DATA 624 Fall 2017: Homework 1

Dan Smilowitz

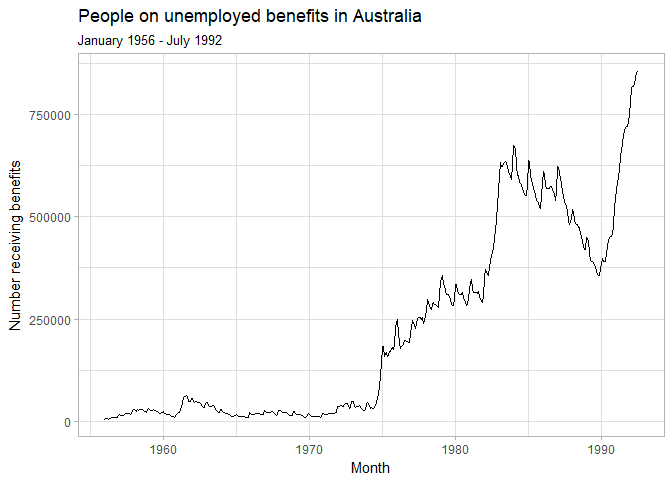
October 17, 2017

# HA 2.1

library(fma)  
data("dole")  
data("usdeaths")  
data("bricksq")

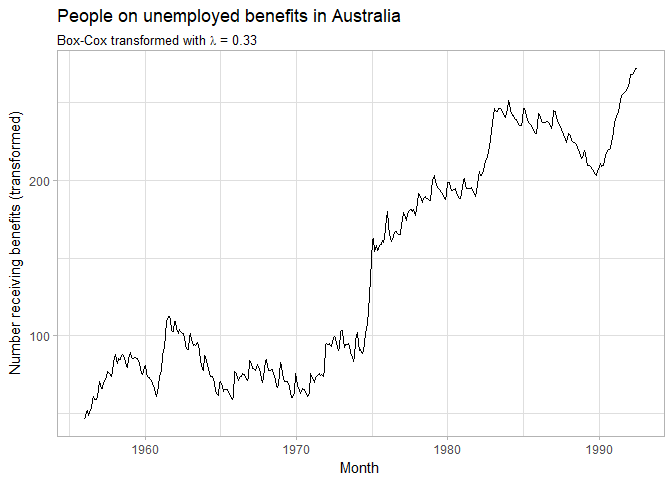
## Unemployment Benefits

autoplot(dole) +  
 labs(title = "People on unemployed benefits in Australia",  
 subtitle = "January 1956 - July 1992",  
 x = "Month", y = "Number receiving benefits")



Given the rapid growth after 1975, a transformation seems reasonable -- this plot is presented below using a base-10 logarithm:

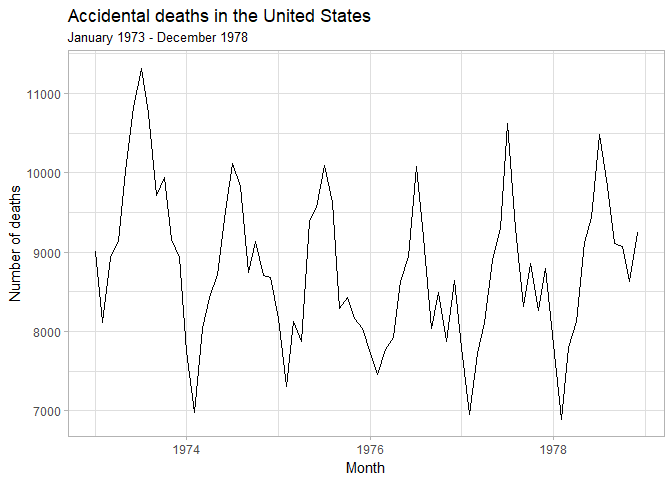
lambda\_dole <- round(BoxCox.lambda(dole), 2)  
autoplot(BoxCox(dole, lambda\_dole)) +  
 labs(title = "People on unemployed benefits in Australia",  
 subtitle = substitute("Box-Cox transformed with "\*lambda\*" = "\*l,  
 list(l = lambda\_dole)),  
 x = "Month", y = "Number receiving benefits (transformed)")



This transformation provides greater insight into variation in the data before 1975 -- in the un-transformed plot, the data appeared to be nearly zero, but here it can be clearly seen that there is variation of non-trivial degree in this period. The significant growth in values after 1975 is still apparent, as each horizontal line in the grid represents values more than tripling.

## Accidental Deaths

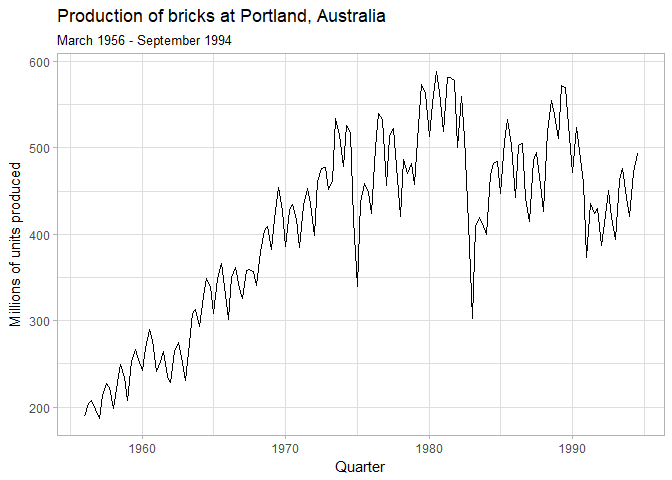
autoplot(usdeaths) +  
 labs(title = "Accidental deaths in the United States",  
 subtitle = "January 1973 - December 1978",  
 x = "Month", y = "Number of deaths")



In this case, since the magnitude of observations does not significantly change over time, a transformation is not needed.

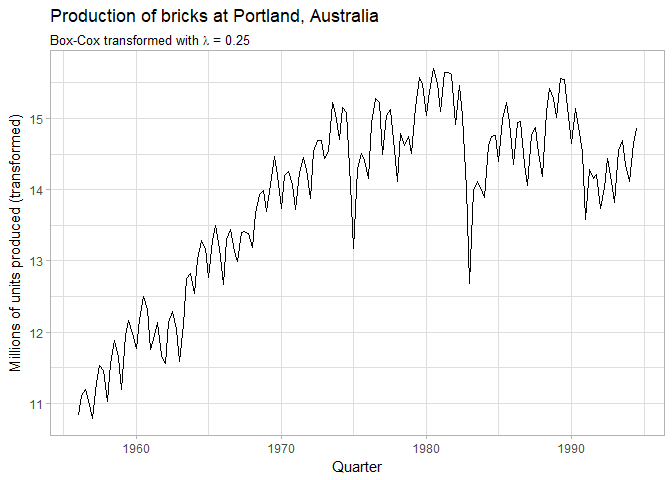
## Brick Production

autoplot(bricksq) +  
 labs(title = "Production of bricks at Portland, Australia",  
 subtitle = "March 1956 - September 1994",  
 x = "Quarter", y = "Millions of units produced")



Because of the difference in series behavior before and after 1975 (steady growth with seasonal variation followed by more stable behavior with seasonal variation), a transformation may be insightful.

lambda\_bricksq <- round(BoxCox.lambda(bricksq), 2)  
autoplot(BoxCox(bricksq, lambda\_bricksq)) +  
 labs(title = "Production of bricks at Portland, Australia",  
 subtitle = substitute("Box-Cox transformed with "\*lambda\*" = "\*l,  
 list(l = lambda\_bricksq)),  
 x = "Quarter", y = "Millions of units produced (transformed)")



This transformation does not provide any significant differences in the way the data is displayed; it is likely that no new inferences will be developed compared to the original plot.

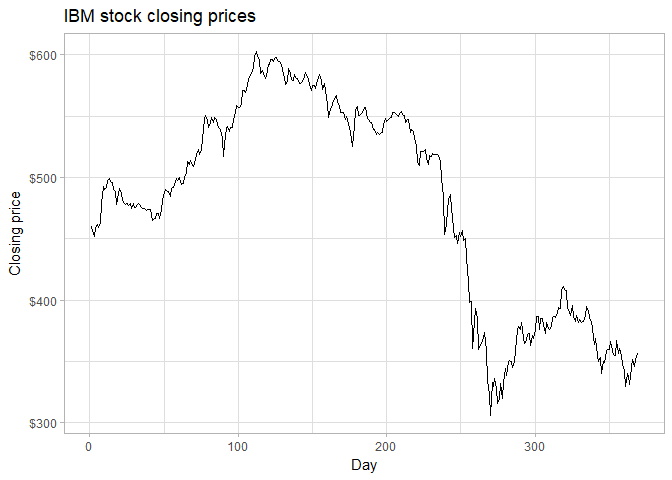
# HA 2.3

data("ibmclose")

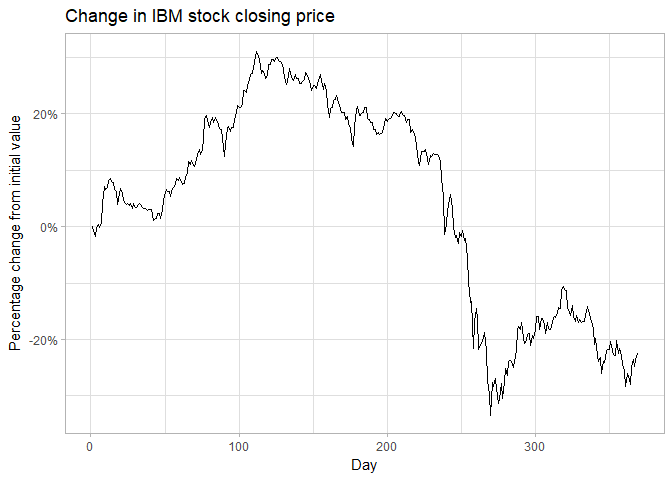
## Part a

The plots below show the IBM closing stock prices, change relative to the first value, and as the daily percentage change:

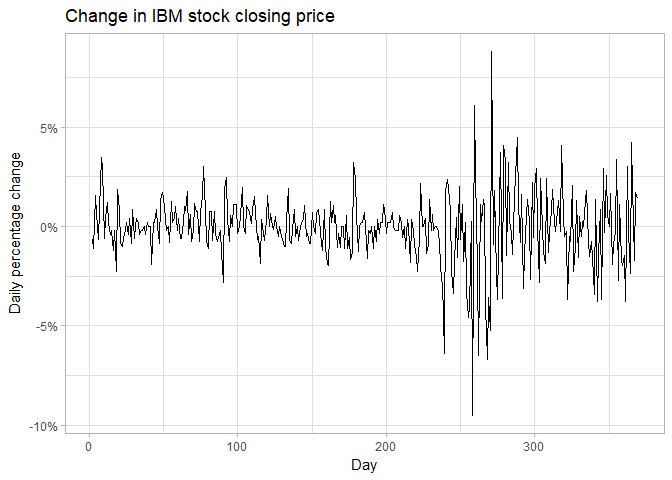
ibm\_tot\_diff <- ibmclose / ibmclose[1] - 1  
ibm\_day\_diff <- ts(ibmclose[2:369] / ibmclose[1:368] - 1,  
 start = 2, end = 369, frequency = 1)  
  
autoplot(ibmclose) +  
 labs(title = "IBM stock closing prices",  
 x = "Day",  
 y = "Closing price") +  
 scale\_y\_continuous(labels = dollar\_format())



autoplot(ibm\_tot\_diff) +  
 labs(title = "Change in IBM stock closing price",  
 x = "Day",  
 y = "Percentage change from initial value") +  
 scale\_y\_continuous(labels = percent)



autoplot(ibm\_day\_diff) +  
 labs(title = "Change in IBM stock closing price",  
 x = "Day",  
 y = "Daily percentage change") +  
 scale\_y\_continuous(labels = percent)



The plots above show a steady rise in price, followed by a slight decline then a steep decline, all showing daily variation generally between -2.5% and 2.5%. Near the end of this decline, and into the subsequent steady increase, the daily returns become increasingly volatile. The price then decreases again slowly with a volatility lower than the period immediately preceding it, but still higher than in the initial period of 200 days.

## Part b

The data is split into a training and testing set with 300 and 69 observations, respectively:

ibm\_train <- ibmclose[1:300]  
ibm\_test <- ibmclose[301:369]

## Part c

Three forecasts are performed on the split data:

* Random walk with drift
* Mean
* Naive

ibm\_drift <- rwf(ibm\_train, 69, drift = TRUE)  
ibm\_mean <- meanf(ibm\_train, 69)  
ibm\_naive <- naive(ibm\_train, 69)

The accuracy of this forecasts is calculated against the test data using the accuracy function and three measures of accuracy:

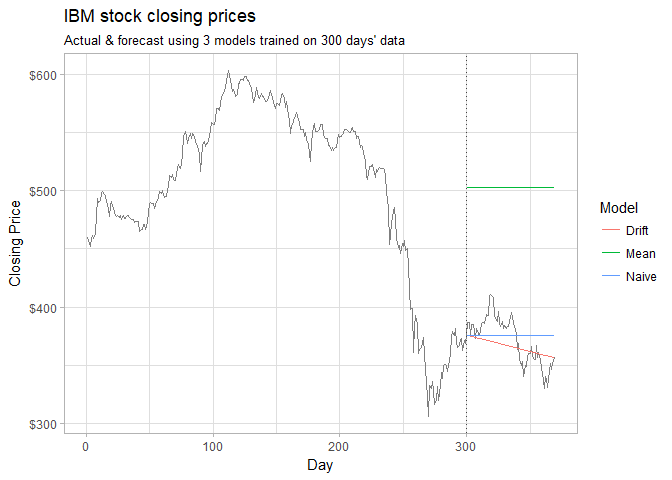
ibm\_acc <- as.data.frame(rbind(  
 accuracy(ibm\_drift, ibm\_test)["Test set", c("ME", "RMSE", "MAE", "MPE", "MAPE", "MASE")],  
 accuracy(ibm\_mean, ibm\_test)["Test set", c("ME", "RMSE", "MAE", "MPE", "MAPE", "MASE")],  
 accuracy(ibm\_naive, ibm\_test)["Test set", c("ME", "RMSE", "MAE", "MPE", "MAPE", "MASE")]),  
 row.names = c("Drift", "Mean", "Naive"))

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ME | RMSE | MAE | MPE | MAPE | MASE |
| **Drift** | 6.108 | 17.07 | 13.97 | 1.419 | 3.708 | 2.742 |
| **Mean** | -130.6 | 132.1 | 130.6 | -35.48 | 35.48 | 2.224 |
| **Naive** | -3.725 | 20.25 | 17.03 | -1.294 | 4.668 | 3.341 |

By the Root Mean Squared Error, Mean Absolute Error, and Mean Absolute Percentage Error benchmarks, the random walk with drift method is most accurate. By the Mean Absolute Scaled Error benchmark, the mean method is most accurate. By the Mean Error and Mean Percentage error benchmarks, the naive method is most accurate.

Due to the split in which model "did best" according to various benchmarks, a determination of which measure is most appropriate (scale-dependent, percent, or absolute) to determine suitability of a forecast. In the absence of this knowledge, a plot of the forecasts vs. the data may provide insight:

data\_frame(Day = 1:369,  
 Data = ibmclose,  
 Drift = c(rep(NA, 300), ibm\_drift$mean),  
 Mean = c(rep(NA, 300), ibm\_mean$mean),  
 Naive = c(rep(NA, 300), ibm\_naive$mean)) %>%  
 gather(Model, Value, c(Drift, Mean, Naive)) %>%   
 ggplot(aes(Day)) +  
 geom\_line(aes(y = Data), col = "grey50") +  
 geom\_line(aes(y = Value, col = Model)) +  
 geom\_vline(xintercept = 300, lty = 3, col = "grey25") +  
 scale\_x\_continuous() +  
 scale\_y\_continuous("Closing Price", labels = dollar\_format()) +  
 labs(title = "IBM stock closing prices",  
 subtitle = "Actual & forecast using 3 models trained on 300 days' data")



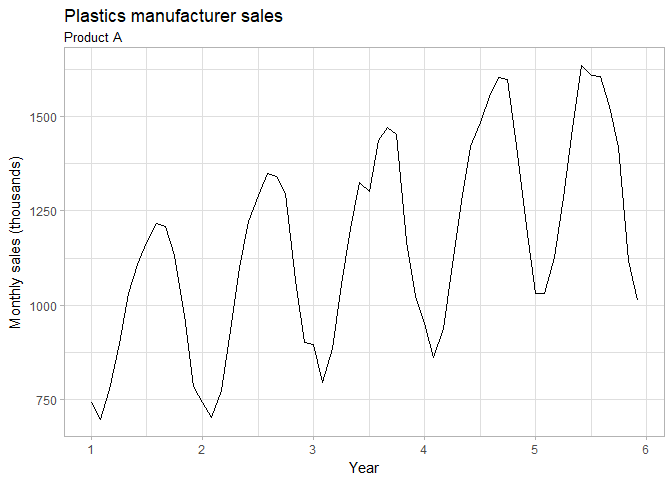
From this plot, it is clear that the naive model fails to capture overall trend of the observations beyond the training set, and the mean model is rather inappropriate. The random walk with drift model best captures the data, and performed best according to 3 of 6 benchmarks utilized --- it is the most appropriate model for this data.

# HA 6.2

data("plastics")

## Part a

autoplot(plastics) +  
 labs(title = "Plastics manufacturer sales",  
 subtitle = "Product A",  
 x = "Year",  
 y = "Monthly sales (thousands)")



From this plot, there is a clear annual seasonality where sales rise from roughly February through roughly July or August, then fall through the end of the year through roughly the following February. A clear increasing trend can be seen as well -- outside of the first few months of year 2, each month's sales show an increase from the same month in the prior year.

## Part b

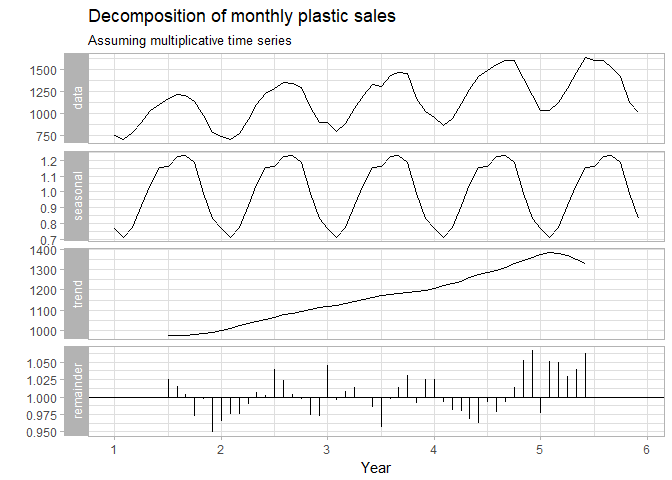
The multiplicative decomposition, and resulting seasonal and trend indices, are presented below:

fit\_plastics <- decompose(plastics, "m")  
fit\_plastics[c("seasonal", "trend")]

$seasonal  
 Jan Feb Mar Apr May Jun Jul  
1 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317  
2 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317  
3 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317  
4 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317  
5 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317  
 Aug Sep Oct Nov Dec  
1 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834  
2 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834  
3 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834  
4 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834  
5 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834  
  
$trend  
 Jan Feb Mar Apr May Jun Jul  
1 NA NA NA NA NA NA 976.9583  
2 1000.4583 1011.2083 1022.2917 1034.7083 1045.5417 1054.4167 1065.7917  
3 1117.3750 1121.5417 1130.6667 1142.7083 1153.5833 1163.0000 1170.3750  
4 1208.7083 1221.2917 1231.7083 1243.2917 1259.1250 1276.5833 1287.6250  
5 1374.7917 1382.2083 1381.2500 1370.5833 1351.2500 1331.2500 NA  
 Aug Sep Oct Nov Dec  
1 977.0417 977.0833 978.4167 982.7083 990.4167  
2 1076.1250 1084.6250 1094.3750 1103.8750 1112.5417  
3 1175.5000 1180.5417 1185.0000 1190.1667 1197.0833  
4 1298.0417 1313.0000 1328.1667 1343.5833 1360.6250  
5 NA NA NA NA NA

## Part c

autoplot(fit\_plastics) +  
 labs(title = "Decomposition of monthly plastic sales",  
 subtitle = "Assuming multiplicative time series",  
 x = "Year")

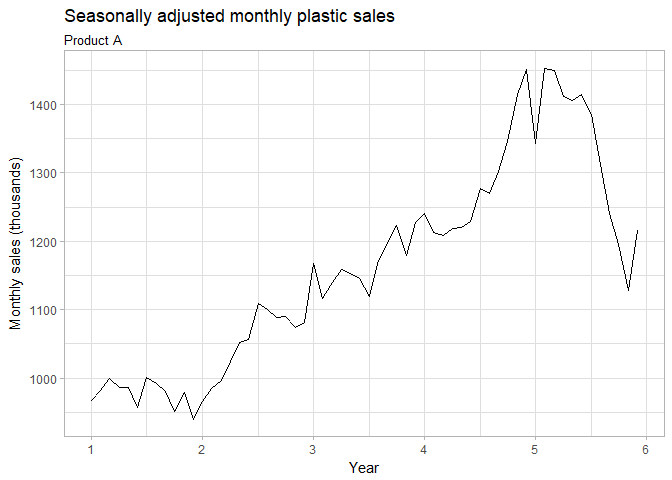


The seasonal and trend indices, as shown above and illustrated in the above graph, support the interpretation from part a -- there is an annual rising-then-falling seasonality running from February-February and a clear upward trend.

## Part d

For a multiplicative time series, the seasonally adjusted data is given by dividing the original data by the seasonal component ()

adj\_plastics <- plastics / fit\_plastics[["seasonal"]]  
autoplot(adj\_plastics) +  
 labs(title = "Seasonally adjusted monthly plastic sales",  
 subtitle = "Product A",  
 x = "Year",  
 y = "Monthly sales (thousands)")



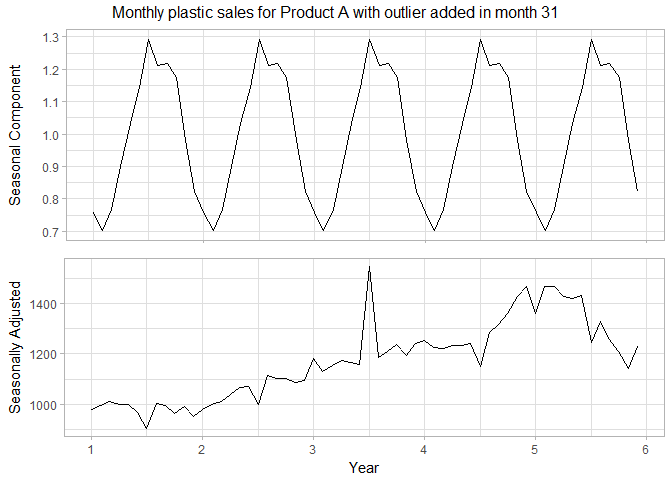
## Part e

The 31st observation, corresponding to 2.5 years into the time window, is increased from 1303 to 2000:

out\_plastics <- plastics  
out\_plastics[31] <- 2000

The seasonal component and seasonally-adjusted data are recalculated, using the seasadj function for brevity:

fit\_out\_plastics <- decompose(out\_plastics, "m")  
gridExtra::grid.arrange(  
 autoplot(fit\_out\_plastics$seasonal) +   
 labs(y = "Seasonal Component\n", x = NULL) +  
 scale\_x\_continuous(labels = NULL, minor\_breaks = seq(1, 6, 0.5)),  
 autoplot(seasadj(fit\_out\_plastics)) +   
 labs(y = "Seasonally Adjusted", x = "Year"),  
 top = grid::textGrob("Monthly plastic sales for Product A with outlier added in month 31"))

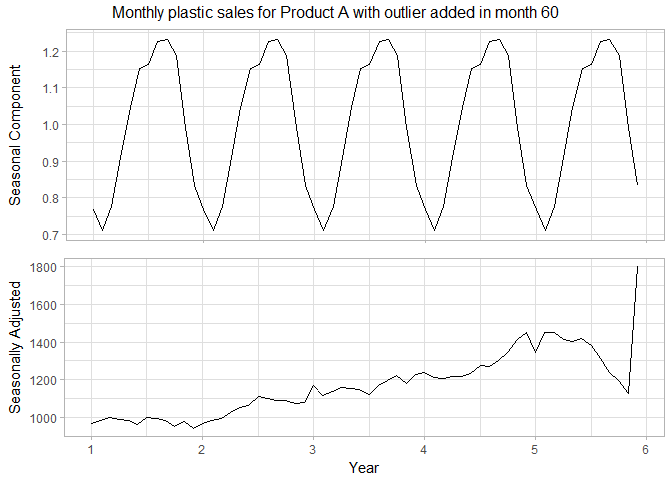


The addition of this outlier is observed in the seasonally adjusted data. The change at in the seasonally adjusted data is less than the raw change to the value, indicating that some of the change is incorporated into the seasonal data.

## Part f

Performing a similar manipulation as in parts d and e, this time changing the final value of the initial time series from 1013 to 1500:

out\_end\_plastics <- plastics  
out\_end\_plastics[length(plastics)] <- 1500  
  
fit\_out\_end\_plastics <- decompose(out\_end\_plastics, "m")  
gridExtra::grid.arrange(  
 autoplot(fit\_out\_end\_plastics$seasonal) +   
 labs(y = "Seasonal Component\n", x = NULL) +  
 scale\_x\_continuous(labels = NULL, minor\_breaks = seq(1, 6, 0.5)),  
 autoplot(seasadj(fit\_out\_end\_plastics)) +   
 labs(y = "Seasonally Adjusted", x = "Year"),  
 top = grid::textGrob("Monthly plastic sales for Product A with outlier added in month 60"))

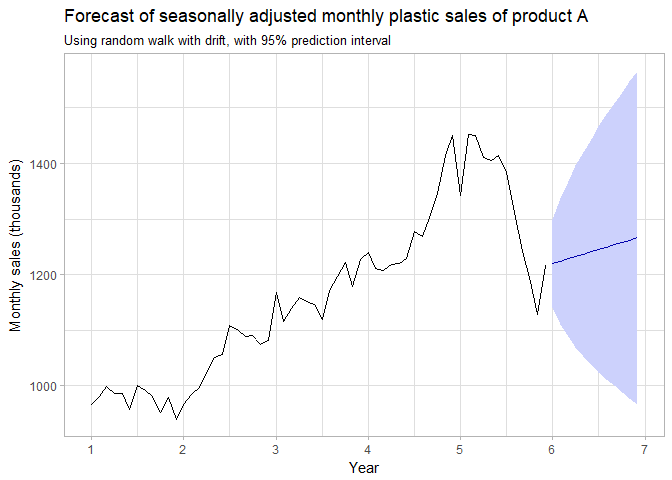


In this case, the seasonally adjusted data absorbs almost all of the change in the data, showing that the outlier at end of the period affects the seasonality of the time series less strongly than an outlier in the middle.

## Part g

A random walk with drift is performed on the seasonally adjusted plastic sales data from part d to forecast the next 12 months of seasonally adjusted data

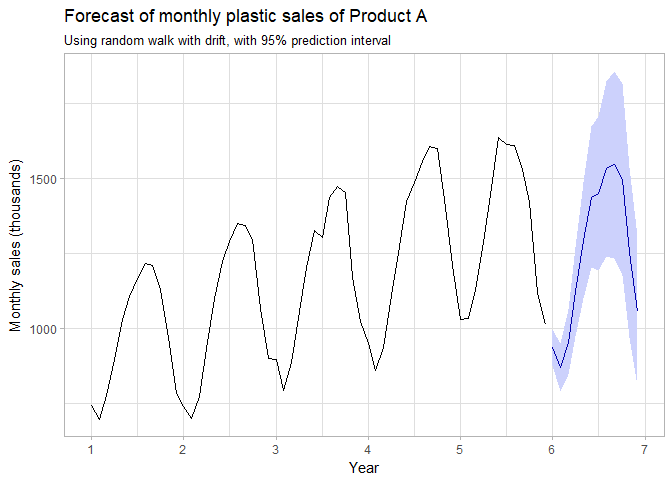
drift\_plastics <- rwf(adj\_plastics, drift = TRUE, h = 12, level = c(95))  
autoplot(drift\_plastics) +  
 scale\_x\_continuous(breaks = 1:7) +  
 labs(title = "Forecast of seasonally adjusted monthly plastic sales of product A",  
 subtitle = "Using random walk with drift, with 95% prediction interval",  
 x = "Year",  
 y = "Monthly sales (thousands)") +  
 theme(legend.position = "none")



## Part h

The reseasonalized results are given by multiplying the seasonal component (figure) of the decomposed series by multiplying the values (mean), and confidence bounds (lower and upper) of the forecast. These are plotted alongside the original values below:

reseas\_plastics <- drift\_plastics  
reseas\_plastics$x <- plastics  
reseas\_plastics$mean <- drift\_plastics$mean \* fit\_plastics$figure  
reseas\_plastics$upper <- drift\_plastics$upper \* fit\_plastics$figure  
reseas\_plastics$lower <- drift\_plastics$lower \* fit\_plastics$figure  
  
autoplot(reseas\_plastics) +  
 scale\_x\_continuous(breaks = 1:7) +  
 labs(title = "Forecast of monthly plastic sales of Product A",  
 subtitle = "Using random walk with drift, with 95% prediction interval",  
 x = "Year",  
 y = "Monthly sales (thousands)") +  
 theme(legend.position = "none")



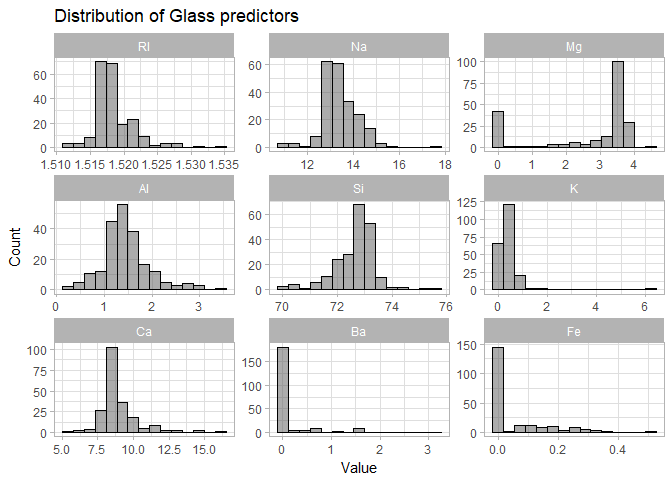
# KJ 3.1

data("Glass", package = "mlbench")

## Parts a & b

The distributions of each predictor are shown below:

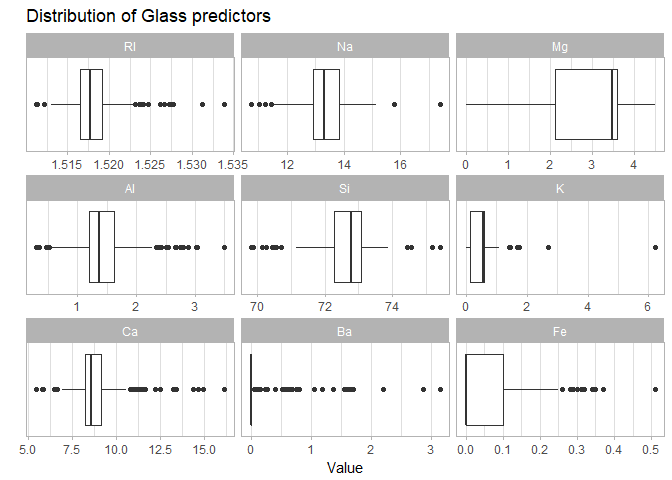
Glass %>%  
 gather(Predictor, Value, -Type) %>%  
 mutate(Predictor = forcats::as\_factor(Predictor)) %>%  
 ggplot(aes(x = Value)) +  
 geom\_histogram(bins = 15, alpha = 0.5, col = "black") +  
 facet\_wrap(~ Predictor, scales = "free") +  
 labs(title = "Distribution of Glass predictors",  
 y = "Count")



It can be seen that two predictors (Ba and Fe) have a very high number of zero or near-zero observations and are heavily right-skewed. This same right-skewness is shown in predictor K, and to a lesser degree by predictor Ca. Predictors RI and NA show a much smaller possible right-skewness. Predictor Mg shows a left-skewness accompanied by a large number of zero-value observations.

It appears that there may be outliers in the extremely skewed variables. Specifically, the strongly right-skewed K predictor ranges between 0 and 6, but only 3 of the 219 observations are greater than 3. This can be further investigated using boxplots:

Glass %>%  
 gather(Predictor, Value, -Type) %>%  
 mutate(Predictor = forcats::as\_factor(Predictor)) %>%  
 ggplot(aes(x = Predictor, y = Value)) +  
 geom\_boxplot() + coord\_flip() +  
 facet\_wrap(~ Predictor, scales = "free") +  
 ggtitle("Distribution of Glass predictors") +  
 scale\_x\_discrete("", breaks = NULL, labels = NULL)



The boxplot highlights that there appear to be a number of outliers in each predictor except Mg. The number of outliers by predictor is calculated, where an outlier *x* is defined by the criteria , where *Q1*, *Q3* and *IQR* are the first quartile, third quartile, and interquartile range, respectively, of the predictor *X*.

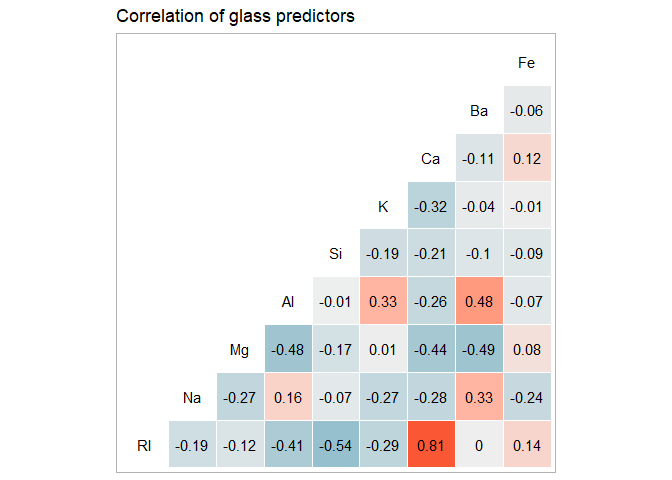
outliers\_glass <- Glass %>%   
 select(-Type) %>%   
 summarise\_all(funs(sum(  
 . < quantile(., 0.25) - 1.5 \* IQR(.) | . > quantile(., 0.75) + 1.5 \* IQR(.)  
 )))

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RI | Na | Mg | Al | Si | K | Ca | Ba | Fe |
| 17 | 7 | 0 | 18 | 12 | 7 | 26 | 38 | 12 |

This table shows that six predictors -- RI, Al, Si, Ca, Ba, and Fe -- have more than 11 outliers, corresponding to over 5% of observations.

The correlation between the predictors are shown in the plot below:

Glass %>%  
 GGally::ggcorr(label = TRUE, label\_round = 2) +  
 labs(title = "Correlation of glass predictors") +  
 theme(legend.position = "none")



From this visualization, it can be seen that there is a high positive correlation (0.81) between RI and Ca; the strongest positive correlation not involving the RI predictor is 0.48 between Al and Ba. The strongest negative correlation (-0.54) exists between RI and Si, followed by -0.49 between Ba and Mg. There are five predictor pairs with correlation of absolute value less than 0.05.

## Part c

For the skewed predictors identified above, a Box-Cox transformation may improve the classification model by reducing or removing the skew. For the six predictors with high concentrations of outliers, a spatial sign transformation may be useful if the model is sensitive to outliers.

# KJ 3.2

data("Soybean", package = "mlbench")

## Part a

The following predictors meet the degeneracy criteria set forth in the text:

* Fraction of unique values over sample size (Unique\_Ratio) under 10%
* Ratio of the most frequent value to second-most frequent (Freq\_Ratio) over 20

degen\_soybean <- Soybean %>%   
 # convert predictors to tidy format  
 select(-Class, -date) %>%   
 gather(Predictor, Value) %>%   
 # only consider complete cases  
 drop\_na() %>%  
 group\_by(Predictor, Value) %>%   
 # get ratio of most prevalent to second-most prevalent by variable  
 summarise(n = n()) %>%   
 mutate(Freq\_Ratio = max(n) / sort(n, decreasing = TRUE)[2]) %>%   
 # get fraction of unique values over number of observations  
 group\_by(Predictor, Freq\_Ratio) %>%   
 summarise(Unique\_Ratio = length(unique(Value)) / nrow(Soybean)) %>%   
 # filter for degenerate predictors  
 ungroup() %>%   
 filter(Freq\_Ratio > 20, Unique\_Ratio < 0.10)

|  |  |  |
| --- | --- | --- |
| **Predictor** | **Freq\_Ratio** | **Unique\_Ratio** |
| leaf.mild | 26.75 | 0.004392 |
| mycelium | 106.5 | 0.002928 |
| sclerotia | 31.25 | 0.002928 |

## Part b

The predictors are shown below by the portion of observations with missing data:

missing\_predictor <- Soybean %>%   
 # convert to tidy format  
 select(-Class) %>%   
 gather(Predictor, Value) %>%   
 # get number missing by predictor  
 group\_by(Predictor) %>%   
 summarize(Missing = sum(is.na(Value))) %>%   
 # divide by total observations  
 mutate(Missing = Missing / nrow(Soybean)) %>%   
 arrange(desc(Missing))

|  |  |
| --- | --- |
| **Predictor** | **Missing** |
| hail | 0.1772 |
| lodging | 0.1772 |
| seed.tmt | 0.1772 |
| sever | 0.1772 |
| germ | 0.164 |
| leaf.mild | 0.1581 |
| fruit.spots | 0.1552 |
| fruiting.bodies | 0.1552 |
| seed.discolor | 0.1552 |
| shriveling | 0.1552 |
| leaf.shread | 0.1464 |
| mold.growth | 0.1347 |
| seed | 0.1347 |
| seed.size | 0.1347 |
| fruit.pods | 0.123 |
| leaf.halo | 0.123 |
| leaf.malf | 0.123 |
| leaf.marg | 0.123 |
| leaf.size | 0.123 |
| canker.lesion | 0.05564 |
| ext.decay | 0.05564 |
| int.discolor | 0.05564 |
| mycelium | 0.05564 |
| precip | 0.05564 |
| sclerotia | 0.05564 |
| stem.cankers | 0.05564 |
| plant.stand | 0.05271 |
| roots | 0.04539 |
| temp | 0.04392 |
| crop.hist | 0.02343 |
| plant.growth | 0.02343 |
| stem | 0.02343 |
| area.dam | 0.001464 |
| date | 0.001464 |
| leaves | 0 |

The predictors hail, lodging, seed.tmt, and sever have the highest proportion of missing values (17.72%), closely followed by germ (16.4%). The leaves predictor has no missing values, while area.dam and date have roughly 0.15% missing, corresponding to a single missing value.

Similarly, the proportion of missing predictor values with missing data is shown below by class:

missing\_class <- Soybean %>%   
 # convert to tidy format  
 gather(Predictor, Value, -Class) %>%   
 # get number missing by class  
 group\_by(Class) %>%   
 summarize(Missing = sum(is.na(Value))) %>%   
 # divide by observations \* predictors  
 mutate(Missing = Missing / (nrow(Soybean) \* 35)) %>%   
 arrange(desc(Missing))

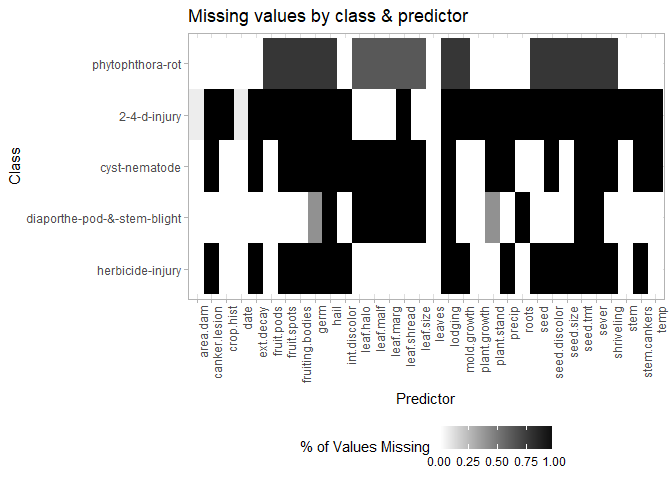
|  |  |
| --- | --- |
| **Class** | **Missing** |
| phytophthora-rot | 0.05078 |
| 2-4-d-injury | 0.01882 |
| cyst-nematode | 0.01406 |
| diaporthe-pod-&-stem-blight | 0.007404 |
| herbicide-injury | 0.006693 |
| alternarialeaf-spot | 0 |
| anthracnose | 0 |
| bacterial-blight | 0 |
| bacterial-pustule | 0 |
| brown-spot | 0 |
| brown-stem-rot | 0 |
| charcoal-rot | 0 |
| diaporthe-stem-canker | 0 |
| downy-mildew | 0 |
| frog-eye-leaf-spot | 0 |
| phyllosticta-leaf-spot | 0 |
| powdery-mildew | 0 |
| purple-seed-stain | 0 |
| rhizoctonia-root-rot | 0 |

The *phytophthora-rot* shows over 5% missing values; the *2-4-d-injury* and *cyst-nematode* show between 1-2% missing values; the *diaporthe-pod-&-stem-blight* and *herbicide-injury* show under 1% missing values; the remaining 14 classes show zero missing values. Note that these percentages are an indication of the number of observations of each variable missing, i.e. the number of missing data points rather than observations that contain missing data points.

## Part c

As outlined in Part a, the variables *leaf.mild*, *mycelium*, and *sclerotia* are degenerate -- these variables should be removed. To investigate the appropriate approach for additional missing variables, the patterns of missing values across predictors and classes are investigated:

# get degenerative predictors  
predictors\_degen <- degen\_soybean %>% pull(Predictor)  
  
# get classes with missing values  
classes\_missing <- missing\_class %>% filter(Missing > 0) %>%  
 pull(Class) %>% as.character()  
  
# calculate missing values by predictor & class, then plot  
Soybean %>%   
 # convert to tidy format, select missing classes, exclude degen predictors  
 gather(Predictor, Value, -Class) %>%   
 filter(Class %in% classes\_missing,  
 !Predictor %in% predictors\_degen) %>%   
 # get number missing by predictor & class  
 group\_by(Predictor, Class) %>%   
 summarize(n = n(), Missing = sum(is.na(Value))) %>%   
 # divide by observations \* predictors  
 mutate(Missing = Missing / n) %>%   
 ungroup() %>%  
 # plot tiled results (heatmap)  
 ggplot(aes(x = Predictor, y = Class, fill = Missing)) +  
 geom\_tile() +  
 # tweak appearance  
 scale\_y\_discrete(limits = rev(classes\_missing)) +  
 scale\_fill\_gradient(name = "% of Values Missing", low = "white", high = "black") +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1),  
 legend.position = "bottom") +  
 ggtitle("Missing values by class & predictor")



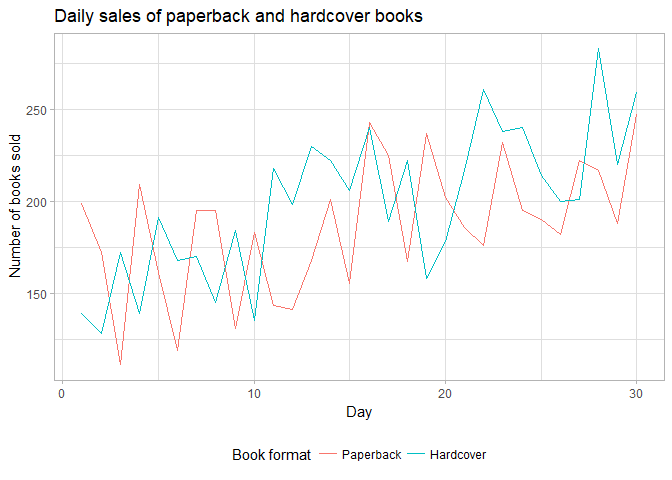
The classes *2-4-d-injury*, *cyst-nematode*, and *herbicide-injury* show nearly 100% missing values across all predictors. Imputation of values for these classes will likely not be helpful since so many of the observations are missing; as such, cases associated with these three classes should be removed from the dataset. Given the more moderate nature of the missing data patterns for the *phytophthora-rot* and *diaporthe-pod-&-stem-blight* classes, imputation for the missing values may be more informative and should be performed.

# HA 7.1

data("books")

## Part a

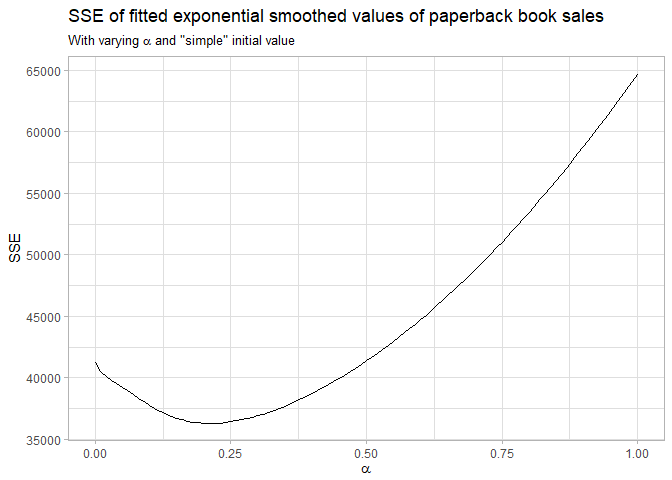
autoplot(books) +  
 labs(title = "Daily sales of paperback and hardcover books",  
 x = "Day", y = "Number of books sold") +  
 scale\_color\_discrete("Book format") +  
 theme(legend.position = "bottom")



The data for both book formats show an upward trend; a slight cyclical behavior is also apparent, though it does not appear to be seasonal.

## Part b

# separate paperback series  
paperback <- books[, "Paperback"]  
# create containers for results  
alpha\_paperback <- numeric()  
sse\_paperback <- numeric()  
# loop through possible values of alpha in increments of 0.01  
for (alpha in seq(0, 1, 0.01)) {  
 # get fit and associated SSE; save to list  
 fit <- ses(paperback, initial = "simple", alpha = alpha)  
 SSE <- sum((paperback - fit$fitted)^2)  
 # store alpha and SSE values  
 alpha\_paperback <- c(alpha\_paperback, alpha)  
 sse\_paperback <- c(sse\_paperback, SSE)  
}  
# combine into data frame and plot  
fits\_paperback <- data\_frame(alpha = alpha\_paperback, SSE = sse\_paperback)  
ggplot(fits\_paperback, aes(x = alpha, y = SSE)) +  
 geom\_line() +  
 labs(title = "SSE of fitted exponential smoothed values of paperback book sales",  
 subtitle = expression("With varying "\*alpha\*" and \"simple\" initial value"),  
 x = expression(alpha))



It is clear from the plot that the value of α affects the accuracy of forecasts. From the plot, it appears that the best value of α (corresponding to the lowest SSE) is between 0.125 and 0.25. This can be calculated to show that the SSE is optimized with α=0.21:

(paperback\_loop\_alpha <- fits\_paperback %>% slice(which.min(SSE)))

# A tibble: 1 x 2  
 alpha SSE  
 <dbl> <dbl>  
1 0.21 36314.59

## Parts c & d

Using ses without specifying an parameter returns a value similar to that observed above:

paperback\_simple <- ses(paperback, initial = "simple", h = 4)  
(paperback\_simple\_alpha <- paperback\_simple$model$par["alpha"])

alpha   
0.2125115

Performing this again with initial="optimal" yields a result that is a fair deal off:

paperback\_opt <- ses(paperback, initial = "optimal", h = 4)  
(paperback\_opt\_alpha <- paperback\_opt$model$par["alpha"])

alpha   
0.1685384

A comparison of the values and associated SSEs is presented below:

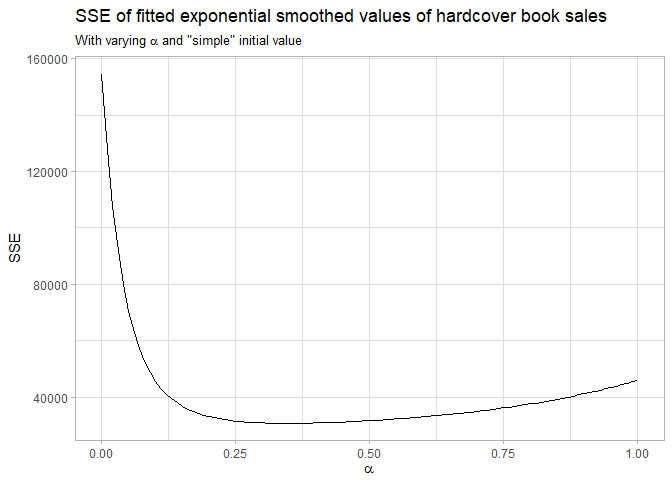
emphasize.strong.cols(1)  
data\_frame(Method = c("Loop", "SES simple", "SES optimal"),  
 alpha = c(paperback\_loop\_alpha$alpha,  
 paperback\_simple\_alpha,  
 paperback\_opt\_alpha),  
 SSE = c(paperback\_loop\_alpha$SSE,  
 sum((paperback\_simple$fitted - paperback)^2),  
 sum((paperback\_opt$fitted - paperback)^2))) %>%   
 pander()

|  |  |  |
| --- | --- | --- |
| **Method** | **alpha** | **SSE** |
| **Loop** | 0.21 | 36315 |
| **SES simple** | 0.2125 | 36314 |
| **SES optimal** | 0.1685 | 33945 |

It can be seen that although the values of differ between methods, the SSE does not vary tremendously.

## Hardcover

# separate paperback series  
hardcover <- books[, "Hardcover"]  
  
## part b -- loop approach  
# create containers for results  
alpha\_hardcover <- numeric()  
sse\_hardcover <- numeric()  
# loop through possible values of alpha in increments of 0.01  
for (alpha in seq(0, 1, 0.01)) {  
 # get fit and associated SSE; save to list  
 fit <- ses(hardcover, initial = "simple", alpha = alpha)  
 SSE <- sum((hardcover - fit$fitted)^2)  
 # store alpha and SSE values  
 alpha\_hardcover <- c(alpha\_hardcover, alpha)  
 sse\_hardcover <- c(sse\_hardcover, SSE)  
}  
# combine into data frame and plot  
fits\_hardcover <- data\_frame(alpha = alpha\_hardcover, SSE = sse\_hardcover)  
ggplot(fits\_hardcover, aes(x = alpha, y = SSE)) +  
 geom\_line() +  
 labs(title = "SSE of fitted exponential smoothed values of hardcover book sales",  
 subtitle = expression("With varying "\*alpha\*" and \"simple\" initial value"),  
 x = expression(alpha))



# get best value  
hardcover\_loop\_alpha <- fits\_hardcover %>% slice(which.min(SSE))  
  
## parts c & d -- ses selection  
# with initial = "simple"  
hardcover\_simple <- ses(hardcover, initial = "simple", h = 4)  
hardcover\_simple\_alpha <- hardcover\_simple$model$par["alpha"]  
# with initial = "optimal"  
hardcover\_opt <- ses(hardcover, initial = "optimal", h = 4)  
hardcover\_opt\_alpha <- hardcover\_opt$model$par["alpha"]

The resulting values of and associated SSE are shown below:

|  |  |  |
| --- | --- | --- |
| **Method** | **alpha** | **SSE** |
| **Loop** | 0.35 | 30758 |
| **SES simple** | 0.3473 | 30758 |
| **SES optimal** | 0.3283 | 30588 |

As for the paperback series, the value of with initial="simple" is very close to the value derived by iterating through a loop, and the value with initial="optimal" is a bit different than the other two, though to a lesser degree in this case. The SSE values vary even less for the hardcover series, and are nearly identical for the loop & SES simple methods.

# HA 7.2

# get linear holt fits for each series with 4-day forecast & default arguments  
paperback\_holt <- holt(paperback, h = 4)  
hardcover\_holt <- holt(hardcover, h = 4)

## Part a

The SSE measures for the above forecasts can be compared to the exponential smoothing forecasts prepared in part d of question 7.1 by squaring and summing the residuals:

# get SSE values for paperback forecasts  
sse\_opt\_paperback <- sum(paperback\_opt$residuals^2)  
sse\_holt\_paperback <- sum(paperback\_holt$residuals^2)  
  
# get SSE values for hardcover forecasts  
sse\_opt\_hardcover <- sum(hardcover\_opt$residuals^2)  
sse\_holt\_hardcover <- sum(hardcover\_holt$residuals^2)  
  
# gather results into matrix  
holt\_v\_opt <- matrix(c(sse\_opt\_paperback, sse\_holt\_paperback,  
 sse\_opt\_hardcover, sse\_holt\_hardcover),  
 nrow = 2, byrow = TRUE,  
 dimnames = list(c("Paperback", "Hardcover"), c("SES", "Holt")))

Comparison of SSE measure for SES & Holt methods

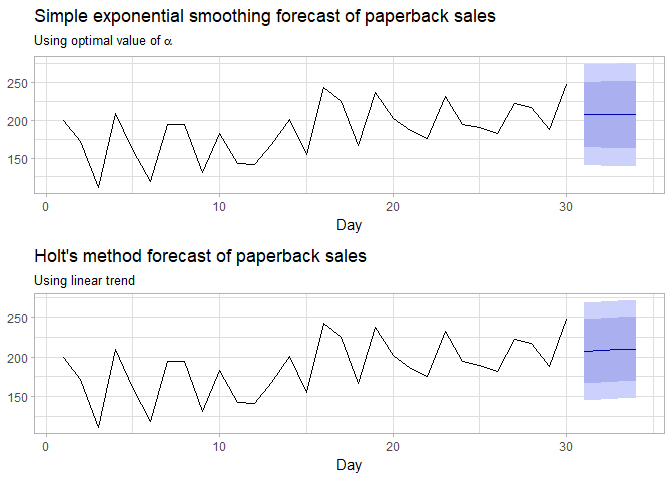
|  |  |  |
| --- | --- | --- |
|  | **SES** | **Holt** |
| **Paperback** | 33945 | 30074 |
| **Hardcover** | 30588 | 22582 |

The table clearly shows that Holt's method yields a much slower error in the forecast. This method is more appropriate for these two data series, as it is able to capture the upwards trend noted in part a of the previous question.

## Part b

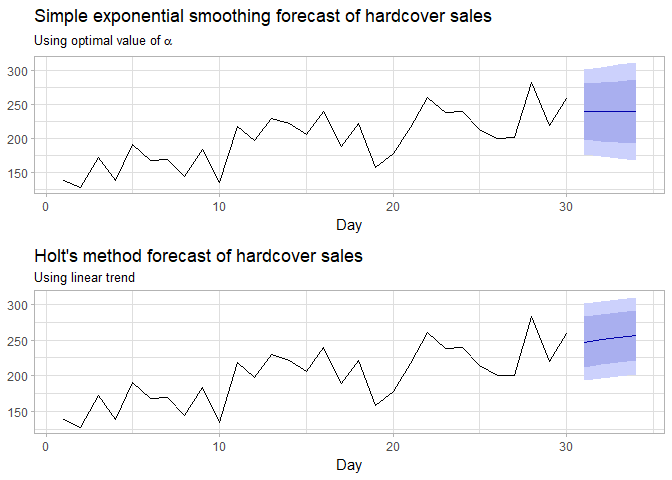
The two forecasts for the paperback series are presented below:

gridExtra::grid.arrange(  
 autoplot(paperback\_opt) +  
 labs(title = "Simple exponential smoothing forecast of paperback sales",  
 subtitle = expression("Using optimal value of "\*alpha),  
 x = "Day", y = NULL) +  
 theme(legend.position = "none"),  
 autoplot(paperback\_holt) +  
 labs(title = "Holt's method forecast of paperback sales",  
 subtitle = "Using linear trend",  
 x = "Day", y = NULL) +  
 theme(legend.position = "none"),  
 ncol = 1)



The two forecasts for the hardcover series are presented below:

gridExtra::grid.arrange(  
 autoplot(hardcover\_opt) +  
 labs(title = "Simple exponential smoothing forecast of hardcover sales",  
 subtitle = expression("Using optimal value of "\*alpha),  
 x = "Day", y = NULL) +  
 theme(legend.position = "none"),  
 autoplot(hardcover\_holt) +  
 labs(title = "Holt's method forecast of hardcover sales",  
 subtitle = "Using linear trend",  
 x = "Day", y = NULL) +  
 theme(legend.position = "none"),  
 ncol = 1)



For both series, the Holt's method linear forecast appears to better capture the upward trend in the data, as discussed in part b, while the simple exponential smoothing forecast fails to do so.

## Part c

The 95% prediction interval can be generated using the equation for each of the four forecasts:

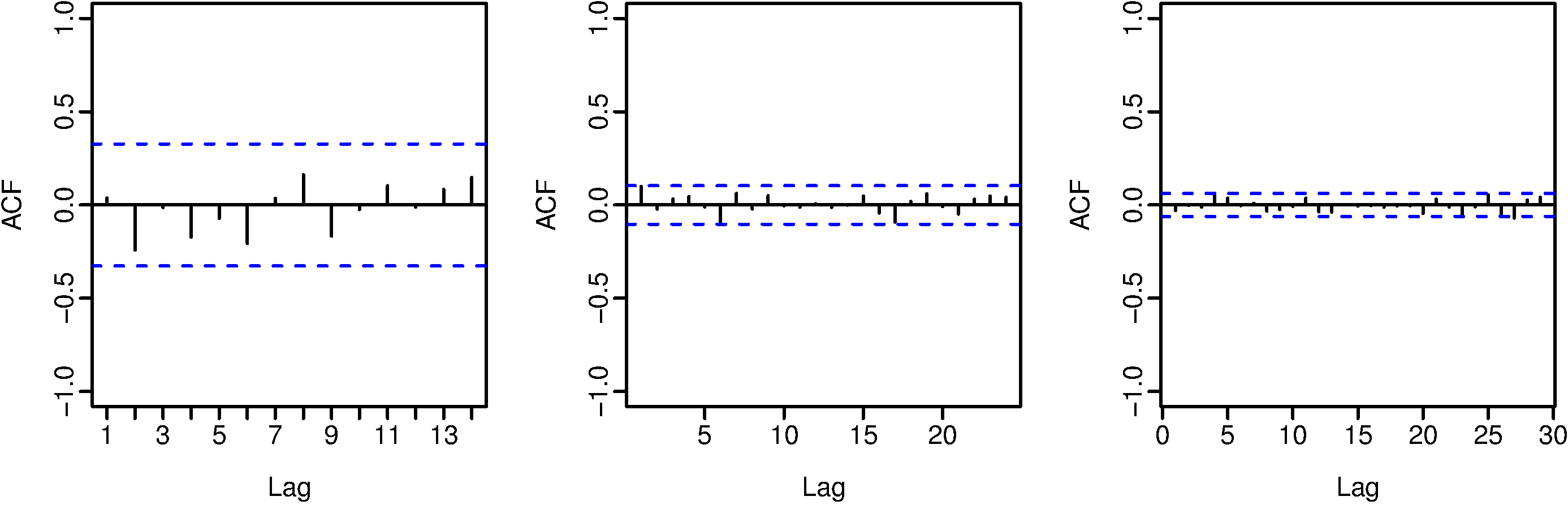
# manually calculate prediction intervals  
paper\_opt\_pi\_man <- c(paperback\_opt$mean[1] - 1.96 \* sqrt(paperback\_opt$model$sigma2),  
 paperback\_opt$mean[1] + 1.96 \* sqrt(paperback\_opt$model$sigma2))  
paper\_holt\_pi\_man <- c(paperback\_holt$mean[1] - 1.96 \* sqrt(paperback\_holt$model$sigma2),  
 paperback\_holt$mean[1] + 1.96 \* sqrt(paperback\_holt$model$sigma2))  
hard\_opt\_pi\_man <- c(hardcover\_opt$mean[1] - 1.96 \* sqrt(hardcover\_opt$model$sigma2),  
 hardcover\_opt$mean[1] + 1.96 \* sqrt(hardcover\_opt$model$sigma2))  
hard\_holt\_pi\_man <- c(hardcover\_holt$mean[1] - 1.96 \* sqrt(hardcover\_holt$model$sigma2),  
 hardcover\_holt$mean[1] + 1.96 \* sqrt(hardcover\_holt$model$sigma2))  
  
# extract prediction intervals calculated by R  
paper\_opt\_pi\_r <- unname(c(paperback\_opt$lower[1, "95%"], paperback\_opt$upper[1, "95%"]))  
paper\_holt\_pi\_r <- unname(c(paperback\_holt$lower[1, "95%"], paperback\_holt$upper[1, "95%"]))  
hard\_opt\_pi\_r <- unname(c(hardcover\_opt$lower[1, "95%"], hardcover\_opt$upper[1, "95%"]))  
hard\_holt\_pi\_r <- unname(c(hardcover\_holt$lower[1, "95%"], hardcover\_holt$upper[1, "95%"]))

# collect into matrix  
pi\_matrix <- matrix(c(paper\_opt\_pi\_man, paper\_opt\_pi\_r,  
 paper\_holt\_pi\_man, paper\_holt\_pi\_r,  
 hard\_opt\_pi\_man, hard\_opt\_pi\_r,  
 hard\_holt\_pi\_man, hard\_holt\_pi\_r),  
 ncol = 4, byrow = TRUE,  
 dimnames = list(c("Paperback SES", "Paperback Holt",  
 "Hardcover SES", "Hardcover Holt"),  
 c("Manual Lower", "Manual Upper",  
 "R Lower", "R Upper")))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Manual Lower** | **Manual Upper** | **R Lower** | **R Upper** |
| **Paperback SES** | 141.2 | 273 | 141.2 | 273 |
| **Paperback Holt** | 145 | 269.1 | 145 | 269.1 |
| **Hardcover SES** | 177 | 302.1 | 177 | 302.1 |
| **Hardcover Holt** | 193.6 | 301.1 | 193.6 | 301.1 |

As seen in the table above, the manually-calculated prediction intervals are equivalent to those calculated by R.

# HA 8.1



The three ACF figures show different critical values indicating autocorrelation, as well as a differing number of lags along the x axis. Both of these differences are due to the number of observations in the samples -- the critical values scale by a factor of .

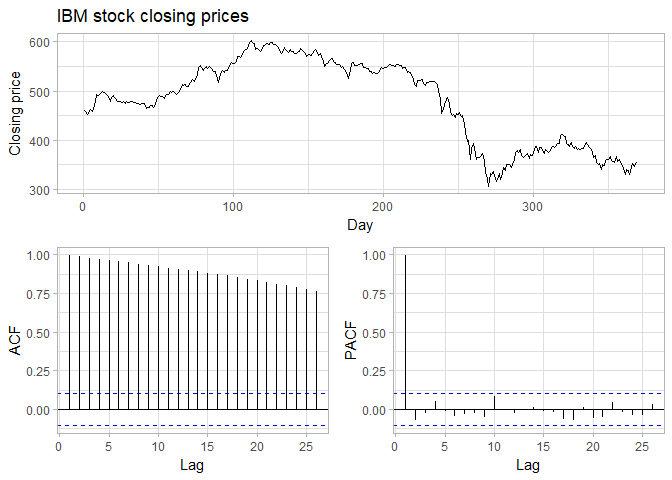
All three figures indicate that the associated data are white noise, as all vertical lines fall within the critical values indicated by the dotted blue lines.

# HA 8.2

Explain how each plot shows the series is non-stationary and should be differenced.

The plot of IBM closing stock prices, as well as the ACF and PACF of the series are presented below:

ggtsdisplay(ibmclose, main = "IBM stock closing prices", points = FALSE,  
 xlab = "Day", ylab = "Closing price")



The plot of the time series shows some clear trends -- first upwards, then downwards, then upwards again followed by a decline near the end; these trends make the series non-stationary. The ACF plot shows all values below the critical value; this is indicative of a random walk and therefore non-stationary.

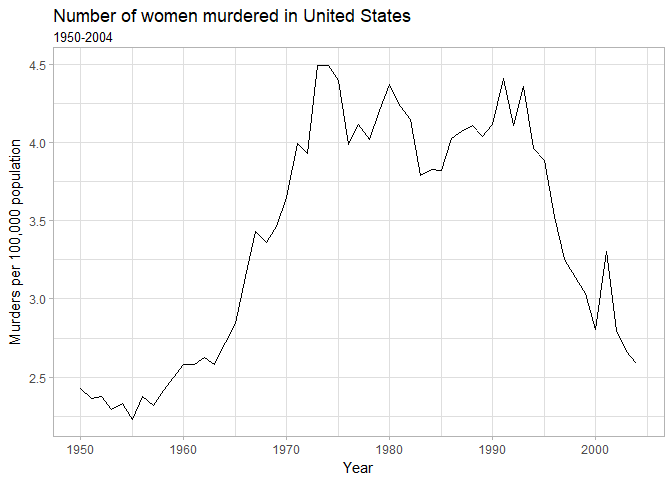
# HA 8.6

data("wmurders", package = "fpp")

## Parts a-c

The time series is plotted below:

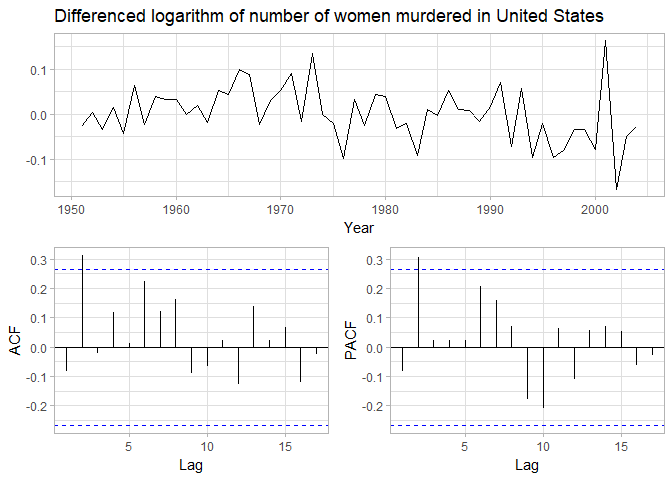
autoplot(wmurders) +  
 labs(title = "Number of women murdered in United States", subtitle = "1950-2004",  
 x = "Year", y = "Murders per 100,000 population")



The variance of the series appears to increase with the level of the time series -- as such, the series will be log-transformed. The series is clearly non-stationary, as it displays trend behavior. There is no apparent seasonality, so a simple difference is taken.

The transformed, differenced series is plotted below, alongside ACF & PACF figures:

ggtsdisplay(diff(log(wmurders)),  
 main = "Differenced logarithm of number of women murdered in United States",  
 xlab = "Year", points = FALSE)



This series appears sufficiently stationary. There are spikes at in both the ACF and PACF figures, suggesting that a AR(2) or MA(2) model is appropriate for the differenced data. The former is selected, yielding an ARIMA(2, 1, 0) model. There does not appear to be any drift in the original dataset, so a constant is not included in the model. This model can be written in backshift notation as

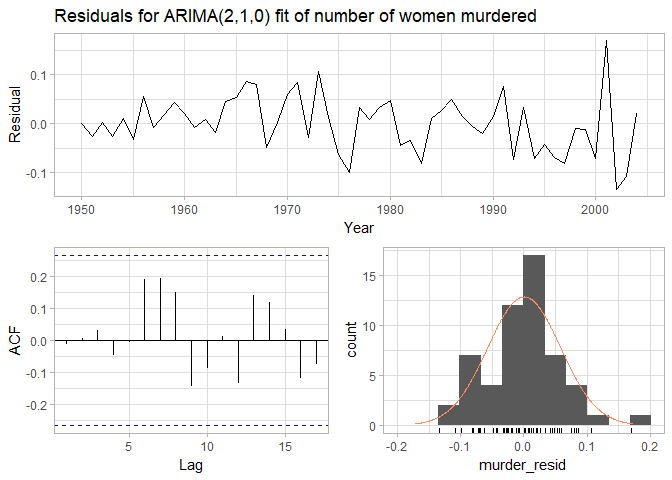
## Part d

A log transformation is equivalent to a Box-Cox transformation with , so this parameter is fed into the Arima function:

murder\_fit <- Arima(wmurders, order = c(2, 1, 0), lambda = 0)

The residuals of this fit are explored below:

murder\_resid <- residuals(murder\_fit)  
ggtsdisplay(murder\_resid, plot.type = "histogram", points = FALSE,  
 main = "Residuals for ARIMA(2,1,0) fit of number of women murdered",  
 xlab = "Year", ylab = "Residual")



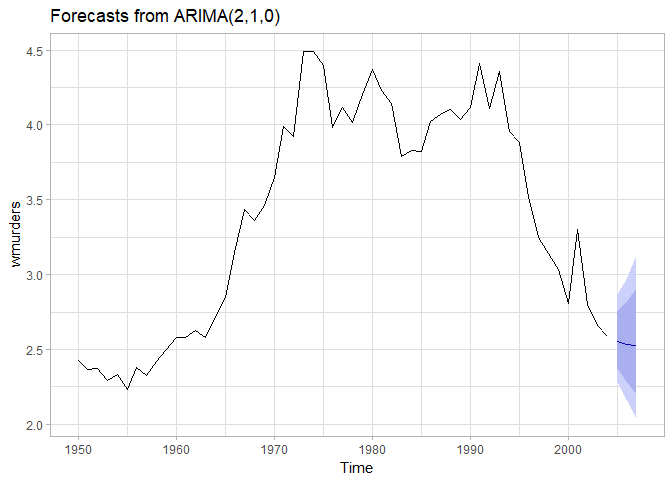
From the top plot, it can be seen that the residuals have a near-zero mean, and the variance in the residuals is near-constant; the ACF plot shows there is no significant correlation between residuals; the histogram and accompanying density plot appear to approximate a normal distribution. From these results, the model appears satisfactory -- this is confirmed using a portmanteau test (using :

Box.test(murder\_resid, type = "L", fitdf = 2, lag = 15)

Box-Ljung test  
  
data: murder\_resid  
X-squared = 12.39, df = 13, p-value = 0.496

## Parts e & f

murder\_fcast <- forecast(murder\_fit, 3)  
autoplot(murder\_fcast) + theme(legend.position = "none")



The forecast values, rounded to 3 decimal places, are given below:

|  |  |
| --- | --- |
|  | Value |
| **2005** | 2.555 |
| **2006** | 2.535 |
| **2007** | 2.526 |

### Manual Calculation

Each of the forecasts are checked manually below:

The equation from part c can be expanded to be rewritten as

The backshift operator can be applied and the equation rearranged to give

This can now be rearranged to isolate for and used to forecast for the next 3 periods, using the log-transformed series :

To transfer back to the original scale,

For the next prediction,

Finally,

It can be seen that the derived forecasts match those generated by forecast.

## Part g

An auto.arima fit is created for the data, setting to log-transform the data setting the seasonal option to false:

(murder\_auto <- auto.arima(wmurders, lambda = 0, seasonal = FALSE))

Series: wmurders   
ARIMA(1,2,1)   
Box Cox transformation: lambda= 0   
  
Coefficients:  
 ar1 ma1  
 -0.2954 -0.7892  
s.e. 0.1533 0.1190  
  
sigma^2 estimated as 0.003613: log likelihood=74.09  
AIC=-142.17 AICc=-141.68 BIC=-136.26

The auto.arima model fit is an ARIMA(1, 2, 1) consistently with modifying function arguments. This obviously differs from the ARIMA(2, 1, 0) used in this problem.

The ARIMA(2, 1, 0) model shows a better and the auto.arima model has , this comparison is not informative.

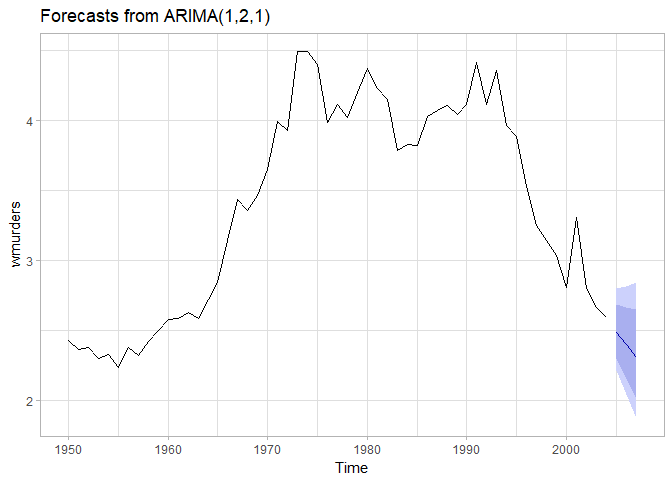
To compare the validity of the forecasts of two ARIMA models despite different model characteristics, a test set is used to calculate the RMSE on the tail portion of the data. The getrmse function from the text is used for this:

# definte RMSE function  
getrmse <- function(x, h, ...) {  
 train.end <- time(x)[length(x) - h]  
 test.start <- time(x)[length(x) - h + 1]  
 train <- window(x, end = train.end)  
 test <- window(x, start = test.start)  
 fit <- Arima(train, ...)  
 fc <- forecast(fit, h = h)  
 accuracy(fc, test)[2, "RMSE"]  
}  
  
# get RMSEs using test set of last 5 years  
RMSEs <- c(getrmse(wmurders, 3, order = c(2, 1, 0), lambda = 0),  
 getrmse(wmurders, 3, order = c(1, 2, 1), lambda = 0))  
names(RMSEs) <- c("manually selected", "auto.arima selected")

|  |  |
| --- | --- |
| **manually selected** | **auto.arima selected** |
| 0.6537 | 0.4591 |

The table above shows that the auto.arima selected model performs better on the test set, but inspection of a forecast generated by the model reveals a potential issue:

autoplot(forecast(murder\_auto, 3)) + theme(legend.position = "none")



From the plot above, it can be seen that the ARIMA(1, 2, 1) model extends the drop in rate in recent years continuously. This is probably less realistic than the ARIMA(2, 1, 0) model's prediction of a steadying of the rate. It also may explain the higher RMSE of this model against the test set, since the years of the test set illustrate a downward trend. For the sake of practicality and extensibility, the ARIMA(2, 1, 0) model should be used, even if it means a slight tradeoff in accuracy.

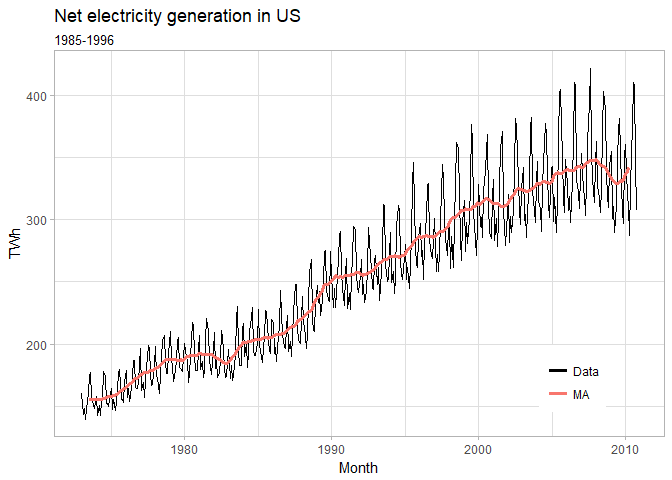
# HA 8.8

data("usmelec", package = "fpp")

## Part a

The raw data and the moving average are presented together in the plot below:

data\_frame(Month = time(usmelec),  
 TWh = as.numeric(usmelec),  
 MA = ma(usmelec, order = 12)) %>%   
 ggplot(aes(x = Month)) +  
 geom\_line(aes(y = TWh, col = "Data")) +  
 geom\_line(aes(y = MA, col = "MA"), lwd = 1.25) +  
 scale\_color\_manual(NULL, breaks = c("Data", "MA"), values = c("black", "#F8766D")) +  
 theme(legend.position = c(0.85, 0.15)) +  
 ggtitle(label = "Net electricity generation in US", subtitle = "1985-1996")



It is clear from the moving average in the plot that electricity generation has a stable upward trend.

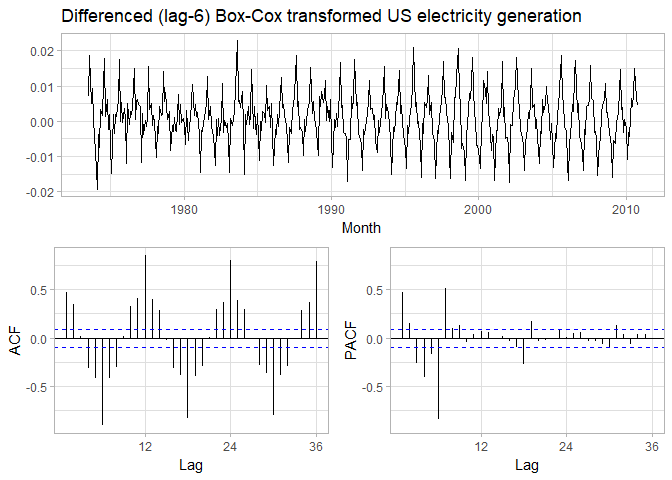
## Parts b & c

The variation in the data appears to increase; therefore a transformation is investigated:

elec\_lambda <- BoxCox.lambda(usmelec)  
elec\_trans <- BoxCox(usmelec, elec\_lambda)

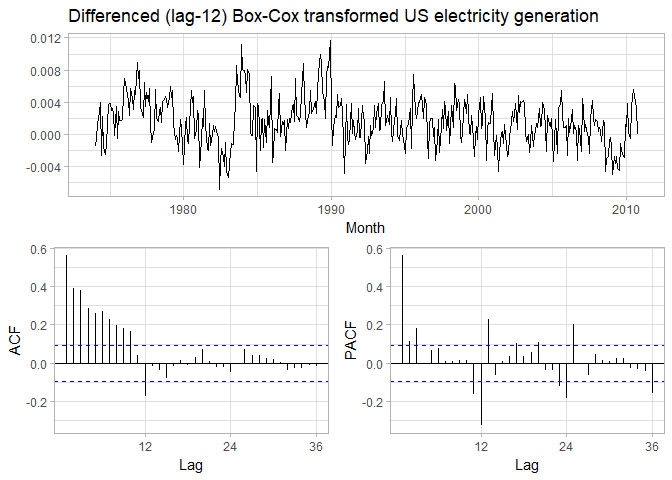
The data is transformed using the recommended pattern of . As identified above, the data is not stationary -- this will not be fixed by the Box-Cox transformation, so differencing is required. As stated in the text, the series shows two annual peaks; therefore a lag-6 differencing is attempted first:

elec\_diff <- diff(elec\_trans, 6)  
ggtsdisplay(elec\_diff, points = FALSE,  
 main = "Differenced (lag-6) Box-Cox transformed US electricity generation",  
 xlab = "Month")



There is still apparent seasonality present in this plot, and the sinusoidal behavior of the ACF plot clearly indicates a lack of stationary behavior; as such, lag-12 differencing is used and checked for stationarity:

elec\_diff <- diff(elec\_trans, 12)  
ggtsdisplay(elec\_diff, points = FALSE,  
 main = "Differenced (lag-12) Box-Cox transformed US electricity generation",  
 xlab = "Month")



This lag has successfully eliminated the seasonality and created a stationary time series (as seen in the decaying behavior of the ACF plot). This makes sense, as the two spikes observed in a given year are not of equal magnitude.

## Part d

The PACF plot above shows a large spike at

The models are calculated and their AIC values returned:

# define function to create models & return AIC values  
elec\_aic <- function(p, d, q, P, D, Q) {  
 # create model with Box-Cox and specified ARIMA parameters; extract AIC  
 AIC(Arima(usmelec, order = c(p, d, q), seasonal = c(P, D, Q), lambda = elec\_lambda))  
}  
# create possible combinations of p & q  
elec\_pq <- expand.grid(p = 0:3, q = 0:3) %>% filter(p > 0 | q > 0)  
# calculate AIC for ARIMA(p, 0, q)(0, 1, 1)  
elec\_pq <- elec\_pq %>% mutate(aic = pmap\_dbl(list(p, 0, q, 0, 1, 1), elec\_aic))  
# return best AIC  
elec\_pq %>% slice(which.min(aic))

# A tibble: 1 x 3  
 p q aic  
 <int> <int> <dbl>  
1 2 1 -4266.492

The lowest AIC returned was for , so the best model is ARIMA(2, 0, 1)(0, 1, 1). This model is used going forward:

elec\_fit <- Arima(usmelec, order = c(2, 0, 1), seasonal = c(0, 1, 1), lambda = elec\_lambda)

## Part e

The parameters of the model, returned by elec\_fit$coef, are as follows:

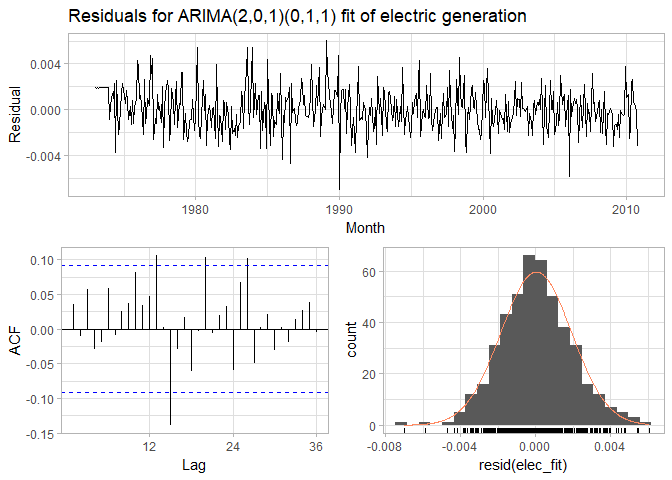
A Ljung-Box test returns a p-value > 0.05, suggesting that the residuals may be white noise:

Box.test(resid(elec\_fit), type = "L", fitdf = 4, lag = 5)

Box-Ljung test  
  
data: resid(elec\_fit)  
X-squared = 2.6494, df = 1, p-value = 0.1036

The residuals are further explored via diagnostic plotting:

ggtsdisplay(resid(elec\_fit), points = FALSE, plot.type = "histogram",  
 main = "Residuals for ARIMA(2,0,1)(0,1,1) fit of electric generation",  
 xlab = "Month", ylab = "Residual")

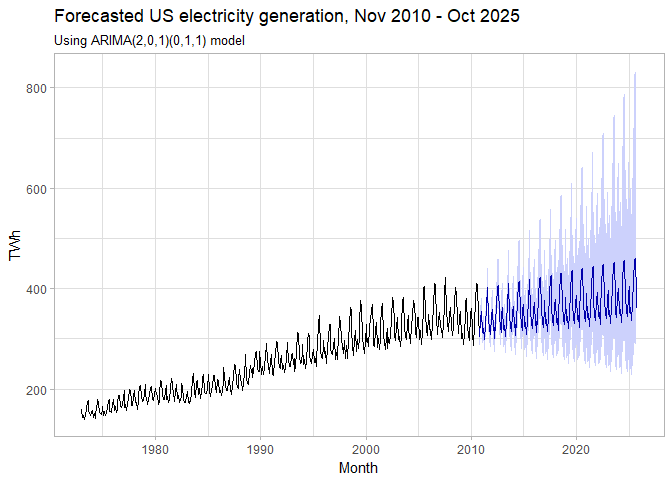


The residuals appear to be approximately normally distributed with a mean around zero. They do not appear to be autocorrelated, but there is a large spike at .

## Parts f & g

The forecast for the next 15 years (180 months) is created and plotted below:

elec\_fcast <- forecast(elec\_fit, 180, level = 95)  
autoplot(elec\_fcast) + theme(legend.position = "none") +  
 labs(title = "Forecasted US electricity generation, Nov 2010 - Oct 2025",  
 subtitle = "Using ARIMA(2,0,1)(0,1,1) model",  
 x = "Month", y = "TWh")



The forecast shows continued steady growth with the same seasonality observed in the data, with the 95% prediction interval showing the possibility of expanded, possibly exponential, growth or a slight decline.

The link provided in the text only provides data though June 2013; however, its original source, the [US EIA](https://www.eia.gov/totalenergy/data/monthly/#electricity) provides data through present day, accessed via Topic 7.2a. This data is read in and used to test for accuracy.

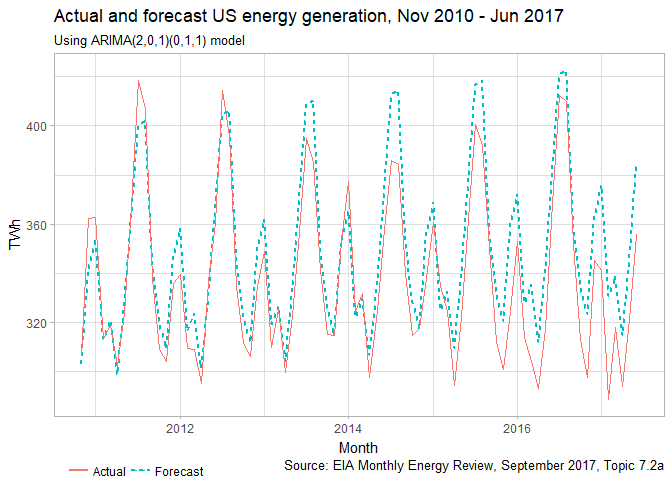
# read in data from EIA website  
elec\_new <- read\_csv("https://www.eia.gov/totalenergy/data/browser/csv.php?tbl=T07.02A")  
# extract relevant data  
elec\_new <- elec\_new %>%   
 # filter for dates after usmelec & generation by all fuel sources (Column\_Order 13)  
 # also remove annual totals (denoted by month 13)  
 filter(YYYYMM > 201010, Column\_Order == 13, substr(YYYYMM, 5, 6) != "13") %>%   
 # select needed columns: MSN & Desciprtion are redundant to Column\_Name; units identical  
 select(YYYYMM, Value) %>%   
 # convert from millions of kWh (GWh) to billions of kWh (TWh)  
 mutate(Value = Value / 1000)  
# convert to timeseries  
elec\_new <- ts(elec\_new$Value, start = c(2010, 11), frequency = 12)

The accuracy of the forecast can now be compared to the actuals (through June 2016) using the accuracy function:

|  |  |  |
| --- | --- | --- |
|  | **Training set** | **Test set** |
| **ME** | -0.07943 | -10.66 |
| **RMSE** | 7.098 | 16.09 |
| **MAE** | 5.21 | 13.35 |
| **MPE** | -0.006123 | -3.245 |
| **MAPE** | 2.017 | 3.975 |
| **MASE** | 0.5722 | 1.466 |
| **ACF1** | -0.01407 | 0.6098 |
| **Theil's U** | NA | 0.5239 |

The accuracy of these forecasts through the period containing available data is fairly good, with an RMSE on the test set of roughly 16 and an MAPE of 3.98%. When plotted, the quality of the forecast is also apparent:

# create data frame  
data\_frame(Month = time(elec\_new),  
 Actual = elec\_new,  
 Forecast = window(elec\_fcast$mean, end = max(time(elec\_new)))) %>%   
 ggplot(aes(x = Month)) +  
 geom\_line(aes(y = Forecast, col = "Forecast", lty = "Forecast"), lwd = 1) +  
 geom\_line(aes(y = Actual, col = "Actual", lty = "Actual")) +  
 scale\_color\_discrete("") + scale\_linetype\_discrete("") +  
 theme(legend.position = c(0, -0.15), legend.direction = "horizontal",  
 legend.justification = c(0, 0.5), legend.background = element\_blank()) +  
 labs(title = "Actual and forecast US energy generation, Nov 2010 - Jun 2017",  
 subtitle = "Using ARIMA(2,0,1)(0,1,1) model", y = "TWh",  
 caption = "Source: EIA Monthly Energy Review, September 2017, Topic 7.2a")



While the forecast is fairly accurate in the time window utilized here, it can be seen that it is becoming less accurate with time. It is likely that the forecast should not be used beyond the roughly 6-year horizon shown in the plot above, and almost definitely not to the full 15-year window generated here. At this duration, the forecast may be informative, but will likely not be accurate enough to be usable.

# KJ 6.3

data("ChemicalManufacturingProcess", package = "AppliedPredictiveModeling")  
*# split into predictors & response*  
chem\_pred <- ChemicalManufacturingProcess %>% select(-Yield)  
chem\_yield <- ChemicalManufacturingProcess %>% select(Yield)

## Parts b & c

### Imputation & Pre-Processing

First the total percentage of predictor cells is calculated:

sum(is.na(chem\_pred))

[1] 106

These 106 values correspond to roughly 1.06% of all predictor observations.

Following the practices established in Chapter 3, the predictors are investigated to determine if any have a relatively high number of missing values

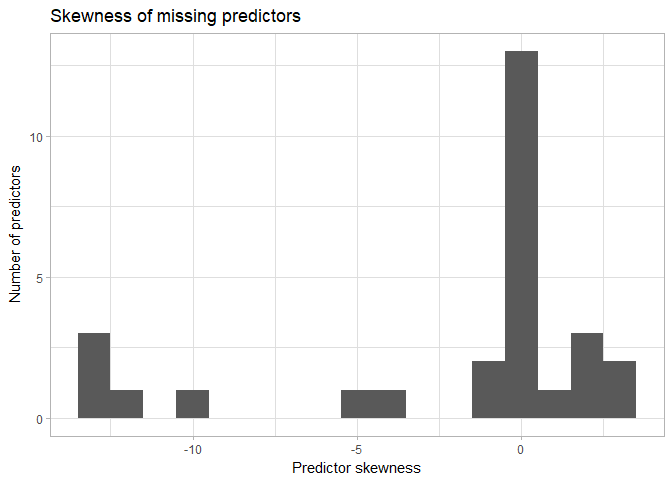
missing\_pred <- chem\_pred %>%  *# convert to tidy format* **gather**(Predictor, Value) %>%  *# get number missing by predictor* **group\_by**(Predictor) %>%  **summarize**(Missing = **sum**(**is.na**(Value))) %>%  **filter**(Missing > 0) %>%  **arrange**(**desc**(Missing))

|  |  |
| --- | --- |
| Predictor | Missing |
| **ManufacturingProcess03** | 15 |
| **ManufacturingProcess11** | 10 |
| **ManufacturingProcess10** | 9 |
| **ManufacturingProcess25** | 5 |
| **ManufacturingProcess26** | 5 |
| **ManufacturingProcess27** | 5 |
| **ManufacturingProcess28** | 5 |
| **ManufacturingProcess29** | 5 |
| **ManufacturingProcess30** | 5 |
| **ManufacturingProcess31** | 5 |
| **ManufacturingProcess33** | 5 |
| **ManufacturingProcess34** | 5 |
| **ManufacturingProcess35** | 5 |
| **ManufacturingProcess36** | 5 |
| **ManufacturingProcess02** | 3 |
| **ManufacturingProcess06** | 2 |
| **ManufacturingProcess01** | 1 |
| **ManufacturingProcess04** | 1 |
| **ManufacturingProcess05** | 1 |
| **ManufacturingProcess07** | 1 |
| **ManufacturingProcess08** | 1 |
| **ManufacturingProcess12** | 1 |
| **ManufacturingProcess14** | 1 |
| **ManufacturingProcess22** | 1 |
| **ManufacturingProcess23** | 1 |
| **ManufacturingProcess24** | 1 |
| **ManufacturingProcess40** | 1 |
| **ManufacturingProcess41** | 1 |

From the table above, it can be seen that three values (*ManufacturingProcess03*, *ManufacturingProcess11*, and *ManufacturingProcess10*) have the largest number of missing values (15, 10, 9), respectively, all representing over 5% of values missing. It is worth noting that all 9 observations missing *ManufacturingProcess10* are also missing the *ManufacturingProcess03* and *ManufacturingProcess11* -- the physical meanings of these variables may suggest further investigation into the relationship between these variables. There are another 11 predictors with 5 missing values and 12 predictors with a single missing value. The remaining 29 predictors have no missing values. Notably, all 28 predictors with missing values describe manufacturing processes.

To investigate if the simple median imputation method will be sufficient for the predictors with missing values, the skewness of the predictors is investigated:

chem\_pred %>% **summarize\_at**(**vars**(missing\_pred$Predictor), **funs**(e1071::**skewness**(., na.rm = TRUE))) %>% **slice**(1) %>% **unlist**() %>% **qplot**(binwidth = 1) + **labs**(title = "Skewness of missing predictors", x = "Predictor skewness", y = "Number of predictors")



The plot above shows that many of the missing predictors show a negative skewness, with some being quite strong. Because of this, imputation using the median may not accurately capture variability in the data. Therefore, k-nearest neighbor imputation will be performed to better capture patterns in the data.

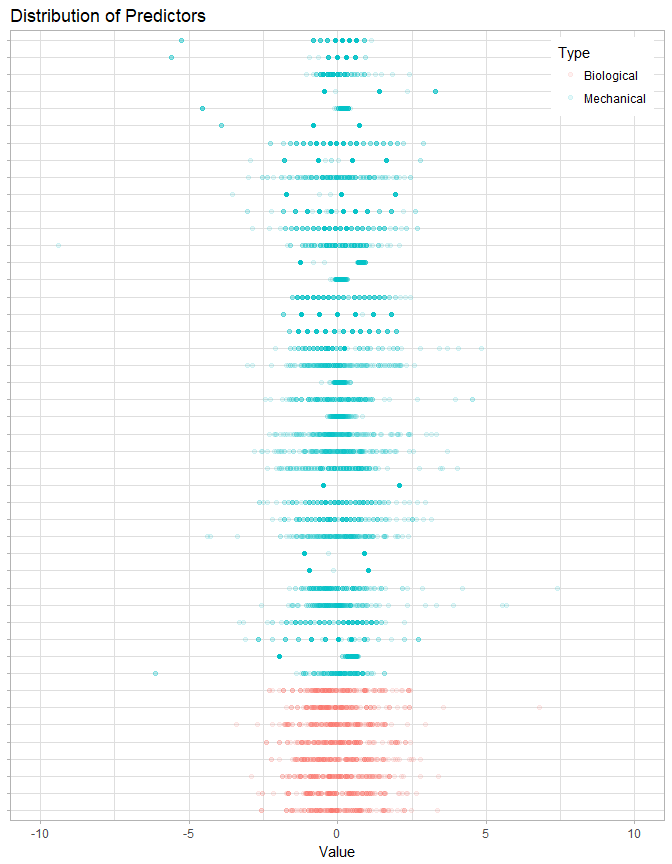
Before imputation, the data will be pre-processed. As shown above, many of the predictors are heavily skewed. To lessen the effects of this skewness on the model that will be developed, the data (both predictors and yield) will be Box-Cox transformed, centered, and scaled. Due to the large number of predictors, predictors with high correlation and/or near-zero variance will also be eliminated. The preProcess function is capable of performing all of these operations -- per the documentation for this function:

The operations are applied in this order: zero-variance filter, near-zero variance filter, correlation filter, Box-Cox/Yeo-Johnson/exponential transformation, centering, scaling, range, imputation, PCA, ICA then spatial sign.

library(caret)  
*# set up pre-processing transformation*  
chem\_preproc <- preProcess(chem\_pred, method = c("knnImpute", "center", "scale",  
 "nzv", "corr"))  
*# apply pre-processing to data*  
chem\_pred\_trans <- predict(chem\_preproc, chem\_pred)

Of the original 57 predictors, 46 remain for the model. These pre-processed predictors can be viewed to confirm the removal of skewness:

chem\_pred\_trans %>% **gather**(Predictor, Value) %>% **mutate**(Type = **ifelse**(**substr**(Predictor, 1, 1) == "B", "Biological", "Mechanical"), Number = **substr**(Predictor, **nchar**(Predictor) - 1, **nchar**(Predictor)))%>% **ggplot**(**aes**(x = Predictor, y = Value)) +  **geom\_point**(**aes**(col = Type), alpha = 0.1) + **scale\_x\_discrete**(NULL, labels = NULL) + **scale\_y\_continuous**(limits = **c**(-10, 10)) + **coord\_flip**() + **ggtitle**("Distribution of Predictors") + **theme**(legend.position = **c**(1, 1), legend.justification = **c**(1.1, 1.1))



From the plot two things are clear: skewness has been significantly reduced (though some outliers remain); and some mechanical predictors are discrete, taking only certain values.

### 

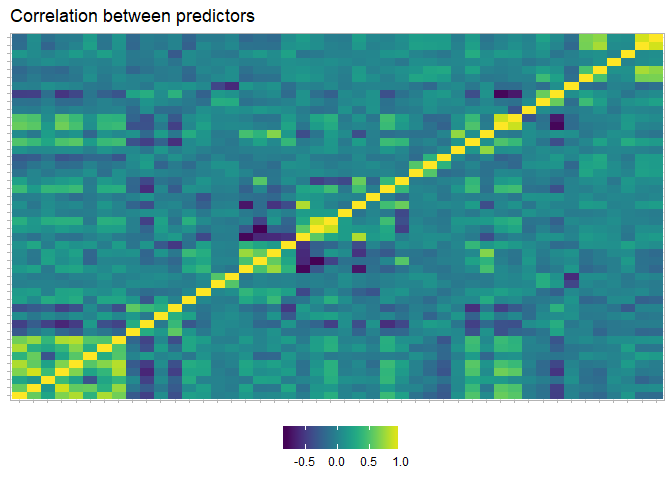
### Model Tuning

Before tuning a model, the predictor and yield data are split into training and test sets:

# get rows for training subsets **set.seed**(42) # for replicabilitytrain\_rows <- **createDataPartition**(chem\_yield$Yield, p = 0.75, list = FALSE)# create training setschem\_pred\_train <- chem\_pred\_trans[train\_rows, ]chem\_yield\_train <- chem\_yield[train\_rows, ]# creae test setschem\_pred\_test <- chem\_pred\_trans[-train\_rows, ]chem\_yield\_test <- chem\_yield[-train\_rows, ]

The correlation plot below shows that, while highly-correlated values were removed in pre-processing, some correlation between variables still exists:

# get correlations **cor**(chem\_pred\_trans) %>% # convert to data frame **as.data.frame**() %>% # change row names to variable **rownames\_to\_column**("PredictorX") %>% # convert to tidy format for plotting **gather**(PredictorY, Cor, -PredictorX) %>% # plot  **ggplot**(**aes**(PredictorX, PredictorY)) + **geom\_raster**(**aes**(fill = Cor)) + **scale\_x\_discrete**(NULL, labels = NULL) + **scale\_y\_discrete**(NULL, labels = NULL) + viridis::**scale\_fill\_viridis**(NULL) + **ggtitle**("Correlation between predictors") + **theme**(legend.position = "bottom")

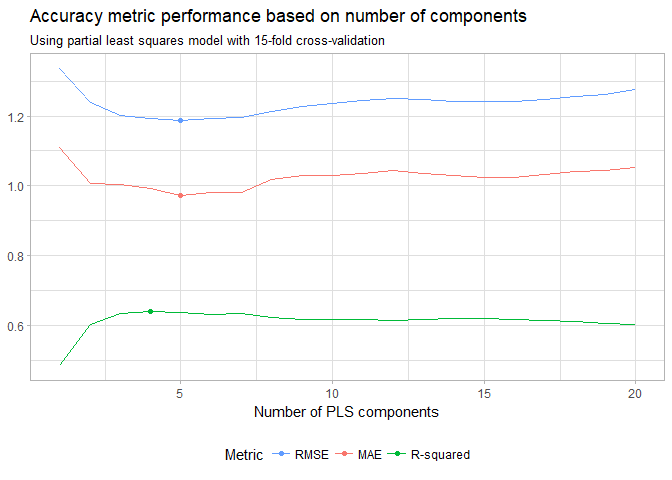


Because of this correlation, a dimensionality reduction may yield a more parsimonious model. For interpretability, a partial least squares model is used.

# use 15-fold cross-validation **set.seed**(100) # for replicabilitypls\_ctrl <- **trainControl**(method = "cv", number = 15)# train model; investigate up to 20 componentschem\_pls <- **train**(x = chem\_pred\_train, y = chem\_yield\_train, method = "pls", tuneLength = 20, trControl = pls\_ctrl)

The optimal value of the RMSE metric used to select the optimal model is 1.1859, which is achieved using 5 PLS components. This is illustrated in the plot of 3 performance metrics below:

chem\_pls$results %>%  **ggplot**(**aes**(x = ncomp)) + **geom\_line**(**aes**(y = RMSE, col = "RMSE")) + **geom\_line**(**aes**(y = Rsquared, col = "R-squared")) + **geom\_line**(**aes**(y = MAE, col = "MAE")) + **geom\_point**(data = **data\_frame**(Metric = **c**("RMSE", "R-squared", "MAE"), ncomp = **c**(**which.min**(chem\_pls$results$RMSE), **which.max**(chem\_pls$results$Rsquared), **which.min**(chem\_pls$results$MAE)), Optimal = **c**(**min**(chem\_pls$results$RMSE), **max**(chem\_pls$results$Rsquared), **min**(chem\_pls$results$MAE))), **aes**(x = ncomp, y = Optimal, col = Metric), size = 1.5) + **scale\_color\_discrete**("Metric", breaks = **c**("RMSE", "MAE", "R-squared")) + **labs**(title = "Accuracy metric performance based on number of components", subtitle = "Using partial least squares model with 15-fold cross-validation", x = "Number of PLS components", y = NULL) + **theme**(legend.position = "bottom", legend.direction = "horizontal")



## Parts d-f

### Model Performance

The trained model is used to predict the response, and the performance of the predictions is returned:

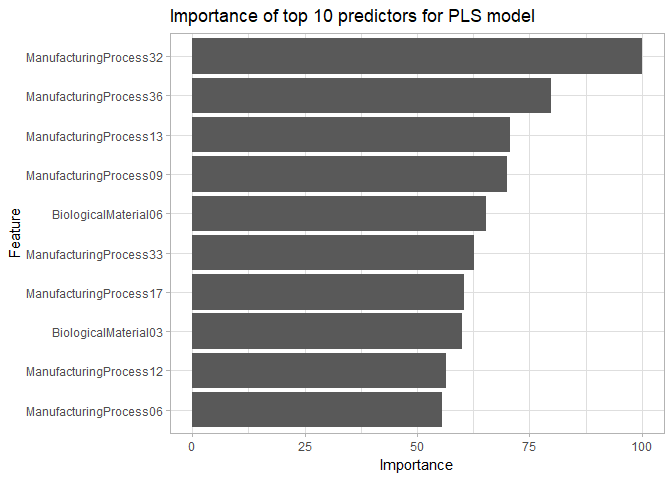
chem\_results <- **predict**(chem\_pls, chem\_pred\_test)chem\_perf <- **defaultSummary**(**data.frame**(obs = chem\_yield\_test, pred = chem\_results))

The RMSE for the test set is 1.1832 – this is slightly better than the RMSE for the resampled training set.

### Predictor Importance & Relationship

The importance of each predictor in the model is displayed graphically below:

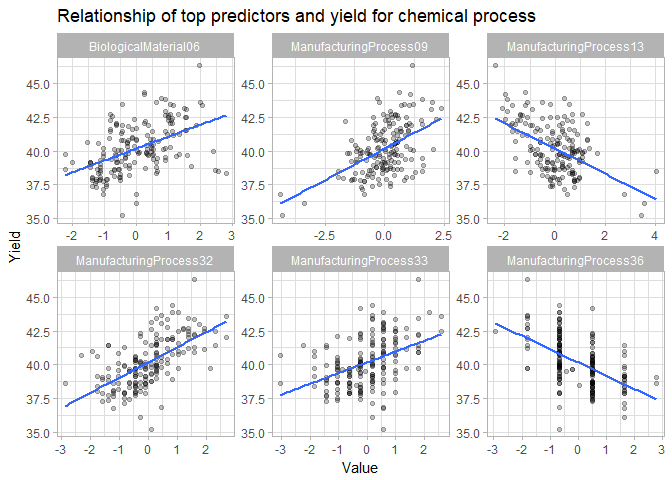
# get variable importancechem\_imp <- **varImp**(chem\_pls)# plot importance **ggplot**(chem\_imp, top = 10) + **ggtitle**("Importance of top 10 predictors for PLS model")



Two of the 10 top predictors are biological; this portion is very near to the share of total predictors available that are biological (21%).

The relationship of the top 6 of these pre-processed predictors is shown below:

# get top 6 predictorschem\_top <- **rownames**(chem\_imp$importance)[**order**(chem\_imp$importance$Overall, decreasing = TRUE)][1:6]# separate top predictors & yield into data framechem\_rel <- **as.data.frame**(**cbind**(chem\_pred\_trans, chem\_yield)[, **c**(chem\_top, "Yield")])# plot to investigate relationshipschem\_rel %>%  # convert to tidy format **gather**(Predictor, Value, -Yield) %>%  # plot **ggplot**(**aes**(x = Value, y = Yield)) + **geom\_point**(alpha = 0.25) + **stat\_smooth**(se = FALSE, method = "glm") + **facet\_wrap**(~ Predictor, nrow = 2, scales = "free") + **labs**(title = "Relationship of top predictors and yield for chemical process")



Looking at the plots above, especially the generalized linear model fit to each plot, it can be seen that four of the variables (*BiologicalMaterial06*, *ManufacturingProcess09*, *ManufacturingProcess32*, and *ManufacturingProcess33*) have positive relationships with yield, while the other two (*ManufacturingProcess13* and *ManufacturingProcess36*) have a negative relationship with yield. The variables with positive relationships should be increased, while those with negative relationships are decreased, in order to improve yield from the manufacturing process. Before acting on this, however, the interactions between these variables and their impact on the response should be investigated.