HW1

Group 4

07/10/2019

## Exercise 2.1

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

library(fpp2)  
help(gold)  
help(woolyrnq)  
help(gas)

Gold - Daily morning gold prices in US dollars for the period from 1 January 1985 to 31 March 1989.

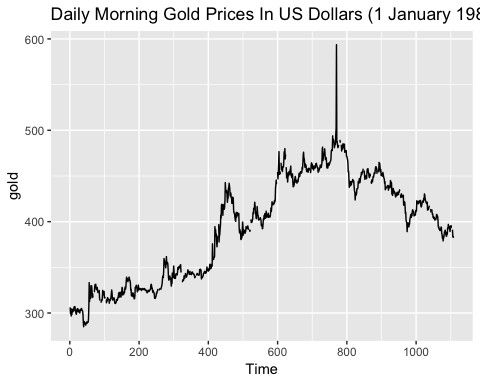
Woolyrnq - Quarterly production of woollen yarn in Australia (tonnes) for the period from Mar 1965 to Sep 1994.

Gas - Australian monthly gas production for the period from 1956 to 1995.

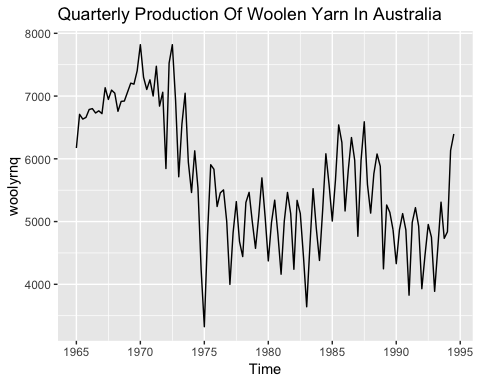
### Exercise 2.1.a

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

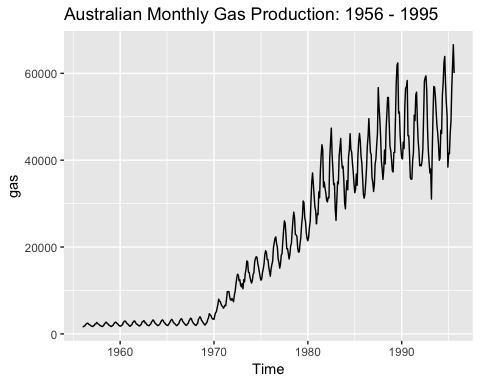
autoplot(gold) + ggtitle("Daily Morning Gold Prices In US Dollars (1 January 1985 – 31 March 1989)")



autoplot(woolyrnq) + ggtitle("Quarterly Production Of Woolen Yarn In Australia")



autoplot(gas) + ggtitle("Australian Monthly Gas Production: 1956 - 1995")



### Exercise 2.1.b

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

frequency(gold)

## [1] 1

frequency(woolyrnq)

## [1] 4

frequency(gas)

## [1] 12

The “frequency” is the number of observations before the seasonal pattern repeats.

The frequency of gold series is 1 (daily)

The frequency of woolyrnq series is 4 (quaterly)

The frequency of gas series is 12 (monthly)

### Exercise 2.1.c

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

which.max(gold)

## [1] 770

Observation number 770 has an outlier in Gold series.

## Exercise 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.

### Exercise 2.3.a

You can read the data into R with the following script:

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

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

### Exercise 2.3.b

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

# Turnover ; New South Wales ; Supermarket and grocery stores  
myts <- ts(retaildata[,"A3349335T"],  
frequency=12, start=c(1982,4))

Turnover of supermarket and grocery stores in New South Wales (column “A3349335T”“) was selected for further analysis.

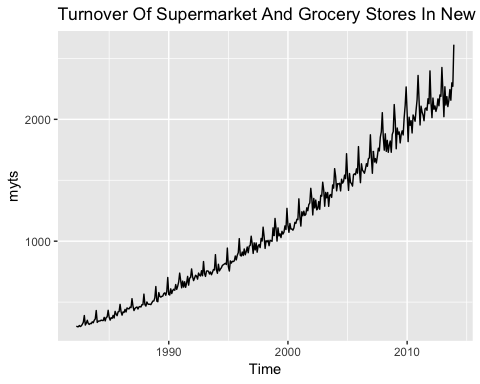
### Exercise 2.3.c

Explore your chosen retail time series using the following functions:

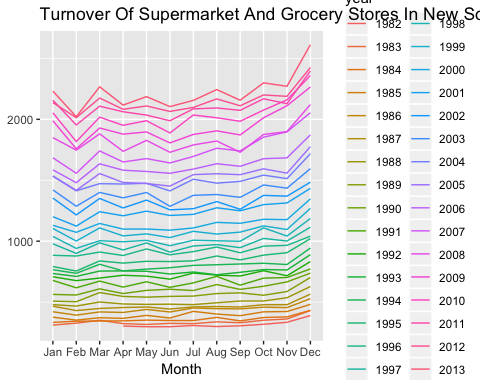
autoplot(), ggseasonplot(), ggsubseriesplot(), gglagplot(), ggAcf()

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

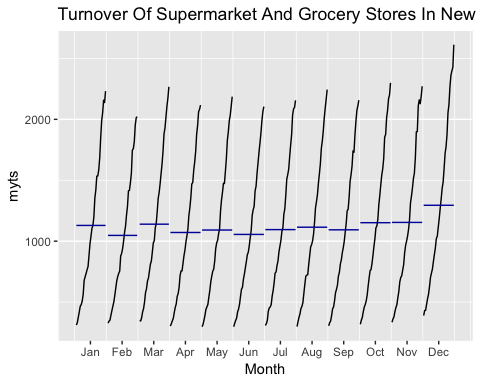
autoplot(myts) + ggtitle("Turnover Of Supermarket And Grocery Stores In New South Wales")



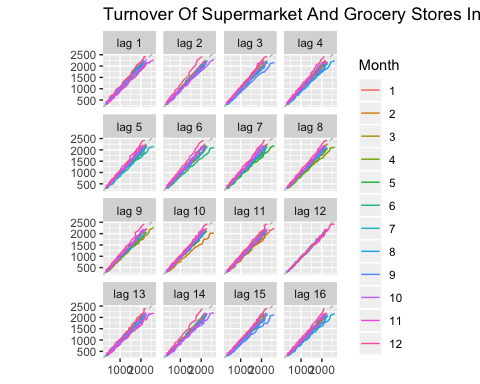
ggseasonplot(myts) + ggtitle("Turnover Of Supermarket And Grocery Stores In New South Wales")



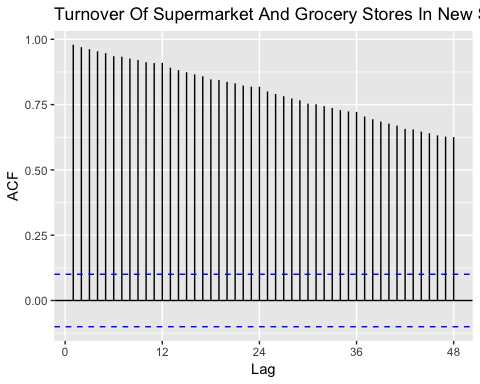
ggsubseriesplot(myts) + ggtitle("Turnover Of Supermarket And Grocery Stores In New South Wales")



gglagplot(myts) + ggtitle("Turnover Of Supermarket And Grocery Stores In New South Wales")



ggAcf(myts, lag = 48) + ggtitle("Turnover Of Supermarket And Grocery Stores In New South Wales")



Turnover of supermarket and grocery stores shows a strong increasing trend. There is no evidence of any cyclic behaviour here, though perhaps the data only demonstrates a portion of the cycling.

It is clear that there is a large jump in sales in December each year which is possibly due to Christmas and New Year holidays. February and June shows the lowest turnover during the year.

The data have a trend, so the autocorrelations for small lags tend to be large and positive because observations nearby in time are also nearby in size. So the ACF of trended time series has positive values that slowly decrease as the lags increase.

Overall turnover increasing over the years.

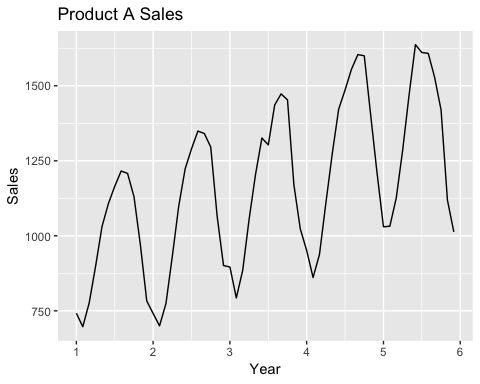
## Exercise 6.2

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

### Exercise 6.2.a

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

library(fpp2)  
autoplot(plastics)+ ggtitle("Product A Sales") + xlab("Year") + ylab("Sales")



This Time Plot shows very clearly that the data have upward trend and seasonal fluctuations peaking midway through the year.

### Exercise 6.2.b

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

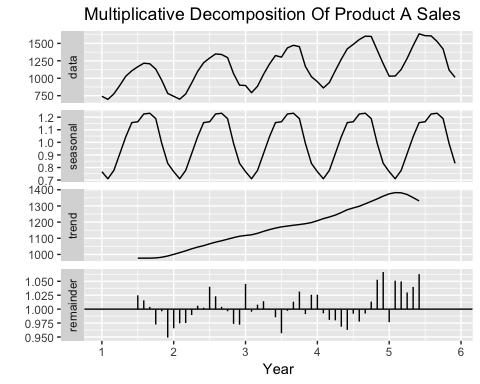
data\_decomp <- decompose(plastics, type = "multiplicative")  
# trend-cycle indices  
data\_decomp$trend

## Jan Feb Mar Apr May Jun Jul  
## 1 NA NA NA NA NA NA 976.9583  
## 2 1000.4583 1011.2083 1022.2917 1034.7083 1045.5417 1054.4167 1065.7917  
## 3 1117.3750 1121.5417 1130.6667 1142.7083 1153.5833 1163.0000 1170.3750  
## 4 1208.7083 1221.2917 1231.7083 1243.2917 1259.1250 1276.5833 1287.6250  
## 5 1374.7917 1382.2083 1381.2500 1370.5833 1351.2500 1331.2500 NA  
## Aug Sep Oct Nov Dec  
## 1 977.0417 977.0833 978.4167 982.7083 990.4167  
## 2 1076.1250 1084.6250 1094.3750 1103.8750 1112.5417  
## 3 1175.5000 1180.5417 1185.0000 1190.1667 1197.0833  
## 4 1298.0417 1313.0000 1328.1667 1343.5833 1360.6250  
## 5 NA NA NA NA NA

# seasonal indices  
data\_decomp$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

# multiplicative decomposition (plot)  
autoplot(data\_decomp) + xlab("Year") +  
 ggtitle("Multiplicative Decomposition Of Product A Sales")



### Exercise 6.2.c

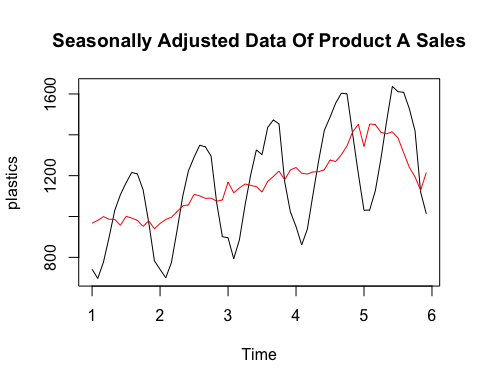
Do the results support the graphical interpretation from part a?

Yes, the trend-cycle indices show a steady rise over the years. Seasonal indices confirm stable repeating fluctuations during the years with peaks in summer months.

### Exercise 6.2.d

Compute and plot the seasonally adjusted data.

plot(plastics, col="black", main="Seasonally Adjusted Data Of Product A Sales")  
lines(seasadj(data\_decomp), col="red")

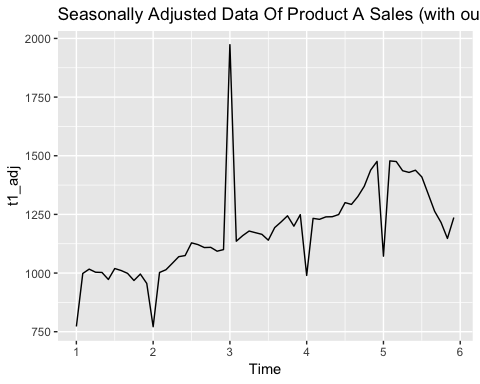


The resulting plot shows the seasonally adjusted data represented by a red line which includes trend-cycle and the remainder components.

### Exercise 6.2.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?

t1 <- plastics  
t1[25] = t1[25] + 1000  
  
t1\_decomp <- decompose(t1, type="multiplicative")  
t1\_adj <- seasadj(t1\_decomp)  
  
plot1<-autoplot(t1\_adj) + ggtitle("Seasonally Adjusted Data Of Product A Sales (with outlier in the middle)")  
plot1

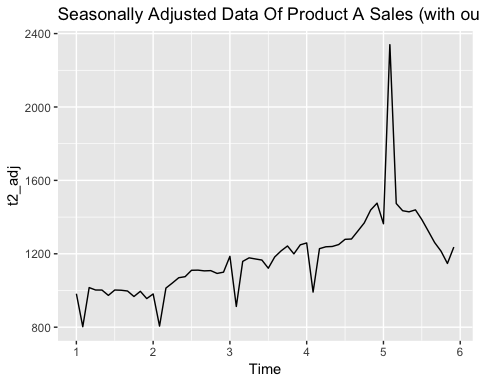


Seasonally adjusted data has been affected by the outlier in the middle.

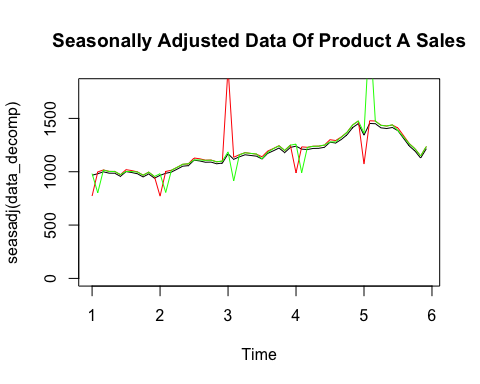
### Exercise 6.2.f

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

t2 <- plastics  
t2[50] <- t2[50] + 1000  
  
t2\_decomp <- decompose(t2, type="multiplicative")  
t2\_adj <- seasadj(t2\_decomp)  
autoplot(t2\_adj) + ggtitle("Seasonally Adjusted Data Of Product A Sales (with outlier at the end)")



plot(seasadj(data\_decomp), col="black", main="Seasonally Adjusted Data Of Product A Sales", ylim=c(0, 1800))  
lines(seasadj(t1\_decomp), col="red", ylim=c(0, 1800))  
lines(seasadj(t2\_decomp), col="green",ylim=c(0, 1800))



black line: seasonally adjusted data without outlier

red line: seasonally adjusted data with outlier at the middle

green line: seasonally adjusted data with outlier at the end

From the graph above we can see that seasonally adjusted data has been affected by the outlier in the middle and at the end, but there is no apparent difference if the outlier is near the end rather than in the middle of the time series.

Although we expected that the outlier at end of the period affects the seasonality of the time series less strongly than an outlier in the middle.

## Exercise 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)   
library(dplyr)  
library(psych)  
library(corrplot)   
library(e1071)  
library(car)  
library(caret)  
library(tidyr)  
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 ...

Nine chemery predictors are numerical data, and column “type” is factors with six levels: 1,2,3,5,6,and 7. There are no missing data in the table.

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

Si has the highest percentage usage among all elements (69.81% to 75.41%). Fe has the lowest percentage usage among all elements (0% to 0.51%).

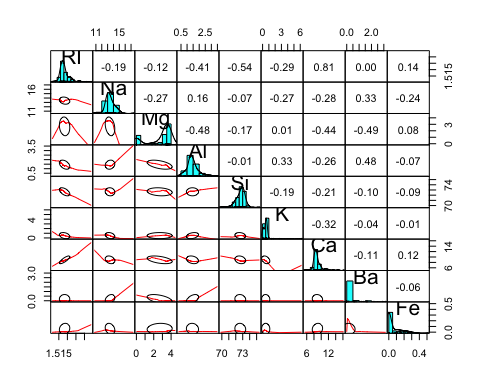
Type “1” and “2” glasses have 70 and 76 samples which are 65% of total sample size. Type “6” glass has 9 sample size which is the smallest among six types of glasses.

### Exercise 3.1.a

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

The relationship of predictors can be shown by several pairwise plots.

pairs.panels(Glass[,-10],show.points=FALSE,gap=FALSE)



“Na” and “Si” seems close to normally distributed.

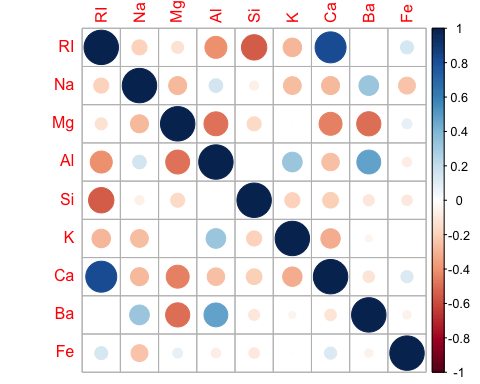
“RI”, “Al” are slightly right-skewed.

“Fe”, “K”, “Ba”, “Ca” are strongly right-skewed.

“Mg” does not seem to be normally distributed.

Corrplot and pairwise plots allow to detect elements which correlate significantly between each other.

corrplot(cor(Glass[,-10]))



“RI” - “CA” (0.81)

“RI” - “SI” (-0.54)

Potentially it can cause a collinearity problem during model building process.

cor(Glass[,-10], as.numeric(Glass[,10]))

## [,1]  
## RI -0.168739357  
## Na 0.506424080  
## Mg -0.728159518  
## Al 0.591197598  
## Si 0.149690687  
## K -0.025834560  
## Ca -0.008997841  
## Ba 0.577676375  
## Fe -0.183206747

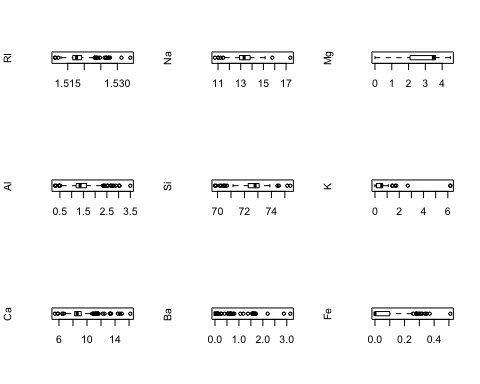
Correlation between each elements and a glass type indicates that “Na”, “Mg”, “Al”, “Ba” strongly correlate with glass type (correlation coefficient more than 0.5) potentially making them a good predictors of glass type.

### Exercise 3.1.b

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

The Boxplot displays ouliers and any values outside the whiskers are considered outliers.

data <-Glass[,-10]  
par(mfrow = c(3, 3))  
for (i in 1:ncol(data)) {  
 boxplot(data[ ,i], ylab = names(data[i]), horizontal=T)  
}



apply(Glass[,-10],2,skewness)

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

Box Plots show that all elements except Mg have outliers. Outliers identification and considering if they are influential points or not are the important part of any modeling process.

Computed skewness of the elements allowed us to confirm the findings discussed abouve: higly skewd elements are Fe, Ba, K, Ca

### Exercise 3.1.c

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

Yeo Johnson transformation was selected as a normalizing transformation. The Yeo-Johnson transformation can be thought of as an extension of the Box-Cox transformation. It handles both positive and negative values, whereas the Box-Cox transformation only handles positive values. Both can be used to transform the data so as to improve normality.

# YeoJohnson tranformation  
glass\_trans=preProcess(Glass[,-10], method=c("YeoJohnson"))  
pred=predict(glass\_trans,Glass[-10])  
  
# checking skewness after YeoJohnson transformation  
apply(pred,2,skewness)

## RI Na Mg Al Si   
## 1.6027150827 -0.0088476749 -0.8770969306 0.0002128304 -0.7202392108   
## K Ca Ba Fe   
## -0.0708227694 -0.2063893005 3.3686799688 1.7298107096

Yeo Johnson method allowed to normalize the variables, but not all of them. The following elements were transformed to approximately normal distribution: Na, Al, K, Ca.

## Exercise 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.

### Exercise 3.2.a

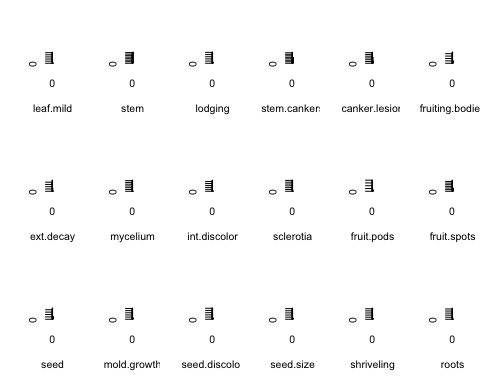
Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?

library(mlbench)   
data(Soybean)   
str(Soybean)

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

As we see all variable are factors.

X <- Soybean[,1:36]  
par(mfrow = c(3, 6))  
for (i in 1:ncol(X)) {  
 barplot(table(Soybean[,i]),xlab = names(X[i]))  
}



A degenerate distribution is a probability distribution in a space (discrete or continuous) with support only on a space of lower dimension. As it is said in the book: “Some models can be crippled by predictors with degenerate distributions. In these cases, there can be a significant improvement in model performance and/or stability without the problematic variables… such an uninformative variable may have little effect on the calculations.”

The plots indicate that we have the following variables such as “sclerotia”, “leaf.mild” and “mycelium” that are close to zero variance predictors (a predictor variable that has a single unique value).

Using nearZeroVar() we can confirm which variables are close to zero-variance predictors.

nearZeroVar detects predictors that have one unique value (i.e. are zero variance predictors) or predictors that have both of the following characteristics: they have very few unique values relative to the number of samples and the ratio of the frequency of the most common value to the frequency of the second most common value is large.

<https://www.rdocumentation.org/packages/caret/versions/6.0-84/topics/nearZeroVar>

# Near zero variance predictors  
library(caret)   
  
nearZeroVar(Soybean)

## [1] 19 26 28

nearZeroVar(Soybean, names = TRUE)

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

nearZeroVar() has confirmed that the following variables are zero variance predictors: “sclerotia”, “leaf.mild” and “mycelium”. These variables can be removed before model building process.

### Exercise 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?

% of missing values per each variables were calculated below:

sapply(Soybean, function(y) sum(length(which(is.na(y)))))/nrow(Soybean)\*100

## Class date plant.stand precip   
## 0.0000000 0.1464129 5.2708638 5.5636896   
## temp hail crop.hist area.dam   
## 4.3923865 17.7159590 2.3426061 0.1464129   
## sever seed.tmt germ plant.growth   
## 17.7159590 17.7159590 16.3982430 2.3426061   
## leaves leaf.halo leaf.marg leaf.size   
## 0.0000000 12.2986823 12.2986823 12.2986823   
## leaf.shread leaf.malf leaf.mild stem   
## 14.6412884 12.2986823 15.8125915 2.3426061   
## lodging stem.cankers canker.lesion fruiting.bodies   
## 17.7159590 5.5636896 5.5636896 15.5197657   
## ext.decay mycelium int.discolor sclerotia   
## 5.5636896 5.5636896 5.5636896 5.5636896   
## fruit.pods fruit.spots seed mold.growth   
## 12.2986823 15.5197657 13.4699854 13.4699854   
## seed.discolor seed.size shriveling roots   
## 15.5197657 13.4699854 15.5197657 4.5387994

The following variables have the lagest % of missing values: “sever”, “lodging”, “hail”, “seed.tmt”. Hence we can assume that these variables are more likely to be missing.

Apart from checking NA’s in each predictors, we also checked if some classes have missing data.

Soybean %>%  
 filter(!complete.cases(.)) %>%   
group\_by(Class) %>%   
 summarise(na = n()) %>%  
 select(Class, na) %>%   
 arrange(desc(na))

## # A tibble: 5 x 2  
## Class na  
## <fct> <int>  
## 1 phytophthora-rot 68  
## 2 2-4-d-injury 16  
## 3 diaporthe-pod-&-stem-blight 15  
## 4 cyst-nematode 14  
## 5 herbicide-injury 8

“phytophthora-rot” has the most NA’s. “2-4-d-injury”, “diaporthe-pod-&-stem-blight” and “cyst-nematode” also have missing values, but significantly less. We can conclude that the pattern of missing data is related to the classes.

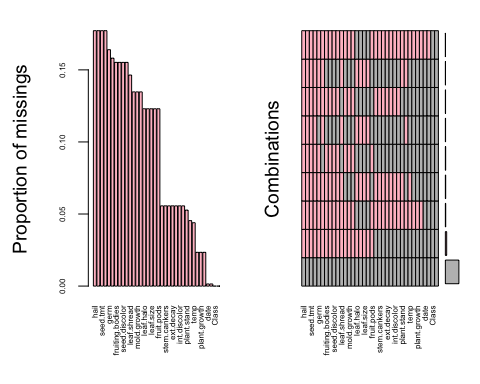
### Exercise 3.2.c

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

If missingness is not informative or close to 100% we can potentially remove predictors. Let’s check missing values distribution further with aggr() from VIF package. This function allows us to plot the amount of missing/imputed values in each variable and the amount of missing/imputed values in certain combinations of variables.

library(VIM)  
aggr(Soybean, col=c('grey','pink'), sortVars=T,numbers=T, cex.axis=0.5)

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



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

There are lot of missing data for some predictors. It may be possible to remove the predictors with the largest number of missing values. There are several techniques that hepl to deal with missing values. We are going to apply one of them - kNN based method. The assumption behind using KNN for missing values is that a point value can be approximated by the values of the points that are closest to it, based on other variables.

kNN() performs imputation of missing data in a data frame using the k-Nearest Neighbour algorithm.

Soybean\_imp<- kNN(Soybean, variable = c("hail"), k =5)  
summary(Soybean\_imp)

## 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   
## 0:551 0 : 65 0 :123 0 :195 0 :305 0 :165   
## 1:132 1 :165 1 :227 1 :322 1 :222 1 :213   
## 2 :219 2 :145 2 : 45 2 : 35 2 :193   
## 3 :218 3 :187 NA's:121 NA's:121 NA's:112   
## NA's: 16 NA's: 1   
##   
##   
## plant.growth leaves leaf.halo leaf.marg leaf.size leaf.shread  
## 0 :441 0: 77 0 :221 0 :357 0 : 51 0 :487   
## 1 :226 1:606 1 : 36 1 : 21 1 :327 1 : 96   
## NA's: 16 2 :342 2 :221 2 :221 NA's:100   
## NA's: 84 NA's: 84 NA's: 84   
##   
##   
##   
## leaf.malf leaf.mild stem lodging stem.cankers canker.lesion  
## 0 :554 0 :535 0 :296 0 :520 0 :379 0 :320   
## 1 : 45 1 : 20 1 :371 1 : 42 1 : 39 1 : 83   
## NA's: 84 2 : 20 NA's: 16 NA's:121 2 : 36 2 :177   
## NA's:108 3 :191 3 : 65   
## NA's: 38 NA's: 38   
##   
##   
## fruiting.bodies ext.decay mycelium int.discolor sclerotia fruit.pods  
## 0 :473 0 :497 0 :639 0 :581 0 :625 0 :407   
## 1 :104 1 :135 1 : 6 1 : 44 1 : 20 1 :130   
## NA's:106 2 : 13 NA's: 38 2 : 20 NA's: 38 2 : 14   
## NA's: 38 NA's: 38 3 : 48   
## NA's: 84   
##   
##   
## fruit.spots seed mold.growth seed.discolor seed.size shriveling  
## 0 :345 0 :476 0 :524 0 :513 0 :532 0 :539   
## 1 : 75 1 :115 1 : 67 1 : 64 1 : 59 1 : 38   
## 2 : 57 NA's: 92 NA's: 92 NA's:106 NA's: 92 NA's:106   
## 4 :100   
## NA's:106   
##   
##   
## roots hail\_imp   
## 0 :551 Mode :logical   
## 1 : 86 FALSE:562   
## 2 : 15 TRUE :121   
## NA's: 31   
##   
##   
##

As we see there is no missing values in “hail” variable now. We can apply the same approach to other variables.

library(fpp2)

## Exercise 7.1

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

### Exercise 7.1.a

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

ses\_m1<- ses(pigs, h = 4)  
  
# finding optimal parameters  
ses\_m1$model$par

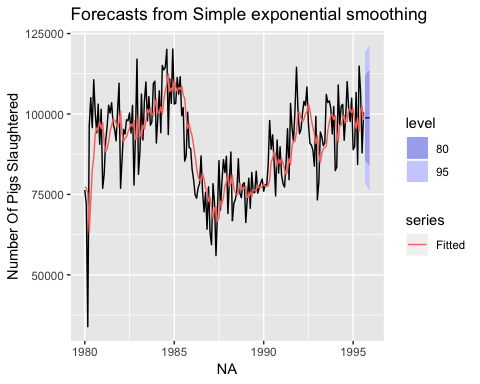
## alpha l   
## 2.971488e-01 7.726006e+04

summary(ses\_m1)

##   
## 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  
## Training set 385.8721 10253.6 7961.383 -0.922652 9.274016 0.7966249  
## ACF1  
## Training set 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

We can see that ses() has calculated the following optimal parametrs: α = 0.2971 and l0 = 77260.1 As α is small, more weight is given to observations from the more distant past.

autoplot(ses\_m1) + autolayer(fitted(ses\_m1), series = "Fitted") +  
 ylab("Number Of Pigs Slaughtered")



### Exercise 7.1.b

Compute a 95% prediction interval for the first forecast using ^y±1.96s where s is the standard deviation of the residuals. Compare your interval with the interval produced by R.

# calculating 95% prediction interval for the first forecast produced by R  
ses\_m1$upper[1, "95%"]

## 95%   
## 119020.8

ses\_m1$lower[1, "95%"]

## 95%   
## 78611.97

# calculating the standard deviation of the residuals  
s <- sd(ses\_m1$residuals)  
  
# calculating 95% prediction interval for the first forecast using formula ^y±1.96s  
ses\_m1$mean[1] + 1.96\*s

## [1] 118952.8

ses\_m1$mean[1] - 1.96\*s

## [1] 78679.97

A 95% prediction interval calculated using formula ^y±1.96s and produced by R are very similar.

## Exercise 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?

Solution:

The unknown parameters and the initial values for any exponential smoothing method can be estimated by minimising the SSE. Hence, we find the values of the unknown parameters and the initial values that minimise the SSE.

# function takes argument as a vector, because optim() requires vector as the first argument of the function.  
sse\_fn <- function( pars=c(alpha,l0), y){  
   
 # assigning the initial sse value  
 sse <- 0  
 # assigning first parameter to alpha  
 alpha <- pars[1]  
 # assigning second parametr to l0   
 l0 <- pars[2]  
 y\_hat <- l0  
   
 # calculating sse   
 for(index in 1:length(y)){  
 sse <- sse + (y[index] - y\_hat)^2  
 y\_hat <- alpha\*y[index] + (1 - alpha)\*y\_hat  
 }  
   
 return(sse)  
}

# finding the optimal values of α and ℓ0 using the optim(). Setting initial value of alpha = 0, l0 = first observation value of pigs data set  
opt\_par<- optim(par = c(0, pigs[1]), y = pigs, fn = sse\_fn)  
opt\_par

## $par  
## [1] 0.297086 77272.075070  
##   
## $value  
## [1] 19765613447  
##   
## $counts  
## function gradient   
## 179 NA   
##   
## $convergence  
## [1] 0  
##   
## $message  
## NULL

# finding the optimal values of α and ℓ0 using the optim()  
ses\_m1$model$par

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

Optimal parameters calculated using optim() based on sum of squared errors and produced by R are very similar.

## Excersise 7.5

Data set books contains the daily sales of paperback and hardcover books at the same store. The task is to forecast the next four days’ sales for paperback and hardcover books.

### Excersise 7.5.a

Plot the series and discuss the main features of the data.

library(fpp2)  
library(ggfortify)   
library(tidyverse)

autoplot(books, main = "Daily Sale Of Books",xlab="Day")



The daily sales of paperback and hardcover books show upward trend over the quite short period of time - 30 days. There is no evidence of any cyclic behaviour here, though perhaps the data only demonstrates a portion of the cycling. There is no apparent seasonality.

### Excersise 7.5.b.

Use the ses() function to forecast each series, and plot the forecasts.

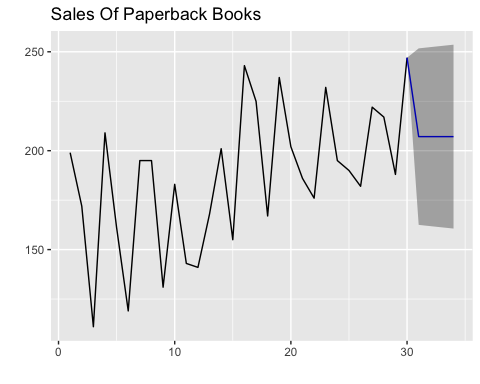
# 4 days forecast for Paperback books sales (SES method)  
ses.paper <- ses(books[, "Paperback"], h = 4)  
ses.paper

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 31 207.1097 162.4882 251.7311 138.8670 275.3523  
## 32 207.1097 161.8589 252.3604 137.9046 276.3147  
## 33 207.1097 161.2382 252.9811 136.9554 277.2639  
## 34 207.1097 160.6259 253.5935 136.0188 278.2005

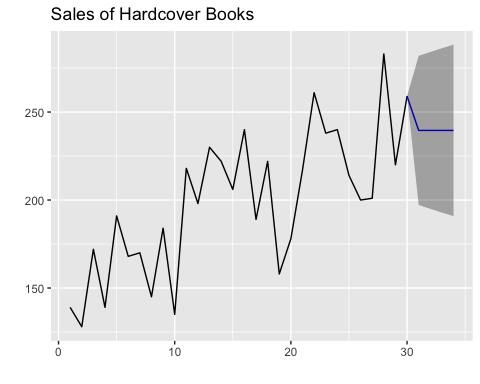
# 4 days forecast for Hardcover books sales (SES method)  
ses.hard <- ses(books[, "Hardcover"], h = 4)  
ses.hard

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 31 239.5601 197.2026 281.9176 174.7799 304.3403  
## 32 239.5601 194.9788 284.1414 171.3788 307.7414  
## 33 239.5601 192.8607 286.2595 168.1396 310.9806  
## 34 239.5601 190.8347 288.2855 165.0410 314.0792

autoplot(ses.paper, main = "Sales Of Paperback Books")



autoplot(ses.hard, main = "Sales of Hardcover Books")



### Excersise 7.5.c.

Compute the RMSE values for the training data in each case.

# RMSE of Paperback forecast on training data (SES method)  
sqrt(mean(ses.paper$residuals^2))

## [1] 33.63769

# RMSE of Hardcover forecast on training data (SES method)  
sqrt(mean(ses.hard$residuals^2))

## [1] 31.93101

Forecast of Hardcover books on training data are more accurate (lower RMSE).

## Excersise 7.6

### Excersise 7.6 a.

Now apply Holt’s linear method to the paperback and hardback series and compute four-day forecasts in each case.

# 4 days forecast for Paperback books sales (Holt's method)  
holt.paper <- holt(books[, "Paperback"], h = 4)  
holt.paper

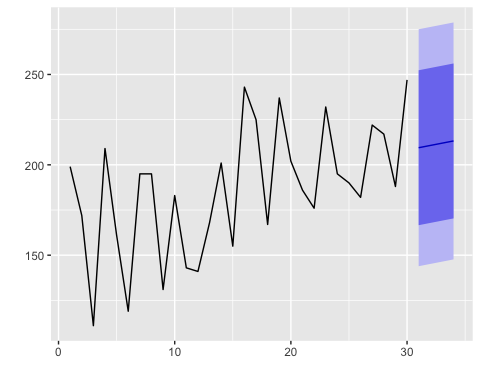
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 31 209.4668 166.6035 252.3301 143.9130 275.0205  
## 32 210.7177 167.8544 253.5811 145.1640 276.2715  
## 33 211.9687 169.1054 254.8320 146.4149 277.5225  
## 34 213.2197 170.3564 256.0830 147.6659 278.7735

# 4 days forecast for Hardcover books sales (Holt's method)  
holt.hard <- holt(books[, "Hardcover"], h = 4)  
holt.hard

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 31 250.1739 212.7390 287.6087 192.9222 307.4256  
## 32 253.4765 216.0416 290.9113 196.2248 310.7282  
## 33 256.7791 219.3442 294.2140 199.5274 314.0308  
## 34 260.0817 222.6468 297.5166 202.8300 317.3334

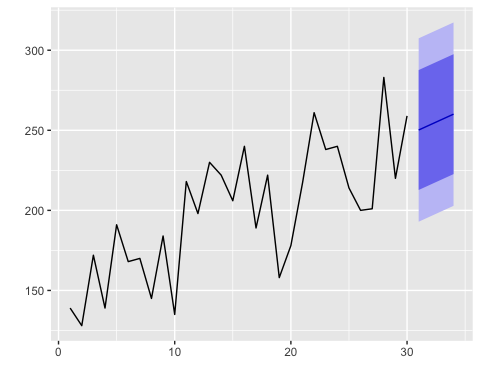
autoplot(books[, "Paperback"]) + autolayer(holt.paper, main = "Sales Of Paperback Books (Holt's method)")

## Warning: Ignoring unknown parameters: main



autoplot(books[, "Hardcover"]) + autolayer(holt.hard, main = "Sales Of Hardcover Books (Holt's method)")

## Warning: Ignoring unknown parameters: main



### Excersise 7.6 b.

Compare the RMSE measures of Holt’s method for the two series to those of simple exponential smoothing in the previous question. (Remember that Holt’s method is using one more parameter than SES.) Discuss the merits of the two forecasting methods for these data sets.

# RMSE of Paperback forecast on training data (Holt's method)  
rmse.holt.paper<- sqrt(mean(holt.paper$residuals^2))  
rmse.holt.paper

## [1] 31.13692

# RMSE of Hardcover forecast on training data (Holt's method)  
rmse.holt.hard<-sqrt(mean(holt.hard$residuals^2))  
rmse.holt.hard

## [1] 27.19358

Holt’s method gives more accurate forecast, it has lower RMSE for both series.

SES method is suitable for forecasting data with no clear trend or seasonal pattern to keep model simplier. Whereas Holt’s method allows forecasting of data with a trend.

### Excersise 7.6 c.

Compare the forecasts for the two series using both methods. Which do you think is best?

Holt’s method looks better because it gave more accurate result (lower RMSE) and it was able to capture the upwards trend of the series.

### Excersise 7.6 d.

Calculate a 95% prediction interval for the first forecast for each series, using the RMSE values and assuming normal errors. Compare your intervals with those produced using ses and holt. `

95% interval of Paperback sales:

# holt function  
holt.paper$upper[1, "95%"]

## 95%   
## 275.0205

holt.paper$lower[1, "95%"]

## 95%   
## 143.913

# RMSE values  
holt.paper$mean[1] + 1.96\*rmse.holt.paper

## [1] 270.4951

holt.paper$mean[1] - 1.96\*rmse.holt.paper

## [1] 148.4384

95% interval of Hardcover sales:

# holt function  
holt.hard$upper[1, "95%"]

## 95%   
## 307.4256

holt.hard$lower[1, "95%"]

## 95%   
## 192.9222

# RMSE values  
holt.hard$mean[1] + 1.96\*rmse.holt.hard

## [1] 303.4733

holt.hard$mean[1] - 1.96\*rmse.holt.hard

## [1] 196.8745

95% prediction interval for each series looks very similar regardless of the calculation method used.

## Excersise 7.10

For this exercise use data set ukcars, the quarterly UK passenger vehicle production data from 1977Q1-2005Q1.

### Excersise 7.10.a.

Plot the data and describe the main features of the series.

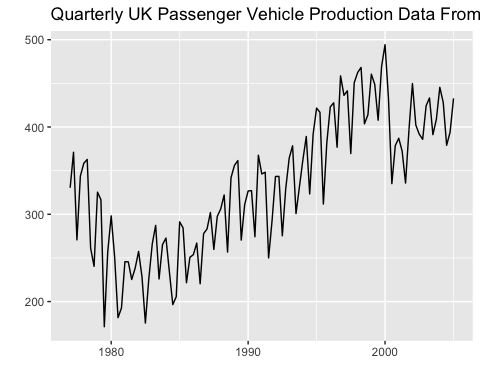
str(ukcars)

## Time-Series [1:113] from 1977 to 2005: 330 371 271 344 358 ...

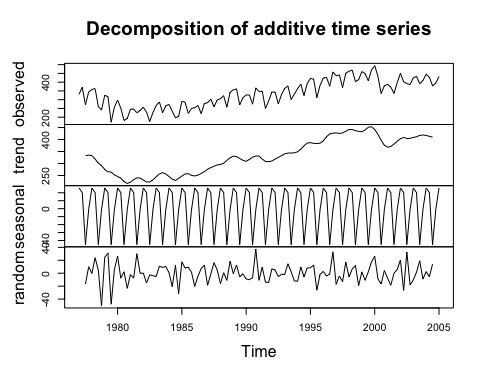
head(ukcars)

## Qtr1 Qtr2 Qtr3 Qtr4  
## 1977 330.371 371.051 270.670 343.880  
## 1978 358.491 362.822

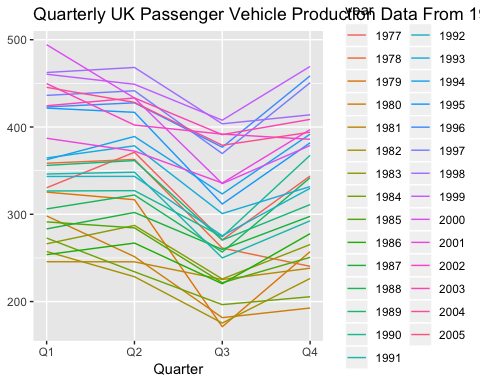
autoplot(ukcars, main = "Quarterly UK Passenger Vehicle Production Data From 1977Q1-2005Q1")



plot(decompose(ukcars))



ggseasonplot(ukcars, main = "Quarterly UK Passenger Vehicle Production Data From 1977Q1-2005Q1")

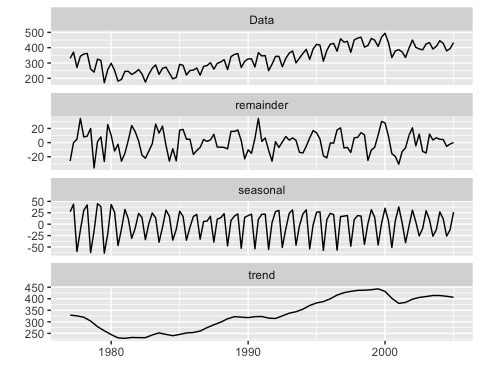


Data has upward trend, seasonal fluctuations (quaterly) and random residuals.

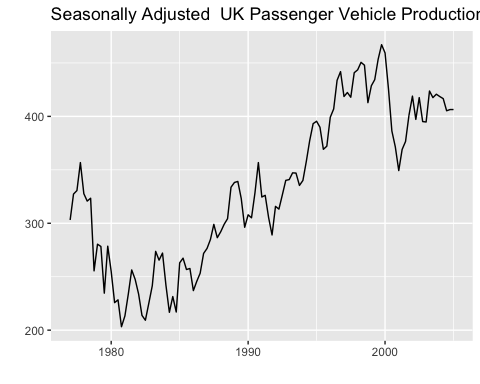
### Excersise 7.10.b.

Decompose the series using STL and obtain the seasonally adjusted data.

# decomposing the series using STL  
ukcars.stl<-stl(ukcars, s.window = 4)  
autoplot(ukcars.stl)



# obtaining seasonally adjusted data  
seasadj.ukcars <-seasadj(ukcars.stl )  
autoplot(seasadj.ukcars, main = "Seasonally Adjusted UK Passenger Vehicle Production Data From 1977-2005")

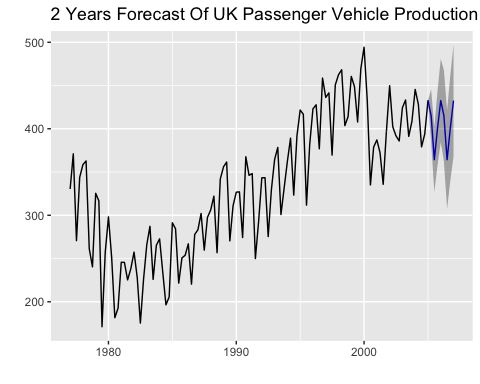


Seasonal adjustment allowed to reduce fluctuations in the data.

### Excersise 7.10.c.

Forecast the next two years of the series using an additive damped trend method applied to the seasonally adjusted data. (This can be done in one step using stlf() with arguments etsmodel=“AAN”, damped=TRUE.)

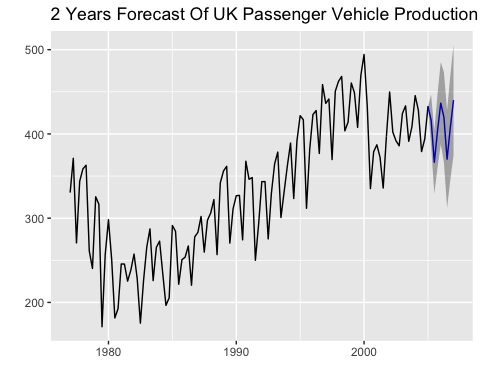
stlf.damped.ukcars<-stlf(ukcars,etsmodel="AAN",damped=TRUE, h=8)  
autoplot(stlf.damped.ukcars, main = "2 Years Forecast Of UK Passenger Vehicle Production (additive damped trend method)")



### Excersise 7.10.d.

Forecast the next two years of the series using Holt’s linear method applied to the seasonally adjusted data (as before but with damped=FALSE).

stlf.ukcars <-stlf(ukcars,etsmodel="AAN",damped=FALSE, h=8)  
autoplot(stlf.ukcars, main = "2 Years Forecast Of UK Passenger Vehicle Production (additive damped trend method)")



### Excersise 7.10.e.

Now use ets() to choose a seasonal model for the data.

ets.ukcars <- ets(ukcars)  
summary(ets.ukcars)

## ETS(A,N,A)   
##   
## Call:  
## ets(y = ukcars)   
##   
## Smoothing parameters:  
## alpha = 0.6199   
## gamma = 1e-04   
##   
## Initial states:  
## l = 314.2568   
## s = -1.7579 -44.9601 21.1956 25.5223  
##   
## sigma: 25.9302  
##   
## AIC AICc BIC   
## 1277.752 1278.819 1296.844   
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 1.313884 25.23244 20.17907 -0.1570979 6.629003 0.6576259  
## ACF1  
## Training set 0.02573334

ets() function does not produce forecasts. It estimates the model parameters and returns information about the fitted model. By default it uses the AICc to select an appropriate model, although other information criteria can be selected. With the ets() function, the default estimation method is maximum likelihood rather than minimum sum of squares.

### Excersise 7.10.f.

Compare the RMSE of the ETS model with the RMSE of the models you obtained using STL decompositions. Which gives the better in-sample fits?

# calculating accuracy of additive damped trend method  
accuracy(stlf.damped.ukcars)

## ME RMSE MAE MPE MAPE MASE  
## Training set 1.551267 23.32113 18.48987 0.04121971 6.042764 0.602576  
## ACF1  
## Training set 0.02262668

# calculating accuracy of Holt's linear method applied to the seasonally adjusted data  
accuracy(stlf.ukcars)

## ME RMSE MAE MPE MAPE MASE  
## Training set -0.3412727 23.295 18.1605 -0.5970778 5.98018 0.5918418  
## ACF1  
## Training set 0.02103582

# calculating accuracy of model based on ets()  
accuracy(ets.ukcars)

## ME RMSE MAE MPE MAPE MASE  
## Training set 1.313884 25.23244 20.17907 -0.1570979 6.629003 0.6576259  
## ACF1  
## Training set 0.02573334

Additive damped trend method and Holt’s linear method applied to the seasonally adjusted data produced better accuracy(lower RMSE). The third model produced higher RMSE because it is build based on maximum likelihood rather than minimum sum of squares.

### Excersise 7.10.g.

Compare the forecasts from the three approaches? Which seems most reasonable?

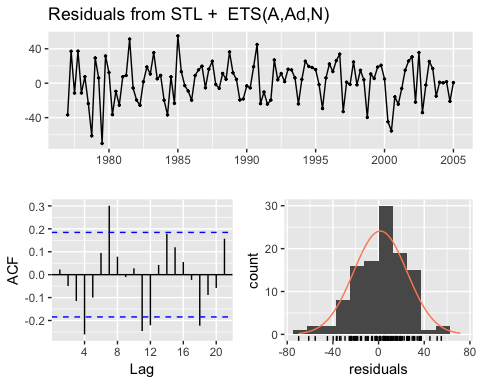
The most reasonable approach seems the first one as it is additive damped trend method applied to the seasonally adjusted data which seems quite robust and gives the lowest RMSE.

### Excersise 7.10.h.

Check the residuals of your preferred model.

checkresiduals(stlf.damped.ukcars)

## Warning in checkresiduals(stlf.damped.ukcars): The fitted degrees of  
## freedom is based on the model used for the seasonally adjusted data.



##   
## Ljung-Box test  
##   
## data: Residuals from STL + ETS(A,Ad,N)  
## Q\* = 24.138, df = 3, p-value = 2.337e-05  
##   
## Model df: 5. Total lags used: 8

The time plot shows some changing variation over time, this heteroscedasticity will potentially make the prediction interval coverage inaccurate. ACF plots shows some autocorrelation in the residuals, because several peaks go beyond the 95% limits defined by the dash blue lines. The residuals look close to normally distributed (slightly skewed). This is not essential for forecasting, but skewness may affect the coverage probability of the prediction intervals.

## Excersise 8.1

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

### Excersise 8.1.a

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

The horizontal axis of an autocorrelation plot shows the size of the lag between the elements of the time series. If a spike (vertical line corresponding to each lag) is significantly different from zero, that is evidence of autocorrelation. A spike that’s close to zero is evidence against autocorrelation. For all the plots the correlations of the lags shown that none of the spikes are larger than the critical value range. It means that data shows no auto-correlation and the above graphs are white noise.

### Excersise 8.1.b

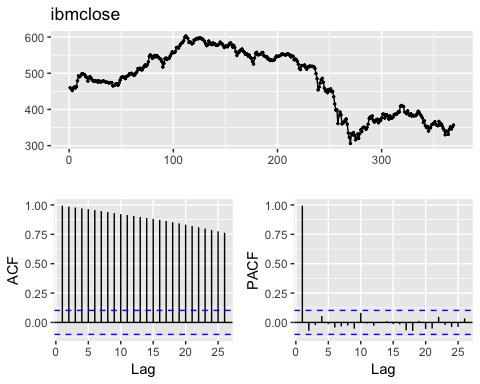
Why are the critical values at different distances from the mean of zero? Why are the autocorrelations different in each figure when they each refer to white noise?

Formula for a critical value is +/- 1.96/(sqrt(T - d)), where T is the sample size. The sample size increases from 36 random numbers to 1000, hence the critical value gets smaller.

## Excersise 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.

library(fpp2)  
ggtsdisplay(ibmclose)



A stationary time series is one whose properties do not depend on the time at which the series is observed. Thus, time series with trends, or with seasonality, are not stationary. IBM stock data is non-stationary for the following reason:

* there is a trend on the time plot of IBM stock prices;
* ACF plot shows that the autocorrelation values are bigger than critical value and decrease slowly. Also, r1 is large(near to 1) and positive. It means that the IBM stock data are non-stationary(that is, predictable using lagged values).
* PACF plot shows that there is a significant correlation at lag 1 followed by correlations that are not significant. This pattern indicates an autoregressive term of order 1. It means that IBM stock data can be predicted by 1 lagged values.

To get stationary data, IBM stock data need differencing. Differencing can help stabilise the mean of a time series by removing changes in the level of a time series, and therefore eliminating (or reducing) trend and seasonality.

## Excersise 8.6

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

### Excersise 8.6.a

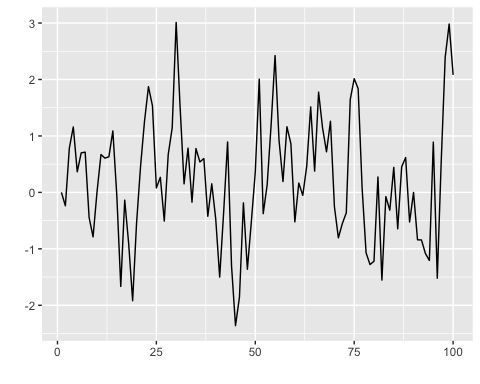
Use the following R code to generate data from an AR(1) model with ϕ1 = 0.6 and σ2 = 1. The process starts with y1=0.

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)  
}

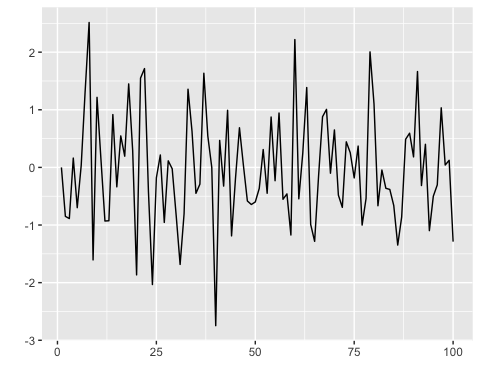
### Excersise 8.6.b.

Produce a time plot for the series. How does the plot change as you change phi1?

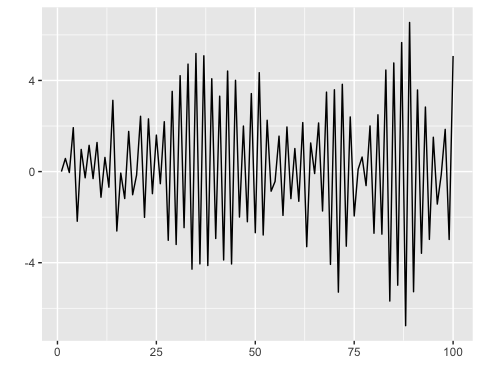
# assigning phi = 0  
autoplot(AR(0.6))



# phi = 0, yt is the equivalent to white noise  
autoplot(AR(0))



# phi <0 oscillate around the mean  
autoplot(AR(-1))



Changing the parameters phi results in different time series patterns. For example:

phi = 0, yt is the equivalent to white noise

phi <0 oscillate around the mean

### Excersise 8.6.c.

Write your own code to generate data from an MA(1) model with θ1=0.6 and σ2=1.

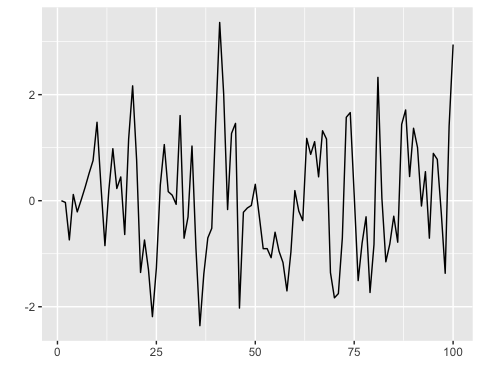
yt=c+εt+θ1εt−1+θ2εt−2+⋯+θqεt−q

MA<- function(theta){  
 y <- ts(numeric(100))  
 e <- rnorm(100)  
 for(i in 2:100){  
 y[i] <- theta\*e[i-1] + e[i]  
 }  
 return(y)  
}

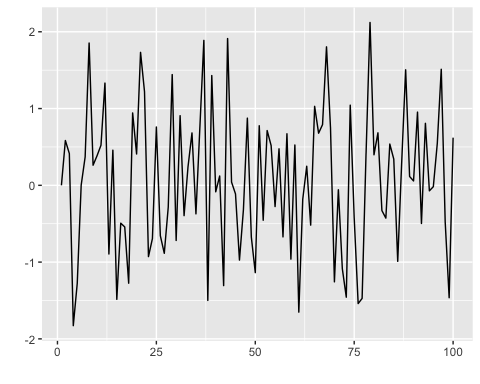
### Excersise 8.6.d.

Produce a time plot for the series. How does the plot change as you change θ1 ?

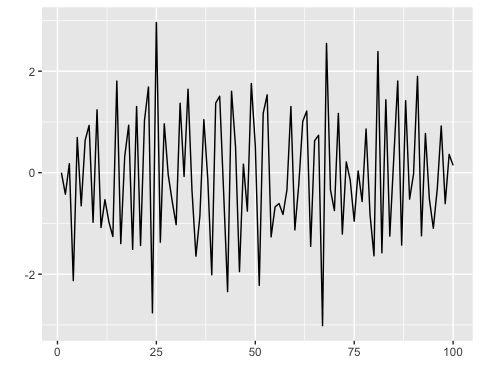
autoplot(MA(0.6))



autoplot(MA(0))



autoplot(MA(-0.6))



Changing the theta results in different time series patterns.

### Excersise 8.6.e.

Generate data from an ARMA(1,1) model with ϕ1=0.6, θ1=0.6 and σ2^=1.

arma1 <- function(n.obs, phi, theta, seed.nr){  
 set.seed(seed.nr)  
 y <- ts(numeric(n.obs))  
 e <- rnorm(n.obs)  
 for (i in 2:n.obs)  
 y[i] <- phi\*y[i-1] + theta\*e[i-1] + e[i]  
 return(y)  
}

### Excersise 8.6.f.

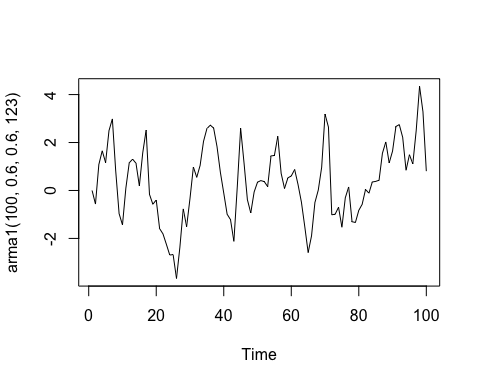
Generate data from an AR(2) model with ϕ1 = −0.8, ϕ2 = 0.3 and σ 2=1 . (Note that these parameters will give a non-stationary series.)

ar2 <- function(n.obs, phi1, phi2, seed.nr){  
 set.seed(seed.nr)  
 y <- ts(numeric(n.obs))  
 e <- rnorm(n.obs)  
 for(i in 3: n.obs)  
 y[i] <- phi1\*y[i-1] + phi2\*y[i-2] + e[i]  
 return(y)  
}

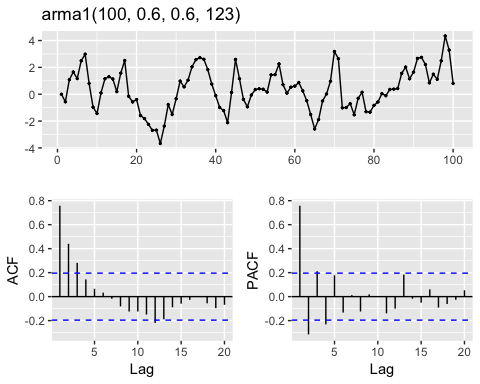
### Excersise 8.6.g.

Graph the latter two series and compare them.

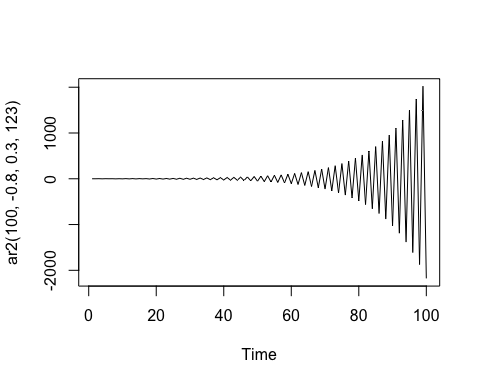
plot(arma1(100, 0.6, 0.6, 123))



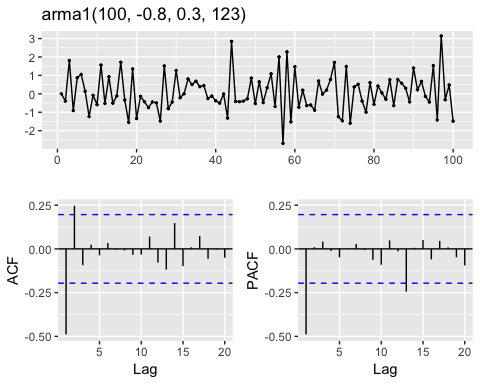
ggtsdisplay(arma1(100, 0.6, 0.6, 123))



plot(ar2(100, -0.8, 0.3, 123))



ggtsdisplay(arma1(100, -0.8, 0.3, 123))



ARMA(1, 1) model looks stationary.

* there is no a clear trend
* ACF plot shows that just several values are bigger than critical value and decrease quickly.
* PACF plot shows that there is a significant correlation at lag 1, but other lags are also quite strongly correlated.

AR(2) model increased with oscillation and non-statutory.

### Excersise 8.8

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

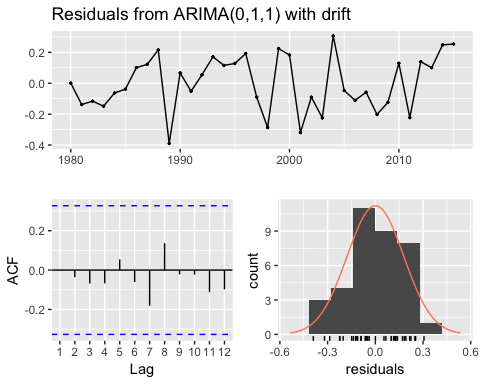
### Excersise 8.8.a

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.

fit<-auto.arima(austa)  
summary(fit)

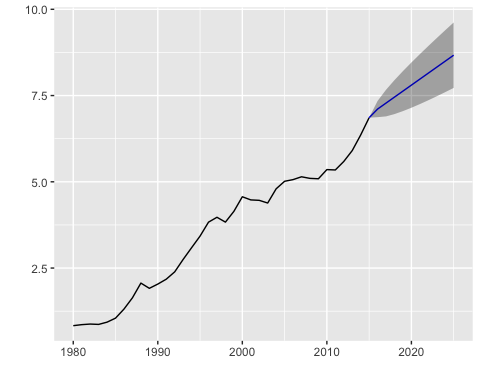
## Series: austa   
## ARIMA(0,1,1) with drift   
##   
## Coefficients:  
## ma1 drift  
## 0.3006 0.1735  
## s.e. 0.1647 0.0390  
##   
## sigma^2 estimated as 0.03376: log likelihood=10.62  
## AIC=-15.24 AICc=-14.46 BIC=-10.57  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE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

# checking the residuals  
checkresiduals(fit, main = "(ARIMA(0,1,1) model residuals")



##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(0,1,1) with drift  
## Q\* = 2.297, df = 5.2, p-value = 0.8266  
##   
## Model df: 2. Total lags used: 7.2

forecast\_1<-forecast(fit, h=10)   
autoplot(forecast\_1)



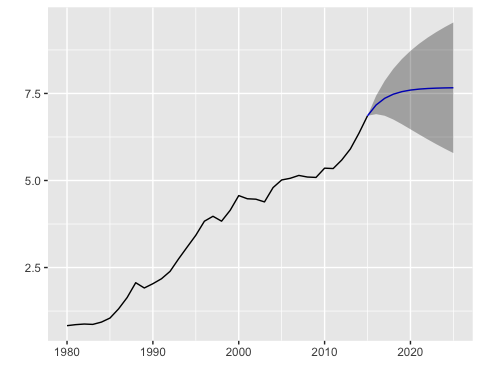
ARIMA(0,1,1) model was selected.

Residuals are roughly normal and approximately independently distributed with a mean of 0 and some constant variance. We can conclude that assumptions are reasonable and the choice of model is appropriate.

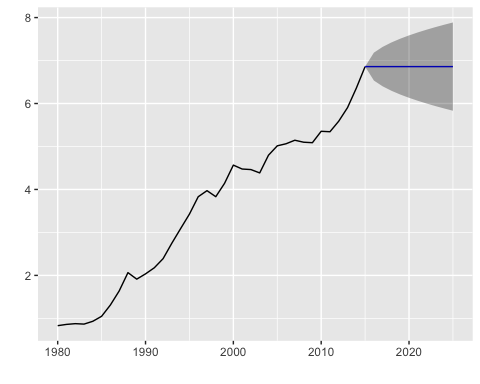
### Excersise 8.8.b

Plot forecasts from an ARIMA(0,1,1) model with no drift and compare these to part a.

# ploting without drift  
fit2<-auto.arima(austa, allowdrift = F)  
forecast\_2<-forecast(fit2, h=10)   
autoplot(forecast\_2)



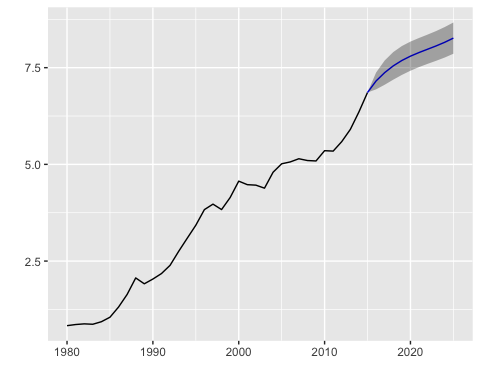
# remove the MA term and plot again.  
  
fit2a<-Arima(austa, order=c(0,1,0), include.drift = F)  
forecast\_2a<-forecast(fit2a,h=10)  
autoplot(forecast\_2a)



### Excersise 8.8.c

Plot forecasts from an ARIMA(2,1,3) model with drift.

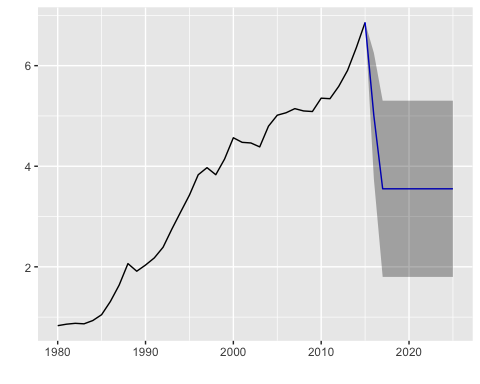
# ploting with drift  
fit3<-Arima(austa, order=c(2,1,3), include.drift = TRUE)  
forecast\_3<-forecast(fit3,h=10)  
autoplot(forecast\_3)



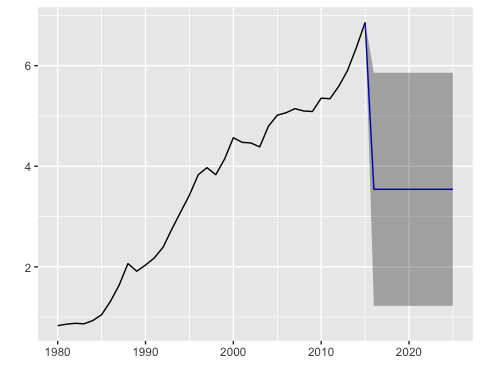
### Excersise 8.8.d

Plot forecasts from an ARIMA(0,0,1) model with a constant.

fit4<-Arima(austa, order=c(0,0,1), include.constant = TRUE)  
forecast\_4<-forecast(fit4,h=10)  
autoplot(forecast\_4)



# remove the MA term and plot again  
  
fit4a<-Arima(austa, order=c(0,0,0), include.constant = TRUE)  
forecast\_4a<-forecast(fit4a,h=10)  
autoplot(forecast\_4a)

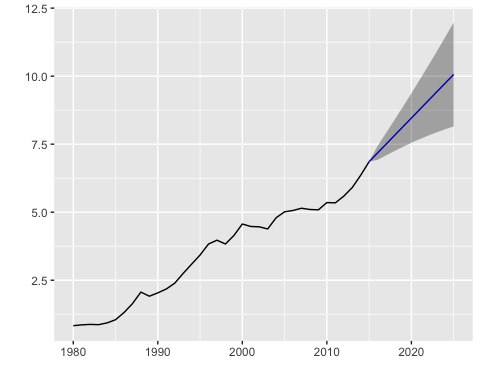


An ARIMA(0,0,0) model only contains a constant and white noise.

### Excersise 8.8.e

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

fit5<-Arima(austa, order=c(0,2,1), include.constant = FALSE)  
forecast\_5<-forecast(fit5,h=10)  
autoplot(forecast\_5)



A model with d=0 and no constant will have forecasts that converge to 0 .

A model with d=0 and a constant will have forecasts that converge to the mean of the data.

A model with d=1 and no constant will have forecasts that converge to a non-zero value close to the last observation.

A model with d=1 and a constant will have forecasts that converge to a linear function with slope based on the whole series.

A model with d=2 and no constant will have forecasts that converge to a linear function with slope based on the last few observations.

The value of d also has an effect on the prediction intervals — the higher the value of d, the more rapidly the prediction intervals increase in size.