DATA 624 HW 1

Daniel Moscoe

6/18/2021

### KJ 3.1

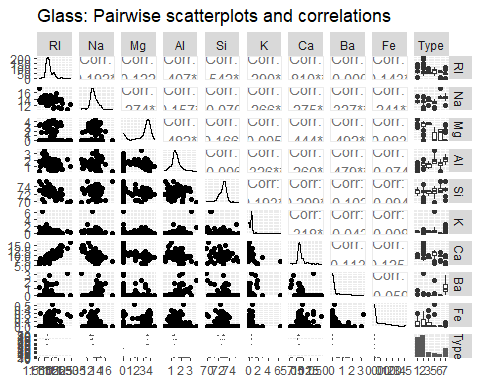
#### Question

"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.

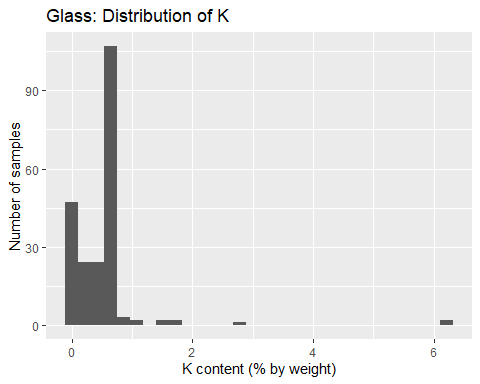
1. Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors."

#### Code

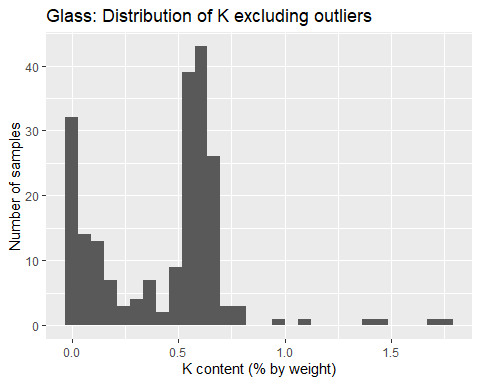
data(Glass)  
ggpairs(Glass) +  
 ggtitle("Glass: Pairwise scatterplots and correlations")



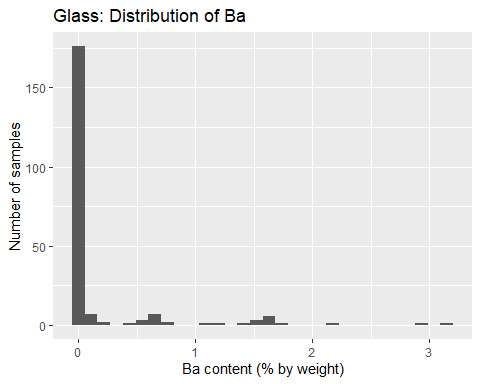
ggplot(data = Glass, aes(x = K)) +  
 geom\_histogram() +  
 ggtitle("Glass: Distribution of K") +  
 xlab("K content (% by weight)") +  
 ylab("Number of samples")



Glass %>%  
 filter(K < 2) %>%  
 ggplot(aes(x = K)) +  
 geom\_histogram() +  
 ggtitle("Glass: Distribution of K excluding outliers") +  
 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

Since the number of variables in this data set is relatively small, we can begin exploring the data by visually inspecting the distributions of all the variables, as well as their relationships to 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. These may be what Kuhn and Johnson describe as low-variance variables.

Several pairs of variables demonstrate large correlation coefficients. RI and Ca have a correlation coefficient of 0.810. 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. The especially large coefficient for RI and Ca suggests that it may be appropriate to omit one of these variables from a predictive model to reduce collinearity.

Based on the histogram for the distribution of K, we see that it clearly exhibits some outlying values greater than 2. The distribution of K excluding outliers exhibits strong peaks near 0 and 0.6.

Based on the histogram for the distribution of Ba, we see that Ba is a low-variance variable. with 176 entries of zero and 38 nonzero entries.

#### Question

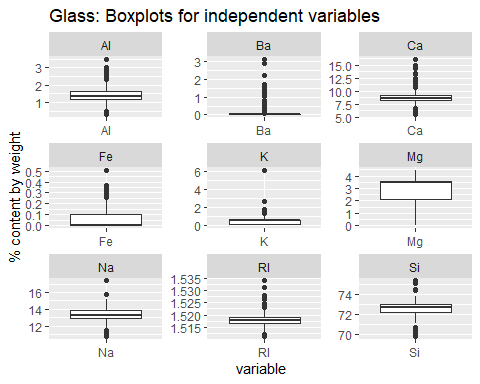
“(b) Do there appear to be any outliers in the data? Are any predictors skewed?”

#### Code

skews <- apply(Glass[,1:9], 2, skewness)  
skews

## 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 first table above shows that especially skewed variables (skewness > 2) are K, Ca, and Ba. Although Fe does not exhibit skewness greater than 2, its large number of zero-values mean that it might also be a good candidate for transformation.

Boxplots show that, with the exception of Mg, every variable contains values that could be considered outliers. Ba, Fe, and K are especially noteworthy, because they combine significant skewness with a large number of outlying values.

#### Question

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

#### Response

Because the number of explanatory variables is small, I would not consider PCA as a means of reducing dimensionality. My first concern would be to deal with variables that contain a large number of zero-values or are highly skewed: Ba, Fe, and K. These variables might be more useful as dummy variables, indicating either the presence or absence of the element. I would also run a Box-Cox transformation on each variable to resolve skewness. Centering and scaling couldn’t hurt, either– although since all these variables are measured in the same units, scaling may not be necessary.

### 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.”

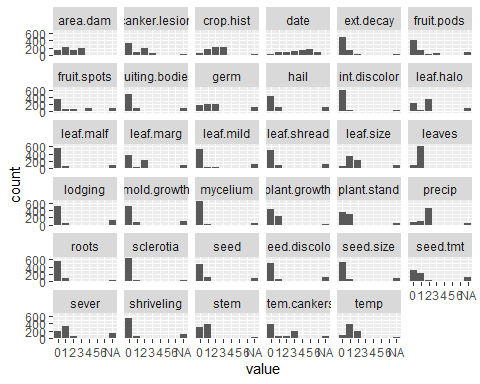
“(a). Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?”

#### Code

data(Soybean)  
summary(Soybean)

## Class date plant.stand precip temp   
## brown-spot : 92 5 :149 0 :354 0 : 74 0 : 80   
## alternarialeaf-spot: 91 4 :131 1 :293 1 :112 1 :374   
## frog-eye-leaf-spot : 91 3 :118 NA's: 36 2 :459 2 :199   
## phytophthora-rot : 88 2 : 93 NA's: 38 NA's: 30   
## anthracnose : 44 6 : 90   
## brown-stem-rot : 44 (Other):101   
## (Other) :233 NA's : 1   
## hail crop.hist area.dam sever seed.tmt germ plant.growth  
## 0 :435 0 : 65 0 :123 0 :195 0 :305 0 :165 0 :441   
## 1 :127 1 :165 1 :227 1 :322 1 :222 1 :213 1 :226   
## NA's:121 2 :219 2 :145 2 : 45 2 : 35 2 :193 NA's: 16   
## 3 :218 3 :187 NA's:121 NA's:121 NA's:112   
## NA's: 16 NA's: 1   
##   
##   
## leaves leaf.halo leaf.marg leaf.size leaf.shread leaf.malf leaf.mild   
## 0: 77 0 :221 0 :357 0 : 51 0 :487 0 :554 0 :535   
## 1:606 1 : 36 1 : 21 1 :327 1 : 96 1 : 45 1 : 20   
## 2 :342 2 :221 2 :221 NA's:100 NA's: 84 2 : 20   
## NA's: 84 NA's: 84 NA's: 84 NA's:108   
##   
##   
##   
## stem lodging stem.cankers canker.lesion fruiting.bodies ext.decay   
## 0 :296 0 :520 0 :379 0 :320 0 :473 0 :497   
## 1 :371 1 : 42 1 : 39 1 : 83 1 :104 1 :135   
## NA's: 16 NA's:121 2 : 36 2 :177 NA's:106 2 : 13   
## 3 :191 3 : 65 NA's: 38   
## NA's: 38 NA's: 38   
##   
##   
## mycelium int.discolor sclerotia fruit.pods fruit.spots seed   
## 0 :639 0 :581 0 :625 0 :407 0 :345 0 :476   
## 1 : 6 1 : 44 1 : 20 1 :130 1 : 75 1 :115   
## NA's: 38 2 : 20 NA's: 38 2 : 14 2 : 57 NA's: 92   
## NA's: 38 3 : 48 4 :100   
## NA's: 84 NA's:106   
##   
##   
## mold.growth seed.discolor seed.size shriveling roots   
## 0 :524 0 :513 0 :532 0 :539 0 :551   
## 1 : 67 1 : 64 1 : 59 1 : 38 1 : 86   
## NA's: 92 NA's:106 NA's: 92 NA's:106 2 : 15   
## NA's: 31   
##   
##   
##

Soybean\_features <- Soybean %>%  
 select(-Class) %>%  
 gather(key = 'variable', value = 'value')  
  
ggplot(data = Soybean\_features, aes(x = value)) +  
 geom\_bar() +  
 facet\_wrap(variable ~ .)



#### Response

A categorical variable might be degenerate if it has extreme class imbalance, such that overall, the variable can be considered “low-variance.” The variable might also be degenerate due to a large number of missing values. Based on the bar plots and summary information above, several variables raise concerns. sclerotia, mycelium, and int.discolor all exhibit extreme class imbalance. hail, sever, seed.tmt, and lodging all have 121 missing values. The number of missing entries across features is not randomly distributed. Instead, there are certain common numbers of missing measurements. 121, 84, and 38 are the most common numbers for missing values in these variables. This pattern in the number of missing values suggests that there may be some meaning in these missing values. More information on how this data was collected could help us discern whether there is meaning in the missing values.

#### Question

“(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?”

#### Code

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

#### Response

The predictors most likely to be missing are hail, sever, seed.tmt, and lodging, each with 121 missing values. There does appear to be a pattern of missing data related to the classes. For example, 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.

#### Question

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

#### Response

Because there seems to be a pattern in the missing data for this data set, the first thing I would do would be to explore further how this data was collected, and whether missing values might be meaningful. After that, I would explore imputation strategies. Because all the variables in this data set are categorical, imputing missing values using an “average” value, like a mean or median, is not feasible. A nearest-neighbors strategy might be useful. However, the class imbalance across a large number of these variables might mean that imputation using nearest-neighbors would be very inaccurate. I would also experiment with treating the missing values as categories themselves within each variable. I expect this last strategy might yield the best results, since the pattern in missingness tells me that missing values might indeed be meaningful for this data set.

### HA 2.1

#### Question

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

#### Code

help(gold)  
help(woolyrnq)  
help(gas)

#### Response

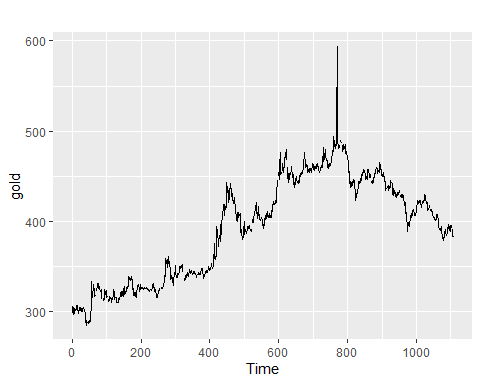
gold is a time-series object containing “daily morning gold prices in US dollars.” The data extends from 1/1/1985 to 3/31/1989. woolyrnq is a time-series object containing “Quarterly production of woollen yarn in Australia” in tonnes. The data extends from 3/1965 to 9/1994. gas is a time-series object containing “Australian monthly gas production.” The unit of measurement isn’t given in the help file or on the plots available from tsdisplay. The data extends from 1956 to 1995.

#### Question

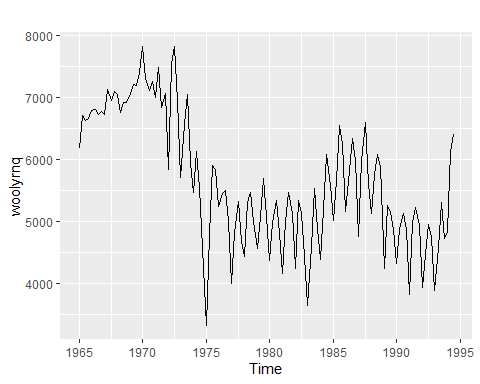
1. “Use autoplot() to plot each of these in separate plots.”

#### Code

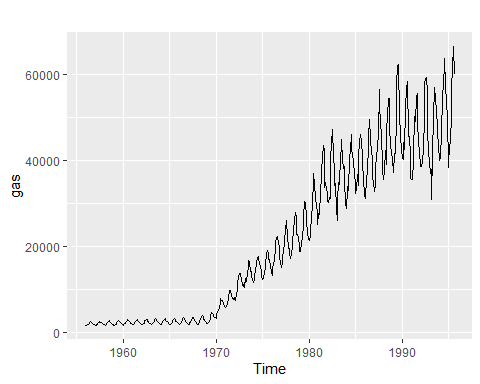
autoplot(gold)



autoplot(woolyrnq)



autoplot(gas)



#### Response

Each of the plots shows a line graph comparing the quantity to time. For gold, time is given in days since the start of data collection. For woolyrnq and gas, time is given in years.

gold exhibits cycles ranging from about 100 days to about 180 days. There is an upward trend from day 0 to around day 775 that then reverses. Based on this graph, I don’t detect any seasonality.

woolyrnq exhibits strong seasonality, peaking around the midpoint of every year, and reaching a seasonal minimum around the end of each year. Based on this graph, there does not appear to be any overall trend. There appears to be some evidence of cycles roughly every 10 years.

gas exhibits a strong increasing trend and strong seasonality, peaking around the midpoint of each year, and reaching seasonal minimums around the end of each year. Based on this graph, there does not appear to be any cyclical behavior. The difference between maximum and minimum gas production is increasing with time.

#### Question

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

#### Code

print(frequency(gold))

## [1] 1

print(frequency(woolyrnq))

## [1] 4

print(frequency(gas))

## [1] 12

#### Response

gold has frequency 1, which would usually mean that one value is recorded per year. However, the help file for gold tells us that it contains daily data. The result of frequency(gold) is incorrect, and should be 365. Working with this data would require transforming the time component to correctly represent the frequency of measurement.

The frequency of woolyrnq is 4, indicating that data is stated quarterly. The frequency of gas is 12, indicating that data is stated monthly.

#### Question

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

#### Code

print(which.max(gold))

## [1] 770

print(gold[which.max(gold)])

## [1] 593.7

print(ymd(19850101) + 769)

## [1] "1987-02-09"

#### Response

The outlier in the gold series is the 770th entry, which is $593.70, and which occurred on 2/9/1987.

### 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.”

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

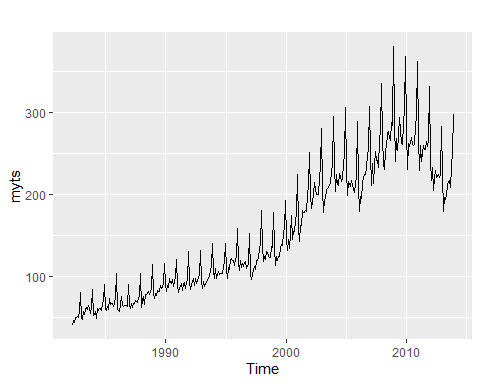
retaildata <- readxl::read\_excel(path, skip=1)

“b. Select one of the time series…”

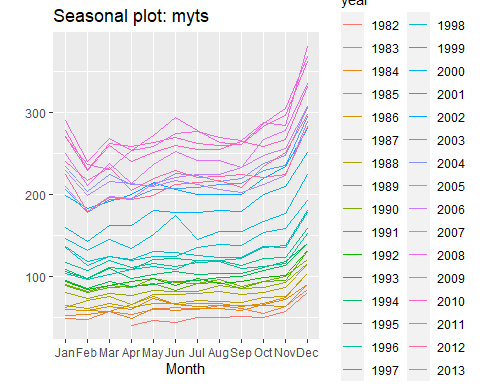
“c. Explore your chosen retail time series using the following functions: autoplot(), ggseasonplot(), ggsubseriesplot(), gglagplot(), ggAcf(). Can you spot any seasonality, cyclicity and trend? What do you learn about the series?”

#### Code

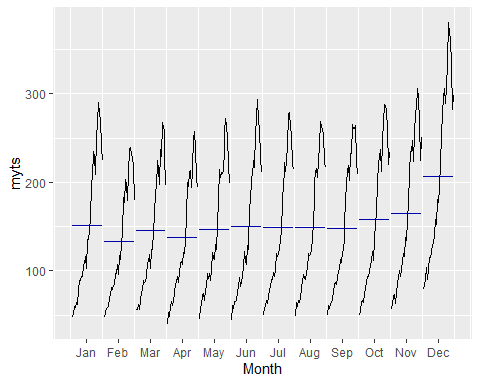
myts <- ts(retaildata[, "A3349503T"],  
 frequency = 12, start = c(1982,4))  
  
autoplot(myts)



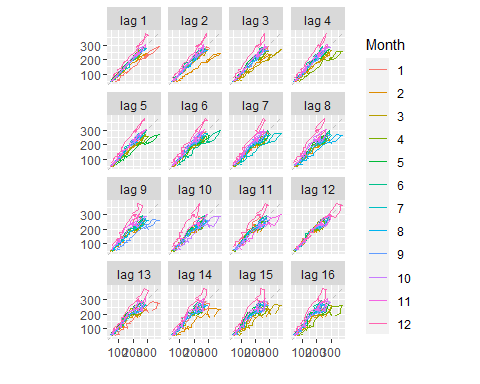
ggseasonplot(myts)



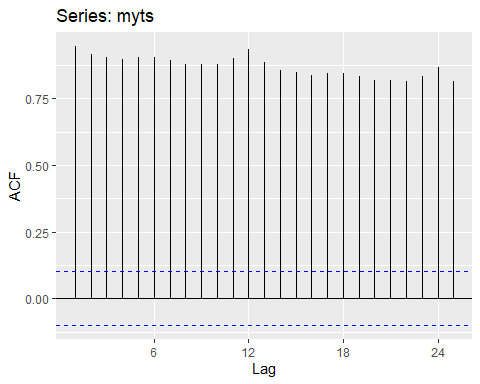
ggsubseriesplot(myts)



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

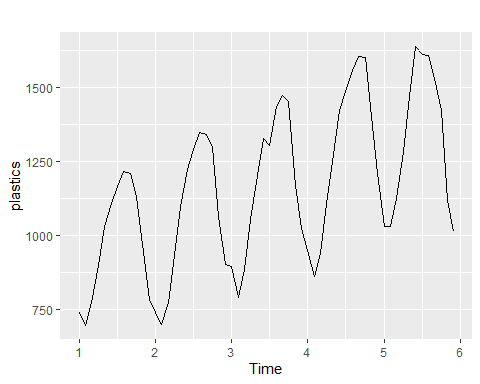
#### Question

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

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

#### Code

autoplot(plastics)



#### Response

The graph above shows the time series for sales of product A. There are clear seasonal fluctuations, with a cycle that oscillates about once per time unit. There is also an upward trend that, over this short timespan, appears linear. Seasonal fluctuations do not appear to vary with the level of the data.

#### Question

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

#### Code

plastics\_decomposition <- decompose(plastics, type = "multiplicative")  
print(plastics\_decomposition["trend"])

## $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

print(plastics\_decomposition["seasonal"])

## $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

#### Response

The trend-cycle component is not defined for the first and last six months. For each other month, the centered 12-month moving average is equivalent to a 13-term weighted moving average, with weight 1/24 for the first and last component, and weight 1/12 for the second through 12th components.

To calculate the seasonal indices, we average the detrended values for each month. Since detrended values for the first and last six months are not defined, the average detrended values for each month will not consider them.

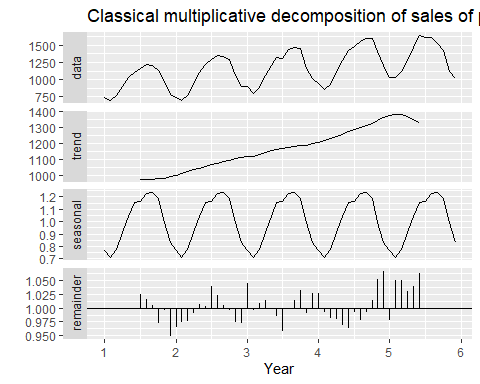
The seasonal indices are the result of finding the mean of the detrended values for each month. These means are then centered to ensure they have a sum of , the seasonal period. For plastics, .

#### Question

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

#### Code

plastics\_decomposition %>%  
 autoplot() + xlab("Year") +  
 ggtitle("Classical multiplicative decomposition of sales of product A")



#### Response

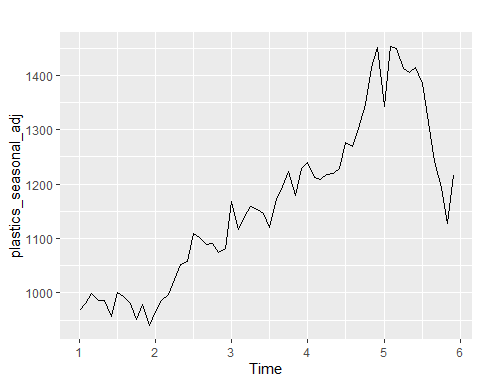
The plot in part a shows that the data contain pronounced seasonal fluctuations that repeat yearly, and that there is an overall upward trend. We can assess these claims by considering plots of just the trend component and just the seasonal component. The plots of these components confirm that there is an overall upward trend, and they reveal a slight downward turn in the trend during year 5. The plots also confirm the presence of seasonal fluctuations.

#### Question

“d. Compute and plot the seasonally adjusted data.”

#### Code

plastics\_seasonal\_adj <- seasadj(plastics\_decomposition)  
autoplot(plastics\_seasonal\_adj)



plastics\_seasonal\_adj

## Jan Feb Mar Apr May Jun Jul  
## 1 967.3468 981.2262 999.3182 986.4758 985.8925 956.7826 1001.1759  
## 2 966.0431 985.4495 996.7427 1023.8257 1051.9377 1057.0417 1108.5982  
## 3 1168.1168 1116.3736 1139.6864 1158.9443 1152.4413 1146.0648 1119.7701  
## 4 1239.8204 1212.1029 1207.9388 1218.2647 1219.4437 1229.0378 1277.0364  
## 5 1342.8129 1452.8342 1450.0416 1411.6051 1405.1361 1414.8628 1384.4587  
## Aug Sep Oct Nov Dec  
## 1 992.4139 981.0263 951.4241 978.9119 939.8819  
## 2 1100.9592 1089.0366 1090.2260 1074.6860 1081.5244  
## 3 1171.9625 1196.2349 1222.2981 1179.5334 1227.9683  
## 4 1269.0820 1302.6210 1345.9580 1414.4319 1451.2352  
## 5 1312.3369 1240.9008 1194.5377 1128.1178 1215.9647

#### Response

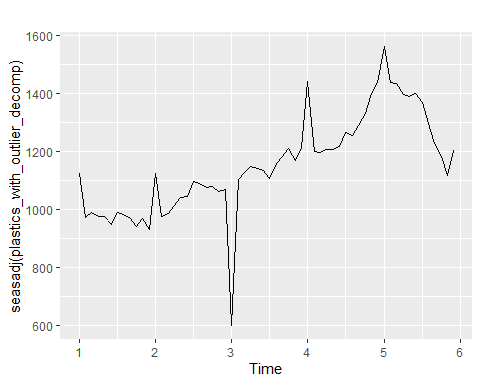
For a multiplicative decomposition, data are seasonally adjusted when they are divided by their seasonal component. The seasonally adjusted data display the upward trend apparent in the original data, but seasonal fluctuations are absent.

#### Question

“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?”

#### Code

plastics\_with\_outlier <- plastics  
plastics\_with\_outlier[25] <- plastics[25] - 500  
  
plastics\_with\_outlier\_decomp <- decompose(plastics\_with\_outlier, type = "multiplicative")  
autoplot(seasadj(plastics\_with\_outlier\_decomp))



#### Response

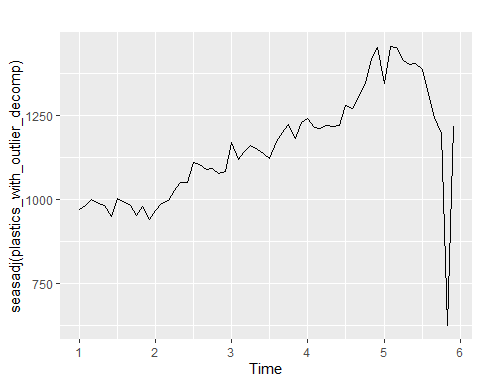
The effect of the outlier is evident throughout the seasonally adjusted time series. The revised value, the first in the third year, is clearly much lower than it was before. But the first value of every year is also affected, by being increased. Since the low outlier reduced the seasonal index for January, all other January values appear higher relative to the new seasonal index.

#### Question

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

#### Code

plastics\_with\_outlier <- plastics  
plastics\_with\_outlier[59] <- plastics[59] - 500  
  
plastics\_with\_outlier\_decomp <- decompose(plastics\_with\_outlier, type = "multiplicative")  
autoplot(seasadj(plastics\_with\_outlier\_decomp))



#### Response

Placing the outlier near the end of the time series does make a difference. When the outlier is at the end, there is no trend component defined for it. Since a detrended value cannot be computed at a position in the time series very near the beginning or end, these values do not contribute to the seasonal indices. Therefore, they do not affect other seasonally adjusted values in the time series.

### HA 7.1

#### Question

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

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

#### Code

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

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

#### 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

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.”

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

#### Response

An autocorrelation plot shows the correlation coefficients between observations and their lagged values. In an ACF plot for white noise, we would expect to see no pattern in the values plotted, and we would expect 95% of those values to lie within the blue dashed lines. All three of the plots here meet those criteria, so all three represent white noise. They differ in the length of the time series examined in each plot. With a larger number of observations in a white noise time series, the variability in autocorrelation coefficients decreases, and the critical values also approach zero. This is exactly what we see in these plots.

#### Question

“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?”

#### Response

The critical values are at different distances from the mean of zero because they correspond to time series of different lengths. In shorter time series, critical values are farther from zero. That’s because random chance can give rise to relatively large autocorrelation coefficients among small subsets of values from white noise. The relationship between the length of the time series and the critical values is reflected in the formula for critical values, , where is the length of the time series. The autocorrelations in each figure are different, even though they all refer to white noise. That’s because in white noise, autocorrelation coefficients are essentially random. Since there is no true relationship between members of the time series and their lagged values, autocorrelation coefficients just fluctuate randomly around their mean of zero, giving rise to different values in different time series.

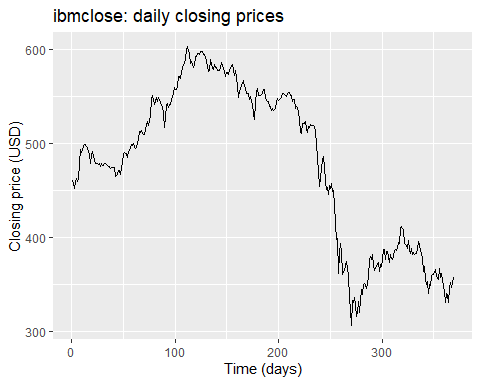
### HA 8.2 [TODO: RESOLVE ISSUE WITH PACF]

#### Question

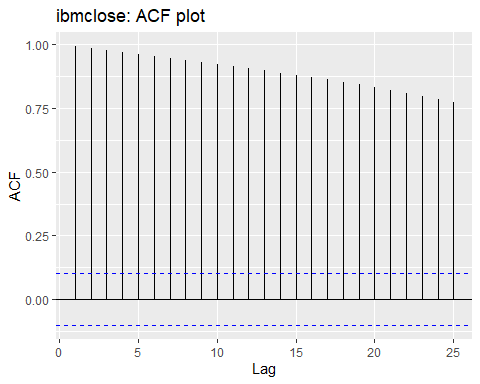
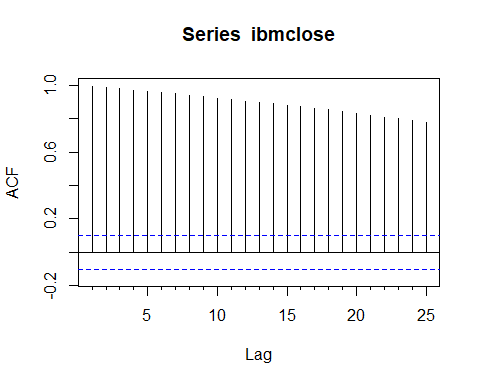
“2. A classic example of a non-stationary series is the daily closing IBM stock price series (data set ibmclose). Use R to plot the daily closing prices for IBM stock and the ACF and PACF. Explain how each plot shows that the series is non-stationary and should be differenced.”

#### Code

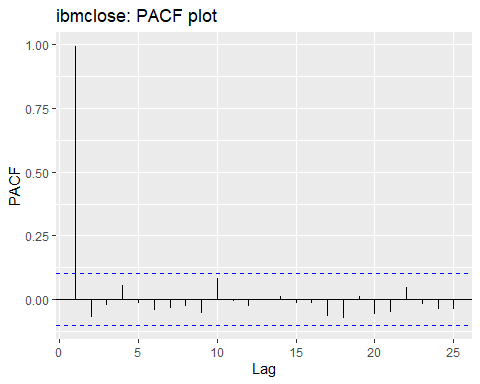
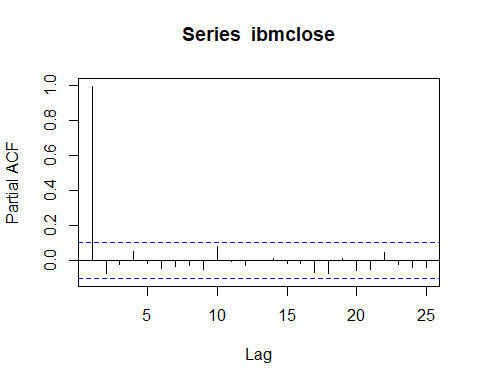
autoplot(ibmclose) +  
 ggtitle("ibmclose: daily closing prices") +  
 xlab("Time (days)") +  
 ylab("Closing price (USD)")



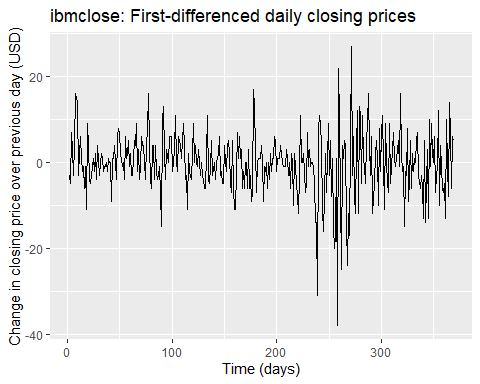
autoplot(Acf(ibmclose)) +  
 ggtitle("ibmclose: ACF plot")



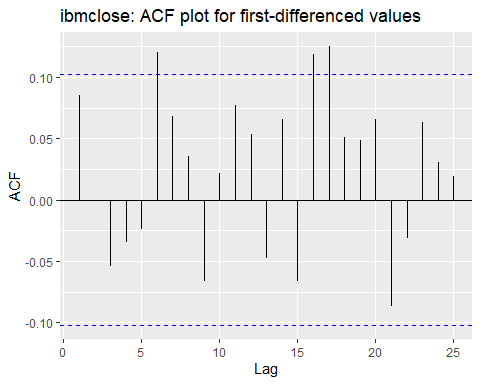
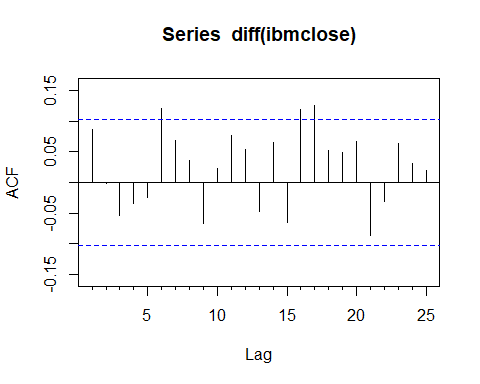
autoplot(pacf(ibmclose)) +  
 ggtitle("ibmclose: PACF plot")



autoplot(diff(ibmclose)) +  
 ggtitle("ibmclose: First-differenced daily closing prices") +  
 xlab("Time (days)") +  
 ylab("Change in closing price over previous day (USD)")



autoplot(Acf(diff(ibmclose))) +  
 ggtitle("ibmclose: ACF plot for first-differenced values")



Box.test(diff(ibmclose), lag = 1, type = 'Ljung-Box')

##   
## Box-Ljung test  
##   
## data: diff(ibmclose)  
## X-squared = 2.717, df = 1, p-value = 0.09929

#### Response

The daily closing prices for IBM stock in the first plot above are non-stationary data. Stationary data shows no trend, but the ibmclose data set shows several distinct regions of trend.

The ACF plot for the ibmclose data shows a very high initial value– nearly 1– that declines slowly as lag increases. This is typical of non-stationary data. If the data were stationary, we would expect to see ACFs within the critical values marked by blue lines on the plot. As in the goog200 data in section 8.1, which shows the closing price of Google stock, differencing could transform the ibmclose data to a stationary time series.

[TODO: How does the PACF plot show that the series is non-stationary and should be differenced?]

The differenced ibmclose data is much more nearly stationary, although the ACF plot for the first-differenced data does show a majority of ACF values that are positive, and it shows several that exceed the critical value. The Box-Ljung test examines the null hypothesis of independence between the differenced values and the lagged differences. The p-value is greater than 0.05, so we do not reject the hypothesis of independence. So the differenced ibmclose data is stationary.

### HA 8.6

#### Question

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

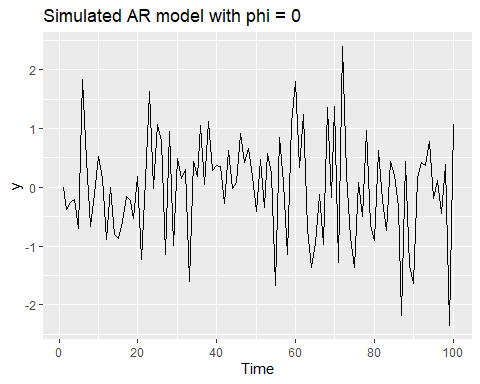
“a. Use the following R code to generate data from an AR(1) model with and . The process starts with .”

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

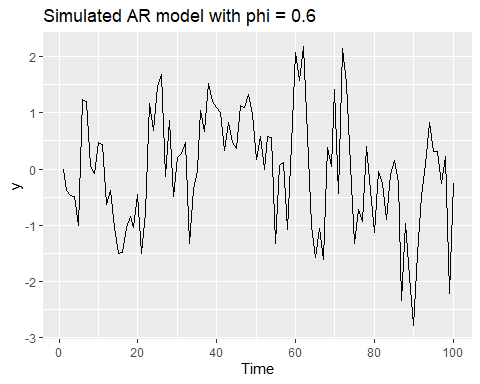
“b. Produce a time plot for the series. How does the plot change as you change ?”

#### Code

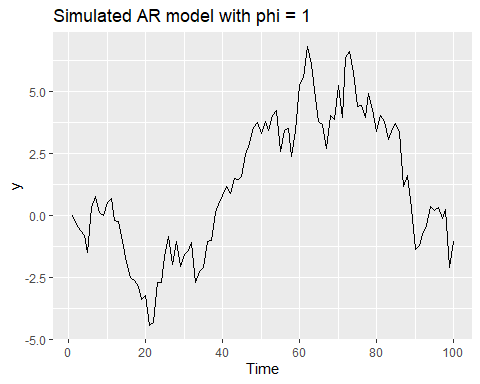
for(i in 2:100){  
 y[i] <- 0\*y[i-1] + e[i]  
}  
autoplot(y) +  
 ggtitle("Simulated AR model with phi = 0")



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



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



#### Response

is the coefficient on the lagged data when a regression model is constructed for the data onto the lagged data. When , the data is white noise. As increases, the relationship between a value and its lag becomes clearer. The time series plot begins to display structure, with most values falling near their lags. With , the time series is a random walk.

#### Question

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

#### Code

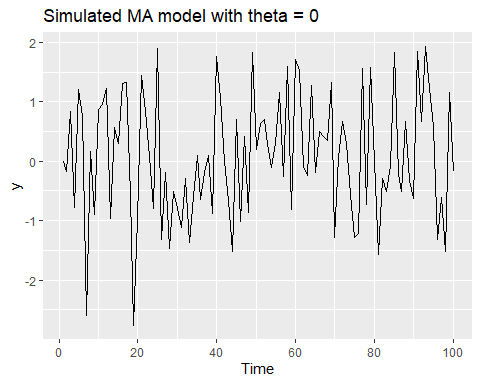
y <- ts(numeric(100))  
e <- rnorm(100)  
for(i in 2:100)  
 y[i] = e[i] + 0.6 \* e[i-1]

#### Question [TODO: MAKE THIS BETTER]

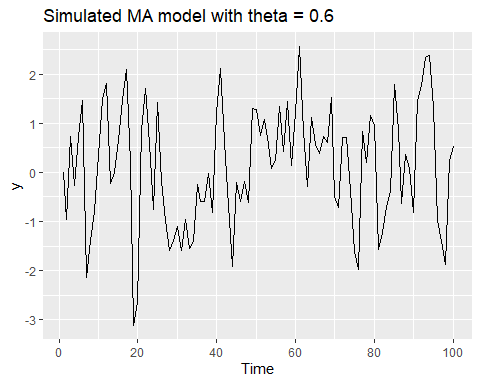
“d. Produce a time plot for the series. How does the plot change as you change ?”

#### Code

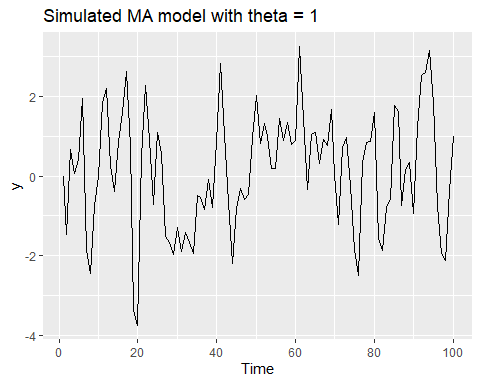
for(i in 2:100){  
 y[i] <- e[i] + 0 \* e[i-1]  
}  
autoplot(y) +  
 ggtitle("Simulated MA model with theta = 0")



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



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



#### Response

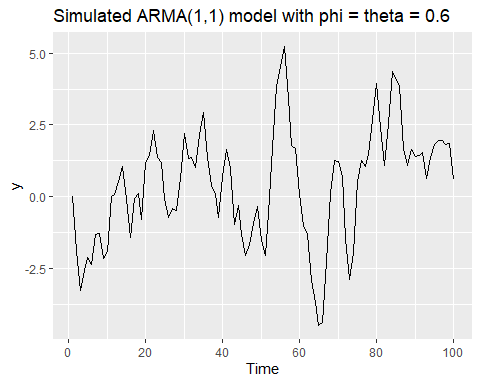
As increases, the variability of the time series appears to increase. [TODO: MAKE THIS BETTER]

#### Question

“e. Generate data from an ARMA(1,1) model with .”

#### Code

y <- ts(numeric(100))  
e <- rnorm(100)  
for(i in 2:100)  
 y[i] <- 0.6 \* y[i-1] + e[i] + 0.6 \* e[i - 1]  
autoplot(y) +  
 ggtitle("Simulated ARMA(1,1) model with phi = theta = 0.6")



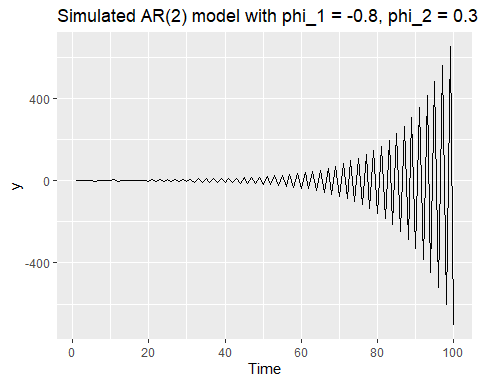
#### Question [TODO: COMPLETE RESPONSE]

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

“g. 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("Simulated AR(2) model with phi\_1 = -0.8, phi\_2 = 0.3")



#### Response

[TODO: RESPOND TO THIS QUESTION]

### HA 8.8

#### Question

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

“a. Use auto.arima() to find an appropriate ARIMA model. What model was selected?”

#### Code

fit1 <- auto.arima(austa)  
summary(fit1)

## 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

#### Response

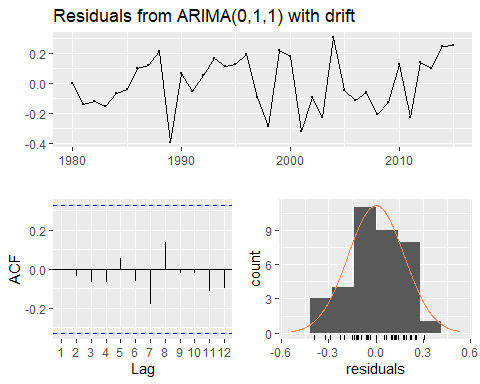
The model selected is ARIMA(0,1,1). This means that there are no autoregression components to the model. The data was subject to first-differencing one time. And there is one moving average term in the model. The model also contains a drift component. [TODO: What is the actual equation for the model?]

#### Question

“Check that the residuals look like white noise.”

#### Code

checkresiduals((fit1))



##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(0,1,1) with drift  
## Q\* = 2.297, df = 5, p-value = 0.8067  
##   
## Model df: 2. Total lags used: 7

#### Response

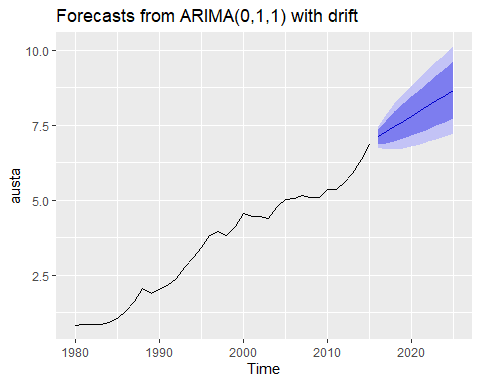
The residuals do appear to be white noise. The time series plot shows random values near a mean of 0. The ACF plot shows no significant spikes. And the histogram shows that the residuals are roughly normally distributed about 0.

#### Question

“Plot forecasts for the next 10 periods.”

#### Code

autoplot(forecast(fit1, h = 10))

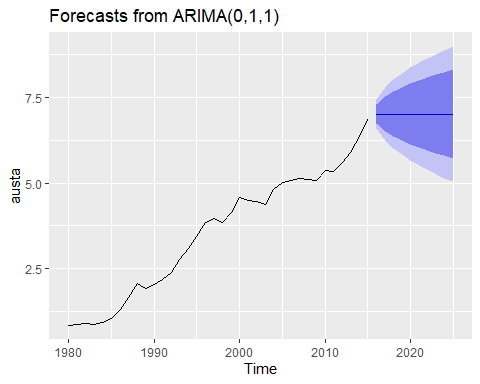


#### Question

“b. Plot forecasts from an ARIMA(0,1,1) model with no drift. Compare these to part a.”

#### Code

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



#### Response

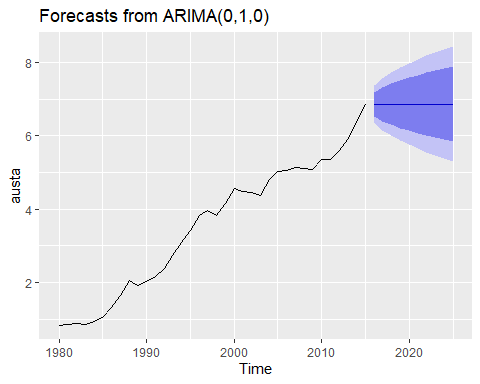
The forecasts in part a show an upward trend because they included a drift component. The forecasts in part b are effectively naive forecasts, equal to the most recent value. They do not demonstrate an upward trend.

#### Question

“Remove the MA term and plot again.”

#### Code

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



print(forecast(fit2, h = 1))

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2016 7.015837 6.734107 7.297567 6.584969 7.446705

print(forecast(fit3, h = 1))

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2016 6.858953 6.534151 7.183755 6.362211 7.355695

#### Response

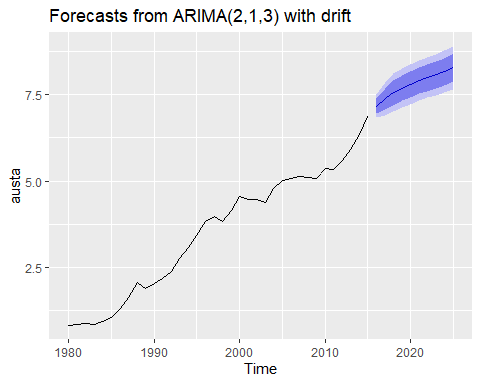
Removing the MA component has no effect on the slope of the forecasts; both are flat. But it does have some effect on the level: point forecasts with no MA term are equal to 6.859 million visitors, while forecasts with an MA term are equal to 7.016 million visitors. However, both forecasts have overlapping confidence intervals, so this difference is not statistically significant.

#### Question

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

#### Code

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

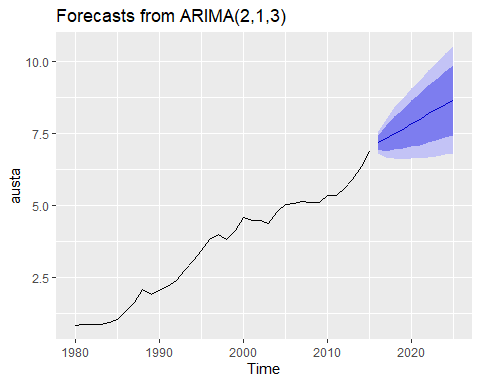


#### Question

“Remove the constant and see what happens.”

#### Code

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



#### Response

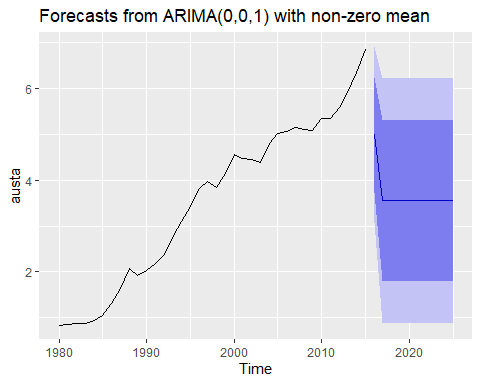
Removing the constant gives forecasts with a similar positive slope as the model with a constant. But confidence intervals for the model with no constant are much wider.

#### Question

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

#### Code

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

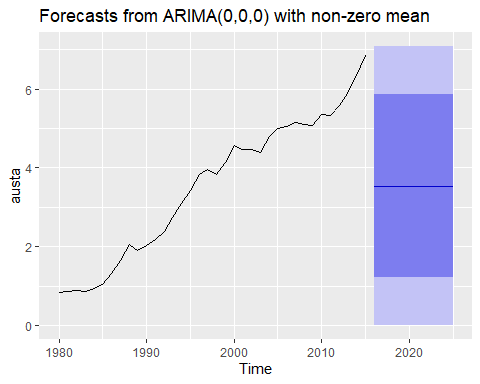


#### Question

“Remove the MA term and plot again.”

#### Code

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



#### Question

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

#### Code

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

