Data624 HW1

Samuel Kigamba, Lin Li, Patrick Maloney, Daniel Moscoe, David Moste

6/27/2021

# HA 2.1

#### Question

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

#### Code

library(forecast)

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

# Use help function to explore the three datasets  
??gold

## starting httpd help server ...

## done

??woolyrnq  
??gas

#### Response

“gold” is a time series object that contains daily morning gold prices in US dollars from January 1 of 1985 through March 31 of 1989.

“woolyrng” is a time series object ofquartely production of woollen yarn in Australia in tonnes from March of 1965 to September of 1994.

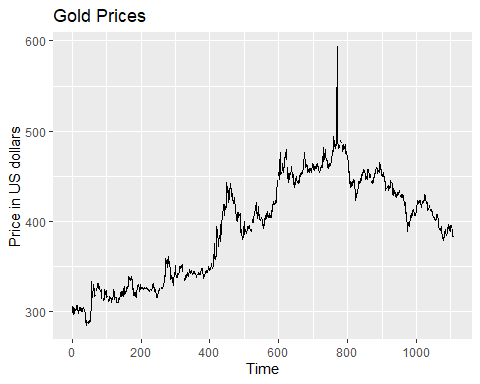
“gas” is a time series object of Australian monthly gas production between 1956 and 1995.

#### Question

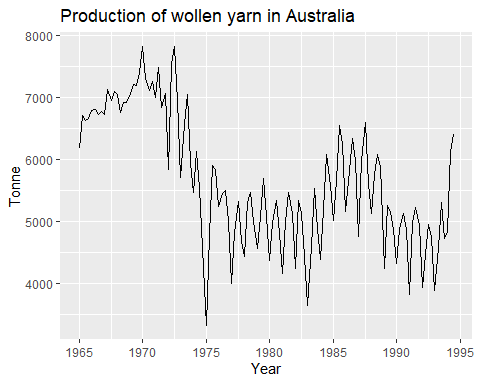
Use autoplot() to plot each of these in separate plots.

#### Code

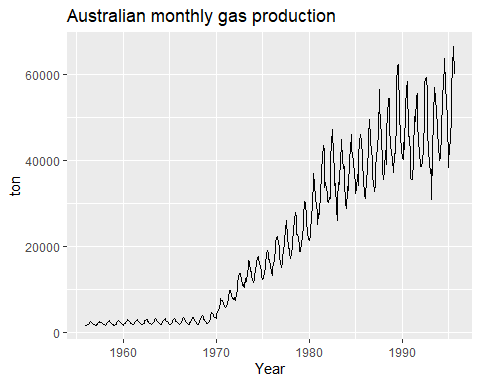
library(forecast)  
library(ggplot2)  
  
# Generate autoplot for "gold"  
autoplot(gold) +   
 ggtitle("Gold Prices") +  
 xlab("Time") +  
 ylab("Price in US dollars")



# Generate autoplot for "woolyrnq"  
autoplot(woolyrnq) +  
 ggtitle("Production of wollen yarn in Australia") +  
 xlab("Year") +  
 ylab("Tonne")



# Generate autoplot for "gas"  
autoplot(gas) +  
 ggtitle("Australian monthly gas production") +  
 xlab("Year") +  
 ylab("ton")



#### Question

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

#### Code

# Get frequency for each series  
frequency(gold)

## [1] 1

frequency(woolyrnq)

## [1] 4

frequency(gas)

## [1] 12

#### Response

The frequency is annual, quarterly, and monthly for gold, woolyrnq and gas, respectively.

#### Question

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

#### Code

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

## [1] 770

#### Response

The outlier is at trading day 770. we can also see a peak in the plot at around day 770.

# HA 2.2

#### Question

Download the file tute1.csv from the book website, open it in Excel (or some other spreadsheet application), and review its contents. You should find four columns of information. Columns B through D each contain a quarterly series, labelled Sales, AdBudget and GDP. Sales contains the quarterly sales for a small company over the period 1981-2005. AdBudget is the advertising budget and GDP is the gross domestic product. All series have been adjusted for inflation.

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

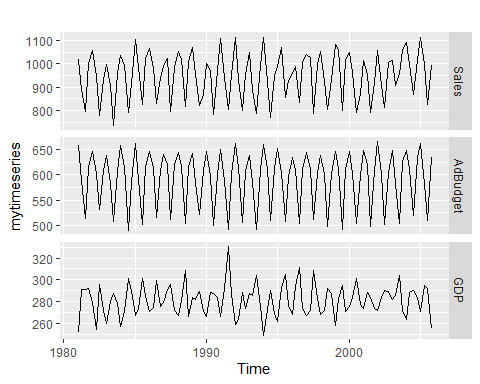
# Get tute1 file  
tute1 <- read.csv("https://raw.githubusercontent.com/nealxun/Forecasting\_Principle\_and\_Practices/master/extrafiles/tute1.csv")  
View(tute1)

1. Convert the data to time series

mytimeseries <- ts(tute1[,-1], start=1981, frequency=4)

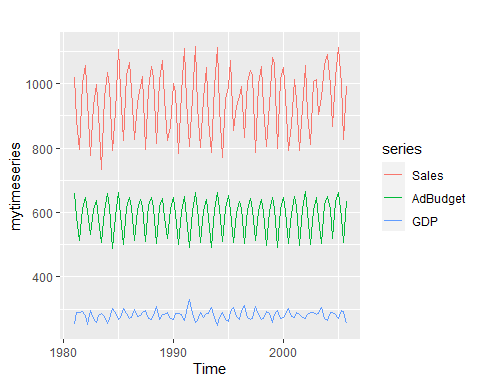
1. Construct time series plots of each of the three series

autoplot(mytimeseries, facets=TRUE)



Check what happens when you don’t include facets = TRUE.

# Plot without facets = TRUE  
autoplot(mytimeseries)



#### Response

The facets = True commend spits the plot into individual time series plots. Without using the function, the data get plotted as a single plot on a single axes.

# HA 2.3

#### Question

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.

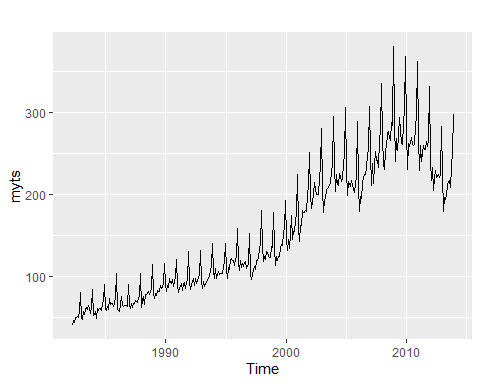
1. You can read the data into R with the following script:

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

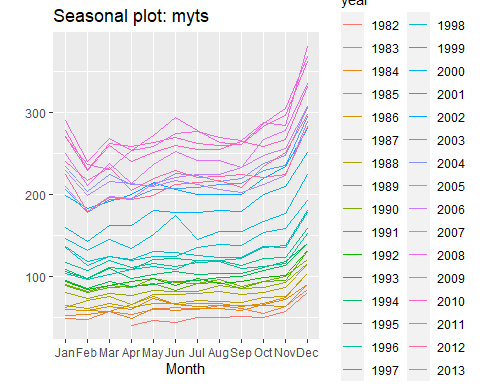
1. Select one of the time series…
2. Explore your chosen retail time series using the following functions: autoplot(), ggseasonplot(), ggsubseriesplot(), gglagplot(), ggAcf(). Can you spot any seasonality, cyclicity and trend? What do you learn about the series?

#### Code

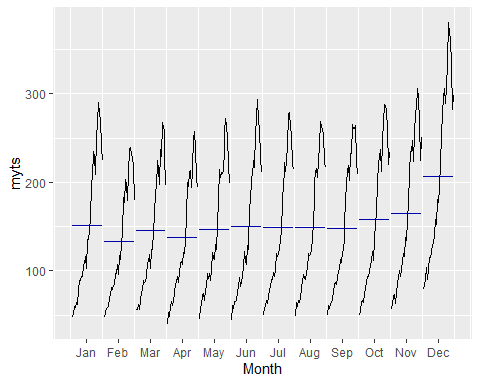
myts <- ts(retaildata[, "A3349503T"],  
 frequency = 12, start = c(1982,4))  
autoplot(myts)



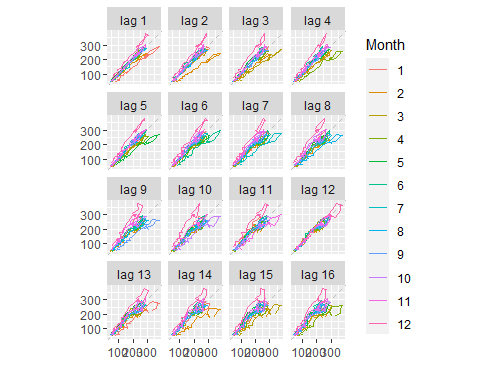
ggseasonplot(myts)



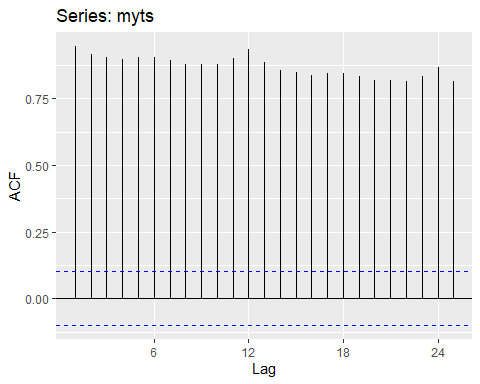
ggsubseriesplot(myts)



gglagplot(myts)



ggAcf(myts)



#### Response

Autoplot() reveals trend and seasonality in the data. From 1982 to 2009, there is an increasing trend that then reverses through 2014. The data exhibit strong yearly seasonality, with the data spiking at the end of each year and then sharply declining. Based on this plot, there is no evidence of cyclicity, although the reversal in trend at 2009 may be part of a cycle with a period greater than the time range of this data.

ggseasonplot shows the increasing trend through 2009, although this plot makes it difficult to see that the trend reverses for 2009-2014. The plot shows the strong spike at the end of each year.

The subseries plot again makes evident the strong seasonality of the data, with a yearly maximum occurring in December, followed by a sharp decline. Perhaps this time-series represents a product or sector that experiences great demand during the holiday season. This plot does a better job than the season plot of showing the reversal in trend that begins in 2009. The blue bars on the plot indicate the mean value for each month. While June through September of each year have roughly constant means, October through April exhibit a rise to the December spike, followed by a decline.

The lag plots support the claim of yearly seasonality, because the plot for lag 12 shows each month’s data tightly clustered around the diagonal. That means measurements within each month tend to be very close to measurements from 1 year prior. Other lag plots still show months centered at the diagonal. But the data in these plots is more widely dispersed, indicating that lags other than 12 months show greater differences between the current and lagged data. A visual inspection suggests that the plot for lag 16 exhibits the greatest spread about the diagonal. The curve for January lies entirely above the diagonal, indicating that January measurements are always greater than those from September of two calendar years prior. The reverse is true for the curve for April, which lies entirely below the diagonal. April measurements are always less than those from December of two calendar years prior.

The autocorrelation plot also shows evidence of seasonality and trend. Lags of multiples of 12 months are peaked, indicating that measurements are most strongly correlated with those of the same month from earlier years. All the autocorrelation coefficients are statistically significant and positive, indicating that these relationships are very likely not the result of mere randomness in the data. There is a general downward trend in the autocorrelation coefficients, indicating that data are most strongly correlated with their recent predecessors.

# HA 6.2

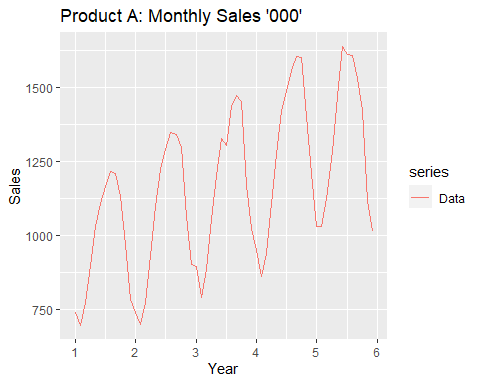
The plastics data set consists of the monthly sales (in thousands) of product A for a plastics manufacturer for five years.

#### Question

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

#### Code

library(fma)  
  
p <- plastics  
autoplot(p, series="Data") + xlab("Year") + ylab("Sales") + ggtitle("Product A: Monthly Sales '000'")



#### Response

The plot above shows that there is a seasonal nature to the plastics data set. It appears that the seasonality is annual will sales peaking at mid year. There trend cycle appears to be positive with strong sales growth every year.

#### Question

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

#### Code

library(fma)  
  
p <- plastics  
  
# Decompose plastics data  
m = decompose(p, type="multiplicative")  
# Calculating trend-cycle  
print(m$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

# Calculating seasonal indices  
print(m$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

# Calculating remainder component  
print(m$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

#### Response

See output above.

Note that the estimate of the trend-cycle is unavailable for the first 6 and last 6 observations and as a result there is also no estimate of the remainder component for the same period.

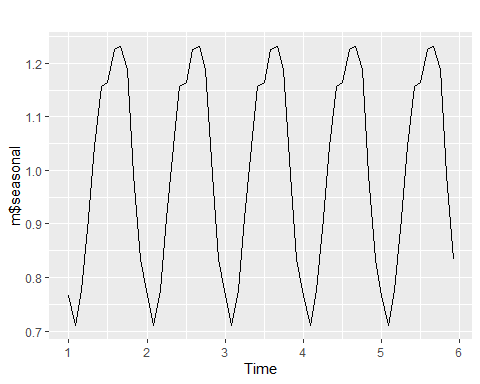
This is a common problem with the classical decomposition.

#### Question

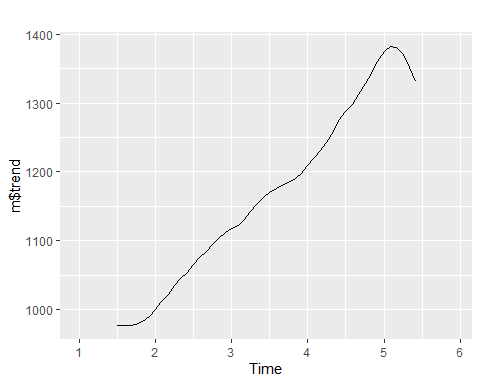
1. Do the results support the graphical interpretation from part a?

#### Code

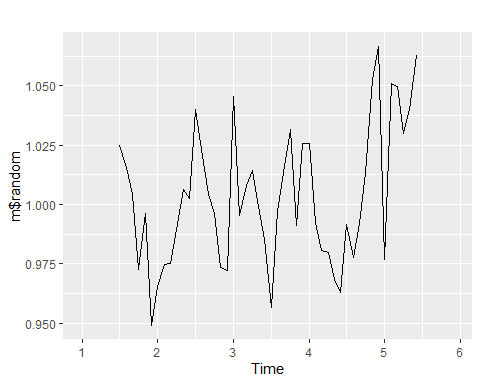
# Plot seasonal indices  
autoplot(m$seasonal)



# plot trend-cycle  
autoplot(m$trend)



# plot trend-cycle  
autoplot(m$random)



#### Response

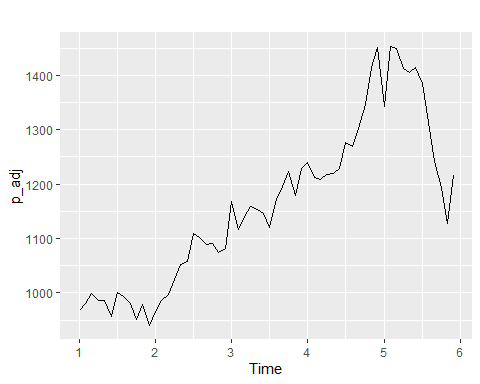
The plot of seasonal decomposition shows a clear pattern of annual seasonality while the trend-cycle plot shows a strong positive trend of growth in sales over the 5 year period. This infact confirms the conclusion made under graph number 1.

#### Question

1. Compute and plot the seasonally adjusted data.

#### Code

# Seasonal adjustment of multiplicative decomposition  
p\_adj <- p/m$seasonal  
autoplot(p\_adj)

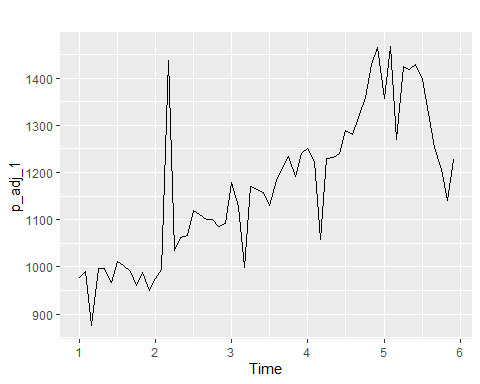


#### Question

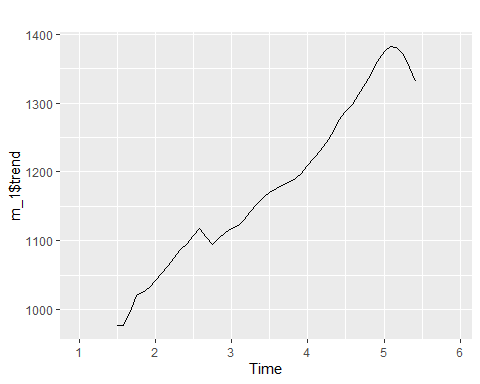
1. Change one observation to be an outlier (e.g., add 500 to one observation), and recompute the seasonally adjusted data. What is the effect of the outlier?

#### Code

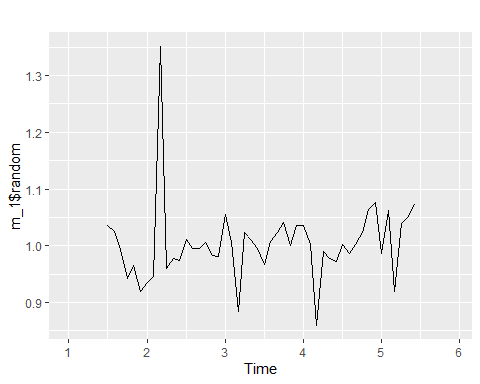
p[15] <- p[15] + 500  
m\_1 <- decompose(p, type = "multiplicative")  
p\_adj\_1 <- p/m\_1$seasonal  
autoplot(p\_adj\_1)



autoplot(m\_1$trend)



autoplot(m\_1$random)



#### Response

The addition of an outlier to the data impacts decomposition because the classical decomposition methods (multiplicative in this case) are unable to capture these seasonal changes over time.

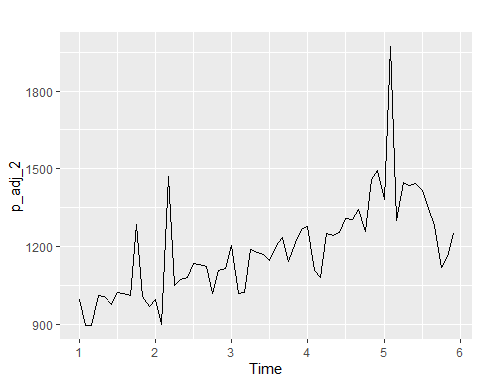
The trend-cycle also tends to oversmooth this rapid rise in the data. This is an inherent weakness of this method.

#### Question

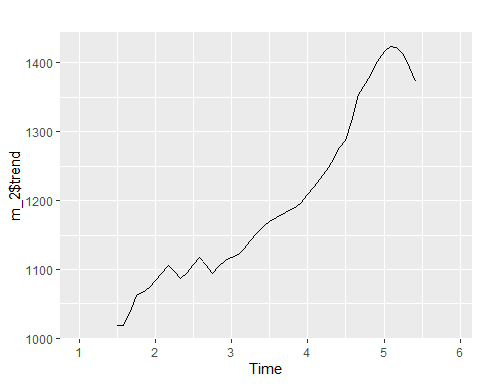
1. Does it make any difference if the outlier is near the end rather than in the middle of the time series?.

#### Code

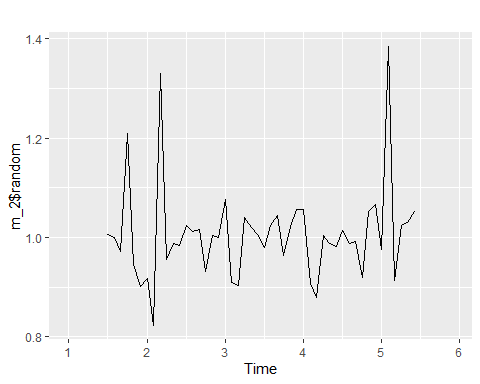
p[10] <- p[10] + 500  
p[50] <- p[50] + 500  
m\_2 <- decompose(p, type = "multiplicative")  
p\_adj\_2 <- p/m\_2$seasonal  
autoplot(p\_adj\_2)



autoplot(m\_2$trend)



autoplot(m\_2$random)



#### Response

The position of the outlier does not seem to make any difference in the decomposition since the classical methods are unable to capture these seasonal changes over time.

Its also evident that the trend-cycle tend to over-smooth the two rapid rises in the data as a result of the two introduced outliers.

# KJ 3.1

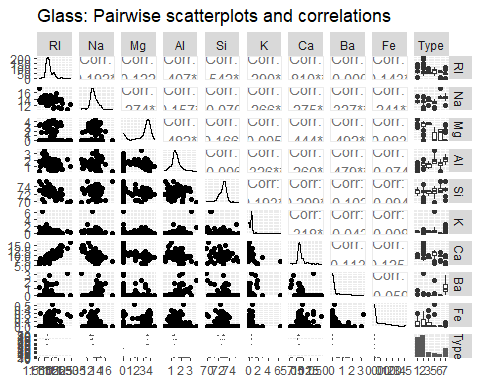
#### Question

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.

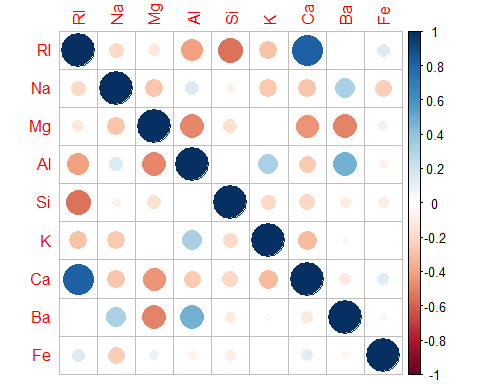
1. Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors.

#### Code

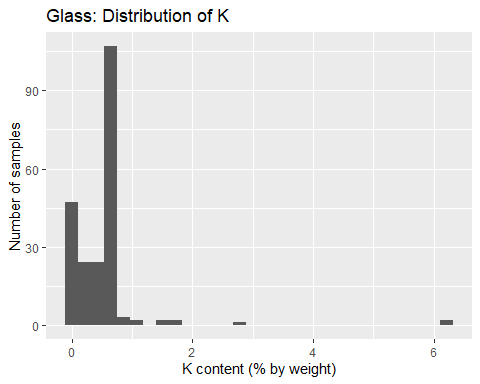
library(ggplot2)  
library(GGally)  
library(corrplot)  
library(mlbench)  
  
data(Glass)  
  
ggpairs(Glass) +  
 ggtitle("Glass: Pairwise scatterplots and correlations")



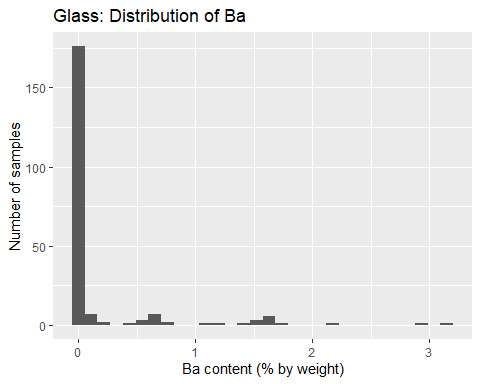
corrplot::corrplot(cor(Glass[1:9]))



ggplot(data = Glass, aes(x = K)) +  
 geom\_histogram() +  
 ggtitle("Glass: Distribution of K") +  
 xlab("K content (% by weight)") +  
 ylab("Number of samples")



ggplot(data = Glass, aes(x = Ba)) +  
 geom\_histogram() +  
 ggtitle("Glass: Distribution of Ba") +  
 xlab("Ba content (% by weight)") +  
 ylab("Number of samples")



#### Response

A visual exploration of our data gives us a quick overview of the distributions of each of the nine variables, as well as how they correlate with each other. The pairwise scatterplots and correlations above, Na, Al, and Si appear approximately normally distributed, with minimal to moderate skew. RI, Mg, and Ca demonstrate significant skew, and may require transformations that mitigate skewness. K, Ba, and Fe appear to be comprised of a large number of low- or zero-values, along with a small number of larger or outlying vales.

There is a clear linear relationship between the amount of Ca in a sample and its Reflective Index, with a correlation coefficient of 0.81. Si and RI have a correlation coefficient of -0.542. Both these linear correlations are clearly visible in the scatterplots for these pairs of variables. In order to avoid collinearity, it may make sense to drop either Ca or RI from the forecast, given the high correlation between the two.

After taking a deeper dive into the variables that don’t follow a normal distribution, we see that the variable K appears bimodally distributed around 0 and 0.6, when outliers are excluded. We also see that Ba is a low-variance variable, with only 38 of 214 samples having a non-zero value.

#### Question

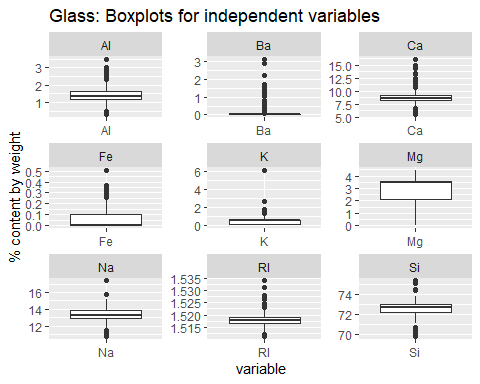
1. Do there appear to be any outliers in the data? Are any predictors skewed?

#### Code

library(ggplot2)  
library(mlbench)  
library(e1071)  
library(tidyr)  
  
skewness <- apply(Glass[,1:9], 2, skewness)  
skewness

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

Glass\_features <- Glass[,1:9] %>%  
 gather(key = 'variable', value = 'value')  
  
ggplot(Glass\_features, aes(variable, value)) +  
 geom\_boxplot() +  
 facet\_wrap(. ~ variable, scales = 'free') +  
 ggtitle("Glass: Boxplots for independent variables") +  
 xlab("variable") +  
 ylab("% content by weight")



#### Response

Yes, the data contains both outliers and skewness. The skewness table shows us that all the variables contain some degree of skewness, but K, Ca, and Ba are significantly skewed (above a 2.0 threshold). These will need to be transformed. Fe comes in below the threshold, but contains many zero values and also would be suitable for transformation.

According to the boxplots, all the variables contain statistical outliers except Mg. K contains the most significant outlier by far, which should definitely be excluded. Ba, Fe, and K are especially noteworthy, because they combine significant skewness with a large number of outlying values.

#### Question

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

#### Response

Centering and scaling would be an appropriate approach given the orders of magnitude between Si and Fe and Ba. Most of the variables would be suitable for a Box-Cox transformation, but especially Ba, Fe, and K. Also, it may be worth considering excluding Ca from the analysis, given its high collinearity with RI, and high level of skewness.

# KJ 3.2

#### Question

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.

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

#### Code

library(mlbench)  
library(caret)  
  
data(Soybean)  
nzv <- nearZeroVar(Soybean, saveMetrics = TRUE)  
nzv[nzv[,"nzv"] == TRUE,]

## freqRatio percentUnique zeroVar nzv  
## leaf.mild 26.75 0.4392387 FALSE TRUE  
## mycelium 106.50 0.2928258 FALSE TRUE  
## sclerotia 31.25 0.2928258 FALSE TRUE

#### Response

Leaf.mild, myselium, and scleorota all have high frequency ratios (more than 20), combined with a low percentage of unique values vs. the overall samples (less than 10%).

#### Question

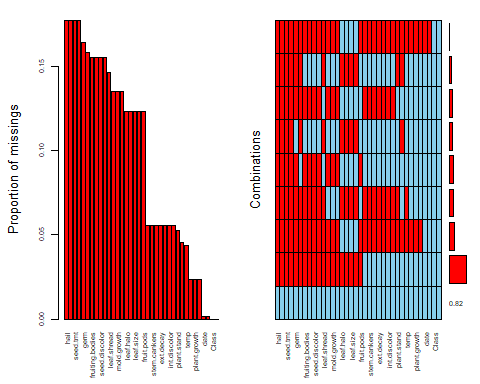
1. Roughly 18% of the data are missing. Are there particular predictors that are more likely to be missing? Is the pattern of missing data related to the classes?

#### Code

library(mlbench)  
library(dplyr)  
library(VIM)  
  
soybean\_na <- sapply(Soybean, function(x) sum(is.na(x)))  
soybean\_na <- data.frame(soybean\_na)  
arrange(soybean\_na, -soybean\_na)

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

par(mfrow = c(1,3))  
aggr(Soybean, only.miss=TRUE, sortVars=TRUE)



##   
## 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

#### Response

The predictors most likely to be missing are hail, sever, seed.tmt, and lodging, each with 121 missing values. As we can see from the missing values plot, there seems to be a pattern with several variables containing the same amount of missing values. For example,in the summation table all variables beginning with leaf. exhibit the same number of missing values, 84. And a group of variables that seems to relate to plant diseases all share the same number of missing values, 38. These variables are stem.cankers, canker.lesion, ext.decay, mycelium, int.discolor, sclerotia. This suggests that perhaps there was an issue with certain facets of the data not being measured or uploaded at the same time as other features.

#### Question

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

#### Response

With 18 percent of the data containing missing values, simply ignoring missing values would result in too much data loss. Also, given the patterns shown in how the data is missing, it makes sense that there could be some meaning in the way the values are missing. Imputation is likely the best way forward, and since the variables are categorical, KNN is the best option. However, it most be noted that some of the variables have large class-imbalance issues, which could affect the imputation accuracy.

# HA 7.1

#### Question

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

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

#### Code

library(forecast)  
library(fma)  
  
pigs <- fma::pigs  
pigs\_ses <- ses(pigs, h = 4, level = 95)  
summary(pigs\_ses)

##   
## Forecast method: Simple exponential smoothing  
##   
## Model Information:  
## Simple exponential smoothing   
##   
## Call:  
## ses(y = pigs, h = 4, level = 95)   
##   
## Smoothing parameters:  
## alpha = 0.2971   
##   
## Initial states:  
## l = 77260.0561   
##   
## sigma: 10308.58  
##   
## AIC AICc BIC   
## 4462.955 4463.086 4472.665   
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set 385.8721 10253.6 7961.383 -0.922652 9.274016 0.7966249 0.01282239  
##   
## Forecasts:  
## Point Forecast Lo 95 Hi 95  
## Sep 1995 98816.41 78611.97 119020.8  
## Oct 1995 98816.41 77738.83 119894.0  
## Nov 1995 98816.41 76900.46 120732.4  
## Dec 1995 98816.41 76092.99 121539.8

#### Response

Output from the ses() function tells us that , and .

As approaches , the effect of earlier measurements on the forecast value decreases. This moderate value of indicates that earlier measurements are important contributors to the forecasted value.

The point forecast for future values is 98816.41.

#### Question

1. Compute a 95% prediction interval for the first forecast using , where is the standard deviation of the residuals. Compare your interval with the interval produced by R.

#### Code

int\_width <- 1.96 \* sd(pigs\_ses$residuals)  
low\_bound <- 98816.41 - int\_width  
up\_bound <- 98816.41 + int\_width  
low\_bound

## [1] 78679.97

up\_bound

## [1] 118952.8

#### Response

Based on the standard deviation of the residuals, a 95% prediction interval for the first forecast is (78,679.97, 118,952.8). The interval produced by R is (78,611.97, 119,020.8). My interval is slightly narrower: it uses the standard deviation of the residuals, which is smaller than the value reported as sigma by the ses() function.

# HA 7.2

#### Question

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()?

#### Code

library(fma)  
  
exp\_smooth\_memo <- memoise::memoise(function(y, alpha, level) {  
 if(length(y) == 0) {  
 return(level)  
 } else {  
 return(alpha \* tail(as.vector(y), 1) + (1 - alpha) \* exp\_smooth\_memo(head(as.vector(y), -1), alpha, level))  
 }  
})  
pigs\_forecast <- exp\_smooth\_memo(pigs, 0.2971, 77260.06)  
pigs\_forecast

## [1] 98816.45

#### Response

This function does return almost exactly the same value as ses().

# HA 7.3

#### Question

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?

#### Code

sum\_sq\_err\_memo\_for\_optim <- function(par = c(alpha, level), y){  
 accumulator <- 0  
 for(i in seq(0, length(y) - 1)) {  
 #print(par[2])  
 accumulator <- accumulator + (y[i + 1] - exp\_smooth\_memo(y[0:i], par[1], par[2]))^2  
 }  
 return(accumulator)  
}  
optim <- optim(par = c(0.5, 50000), fn = sum\_sq\_err\_memo\_for\_optim, y = pigs)  
optim$par

## [1] 0.297214 77263.093727

#### Response

The values returned from optim() are almost identical to those returned by the ses() function.

# HA 8.1

#### Question

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

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

#### Response

The series have different critical values and different autocorrelations. All of these plots indicate white noise since less than 95% of the autocorrelations are below the critical values in each plot and there is no pattern. This is all consistent with white noise.

#### Question

1. 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?

#### Response

The critical values are different because the number of values in each set are different. The shorter time series will have wider ranges for the critical values while the opposite is true for longer time series.

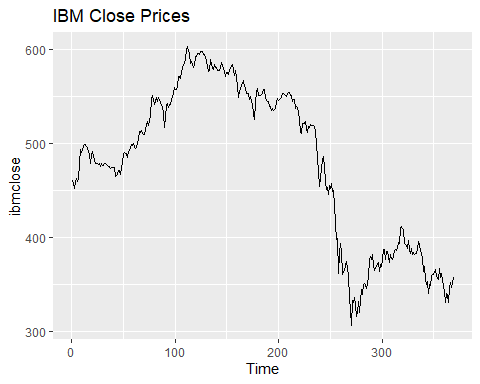
# HA 8.2

#### Question

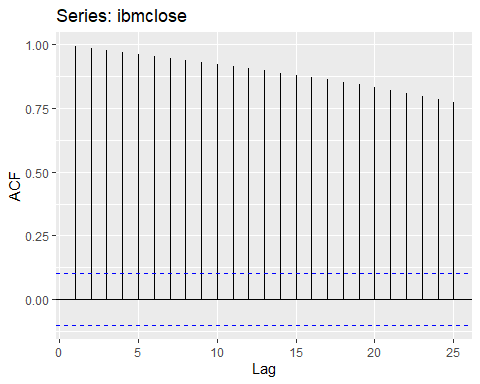
A classic example of a non-stationary series is the daily closing IBM stock price series (data set ibmclose). Use R to plot the daily closing prices for IBM stock and the ACF and PACF. Explain how each plot shows that the series is non-stationary and should be differenced.

#### Code

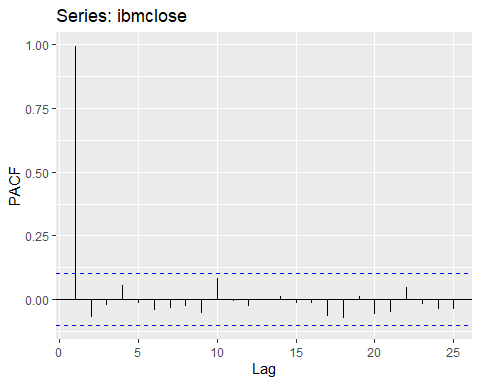
library(forecast)  
library(fma)  
library(ggplot2)  
  
autoplot(ibmclose) +  
 ggtitle("IBM Close Prices")



ggAcf(ibmclose)



ggPacf(ibmclose)



#### Response

The plot of the ibmclose data show clear trends, which immediately indicates the data is non-stationary and should be differenced.

The ACF plot shows that the data is not stationary because the autocorrelations for the lags are well above the critical value for every single lag. This indicates the each value depends on previous values. The values are also decreasing slowly, which indicates there is a trend in the data (not stationary).

# HA 8.6

#### Quesstion

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

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

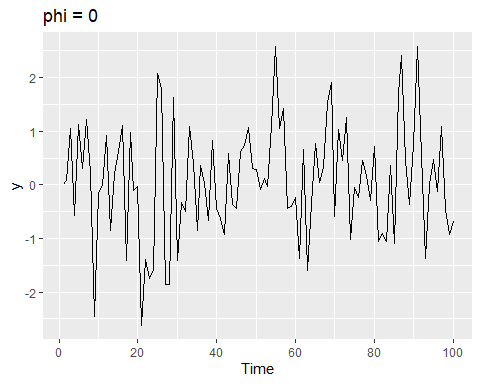
y <- ts(numeric(100))  
e <- rnorm(100)  
  
for(i in 2:100){  
 y[i] <- 0.6\*y[i-1] + e[i]  
}

#### Question

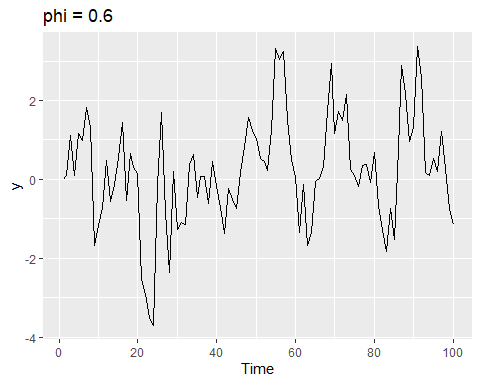
1. Produce a time plot for the series. How does the plot change as you change ?

#### Code

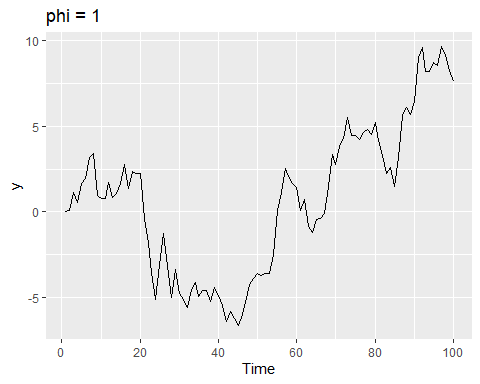
library(ggplot2)  
  
y <- ts(numeric(100))  
e <- rnorm(100)  
  
for(i in 2:100){  
 y[i] <- 0\*y[i-1] + e[i]  
}  
  
autoplot(y) +  
 ggtitle("phi = 0")



for(i in 2:100){  
 y[i] <- 0.6\*y[i-1] + e[i]  
}  
  
autoplot(y) +  
 ggtitle("phi = 0.6")



for(i in 2:100){  
 y[i] <- 1\*y[i-1] + e[i]  
}  
  
autoplot(y) +  
 ggtitle("phi = 1")



#### Response

When is low, the data is just noise, when is close to 1, the series nears a random walk.

#### Question

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

#### Code

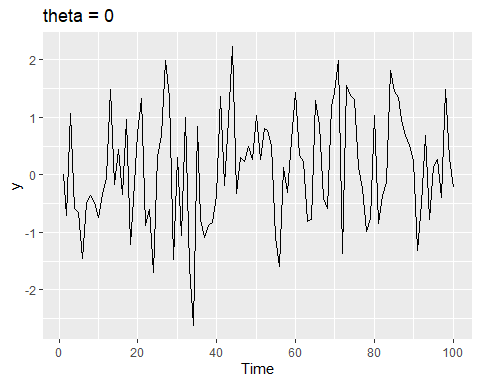
y <- ts(numeric(100))  
e <- rnorm(100)  
for(i in 2:100){  
 y[i] <- 0.6\*e[i-1] + e[i]  
}

#### Question

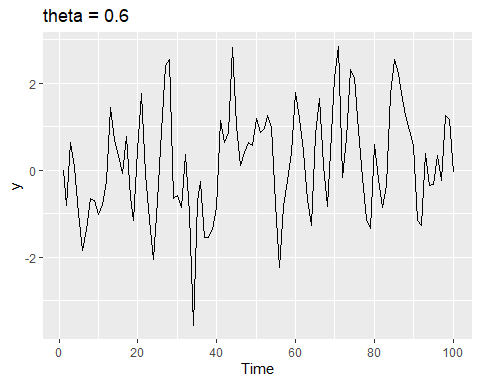
1. Produce a time plot for the series. How does the plot change as you change ?

#### Code

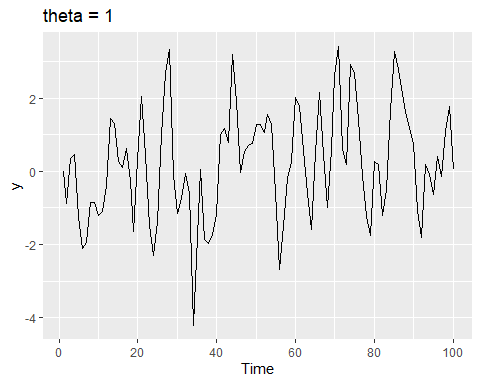
library(forecast)  
library(ggplot2)  
  
y <- ts(numeric(100))  
e <- rnorm(100)  
  
for(i in 2:100){  
 y[i] <- 0\*e[i-1] + e[i]  
}  
  
autoplot(y) +  
 ggtitle("theta = 0")



for(i in 2:100){  
 y[i] <- 0.6\*e[i-1] + e[i]  
}  
  
autoplot(y) +  
 ggtitle("theta = 0.6")



for(i in 2:100){  
 y[i] <- 1\*e[i-1] + e[i]  
}  
  
autoplot(y) +  
 ggtitle("theta = 1")



#### Response

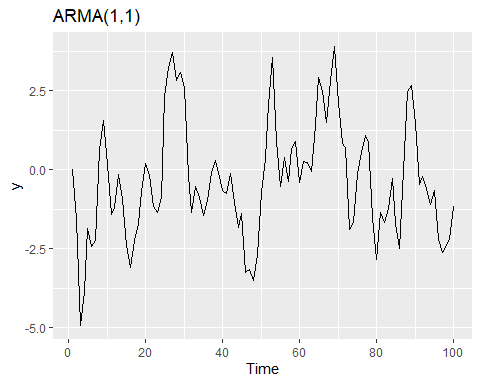
There isn’t much change due to theta, which makes sense since this is a simulation of white noise.

#### Question

1. Generate data from an ARIMA(1,1) model with .

#### Code

library(ggplot2)  
  
y <- ts(numeric(100))  
e <- rnorm(100)  
  
for(i in 2:100){  
 y[i] <- 0.6\*y[i-1] + 0.6\*e[i-1] + e[i]  
}  
  
autoplot(y) +  
 ggtitle("ARMA(1,1)")



#### Question

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

#### Code

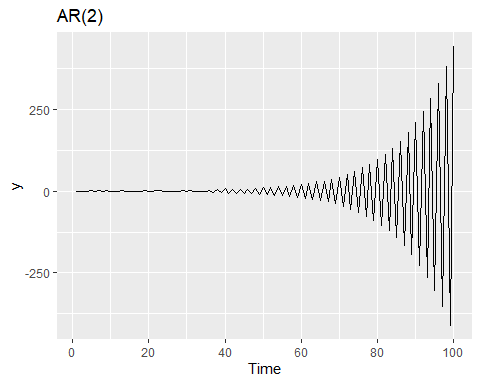
y <- ts(numeric(100))  
e <- rnorm(100)  
  
for(i in 3:100){  
 y[i] <- -0.8\*y[i-1] + 0.3\*y[i-2] + e[i]  
}

#### Question

1. Graph the latter two series and compare them.

#### Code

y <- ts(numeric(100))  
e <- rnorm(100)  
  
for(i in 3:100){  
 y[i] <- -0.8\*y[i-1] + 0.3\*y[i-2] + e[i]  
}  
  
autoplot(y) +  
 ggtitle("AR(2)")



#### Response

The AR(2) model rapdily begins to take-off as time increases, which is the opposite of the decrease in variation of the ARMA(1,1) model.

# HA 8.8

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

#### Question

1. 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.

#### Code

library(fpp)

## Loading required package: expsmooth

## Loading required package: lmtest

## Loading required package: zoo

##   
## Attaching package: 'zoo'

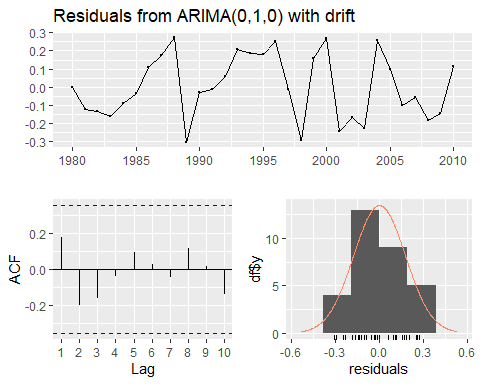
## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

## Loading required package: tseries

fit1 <- auto.arima(austa, stepwise = FALSE, approximation = FALSE)  
  
summary(fit1)

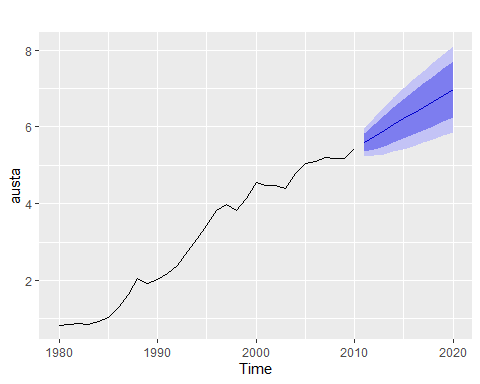
## Series: austa   
## ARIMA(0,1,0) with drift   
##   
## Coefficients:  
## drift  
## 0.1537  
## s.e. 0.0323  
##   
## sigma^2 estimated as 0.03241: log likelihood=9.38  
## AIC=-14.76 AICc=-14.32 BIC=-11.96  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 2.181271e-05 0.1741103 0.1505808 -1.056464 6.160654 0.8013559  
## ACF1  
## Training set 0.1771429

checkresiduals(fit1)



##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(0,1,0) with drift  
## Q\* = 3.8563, df = 5, p-value = 0.5703  
##   
## Model df: 1. Total lags used: 6

fc1 <- forecast(fit1, h = 10)  
autoplot(austa) +  
 autolayer(fc1)



#### Response

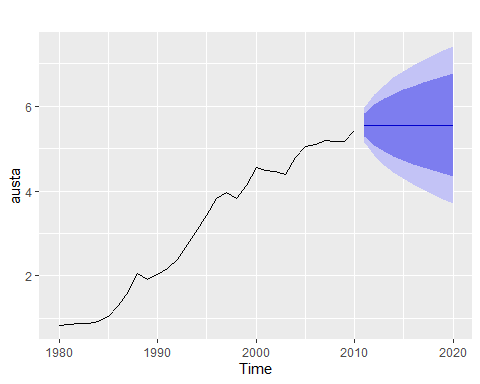
The auto.arima() function selected an ARIMA(0,1,0) with drift. The residuals do indeed look like noise according to the ACF plot (no autocorrelations above critical values).

#### Question

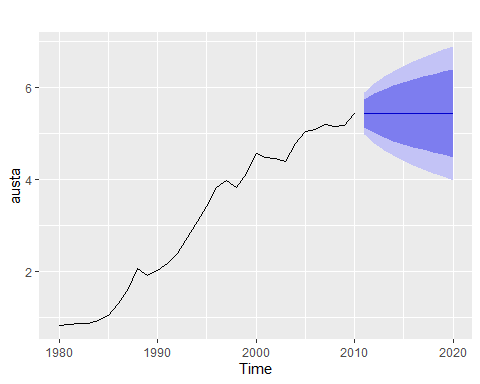
1. Plot forecasts from an ARIMA(0,1,1) model with no drift and compare these to part a. Remove the MA term and plot again.

#### Code

fit2 <- Arima(austa, order=c(0,1,1), include.drift = FALSE)  
  
fc2 <- forecast(fit2, h = 10)  
autoplot(austa) +  
 autolayer(fc2)



fit3 <- Arima(austa, order=c(0,1,0), include.drift = FALSE)  
  
fc3 <- forecast(fit3, h = 10)  
autoplot(austa) +  
 autolayer(fc3)



#### Response

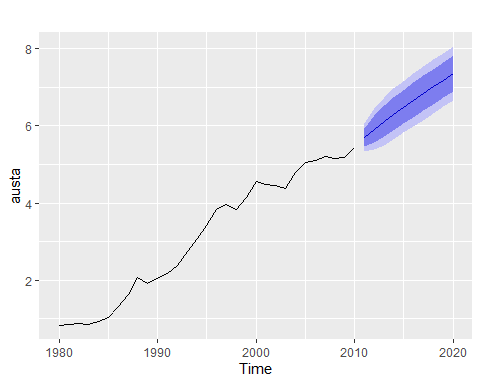
Part (a) forecasts have a trend do to the allwed drift, while these have no trend.

#### Question

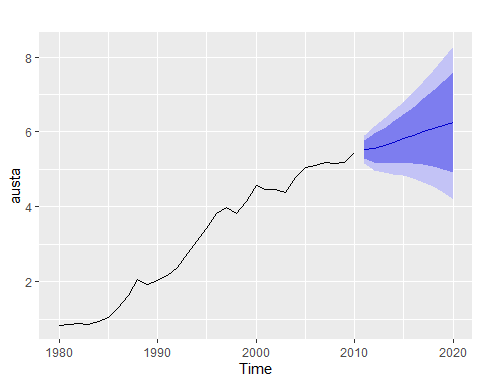
1. Plot forecasts from an ARIMA(2,1,3) model with drift. Remove the constant and see what happens.

#### Code

fit4 <- Arima(austa, order=c(2,1,3), include.drift = TRUE)  
  
fc4 <- forecast(fit4, h = 10)  
autoplot(austa) +  
 autolayer(fc4)



fit5 <- Arima(austa, order=c(2,1,3), include.drift = TRUE, include.constant = FALSE)  
  
fc5 <- forecast(fit5, h = 10)  
autoplot(austa) +  
 autolayer(fc5)



#### Response

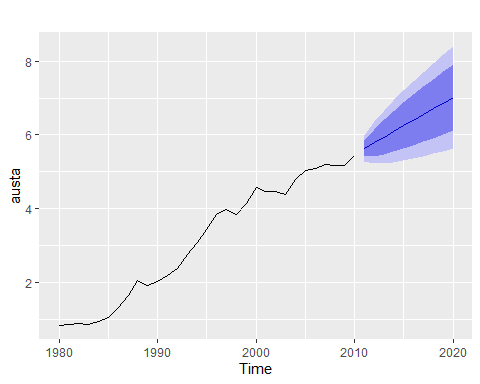
Both of these models include a positve trend, but removing the constant drastically increases the confidence interval.

#### Question

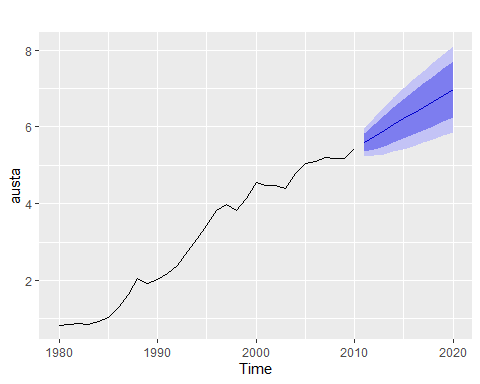
1. Plot forecasts from an ARIMA(0,0,1) model with a constant. Remove the MA term and plot again.

#### Code

fit6 <- Arima(austa, order=c(0,1,1), include.constant = TRUE)  
  
fc6 <- forecast(fit6, h = 10)  
autoplot(austa) +  
 autolayer(fc6)



fit7 <- Arima(austa, order=c(0,1,0), include.constant = TRUE)  
  
fc7 <- forecast(fit7, h = 10)  
autoplot(austa) +  
 autolayer(fc7)



#### Question

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

#### Code

fit8 <- Arima(austa, order=c(0,2,1), include.constant = FALSE)  
  
fc8 <- forecast(fit8, h = 10)  
autoplot(austa) +  
 autolayer(fc8)

