, we can see that the slope of the seasonal effect increases each year.

cyclicity: from the time plot, we see no apparent cyclic behaviour in the time series.

trend: from the time plot, we can see that there is a clear and increasing trend.

autocorrelation: from the lag plot and ACF plot, the current values in the time series are highly correlated with the past values, implying the time series is highly predictable.

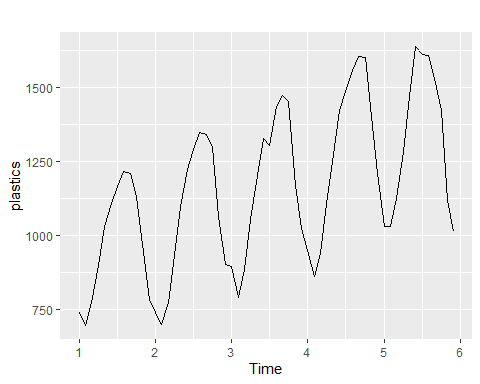
From the time series, we learn that people are spending more in the holiday season in December and the phenomena is growing bigger every year. There is a big drop in sales in April 2011. We may need to perform further research of historical events to find out what was causing that.

# 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?

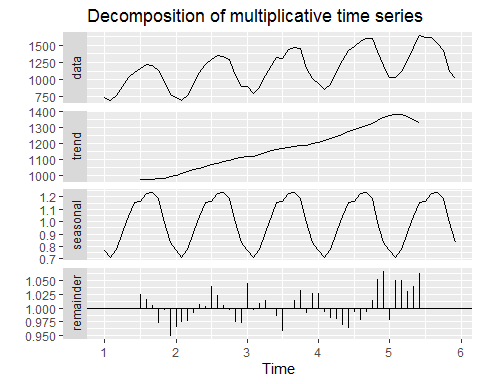
autoplot(plastics)



There is a strong seasonal effect that sales increase from the beginning to the middle of the year then decrease till the end of the year. There is also an upward trend in the whole time series.

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

classical\_multi <- decompose(plastics, type="multiplicative")  
autoplot(classical\_multi)

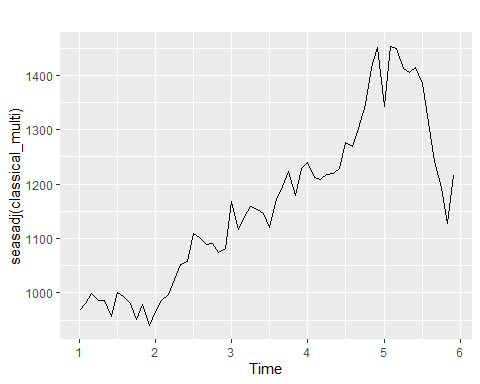


## c. Do the results support the graphical interpretation from part a?

Yes, the results support the graphical interpretation from part a. There is a strong seasonal effect and an upward trend.

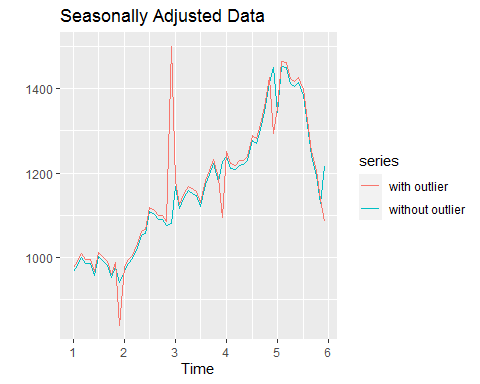
## d. Compute and plot the seasonally adjusted data.

autoplot(seasadj(classical\_multi))



## 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?

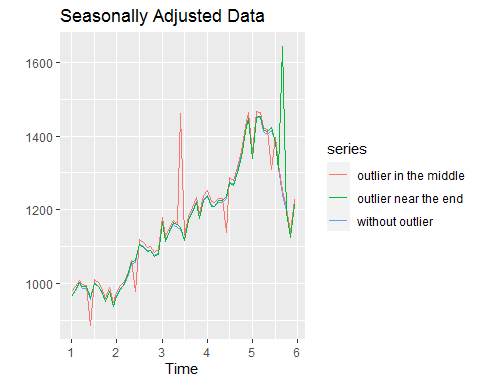
plastics\_adj <- plastics  
plastics\_adj[24] <- plastics\_adj[24] + 500  
  
autoplot(seasadj(classical\_multi), series = "without outlier") +  
 autolayer(seasadj(decompose(plastics\_adj, type="multiplicative")),   
 series = "with outlier") +  
 labs(y="", title="Seasonally Adjusted Data")



The outlier is also present in the seasonally adjusted data. The data at the same month of the outlier in each year have changed notably and all other values in the data have changed slightly.

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

plastics\_adj <- plastics  
plastics\_adj2 <- plastics  
plastics\_adj[round(length(plastics\_adj)/2,0)] <- plastics\_adj[round(length(plastics\_adj)/2,0)] + 500  
plastics\_adj2[length(plastics\_adj2)-3] <- plastics\_adj2[length(plastics\_adj2)-3] + 500  
  
autoplot(seasadj(classical\_multi), series = "without outlier") +  
 autolayer(seasadj(decompose(plastics\_adj, type="multiplicative")),   
 series = "outlier in the middle") +  
 autolayer(seasadj(decompose(plastics\_adj2, type="multiplicative")),   
 series = "outlier near the end") +  
 labs(y="", title="Seasonally Adjusted Data")



The outlier is also present in the seasonally adjusted data regardless the it is in the middle or near the end of the data. When the outlier is in the middle, the data at the same month of the outlier in all years are affected notably. When the outlier is near the end of the series, the values in the others years are not affected.

# 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.

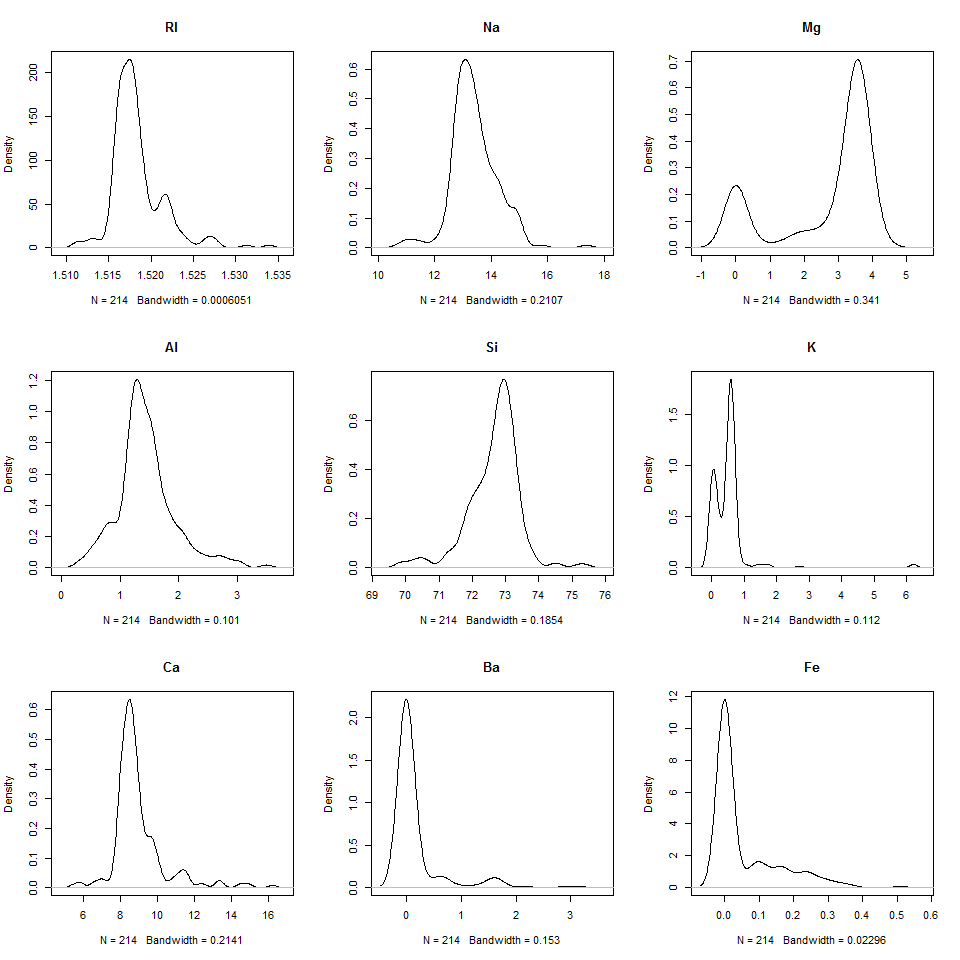
The data can be accessed via:

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.

par(mfrow = c(3, 3))  
  
for (i in (1:9)) {  
 plot(density(Glass[,i]),main=colnames(Glass)[i])  
}



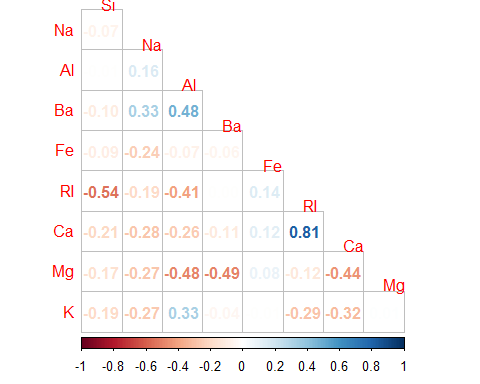
From the density plots, we find:

* **RI**, **Na**, **AI**, and **Si** are unimodal and approximately symmetric.
* **K**, **Ca**, **Ba**, and **Fe** are right skewed.
* **Mg** is bimodal.

library(corrplot)

## corrplot 0.92 loaded

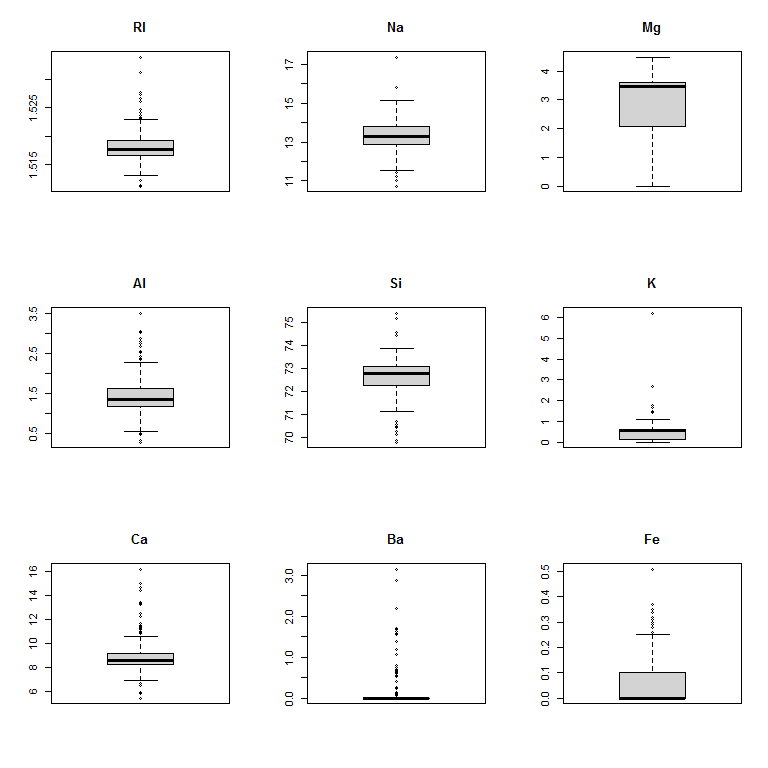
corrplot(cor(Glass[,-10], use = "na.or.complete"), method = 'number', order = "hclust",  
 type = 'lower', diag = FALSE, tl.srt = 0.1)



From the correlation plot, we find that *RI* and *Ca* are highly correlated. The correlations between other predictors are not severe.

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

par(mfrow = c(3, 3))  
  
for (i in (1:9)) {  
 boxplot(Glass[,i],main=colnames(Glass)[i])  
}



Confirmed from the boxplots, **K**, **Ca**, **Ba**, and **Fe** are right skewed which have serious outliers on the right tail. **RI**, **AI** and **Si** are symmetric but they have outliers on both tails.

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

For **K**, **Ca**, **Ba**, and **Fe** that are right skewed, a log-transformation or Box-Cox transformation may be applied to reduce skewness and outliers. For all predictors including the transformed ones, centering and scaling the data into the same standard can also help improving the model, since they are all chemical properties.

# 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)

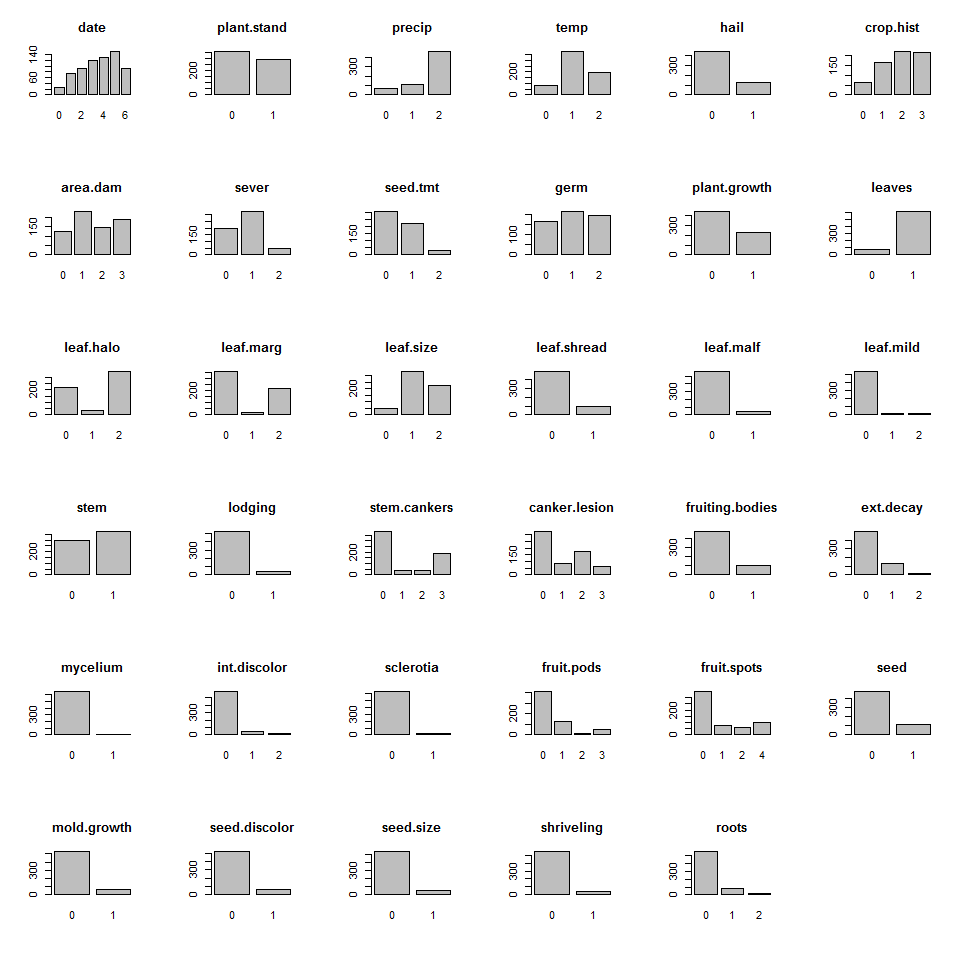
## (a) Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?

sapply(Soybean, is.factor)

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

All predictors are categorical

par(mfrow = c(6, 6))  
  
for (i in (2:36)) {  
 plot(Soybean[,i],main=colnames(Soybean)[i])  
}

 **leaf.mild**, **mycelium**, and **sclerotia** have distributions degenerate in the ways discussed earlier in this chapter. We should consider removing these predictors in our analysis.

This can be confirmed using the nearZeroVar function:

library(caret)

## Loading required package: lattice

colnames(Soybean)[nearZeroVar(Soybean)]

## [1] "leaf.mild" "mycelium" "sclerotia"

## (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?

The following shows the percentage of the missing values in each predictor.

sort(colMeans(is.na(Soybean[-1])), decreasing = TRUE)

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

More than half of the predictors have 10% or more values missing. Among those, **hail**, **sever**, **seed.tmt**, **lodging** have the highly likelihood to be missing.

We check the percentage of rows with at least 1 missing value for each class:

for (class in unique(Soybean$Class)) {  
 print(  
 paste0(class,": ",  
 round(sum(apply(is.na(Soybean[Soybean$Class==class,]),1,any))/  
 nrow(Soybean[Soybean$Class==class,])\*100,2),  
 "%"  
 )  
 )  
}

## [1] "diaporthe-stem-canker: 0%"  
## [1] "charcoal-rot: 0%"  
## [1] "rhizoctonia-root-rot: 0%"  
## [1] "phytophthora-rot: 77.27%"  
## [1] "brown-stem-rot: 0%"  
## [1] "powdery-mildew: 0%"  
## [1] "downy-mildew: 0%"  
## [1] "brown-spot: 0%"  
## [1] "bacterial-blight: 0%"  
## [1] "bacterial-pustule: 0%"  
## [1] "purple-seed-stain: 0%"  
## [1] "anthracnose: 0%"  
## [1] "phyllosticta-leaf-spot: 0%"  
## [1] "alternarialeaf-spot: 0%"  
## [1] "frog-eye-leaf-spot: 0%"  
## [1] "diaporthe-pod-&-stem-blight: 100%"  
## [1] "cyst-nematode: 100%"  
## [1] "2-4-d-injury: 100%"  
## [1] "herbicide-injury: 100%"

Only **phytophthora-rot**, **diaporthe-pod-&-stem-blight**, **cyst-nematode**, **2-4-d-injury**, **herbicide-injury** have missing predictor values

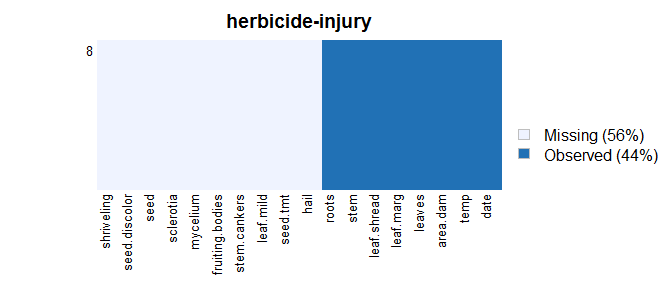
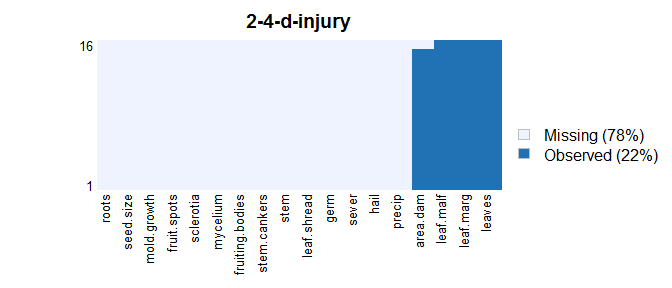
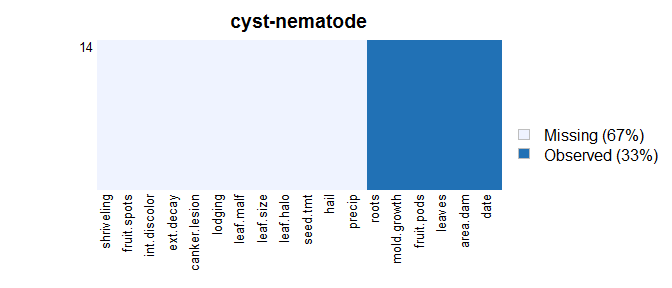
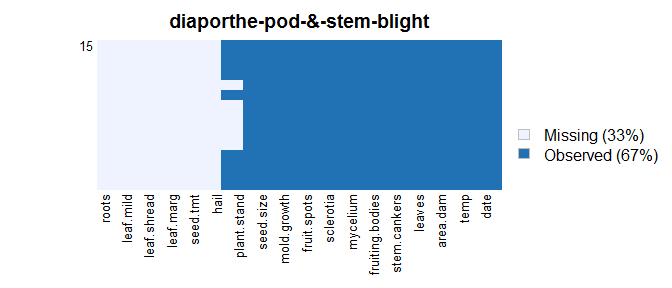
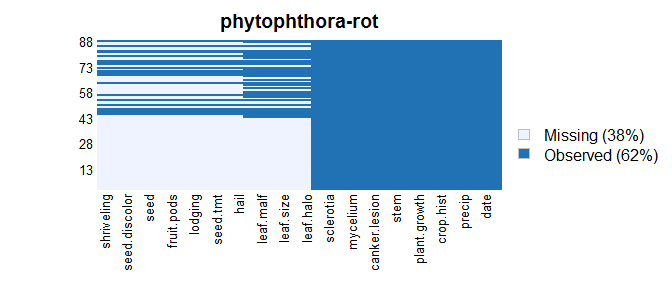
We can also check the missing values in each predictors for each class:

library(Amelia)

## Loading required package: Rcpp

## ##   
## ## Amelia II: Multiple Imputation  
## ## (Version 1.8.0, built: 2021-05-26)  
## ## Copyright (C) 2005-2022 James Honaker, Gary King and Matthew Blackwell  
## ## Refer to http://gking.harvard.edu/amelia/ for more information  
## ##

for (class in c("phytophthora-rot","diaporthe-pod-&-stem-blight", "cyst-nematode",   
 "2-4-d-injury", "herbicide-injury")) {  
 missmap(Soybean[Soybean$Class==class,], main=class)  
}



We can see there is a pattern of missing data related to the classes. The classes have different sets of predictors with missing values. The values are missing for a meaning and do provide predictive power.

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

For predictors with 1 missing value (**date** and **area.dam**), we can simply remove the record from the data.  
For other predictors, as the missing values are related to the classes, we can impute / flag the missing values using the value “unknown”.

# HA 7.1

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

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

ses\_pigs <- ses(pigs, h = 4)  
  
ses\_pigs$model

## 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

The optimal is 0.2971 and the optimal is 77260.0561

The forecasts for the next four months are:

ses\_pigs$mean

## Sep Oct Nov Dec  
## 1995 98816.41 98816.41 98816.41 98816.41

## b. Compute a 95% prediction interval for the first forecast using 1.96 where is the standard deviation of the residuals.

s <- sd(ses\_pigs$residuals)  
ses\_pigs$mean[1] + 1.96\*s

## [1] 118952.8

ses\_pigs$mean[1] - 1.96\*s

## [1] 78679.97

The 95% prediction interval for the first forecast is (78679.97, 118952.8)

## c. Compare your interval with the interval produced by R.

ses\_pigs$upper[1, "95%"]

## 95%   
## 119020.8

ses\_pigs$lower[1, "95%"]

## 95%   
## 78611.97

The interval produced by R is (78611.97, 119020.8)

The calculated interval using the standard deviation of the residuals is slightly different from the interval produced by R. For the simple exponential smoothing model, the forecast variance is where is the variance of the residuals. Hence, the computed results are different.

# 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 ) and level (the initial level ). It should return the forecast of the next observation in the series. Does it give the same forecast as ses?

ses\_cal <- function(pars = c(alpha, level), ts) {  
 ts\_df <- data.frame(Time=c(0), Observation=c(0.0), Level=c(0.0), Forecast=c(0.0))  
 l\_t <- pars[2]  
 alpha <- pars[1]  
 for (t in 1:length(ts)) {  
 ts\_df[t,"Time"] <- t  
 ts\_df[t,"Observation"] <- ts[t]  
 ts\_df[t,"Level"] <- alpha\*ts\_df[t,"Observation"] + (1-alpha)\*l\_t  
 ts\_df[t,"Forecast"] <- l\_t  
 l\_t <- ts\_df[t,"Level"]  
 }  
 return(l\_t) #The forecast from a ses model is simply the last level  
}

Using the and from the model produced by the ses function, the forecast of the next observation from our own function is

ses\_cal(c(ses\_pigs$model$par[1], ses\_pigs$model$par[2]),pigs)

## [1] 98816.41

The forecast of the next observation from the model produced by the ses function is

ses\_pigs$mean[1]

## [1] 98816.41

The two functions produce identical forecast

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

ses\_sse <- function(pars = c(alpha, level), ts) {  
 ts\_df <- data.frame(Time=c(0), Observation=c(0.0), Level=c(0.0), Forecast=c(0.0))  
 l\_t <- pars[2]  
 alpha <- pars[1]  
 for (t in 1:length(ts)) {  
 ts\_df[t,"Time"] <- t  
 ts\_df[t,"Observation"] <- ts[t]  
 ts\_df[t,"Level"] <- alpha\*ts\_df[t,"Observation"] + (1-alpha)\*l\_t  
 ts\_df[t,"Forecast"] <- l\_t  
 l\_t <- ts\_df[t,"Level"]  
 }  
 SSE <- sum((ts\_df$Observation - ts\_df$Forecast)^2)  
 return(SSE)  
}  
  
optimal\_para <- optim(par = c(0.5, pigs[1]), ts = pigs, fn = ses\_sse)  
optimal\_para

## $par  
## [1] 2.990081e-01 7.637927e+04  
##   
## $value  
## [1] 19767049111  
##   
## $counts  
## function gradient   
## 91 NA   
##   
## $convergence  
## [1] 0  
##   
## $message  
## NULL

The optimal is 0.299 and the optimal is 76379.27

ses\_pigs$model$par

## alpha l   
## 2.971488e-01 7.726006e+04

From the ses function, the is 2.971 and the optimal is 77260.06

The values from the optim() and the values from the ses function are not exactly the same but very close.

# HA 8.1

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

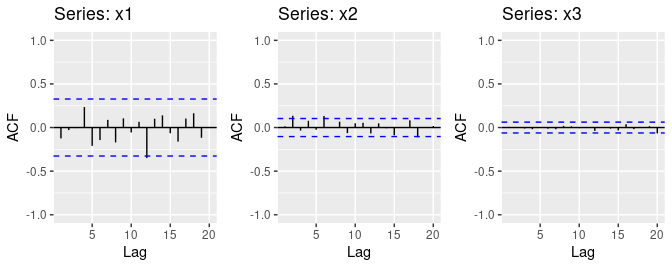


Figure 8.31: Left: ACF for a white noise series of 36 numbers. Middle: ACF for a white noise series of 360 numbers. Right: ACF for a white noise series of 1,000 numbers.

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

The autocorrelations in series x1 are larger than the autocorrelations in series x2 and autocorrelations in series x2 is larger than the autocorrelations in series x3.

The confidence interval (the distance between the blue bars) is larger in x1 than in x2 and it is larger in x2 than in x3. The interval limits are critical values to determine if the autocorrelations are significantly differ from 0.

Since the autocorrelations in all figures are within their confidence intervals, all three time series are considered a 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?

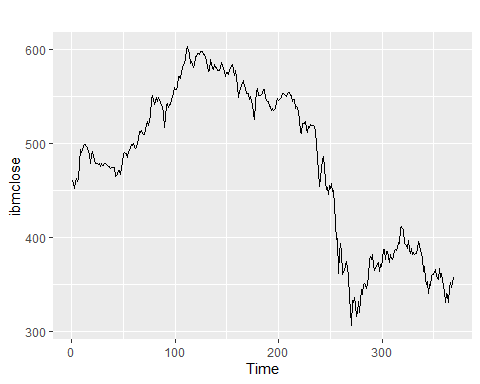
For white noise series, we expect the autocorrelations to be close to zero and within where is the length of the time series. Since the lengths of the three series are significantly different, the critical values are also significantly different.

For white noise series, the data points are considered independent and random. Though they are random, there may be still some notable correlation for some data points just by chance. As the number of data points increase, the effect of the correlated data points diminish. Therefore, the autocorrelations are different in each figure because of the size of the data but in fact they all refer to white noise.

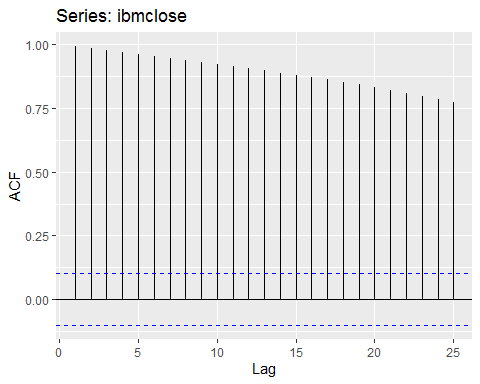
# HA 8.2

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.

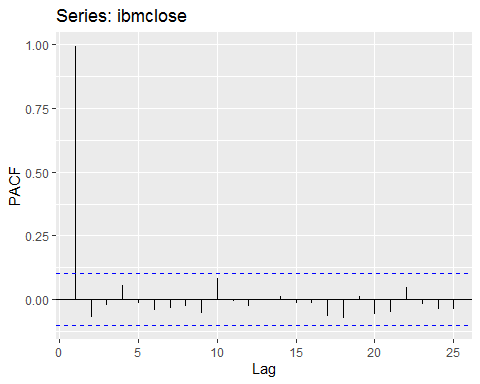
autoplot(ibmclose)



ggAcf(ibmclose)



ggPacf(ibmclose)



From the time plot, we can clearly see there are notable trends and changing levels. From the ACF plot, the autocorrelations at all 25 lags are much higher than the critical value. This implies that the data points are not independent. The PCAF doesn’t show any additional sign of non-stationary but above findings are sufficient evidence to conclude that the series is non-stationary and should be differenced.

# HA 8.6

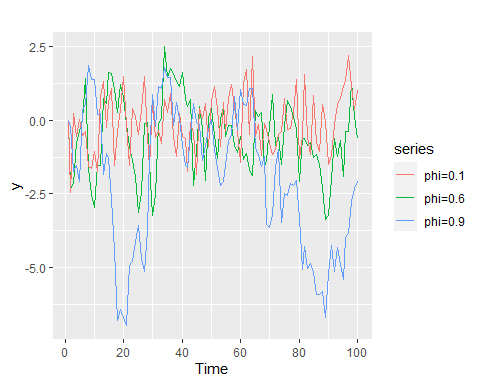
Use R to simulate and plot some data from simple ARIMA models.

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

set.seed(624)  
  
y <- ts(numeric(100))  
e <- rnorm(100)  
for(i in 2:100)  
 y[i] <- 0.6\*y[i-1] + e[i]

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

ts\_generator\_AR <- function(phi) {  
 y <- ts(numeric(100))  
 e <- rnorm(100)  
 for(i in 2:100)  
 y[i] <- phi\*y[i-1] + e[i]  
 return (y)  
}  
  
autoplot(y, series = "phi=0.6") +  
 autolayer(ts\_generator\_AR(0.1), series = paste0("phi=",0.1)) +  
 autolayer(ts\_generator\_AR(0.9), series = paste0("phi=",0.9))



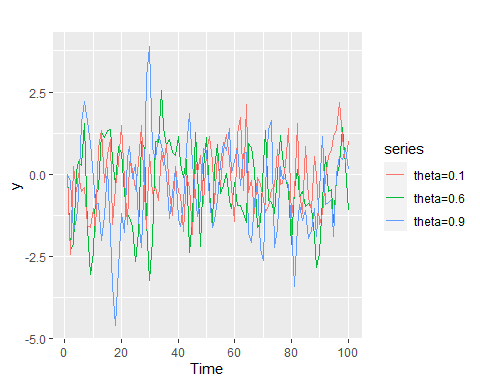
For higher , the fluctuation in the series is stronger since the new value depends more on the last value.

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

ts\_generator\_MA <- function(theta) {  
 y <- ts(numeric(100))  
 e <- rnorm(100)  
 e[1] <- 0  
 for(i in 2:100)  
 y[i] <- theta\*e[i-1] + e[i]  
 return (y)  
}  
  
set.seed(624)  
y <- ts\_generator\_MA(0.6)

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

autoplot(y, series = "theta=0.6") +  
 autolayer(ts\_generator\_MA(0.1), series = paste0("theta=",0.1)) +  
 autolayer(ts\_generator\_MA(0.9), series = paste0("theta=",0.9))



The higher , the higher the variance is in the time series. The new value depends more on the past errors.

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

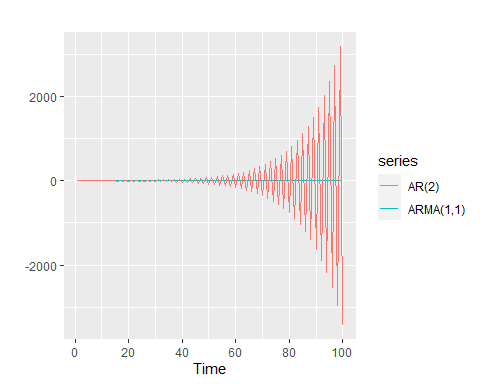
ts\_generator\_ARMA <- function(phi, theta) {  
 y <- ts(numeric(100))  
 e <- rnorm(100)  
 e[1] <- 0  
 for(i in 2:100)  
 y[i] <- phi\*y[i-1] + theta\*e[i-1] + e[i]  
 return (y)  
}  
  
set.seed(624)  
arma\_1\_1 <- ts\_generator\_ARMA(0.6, 0.6)

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

ts\_generator\_AR2 <- function(phi\_1, phi\_2){  
 y <- ts(numeric(100))  
 e <- rnorm(100)  
 for(i in 3:100)  
 y[i] <- phi\_1\*y[i-1] + phi\_2\*y[i-2] + e[i]  
 return(y)  
}  
  
set.seed(624)  
ar\_2 <- ts\_generator\_AR2(-0.8,0.3)

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

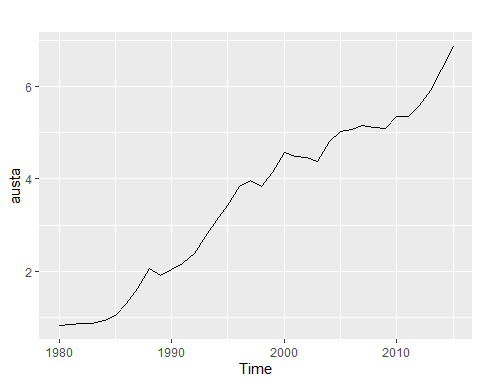
autoplot(arma\_1\_1, series = "ARMA(1,1)") +  
 autolayer(ar\_2, series = "AR(2)") +   
 labs(y="")

 The the magnitude of the AR(2) series increases exponentially with oscillation. The ARMA(1,1) series seems to be stationary / white noise.

# HA 8.8

1. Consider the total international visitors to Australia (in millions) for the period 1980-2015. (Data set austa.)

autoplot(austa)



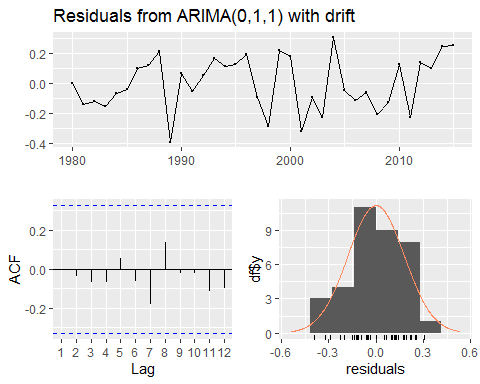
## a. Use auto.arima() to find an appropriate ARIMA model. What model was selected. Check that the residuals look like white noise. Plot forecasts for the next 10 periods.

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

An ARIMA(0,1,1) with drift is select. and . The model suggests a first order differencing with a constant drift, which implies there is a straight-line trend in the time series.

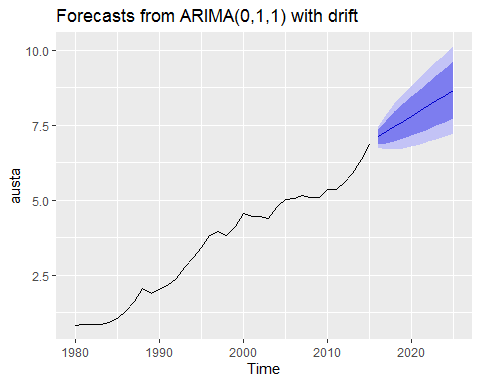
austa\_arima <- forecast(auto.arima(austa), h = 10)  
checkresiduals(austa\_arima)



##   
## 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

The residuals are independent with mean near 0. The residuals look like white noise. The Ljung-Box test has a p-value of 0.8067, implies that it is very **unlikely** that there are autocorrelations in the residuals.

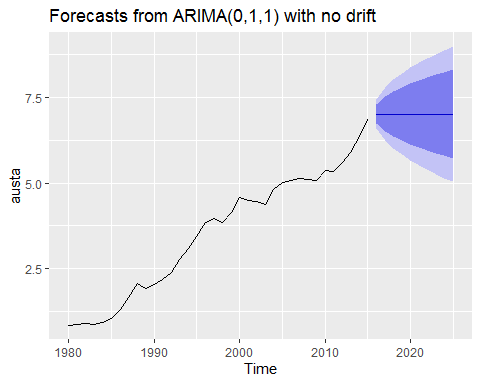
autoplot(austa\_arima)



The forecasts of the next 10 periods look plausible with an increasing trend, which is consistent with the model output.

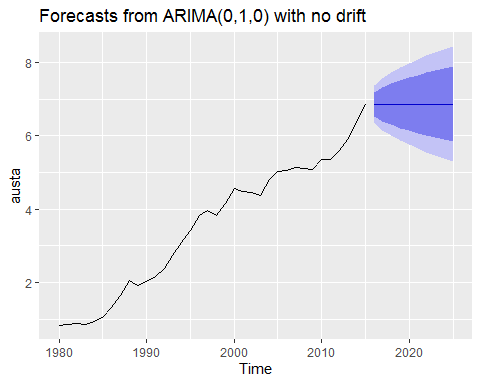
## b. Plot forecasts from an ARIMA(0,1,1) model with no drift and compare these to part (a). Remove the MA term and plot again.

austa\_arima\_nodrift <- forecast(Arima(austa, order = c(0, 1, 1),   
 include.drift = FALSE, method="ML"), h = 10)  
autoplot(austa\_arima\_nodrift) + labs(title="Forecasts from ARIMA(0,1,1) with no drift")



The forecast of the ARIMA(0,1,1) model with no drift has no trend in the forecasted values.

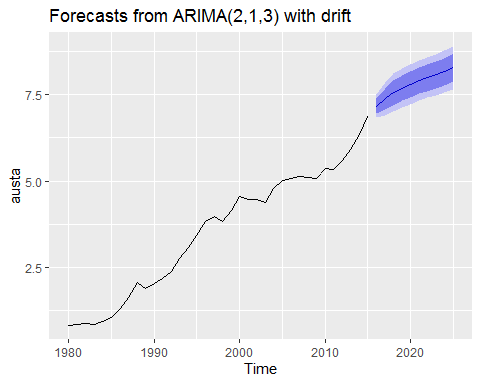
austa\_arima\_nodrift <- forecast(Arima(austa, order = c(0, 1, 0),   
 include.drift = FALSE, method="ML"), h = 10)  
autoplot(austa\_arima\_nodrift) + labs(title="Forecasts from ARIMA(0,1,0) with no drift")



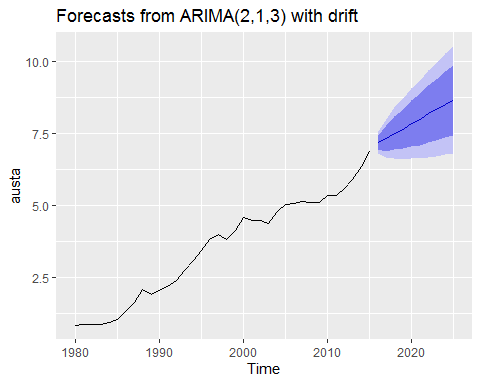
The forecast of the ARIMA(0,1,0) model with no drift also has no trend in the forecasted values and the confidence intervals are smaller than the model with the MA term.

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

austa\_arima\_2.1.3 <- forecast(Arima(austa, order = c(2, 1, 3), include.drift = TRUE, method="ML"),h = 10)  
autoplot(austa\_arima\_2.1.3) + labs(title="Forecasts from ARIMA(2,1,3) with drift")



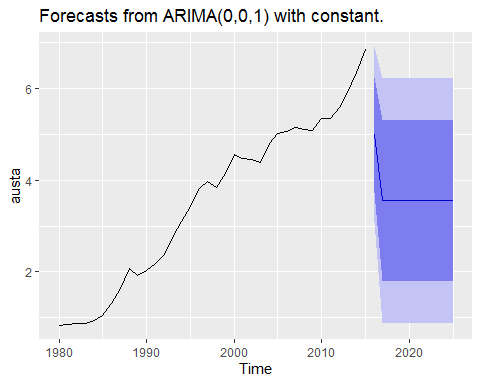
austa\_arima\_2.1.3\_nodrift <- forecast(Arima(austa, order = c(2, 1, 3), include.drift = FALSE, method="ML"),h = 10)  
autoplot(austa\_arima\_2.1.3\_nodrift) + labs(title="Forecasts from ARIMA(2,1,3) with drift")



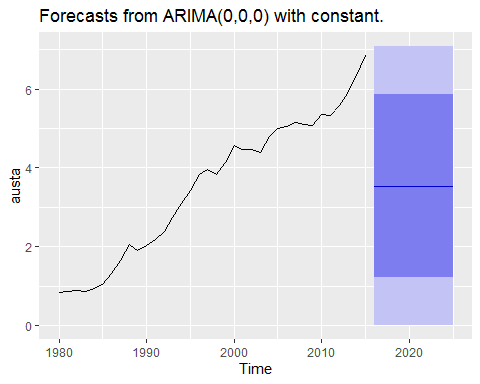
The forecasts of the ARIMA(0,1,0) model with drift have a damped trend. Without the drift, the trend in forecasts seem to be steady.

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

austa\_arima\_0.0.1 <- forecast(Arima(austa, order = c(0, 0, 1), include.constant = TRUE, method="ML"),h = 10)  
autoplot(austa\_arima\_0.0.1) + labs(title="Forecasts from ARIMA(0,0,1) with constant.")



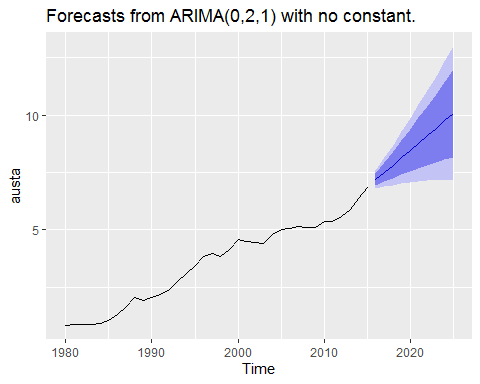
austa\_arima\_0.0.0 <- forecast(Arima(austa, order = c(0, 0, 0), include.constant = TRUE, method="ML"),h = 10)  
autoplot(austa\_arima\_0.0.0) + labs(title="Forecasts from ARIMA(0,0,0) with constant.")



The forecasts of the ARIMA(0,0,1) model decrease quickly and converge to the mean of of past values. The forecasts of the ARIMA(0,0,0) model are simply the mean of the past values, which are the same from the Average method. Also, the confidence intervals in the model ARIMA(0,0,1) are smaller than the model ARIMA(0,0,0).

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

austa\_arima\_0.2.1 <- forecast(Arima(austa, order = c(0, 2, 1), include.constant = FALSE, method="ML"),h = 10)  
autoplot(austa\_arima\_0.2.1) + labs(title="Forecasts from ARIMA(0,2,1) with no constant.")



The forecasts have a steady increasing trend with increasing size of confidence interval.