Data 624 - Homework Part 1 Christophe Hunt

# Packages Used

Please load packages into your environment before running any proceeding code.

library(reshape)  
library(caret)  
library(forecast)  
library(missForest)  
library(fma)  
library(ggfortify)  
library(tidyverse)  
library(psych)  
library(knitr)  
library(mlbench)  
library(Hmisc)  
library(fpp)  
library(psych)

# Week 1

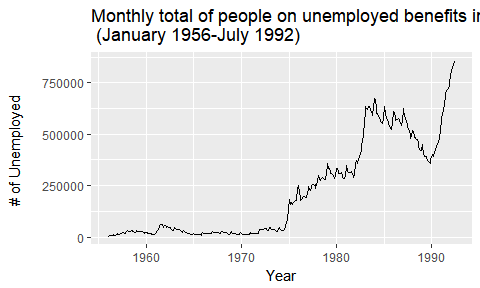
## Problem 2.1

For each of the following series (from the fma package), make a graph of the data. If transforming seems appropriate, do so and describe the effect.

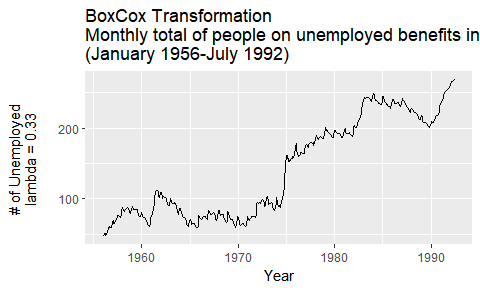
1. Monthly total of people on unemployed benefits in Australia (January 1956-July 1992).

The monthly total of people is graphed below. Caution should be exercised when plotting # totals as it doesn't account for the proportional change. In this case, it could be that the population of Australia also grew and the change in total unemployed is not as dramatic as the chart suggests.

library(fma)  
library(ggfortify)  
data(dole)  
autoplot(dole, xlab = "Year",   
 ylab = "# of Unemployed \n",   
 main = "Monthly total of people on unemployed benefits in Australia \n (January 1956-July 1992)")

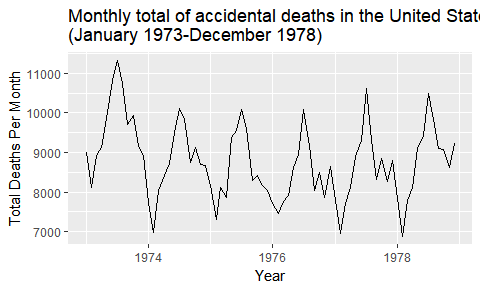


autoplot(BoxCox(dole, BoxCox.lambda(dole)),   
 xlab = "Year",   
 ylab = paste0("# of Unemployed \n", "lambda = ", round(BoxCox.lambda(dole),2), " \n"),  
 main = "BoxCox Transformation \nMonthly total of people on unemployed benefits in Australia \n(January 1956-July 1992)")



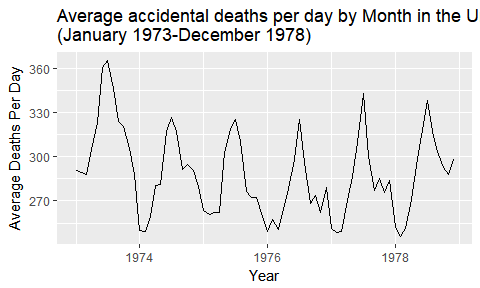
1. Monthly total of accidental deaths in the United States (January 1973-December 1978).

data(usdeaths)  
autoplot(usdeaths,  
 ylab="Total Deaths Per Month",  
 xlab="Year",  
 main="Monthly total of accidental deaths in the United States \n(January 1973-December 1978)")



While the chart is for each month, the days per month are variable so lets adjust for the deaths per day by month to have a more normalized time series.

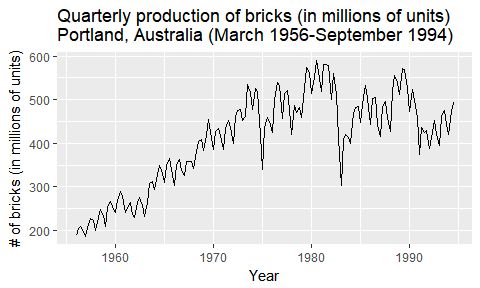
monthdays <- rep(c(31,28,31,30,31,30,31,31,30,31,30,31),6)  
monthdays[38] <- 29 #Leap year adjustment  
  
autoplot(usdeaths/monthdays, ylab="Average Deaths Per Day",  
 xlab="Year",  
 main="Average accidental deaths per day by Month in the United States \n(January 1973-December 1978)")



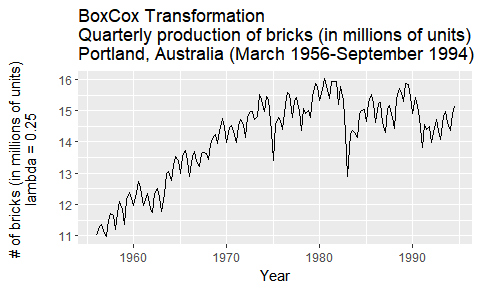
Interestingly, there is some smoothing seen in the earlier years of the time series but more dramatic changes later in the series. Since we adjusted for changes in days per month we can further assume that this is a properly normalized data set month over month.

1. Quarterly production of bricks (in millions of units) at Portland, Australia (March 1956-September 1994).

data(bricksq)  
autoplot(bricksq,  
 xlab = "Year",   
 ylab = paste0("# of bricks (in millions of units)"),  
 main = "Quarterly production of bricks (in millions of units) \nPortland, Australia (March 1956-September 1994)")



autoplot(BoxCox(bricksq, BoxCox.lambda(bricksq)),   
 xlab = "Year",   
 ylab = paste0("# of bricks (in millions of units) \n", "lambda = ", round(BoxCox.lambda(bricksq),2), " \n"),  
 main = "BoxCox Transformation \nQuarterly production of bricks (in millions of units) \nPortland, Australia (March 1956-September 1994)")



## Problem 2.3

Consider the daily closing IBM stock prices (data set ibmclose).

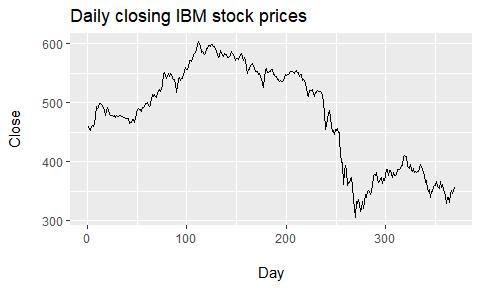
1. Produce some plots of the data in order to become familiar with it.

The sharp downward trend will be challenging for our predictions, it appears to be a shock event and not a normal/seasonal change. We will have to be cautious about our interpretations and predictions when including the shock event.

library(psych)  
library(knitr)  
data(ibmclose)  
summary(ibmclose)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 306.0 387.0 494.0 478.5 549.0 603.0

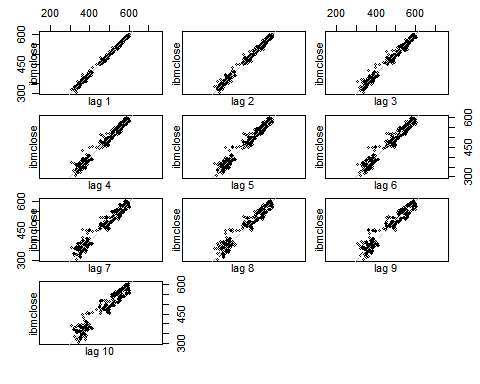
autoplot(ibmclose,   
 xlab = "\nDay",   
 ylab = "Close\n",  
 main = "Daily closing IBM stock prices")



Lagged Plot

The lagged plot shows the correlation over lagged periods, it's not entirely surprisingly that previous day close has more impact on the next day's close than several days later. However, it is useful to visualize the impact when attempting to further model.

lag.plot(ibmclose, lags= 10, do.lines=FALSE)



1. Split the data into a training set of 300 observations and a test set of 69 observations.

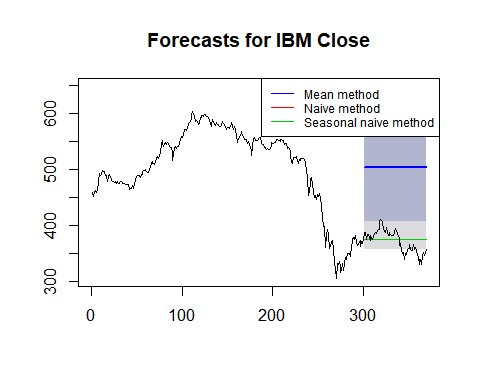
Splitting the data set using the window function for the first 300 observation for training and 69 for the test set.

ibmclose\_train <- window(ibmclose, start=1, end=300)  
ibmclose\_test <- window(ibmclose, start=301)

1. Try various benchmark methods to forecast the training set and compare the results on the test set. Which method did best?

The mean method appears to have performed the worse, which intuitively make sense with the higher performance of the stock before the shock loss event. The Naive and Seasonal Naive performed exactly the same but performed much better looking at the MAE score compared across all three methods. My prefernce would be for naive method based on our analysis of the lag method and how I understand stock performances. It may not be completely appropriate to assume that the stock performs in a seasonal fashion but performance on a short term scale may be indictative of some of the future performance.

ibmclosefit1 <- meanf(ibmclose\_train, h = 69)  
ibmclosefit2 <- rwf(ibmclose\_train, h = 69)  
ibmclosefit3 <- snaive(ibmclose\_train, h = 69)  
  
plot(ibmclosefit1, col = 1, main="Forecasts for IBM Close")  
lines(ibmclosefit2$mean, col=2)  
lines(ibmclosefit3$mean, col=3)  
lines(ibmclose)  
legend("topright",   
 cex = .75,  
 lty=1,   
 col=c(4,2,3),   
 legend=c("Mean method",   
 "Naive method",   
 "Seasonal naive method"))



Mean Method

kable(accuracy(ibmclosefit1, ibmclose\_test))

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
| Training set | 0.000 | 73.61532 | 58.72231 | -2.642058 | 13.03019 | 11.52098 | 0.9895779 | NA |
| Test set | -130.618 | 132.12557 | 130.61797 | -35.478819 | 35.47882 | 25.62649 | 0.9314689 | 19.05515 |

Naive method

kable(accuracy(ibmclosefit2, ibmclose\_test))

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
| Training set | -0.2809365 | 7.302815 | 5.09699 | -0.0826287 | 1.115844 | 1.000000 | 0.1351052 | NA |
| Test set | -3.7246377 | 20.248099 | 17.02899 | -1.2939174 | 4.668186 | 3.340989 | 0.9314689 | 2.973486 |

Seasonal naive method

kable(accuracy(ibmclosefit3, ibmclose\_test))

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
| Training set | -0.2809365 | 7.302815 | 5.09699 | -0.0826287 | 1.115844 | 1.000000 | 0.1351052 | NA |
| Test set | -3.7246377 | 20.248099 | 17.02899 | -1.2939174 | 4.668186 | 3.340989 | 0.9314689 | 2.973486 |

# Week 2

## Problem 6.2

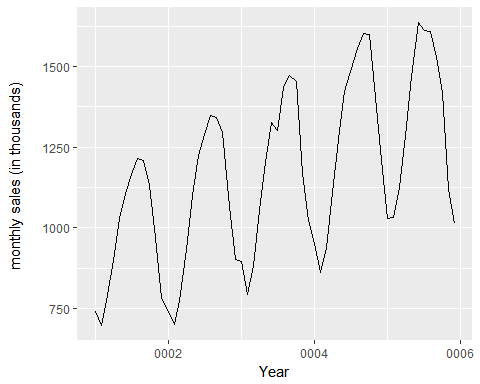
The data below represent the monthly sales (in thousands) of product A for a plastics manufacturer for years 1 through 5 (data set plastics).

library(fma)  
plastics

## Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec  
## 1 742 697 776 898 1030 1107 1165 1216 1208 1131 971 783  
## 2 741 700 774 932 1099 1223 1290 1349 1341 1296 1066 901  
## 3 896 793 885 1055 1204 1326 1303 1436 1473 1453 1170 1023  
## 4 951 861 938 1109 1274 1422 1486 1555 1604 1600 1403 1209  
## 5 1030 1032 1126 1285 1468 1637 1611 1608 1528 1420 1119 1013

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

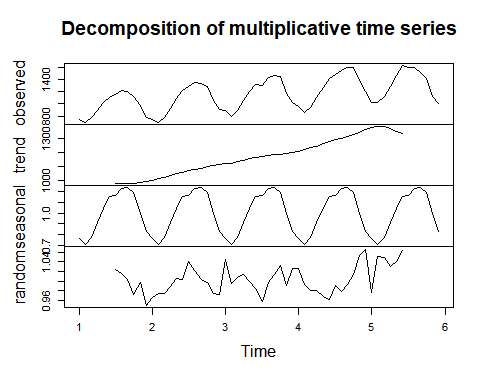
library(ggfortify)  
autoplot(plastics, xlab = "Year", ylab = "monthly sales (in thousands) \n")



There is clearly seasonal fluctuations here, it shows a trend towards high volume in summer and a sharp dip in winter for lower volume.

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

fit <- decompose(plastics, type="multiplicative")  
plot(fit)

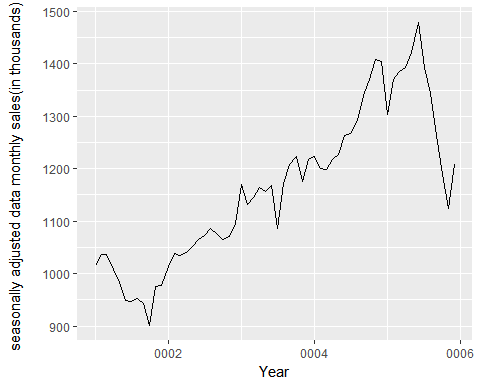


1. Do the results support the graphical interpretation from part (a)?

The results do support the seasonality trend assumption from part (a), it is useful to see the seasonal trend accounting for upward trend in the observed values to highlight the consistent seasonality.

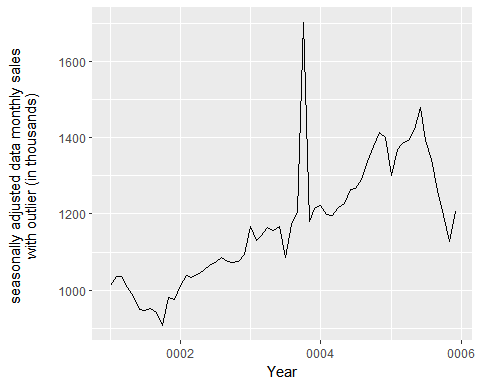
1. Compute and plot the seasonally adjusted data.

fit <- stl(plastics, t.window = 7, s.window="periodic", robust=TRUE)  
autoplot(seasadj(fit), xlab = "Year", ylab = "seasonally adjusted data monthly sales(in thousands) \n")



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

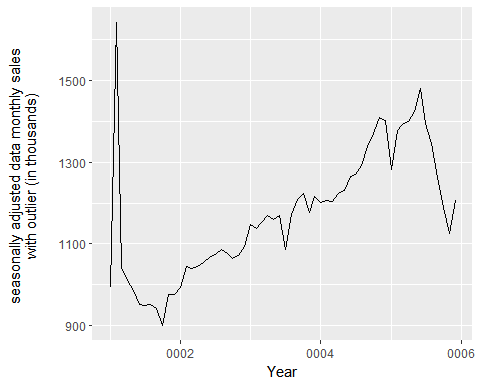
outlier\_plastics <- plastics  
outlier\_plastics[34] <- outlier\_plastics[30] + 600  
fit <- stl(outlier\_plastics, t.window = 7, s.window="periodic", robust=TRUE)  
autoplot(seasadj(fit), xlab = "Year", ylab = "seasonally adjusted data monthly sales \n with outlier (in thousands) \n")



Surpisingly, the effect of an outlier has very little impact on the overall trend of the line but does show considerable impact on the line at the point of the outlier.

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

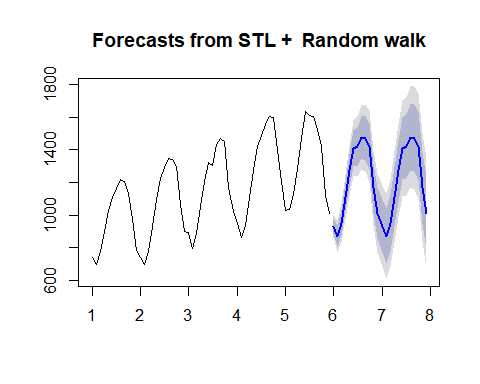
outlier\_plastics <- plastics  
outlier\_plastics[2] <- outlier\_plastics[2] + 600  
fit <- stl(outlier\_plastics, t.window = 7, s.window="periodic", robust=TRUE)  
autoplot(seasadj(fit), xlab = "Year", ylab = "seasonally adjusted data monthly sales \n with outlier (in thousands) \n")



It does appear that there is an initial smoothing of the line from the outlier at the beginning of the series. The line does return to a similar seasonally adjusted line from the original data which makes sense once the line is further out from the outlier. It does seem intiuitive that with an outlier at the beginning of the time series there is less surrounding data to account for the outlier.

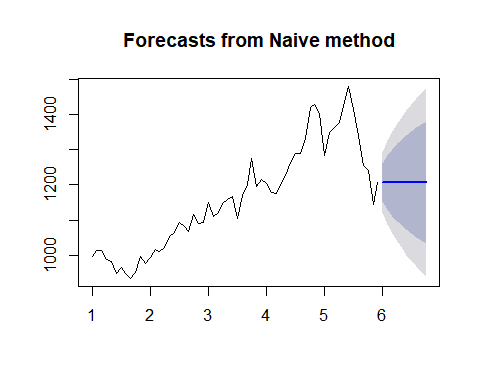
1. Use a random walk with drift to produce forecasts of the seasonally adjusted data.

fc <- stlf(plastics, method="naive")  
plot(fc)



1. Reseasonalize the results to give forecasts on the original scale.

fit <- stl(plastics, s.window="periodic", robust=TRUE)  
fitadj <- seasadj(fit)  
plot(naive(fitadj))



# Week 3

## Problem 3.1

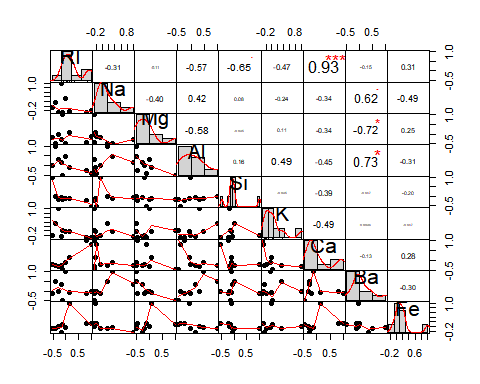
The UC Irvine Machine Learning Repository contains a data set related to glass identification. The data consit of 214 glass samples labeled as one of seven class categories. There are nine predicotrs, including the refrative index and percentages of eight elements: Na, Mg, Al, Si, K, Ca, Ba, and Fe.

library(mlbench)  
library(Hmisc)  
data(Glass)  
describe(Glass)

## Glass   
##   
## 10 Variables 214 Observations  
## ---------------------------------------------------------------------------  
## RI   
## n missing distinct Info Mean Gmd .05 .10   
## 214 0 178 1 1.518 0.003054 1.515 1.516   
## .25 .50 .75 .90 .95   
## 1.517 1.518 1.519 1.522 1.524   
##   
## lowest : 1.51115 1.51131 1.51215 1.51299 1.51316  
## highest: 1.52725 1.52739 1.52777 1.53125 1.53393  
## ---------------------------------------------------------------------------  
## Na   
## n missing distinct Info Mean Gmd .05 .10   
## 214 0 142 1 13.41 0.8685 12.41 12.68   
## .25 .50 .75 .90 .95   
## 12.91 13.30 13.83 14.40 14.85   
##   
## lowest : 10.73 11.02 11.03 11.23 11.45, highest: 14.99 15.01 15.15 15.79 17.38  
## ---------------------------------------------------------------------------  
## Mg   
## n missing distinct Info Mean Gmd .05 .10   
## 214 0 94 0.992 2.685 1.417 0.000 0.000   
## .25 .50 .75 .90 .95   
## 2.115 3.480 3.600 3.757 3.850   
##   
## lowest : 0.00 0.33 0.78 1.01 1.35, highest: 3.90 3.93 3.97 3.98 4.49  
## ---------------------------------------------------------------------------  
## Al   
## n missing distinct Info Mean Gmd .05 .10   
## 214 0 118 1 1.445 0.5311 0.696 0.870   
## .25 .50 .75 .90 .95   
## 1.190 1.360 1.630 2.074 2.394   
##   
## lowest : 0.29 0.34 0.47 0.51 0.56, highest: 2.79 2.88 3.02 3.04 3.50  
## ---------------------------------------------------------------------------  
## Si   
## n missing distinct Info Mean Gmd .05 .10   
## 214 0 133 1 72.65 0.8047 71.31 71.77   
## .25 .50 .75 .90 .95   
## 72.28 72.79 73.09 73.30 73.52   
##   
## lowest : 69.81 69.89 70.16 70.26 70.43, highest: 73.88 74.45 74.55 75.18 75.41  
## ---------------------------------------------------------------------------  
## K   
## n missing distinct Info Mean Gmd .05 .10   
## 214 0 65 0.996 0.4971 0.4385 0.0000 0.0000   
## .25 .50 .75 .90 .95   
## 0.1225 0.5550 0.6100 0.6800 0.7600   
##   
## lowest : 0.00 0.02 0.03 0.04 0.05, highest: 1.46 1.68 1.76 2.70 6.21  
## ---------------------------------------------------------------------------  
## Ca   
## n missing distinct Info Mean Gmd .05 .10   
## 214 0 143 1 8.957 1.334 7.812 7.970   
## .25 .50 .75 .90 .95   
## 8.240 8.600 9.172 10.443 11.561   
##   
## lowest : 5.43 5.79 5.87 6.47 6.65, highest: 13.44 14.40 14.68 14.96 16.19  
## ---------------------------------------------------------------------------  
## Ba   
## n missing distinct Info Mean Gmd .05 .10   
## 214 0 34 0.444 0.175 0.3157 0.00 0.00   
## .25 .50 .75 .90 .95   
## 0.00 0.00 0.00 0.64 1.57   
##   
## lowest : 0.00 0.06 0.09 0.11 0.14, highest: 1.68 1.71 2.20 2.88 3.15  
## ---------------------------------------------------------------------------  
## Fe   
## n missing distinct Info Mean Gmd .05 .10   
## 214 0 32 0.695 0.05701 0.08794 0.000 0.000   
## .25 .50 .75 .90 .95   
## 0.000 0.000 0.100 0.220 0.267   
##   
## lowest : 0.00 0.01 0.03 0.05 0.06, highest: 0.32 0.34 0.35 0.37 0.51  
## ---------------------------------------------------------------------------  
## Type   
## n missing distinct   
## 214 0 6   
##   
## Value 1 2 3 5 6 7  
## Frequency 70 76 17 13 9 29  
## Proportion 0.327 0.355 0.079 0.061 0.042 0.136  
## ---------------------------------------------------------------------------

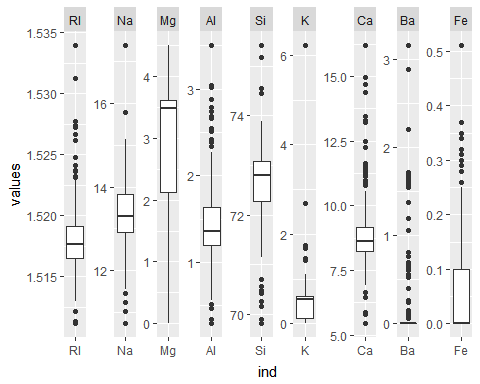
1. Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors.

library(corrplot)  
library(PerformanceAnalytics)  
Glass %>% dplyr::select(-(Type)) %>% cor() %>% chart.Correlation(histogram=TRUE, pch=19)

 > The correlation chart provides a great visualization and provides the interaction of our variables. We see that the refractive index is highly positively correlated with Ca. It also appears that Barium is highly correlated with other elements. Silicon appears to have a very narrow distribution and is negatively correlated with the refractive index.

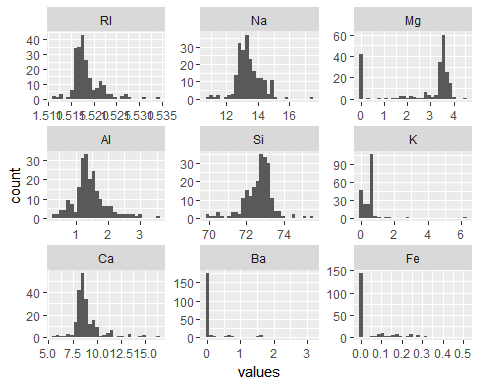
1. Do there appear to be any outliers in the data? Are any predictors skewed?

library(ggplot2)  
ggplot(stack(Glass), aes(x = ind, y = values)) +  
 facet\_wrap(~ind, scales = "free", nrow = 1) +   
 geom\_boxplot()



We can see from the boxplots of the variables that there are indeed outliers at a univarite level. Anything beyond whiskers of our box and whisker plot is potentially an outlier. We would need to further investigate if this is a measurement issue or simply an outlier that is a true observation.

library(ggplot2)  
ggplot(stack(Glass), aes(values)) +  
 facet\_wrap(~ind, scales = "free") +   
 geom\_histogram()



Visually, we can see that a couple of our variables are very skewed. The skewed variables are Mg, K, Ba, and Fe with many observations occuring at 0. Our other variables appear to have some slight skewness but not nearly as dramatic as the other variables.

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

apply(Glass %>% dplyr::select(-(Type)), 2, BoxCoxTrans)

## $RI  
## Box-Cox Transformation  
##   
## 214 data points used to estimate Lambda  
##   
## Input data summary:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.511 1.517 1.518 1.518 1.519 1.534   
##   
## Largest/Smallest: 1.02   
## Sample Skewness: 1.6   
##   
## Estimated Lambda: -2   
##   
##   
## $Na  
## Box-Cox Transformation  
##   
## 214 data points used to estimate Lambda  
##   
## Input data summary:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 10.73 12.91 13.30 13.41 13.82 17.38   
##   
## Largest/Smallest: 1.62   
## Sample Skewness: 0.448   
##   
## Estimated Lambda: -0.1   
## With fudge factor, Lambda = 0 will be used for transformations  
##   
##   
## $Mg  
## Box-Cox Transformation  
##   
## 214 data points used to estimate Lambda  
##   
## Input data summary:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 2.115 3.480 2.685 3.600 4.490   
##   
## Lambda could not be estimated; no transformation is applied  
##   
##   
## $Al  
## Box-Cox Transformation  
##   
## 214 data points used to estimate Lambda  
##   
## Input data summary:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.290 1.190 1.360 1.445 1.630 3.500   
##   
## Largest/Smallest: 12.1   
## Sample Skewness: 0.895   
##   
## Estimated Lambda: 0.5   
##   
##   
## $Si  
## Box-Cox Transformation  
##   
## 214 data points used to estimate Lambda  
##   
## Input data summary:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 69.81 72.28 72.79 72.65 73.09 75.41   
##   
## Largest/Smallest: 1.08   
## Sample Skewness: -0.72   
##   
## Estimated Lambda: 2   
##   
##   
## $K  
## Box-Cox Transformation  
##   
## 214 data points used to estimate Lambda  
##   
## Input data summary:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000 0.1225 0.5550 0.4971 0.6100 6.2100   
##   
## Lambda could not be estimated; no transformation is applied  
##   
##   
## $Ca  
## Box-Cox Transformation  
##   
## 214 data points used to estimate Lambda  
##   
## Input data summary:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 5.430 8.240 8.600 8.957 9.172 16.190   
##   
## Largest/Smallest: 2.98   
## Sample Skewness: 2.02   
##   
## Estimated Lambda: -1.1   
##   
##   
## $Ba  
## Box-Cox Transformation  
##   
## 214 data points used to estimate Lambda  
##   
## Input data summary:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 0.000 0.000 0.175 0.000 3.150   
##   
## Lambda could not be estimated; no transformation is applied  
##   
##   
## $Fe  
## Box-Cox Transformation  
##   
## 214 data points used to estimate Lambda  
##   
## Input data summary:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00000 0.00000 0.00000 0.05701 0.10000 0.51000   
##   
## Lambda could not be estimated; no transformation is applied

From our results, it appears that few variables could benefit from transformations but we should be cautious to not create an overly complex model by applying all the transformations thoughtlessly. The variables that could benefit from transformation are RI, Na, Al, Si, and Ca.

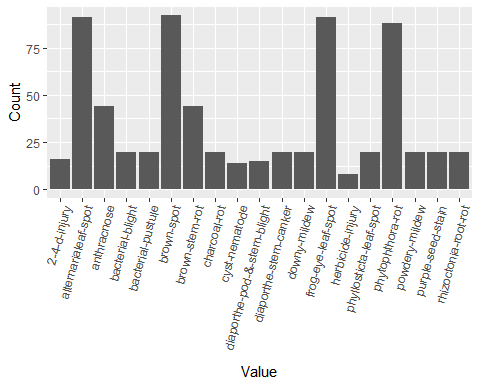
## Problem 3.2

The soybean data can also be found at UC Irvine Machine Learning Repository. Data was collected to predict diease in 683 Syobeas. The 35 predictors are mostly categorical and include information on the environmental conditions (e.g., temperature, precipitation) and plant conditions (e.g., left spots, mold growth). The outcome labels consist of 19 distinct classes. The data can be loaded via:

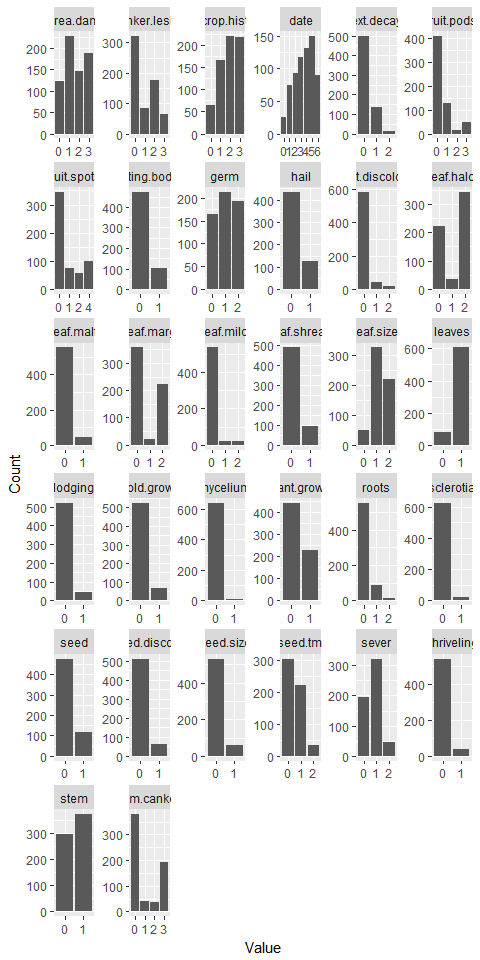
library(mlbench)  
data(Soybean)

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

t <- melt(apply(Soybean %>% dplyr::select(-plant.stand, -precip, -temp), 2, FUN = table)) %>%   
 rename(c("Var.1" = "Value", "value" = "Count", "L1" = "Variable"))  
  
ggplot(t %>% dplyr::filter(Variable == "Class"), aes(x = Value, y = Count, group = Variable)) +  
 geom\_col() +  
 theme(axis.text.x = element\_text(angle = 75, hjust = 1, vjust = 1))



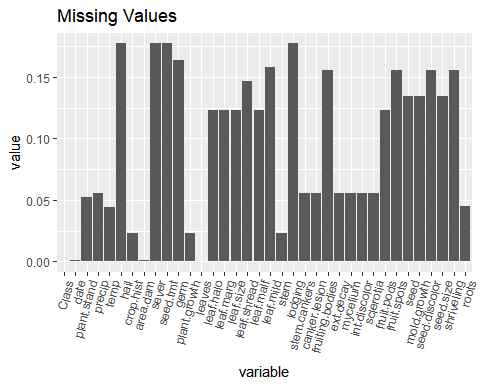
ggplot(t %>% dplyr::filter(Variable != "Class"), aes(x = Value, y = Count, group = Variable)) +  
 facet\_wrap(~Variable, scales = "free") +  
 geom\_col()



A majority of the factor variables show some kind of degeneration from heavy skewness to bimodal distributions. A few like mycelium and sclerotia contain so few observations of another factor it is suspect that the variable contains any predictive value at all.

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

Soybean %>%   
 summarise\_each(funs(sum(is.na(.))/nrow(Soybean))) %>%   
 melt() %>%   
 ggplot(aes(x = variable, y = value)) +  
 geom\_col() +  
 ggtitle("Missing Values") +  
 theme(axis.text.x = element\_text(angle = 75, hjust = 1, vjust = 1))



There are a few predictors that appear to be missing more often, the lodging, server, seed.tmt, and hail all appear to be missing the most often compared to the other predictors.

Soybean %>%   
 group\_by(Class) %>%   
 summarise\_all(funs(sum(is.na(.))/nrow(Soybean))) %>%   
 as.data.frame() %>%  
 melt() %>%  
 ggplot(aes(x = variable, y = value, fill = Class)) +  
 geom\_col() +  
 #facet\_grid(~variable, scales = "free") +  
 ggtitle("Missing Values by Class") +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1, vjust = .1))



When we look at the predictors by class, it does appear that 2-4-d-injury and phyliosticta-leaf-spot accounts for a significant amount of missing data. It would be interesting to see if the observations are missing by chance or these classes have inhereint issues in the sampling process.

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

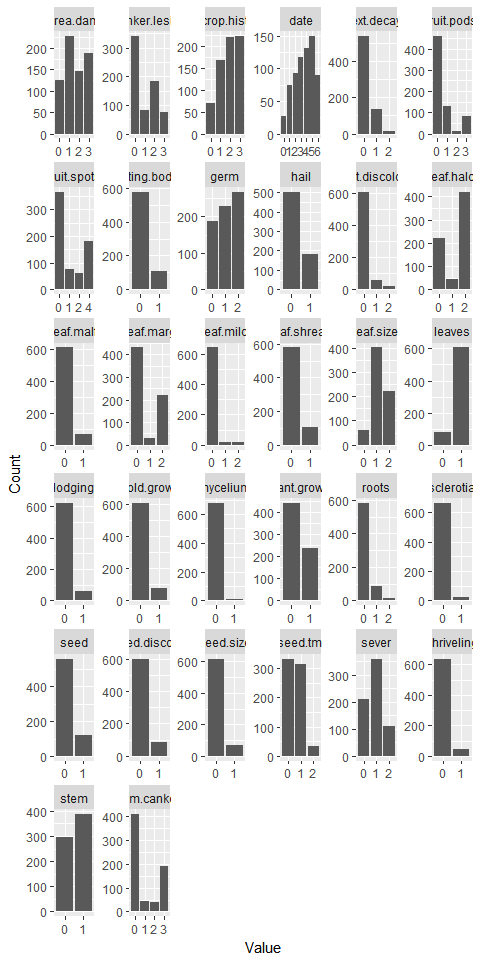
My preferred method is to use the missForest package as it is a nonparametric technique so no assumptions of the actual distribution is made. It is a very robust method that applies the random forest algorithm to imput missing data without the need to eliminate predictors. If I could not use missForest, I would likely exclude the two classes we discussed earlier then imput with the mean for a simple approach.

library(missForest)  
Soybean.imp <- missForest(Soybean)

## missForest iteration 1 in progress...done!  
## missForest iteration 2 in progress...done!  
## missForest iteration 3 in progress...done!  
## missForest iteration 4 in progress...done!

The below is the results of our imputation method.

t <- melt(apply(Soybean.imp$ximp %>%   
 dplyr::select(-plant.stand, -precip, -temp), 2, FUN = table)) %>%   
 rename(c("Var.1" = "Value", "value" = "Count", "L1" = "Variable"))  
  
ggplot(t %>% dplyr::filter(Variable != "Class"), aes(x = Value, y = Count, group = Variable)) +  
 facet\_wrap(~Variable, scales = "free") +  
 geom\_col()



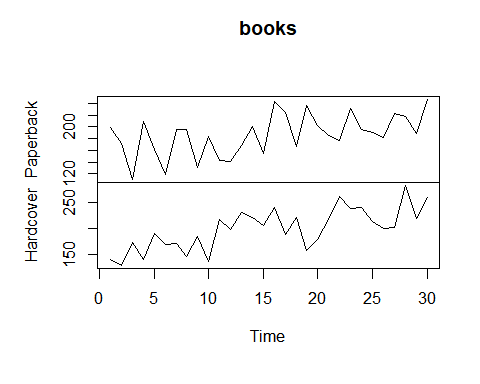
# Week 4

## Problem 7.1

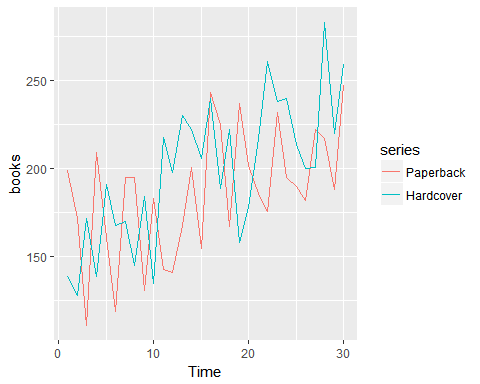
Data set books contains the daily sales of paperback and hardcover books at the same store. The task is to forecast the next four days' sales for paperback and hardcover books (data set books).

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

library(fma)  
library(tidyverse)  
library(ggfortify)  
books <- fma::books  
plot(books)



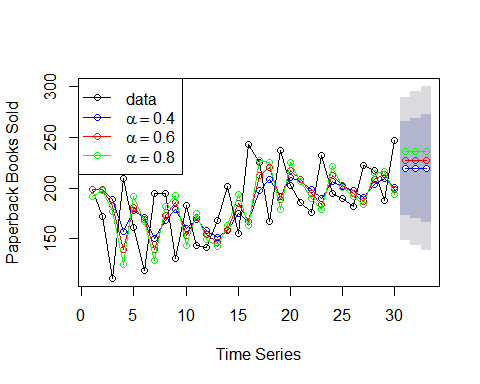
autoplot(books)



The data appears to have some seasonal variance and there appears to be times where Paperback is at the opposite extreme of the Hardcover books.

1. Use simple exponential smoothing with the ses function (setting initial="simple") and explore different values of for the paperback series. Record the within-sample SSE for the one-step forecasts. Plot SSE against and find which value of works best. What is the effect of on the forecasts?

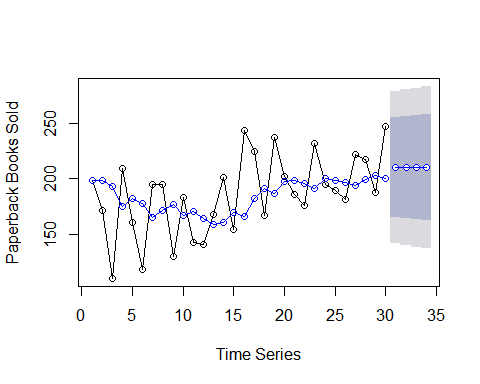
paperback <- books[,1]  
  
fit1 <- ses(paperback, alpha=0.4, initial="simple", h=3)  
fit2 <- ses(paperback, alpha=0.6, initial="simple", h=3)  
fit3 <- ses(paperback, alpha=0.8, h=3)  
  
plot(fit1, ylab="Paperback Books Sold",  
 xlab="Time Series", main="", fcol="white", type="o")  
lines(fitted(fit1), col="blue", type="o")  
lines(fitted(fit2), col="red", type="o")  
lines(fitted(fit3), col="green", type="o")  
lines(fit1$mean, col="blue", type="o")  
lines(fit2$mean, col="red", type="o")  
lines(fit3$mean, col="green", type="o")  
legend("topleft",lty=1, col=c(1,"blue","red","green"),   
 c("data", expression(alpha == 0.4), expression(alpha == 0.6),  
 expression(alpha == 0.8)),pch=1)



Due to so much variability in the data, it's challenging to tell which worked best. The best performance would dependent on the use of the results, I would say looks to be the best predictor but appears to be a more conservative assumption for the next best value.

1. Now let ses select the optimal value of Use this value to generate forecasts for the next four days. Compare your results with 2.

fit\_opt <- ses(paperback, initial="simple", h=4)  
plot(fit\_opt, ylab="Paperback Books Sold",  
 xlab="Time Series", main="", fcol="white", type="o")  
lines(fitted(fit\_opt), col="blue", type="o")  
lines(fit\_opt$mean, col="blue", type="o")



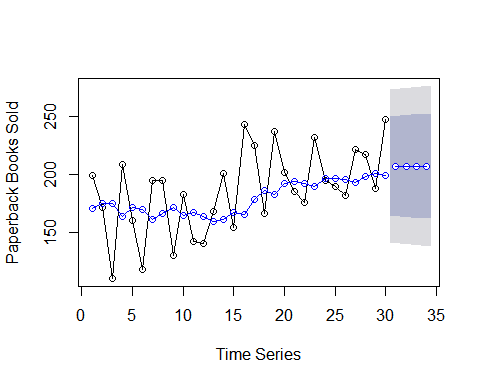
summary(fit\_opt)

##   
## Forecast method: Simple exponential smoothing  
##   
## Model Information:  
## Simple exponential smoothing   
##   
## Call:  
## ses(y = paperback, h = 4, initial = "simple")   
##   
## Smoothing parameters:  
## alpha = 0.2125   
##   
## Initial states:  
## l = 199   
##   
## sigma: 34.7918  
## Error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 1.749509 34.79175 28.64424 -2.770157 16.56938 0.7223331  
## ACF1  
## Training set -0.1268119  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 31 210.1537 165.5663 254.7411 141.9631 278.3443  
## 32 210.1537 164.5706 255.7368 140.4404 279.8671  
## 33 210.1537 163.5962 256.7112 138.9501 281.3573  
## 34 210.1537 162.6418 257.6657 137.4905 282.8170

The optimal alpha for the paperback books is = 0.2125

1. Repeat but with initial="optimal". How much difference does an optimal initial level make?

fit\_opt <- ses(paperback, initial="optimal", h=4)  
plot(fit\_opt, ylab="Paperback Books Sold",  
 xlab="Time Series", main="", fcol="white", type="o")  
lines(fitted(fit\_opt), col="blue", type="o")  
lines(fit\_opt$mean, col="blue", type="o")



summary(fit\_opt)

##   
## Forecast method: Simple exponential smoothing  
##   
## Model Information:  
## Simple exponential smoothing   
##   
## Call:  
## ses(y = paperback, h = 4, initial = "optimal")   
##   
## Smoothing parameters:  
## alpha = 0.1685   
##   
## Initial states:  
## l = 170.8257   
##   
## sigma: 33.6377  
##   
## AIC AICc BIC   
## 318.9747 319.8978 323.1783   
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 7.176212 33.63769 27.8431 0.4737524 15.57782 0.7021303  
## ACF1  
## Training set -0.2117579  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 31 207.1098 164.0013 250.2182 141.1811 273.0384  
## 32 207.1098 163.3934 250.8261 140.2513 273.9682  
## 33 207.1098 162.7937 251.4258 139.3342 274.8853  
## 34 207.1098 162.2021 252.0174 138.4294 275.7901

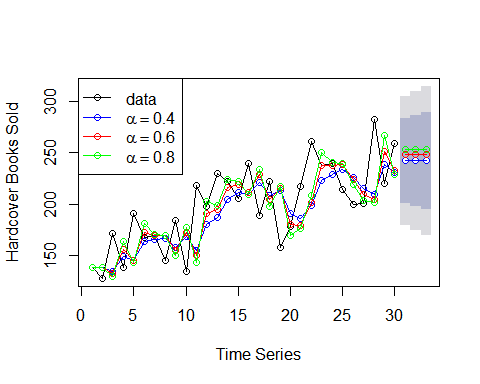
The optimal alpha with the setting of optimal in ses = "optimal" is

1. Repeat steps (b)-(d) with the hardcover series.

### Hardcover books

b2. Use simple exponential smoothing with the ses function (setting initial="simple") and explore different values of for the paperback series. Record the within-sample SSE for the one-step forecasts. Plot SSE against and find which value of works best. What is the effect of on the forecasts?

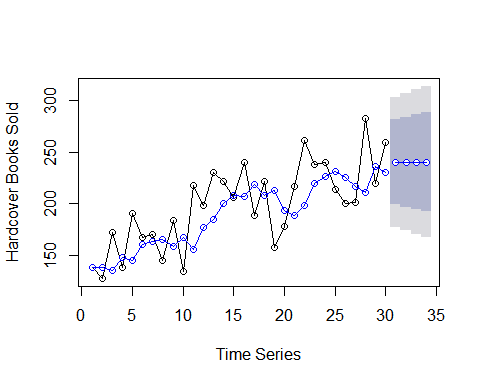
hardcover <- books[,2]  
  
fit1 <- ses(hardcover, alpha=0.4, initial="simple", h=3)  
fit2 <- ses(hardcover, alpha=0.6, initial="simple", h=3)  
fit3 <- ses(hardcover, alpha=0.8, h=3)  
  
plot(fit1, ylab="Hardcover Books Sold",  
 xlab="Time Series", main="", fcol="white", type="o")  
lines(fitted(fit1), col="blue", type="o")  
lines(fitted(fit2), col="red", type="o")  
lines(fitted(fit3), col="green", type="o")  
lines(fit1$mean, col="blue", type="o")  
lines(fit2$mean, col="red", type="o")  
lines(fit3$mean, col="green", type="o")  
legend("topleft",lty=1, col=c(1,"blue","red","green"),   
 c("data", expression(alpha == 0.4), expression(alpha == 0.6),  
 expression(alpha == 0.8)),pch=1)



Due to so much variability in the data, it's challenging to tell which worked best in the hardcover as well. The best performance would be what I was trying to achieve with my results, so I would say appears to be the safest assumption for the next best value but looks like it could be a more accurate prediction.

c2. Now let ses select the optimal value of Use this value to generate forecasts for the next four days. Compare your results with 2.

fit\_opt <- ses(hardcover, initial="simple", h=4)  
plot(fit\_opt, ylab="Hardcover Books Sold",  
 xlab="Time Series", main="", fcol="white", type="o")  
lines(fitted(fit\_opt), col="blue", type="o")  
lines(fit\_opt$mean, col="blue", type="o")



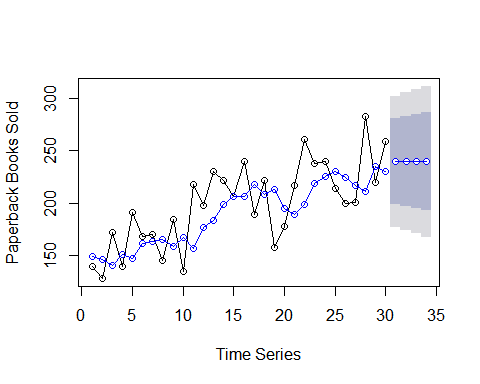
summary(fit\_opt)

##   
## Forecast method: Simple exponential smoothing  
##   
## Model Information:  
## Simple exponential smoothing   
##   
## Call:  
## ses(y = hardcover, h = 4, initial = "simple")   
##   
## Smoothing parameters:  
## alpha = 0.3473   
##   
## Initial states:  
## l = 139   
##   
## sigma: 32.0198  
## Error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 9.72952 32.01982 26.34467 3.104211 13.05063 0.7860035  
## ACF1  
## Training set -0.1629042  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 31 240.3808 199.3457 281.4158 177.6231 303.1385  
## 32 240.3808 196.9410 283.8206 173.9453 306.8162  
## 33 240.3808 194.6625 286.0990 170.4608 310.3008  
## 34 240.3808 192.4924 288.2691 167.1418 313.6197

The optimal alpha for the hardcover books is = 0.3473$ which fits in line with my previous thoughts on best alpha

1. Repeat but with initial="optimal". How much difference does an optimal initial level make?

fit\_opt <- ses(hardcover, initial="optimal", h=4)  
plot(fit\_opt, ylab="Paperback Books Sold",  
 xlab="Time Series", main="", fcol="white", type="o")  
lines(fitted(fit\_opt), col="blue", type="o")  
lines(fit\_opt$mean, col="blue", type="o")



summary(fit\_opt)

##   
## Forecast method: Simple exponential smoothing  
##   
## Model Information:  
## Simple exponential smoothing   
##   
## Call:  
## ses(y = hardcover, h = 4, initial = "optimal")   
##   
## Smoothing parameters:  
## alpha = 0.3283   
##   
## Initial states:  
## l = 149.2836   
##   
## sigma: 31.931  
##   
## AIC AICc BIC   
## 315.8506 316.7737 320.0542   
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 9.166918 31.93101 26.7731 2.636328 13.39479 0.7987858  
## ACF1  
## Training set -0.1417817  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 31 239.5602 198.6390 280.4815 176.9766 302.1439  
## 32 239.5602 196.4905 282.6299 173.6908 305.4297  
## 33 239.5602 194.4443 284.6762 170.5613 308.5591  
## 34 239.5602 192.4869 286.6336 167.5677 311.5527

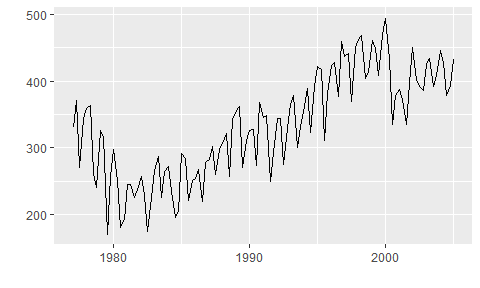
The optimal alpha with the setting of optimal in ses = "optimal" is which is still very close the assumptions made when was set to 0.3283.

## Problem 7.3

For this exercise, use the quarterly UK passenger vehicle production data from 1977:1--2005:1 (data set ukcars).

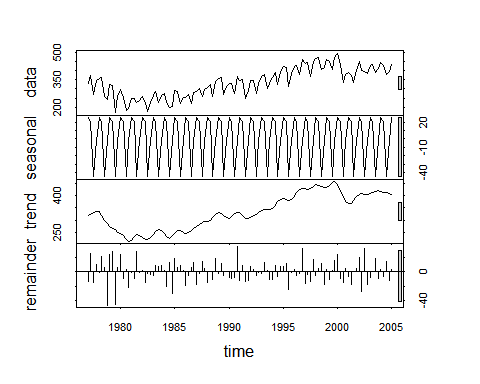
1. Plot the data and describe the main features of the series.

library(expsmooth)  
ukcars <- expsmooth::ukcars  
autoplot(ukcars)

 > The data shows a parabolic curve, it shows a seasonal pattern by quarter and it looks like there stochastic shock at year 2000. More research could provide insight to this shock event, perhaps it was policy related or economic related.

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

fit\_stl <- stl(ukcars, s.window = "periodic")  
plot(fit\_stl)



We see a long gradual upwards trend from ~1980 to ~2000 followed by a drop and gradual trend upwards.

sadj <- seasadj(fit\_stl)  
sfactors <- fit\_stl$time.series[2:11, "seasonal"]

1. Forecast the next two years of the series using an additive damped trend method applied to the seasonally adjusted data. Then reseasonalize the forecasts. Record the parameters of the method and report the RMSE of the one-step forecasts from your method.

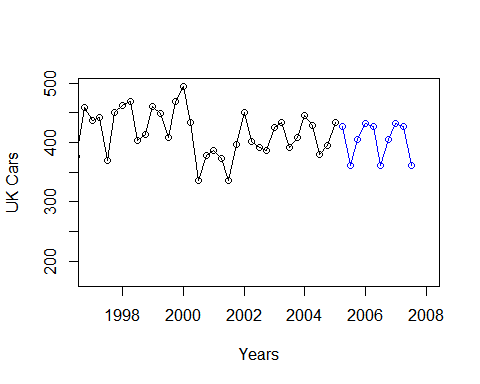
fit\_dpd\_sadj <- holt(sadj, damped = TRUE)  
paste0("Additive damped RMSE : ",round(accuracy(fit\_dpd\_sadj)[2], 3))

## [1] "Additive damped RMSE : 25.14"

rsfit\_dpd\_sadj <- fit\_dpd\_sadj$mean + sfactors   
rsfit\_dpd\_sadj

## Qtr1 Qtr2 Qtr3 Qtr4  
## 2005 427.4678 361.3282 404.8962  
## 2006 432.1189 427.4660 361.3266 404.8947  
## 2007 432.1175 427.4648 361.3255

plot(ukcars,   
 type = "o",  
 xlab = "Years",   
 ylab = "UK Cars",   
 xlim = c(1997, 2008))  
  
lines(rsfit\_dpd\_sadj, type = "o", col = "blue")



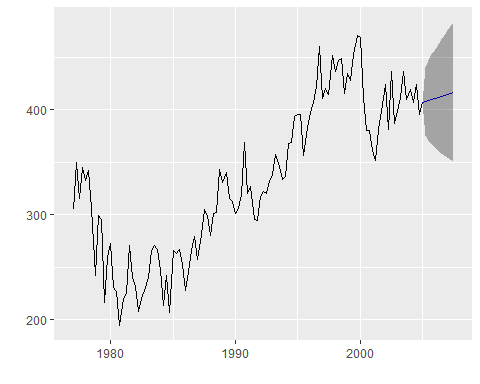
The results are very impressive, it produced a very reasonable forcast excluding any shock events, which would be impossible to predict.

1. Forecast the next two years of the series using Holt's linear method applied to the seasonally adjusted data. Then reseasonalize the forecasts. Record the parameters of the method and report the RMSE of of the one-step forecasts from your method.

fit\_hl <- holt(sadj)  
paste0("Holt Linear Method with seasonal adjustment RMSE: ",round(accuracy(fit\_hl)[2],3))

## [1] "Holt Linear Method with seasonal adjustment RMSE: 25.314"

autoplot(fit\_hl, xlab = "Years", ylab = "UK Car Production")



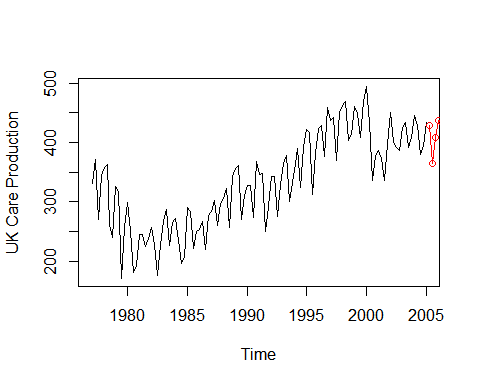
RSME

accuracy(fit\_hl)

## ME RMSE MAE MPE MAPE MASE  
## Training set -0.6725216 25.31401 20.10156 -0.7566151 6.492304 0.6551  
## ACF1  
## Training set 0.03864727

Reseaonalize the forecast

rs\_l <- fit\_hl$mean + sfactors  
plot(ukcars, ylab = "UK Care Production")  
lines(rs\_l, type = "o", col = "red")

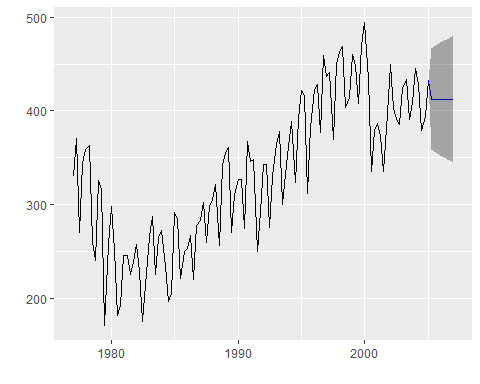


1. Now use ets() to choose a seasonal model for the data.

fit\_ets <- ets(ukcars, model = "ANN")  
summary(fit\_ets)

## ETS(A,N,N)   
##   
## Call:  
## ets(y = ukcars, model = "ANN")   
##   
## Smoothing parameters:  
## alpha = 0.2836   
##   
## Initial states:  
## l = 326.0531   
##   
## sigma: 42.1043  
##   
## AIC AICc BIC   
## 1385.469 1385.689 1393.651   
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 2.703015 42.10435 35.15604 -1.00922 11.5752 1.145718  
## ACF1  
## Training set 0.0830777

autoplot(forecast(fit\_ets), xlab = "Years", ylab = "UK Car Production")



paste0("RMSE from Simple Exp Smoothing Method: ",round(accuracy(fit\_ets)[2], 3))

## [1] "RMSE from Simple Exp Smoothing Method: 42.104"

1. Compare the RMSE of the fitted model with the RMSE of the model you obtained using an STL decomposition with Holt's method. Which gives the better in-sample fits?

paste0("Additive Damped - RMSE : ",round(accuracy(fit\_dpd\_sadj)[2], 3))

## [1] "Additive Damped - RMSE : 25.14"

paste0("Holt - RMSE: ",round(accuracy(fit\_hl)[2],3))

## [1] "Holt - RMSE: 25.314"

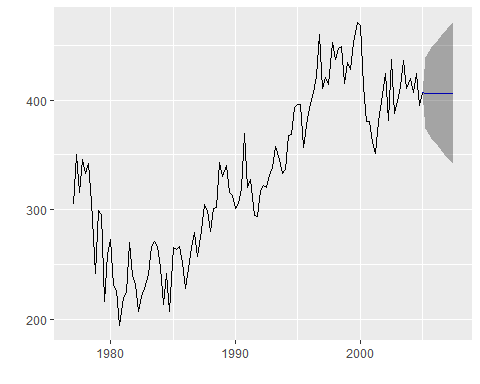
paste0("RMSE from Simple Exp Smoothing Method: ",round(accuracy(fit\_ets)[2], 3))

## [1] "RMSE from Simple Exp Smoothing Method: 42.104"

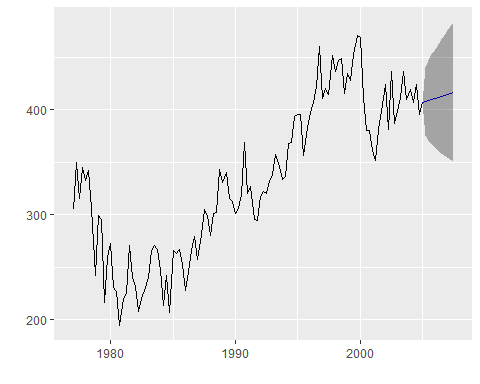
The best in-sample fit is the Additive Damped method based on the RMSE results of our models.

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

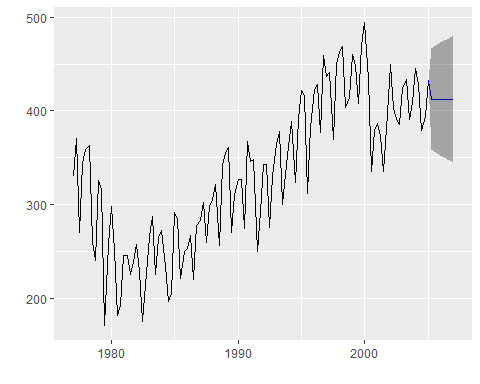
autoplot(forecast(fit\_dpd\_sadj))



autoplot(forecast(fit\_hl))



autoplot(forecast(fit\_ets))



I find this a difficult evaluation on the most appropriate model. Using our evaluation methods the lowest RMSE is the Additive Damped model but visually the second Holt model appears to have the more accurate prediction. I think this is an interesting dilemma because the comparison measure indicates that the first model is best but the visuals of the second model look more sound and there is a conflict with my intiuition and the data. The first model with the lowest RMSE is the most reasonable choice. I did not discuss the simple exponential smoothing method because its performance is clearly subpar.

# Week 6

## Problem 8.1

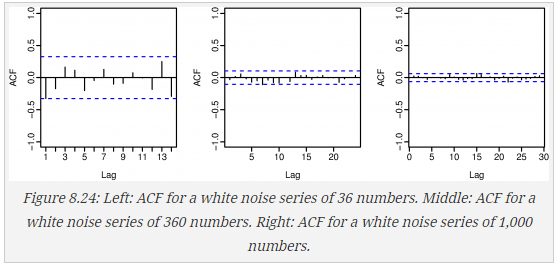


Figure 8.24 shows the ACFs for 36 random numbers, 360 random numbers and for 1,000 random numbers.

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

We can see that the correlation between previous observations are not exponentially decaying which does indicate these are white noise series. All the figures appear to have varying degrees of correlation with the first series having the highest and the last series having the lowest.

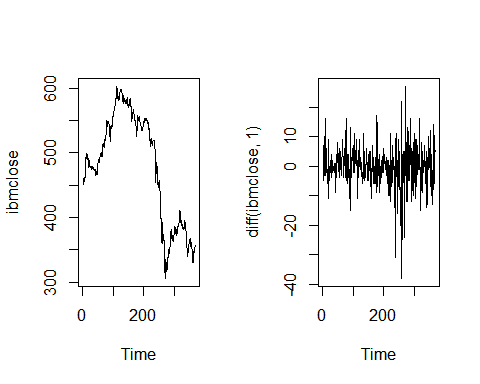
Why are the critical values at different distances from the mean of zero? Why are the auto-correlations different in each figure when they each refer to white noise?

The critical values are determined from the sample size, since the size of each series is different the critical values are also different. The auto-correlations are based on the past values, since the time series are different in sample size the auto-correlations of each series will be different.

## Problem 8.2

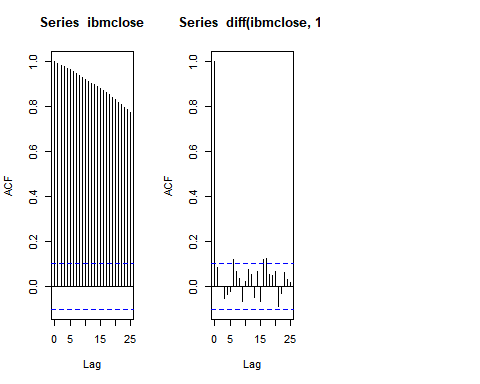
A classic example of a non-stationary series is the daily closing IBM stock prices (data set ibmclose). Use R to plot the daily closing prices for IBM stock and the ACF and PACF. Explain how each plot shows the series is non-stationary and should be differenced.

library(fma)  
data(ibmclose)  
par(mfrow=c(1, 2))  
plot(ibmclose)  
plot(diff(ibmclose,1))



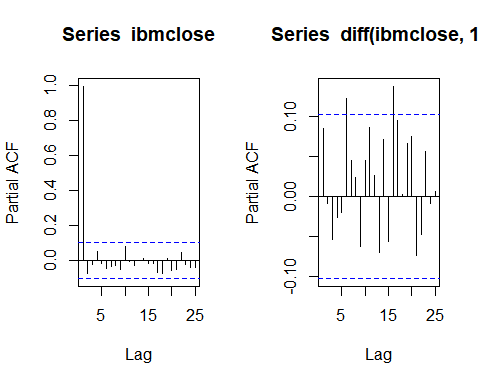
The plot of the time series shows trends over time and patters in the long term. The current value is highly correlated with the previous value. Adding a 1 day lag does should some promise in transforming the time series to reach stationarity but there looks to be some pattern between the 200 and 300 day marks.

par(mfrow=c(1, 3))  
acf(ibmclose)  
acf(diff(ibmclose,1))



The initial graph shows that the data series is most certainly not stationary as we see the exponential decay. Using the differcing of 1 order by 1 day the auto-correlation of the lags shows that the its close to reaching white noise. I have concerns about the marks reaching past the critical values beyond lag 1.

par(mfrow=c(1, 2))  
pacf(ibmclose)  
pacf(diff(ibmclose,1))



Interestingly, the PACF of the original time series suggests an AR(1) model but using differencing the model moves towards a MA(1) model. This is an interesting example of how differencing will transform a time series to be best suited by another model method.

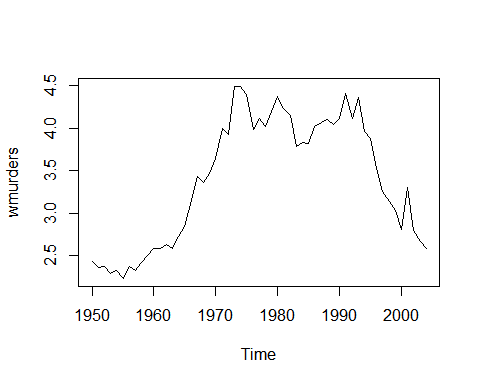
## Problem 8.6

Consider the number of women murdered each year (per 100,000 standard population) in the United States (data set wmurders).

data(wmurders)

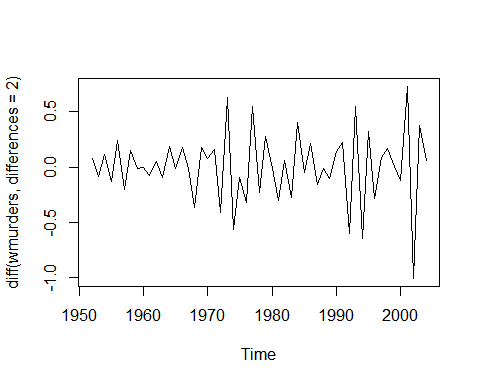
1. By studying appropriate graphs of the series in R, find an appropriate ARIMA(p,d,q) model for these data.

plot(wmurders)

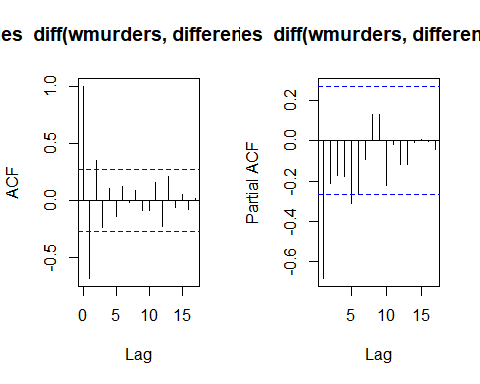


There is a clear pattern in the data so we will need to apply differencing. Using the ndiffs() method we see that 2 order of differences is appropriate.

plot(diff(wmurders, differences = 2))



par(mfrow=c(1, 2))  
acf(diff(wmurders, differences = 2))  
pacf(diff(wmurders, differences = 2))



From the ACF plot and PACF plot we only see a significant positive spike at lag 1 for the ACF so I will conclude that an appropriate model is ARIMA(1,2,0). This seems appropriate as we try to avoid mixed ARIMA models.

arima(wmurders, order = c(1,2,0))

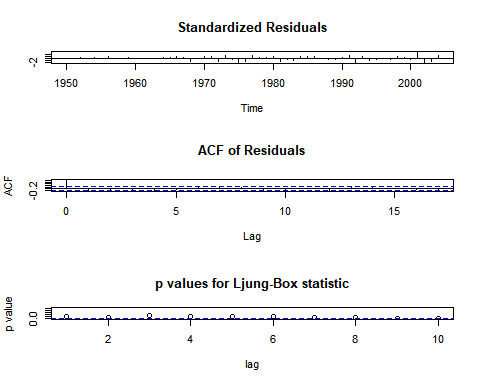
##   
## Call:  
## arima(x = wmurders, order = c(1, 2, 0))  
##   
## Coefficients:  
## ar1  
## -0.6719  
## s.e. 0.0981  
##   
## sigma^2 estimated as 0.05368: log likelihood = 2, aic = 0

1. Should you include a constant in the model? Explain.

A constant is only included when our differencing order is equal to 0, since we have a differencing order of 2 we do not include a constant in the model.

1. Write this model in terms of the backshift operator.
2. Fit the model using R and examine the residuals. Is the model satisfactory?

arima\_1\_2 <- arima(wmurders, order = c(1,2,0))  
tsdiag(arima\_1\_2)



The residuals appear to have non-constant variance, my only concern is the appearance of a pattern from 1980 to 1995. Further investigation may yield more information for this time period.

1. Forecast three times ahead. Check your forecasts by hand to make sure you know how they have been calculated.

forecast(arima\_1\_2, h = 3)

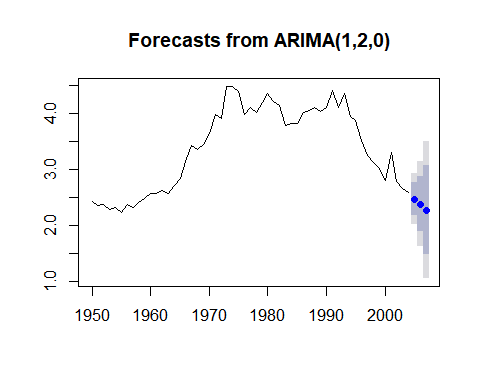
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2005 2.474461 2.177543 2.771379 2.020364 2.928557  
## 2006 2.387811 1.894189 2.881433 1.632881 3.142740  
## 2007 2.282165 1.485116 3.079214 1.063183 3.501147

tail(wmurders)

## Time Series:  
## Start = 1999   
## End = 2004   
## Frequency = 1   
## [1] 3.034263 2.805041 3.304467 2.797697 2.662227 2.589383

1. Create a plot of the series with forecasts and prediction intervals for the next three periods shown.

plot(forecast(arima\_1\_2, h = 3))



1. Does auto.arima give the same model you have chosen? If not, which model do you think is better?

auto.arima(wmurders)

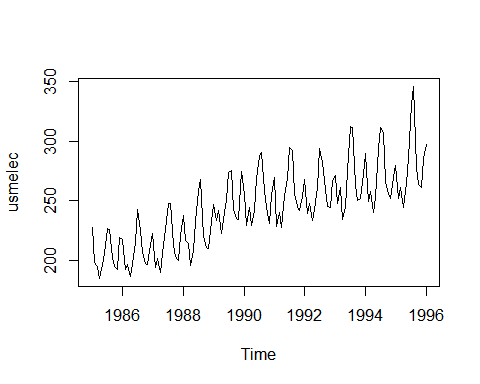
## Series: wmurders   
## ARIMA(1,2,1)   
##   
## Coefficients:  
## ar1 ma1  
## -0.2434 -0.8261  
## s.e. 0.1553 0.1143  
##   
## sigma^2 estimated as 0.04632: log likelihood=6.44  
## AIC=-6.88 AICc=-6.39 BIC=-0.97

The auto.arima method does not give the same model but it does have a lower AICc score so we would choose the auto.arima method if our main concern was predictability. If the use case allowed for lower accuracy but better intereptability I would chose the model I developed since the AICc scores are not dramatically different and we have one less term.

## Problem 8.8

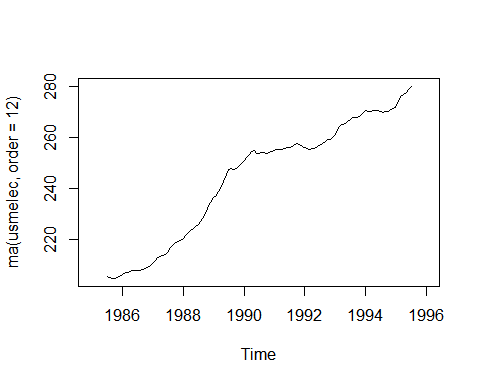
Consider the total net generation of electricity (in billion kilowatt hours) by the U.S. electric industry (monthly for the period 1985-1996). (Data set usmelec.) In general there are two peaks per year: in mid-summer and mid-winter.

data(usmelec)  
usmelec <- window(usmelec, start = 1985, end = 1996)  
plot(usmelec)



1. Examine the 12-month moving average of this series to see what kind of trend is involved.

plot(ma(usmelec, order=12))



The 12 month moving average shows a clear upward trend with somewhat trivial mountains and valleys.

1. Do the data need transforming? If so, find a suitable transformation.

library(forecast)  
BoxCox.lambda(usmelec)

## [1] -0.519621

The data could benefit from a BoxCox transformation to the power of -.5, however, I don't believe that we will see dramatic benefits to the transformation so I will forgo the transformation for the sake for interepetibility. A transformation that would be informative if the data was available would be a population adjustment since the growth of electrical needs could possibly be explained by the growth in population.

1. Are the data stationary? If not, find an appropriate differencing which yields stationary data.

We will start with the kpss test, a p-value lower than .05 indicates that differencing is required.

kpss.test(usmelec)

## Warning in kpss.test(usmelec): p-value smaller than printed p-value

##   
## KPSS Test for Level Stationarity  
##   
## data: usmelec  
## KPSS Level = 3.3024, Truncation lag parameter = 2, p-value = 0.01

paste(nsdiffs(usmelec), "order of seasonal differences")

## [1] "1 order of seasonal differences"

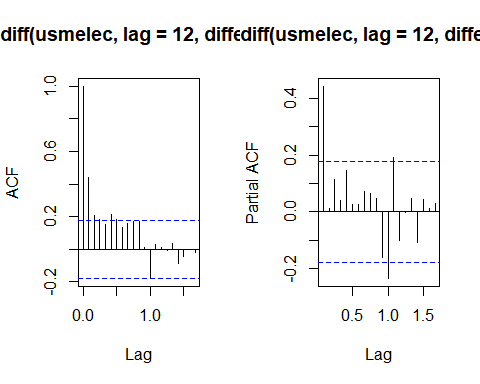
paste(ndiffs(nsdiffs(usmelec)), "order of differences")

## [1] "0 order of differences"

An order of 1 seasonal differencing is necessary to reach stationarity with this time series.

1. Identify a couple of ARIMA models that might be useful in describing the time series. Which of your models is the best according to their AIC values?

par(mfrow=c(1, 2))  
acf(diff(usmelec, lag = 12, differences = 1))  
pacf(diff(usmelec, lag = 12, differences = 1))



Based on the plots the ACF has a postive value at lag 1 and 2 for a non-seasonal MA(2) term, a positive value at PACF at lag 1 for a non-seasonal AR(1), and we know that we will only use a seasonal difference order of 1. The negative values at the first lag for the seasonal PACF and ACF suggest an AR(1) and MA(1) term but the ACF spike is just approaching significance

arima(usmelec, order = c(1,0,2), seasonal = c(1,1,1))

##   
## Call:  
## arima(x = usmelec, order = c(1, 0, 2), seasonal = c(1, 1, 1))  
##   
## Coefficients:  
## ar1 ma1 ma2 sar1 sma1  
## 0.9960 -0.4882 -0.2566 0.0701 -0.7018  
## s.e. 0.0061 0.0881 0.0868 0.1375 0.1136  
##   
## sigma^2 estimated as 57.55: log likelihood = -420.8, aic = 853.6

arima(usmelec, order = c(1,0,2), seasonal = c(1,1,0))

##   
## Call:  
## arima(x = usmelec, order = c(1, 0, 2), seasonal = c(1, 1, 0))  
##   
## Coefficients:  
## ar1 ma1 ma2 sar1  
## 0.9816 -0.5019 -0.2384 -0.3715  
## s.e. 0.0171 0.0915 0.0878 0.0908  
##   
## sigma^2 estimated as 67.75: log likelihood = -428.12, aic = 866.24

arima(usmelec, order = c(1,0,2), seasonal = c(0,1,1))

##   
## Call:  
## arima(x = usmelec, order = c(1, 0, 2), seasonal = c(0, 1, 1))  
##   
## Coefficients:  
## ar1 ma1 ma2 sma1  
## 0.9954 -0.4891 -0.2497 -0.6640  
## s.e. 0.0066 0.0879 0.0849 0.0915  
##   
## sigma^2 estimated as 57.69: log likelihood = -420.93, aic = 851.86

The best ARIMA model is ARIMA(1,0,2)(0,1,1)[12] based on the AIC score and interpetability

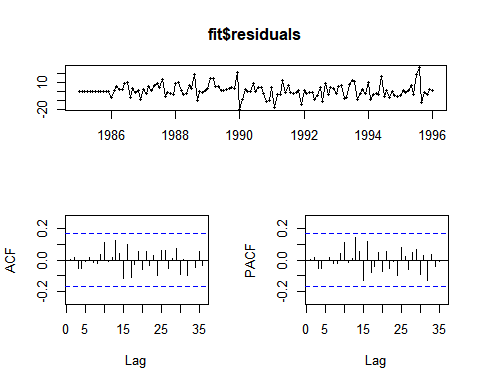
1. Estimate the parameters of your best model and do diagnostic testing on the residuals. Do the residuals resemble white noise? If not, try to find another ARIMA model which fits better.

The parameters of the best model are as follows:

fit <- arima(usmelec, order = c(1,0,2), seasonal = c(0,1,1))  
summary(fit)

##   
## Call:  
## arima(x = usmelec, order = c(1, 0, 2), seasonal = c(0, 1, 1))  
##   
## Coefficients:  
## ar1 ma1 ma2 sma1  
## 0.9954 -0.4891 -0.2497 -0.6640  
## s.e. 0.0066 0.0879 0.0849 0.0915  
##   
## sigma^2 estimated as 57.69: log likelihood = -420.93, aic = 851.86  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.8675686 7.244822 5.131991 0.2835909 2.031825 0.2990197  
## ACF1  
## Training set 0.005297723

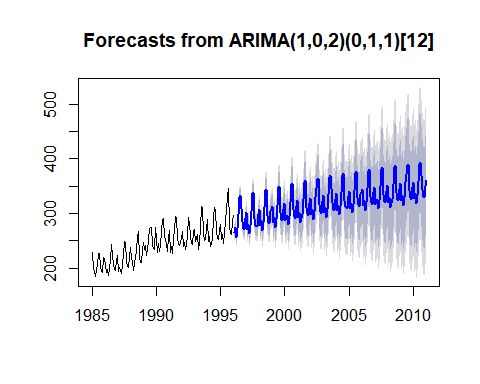
tsdisplay(fit$residuals)

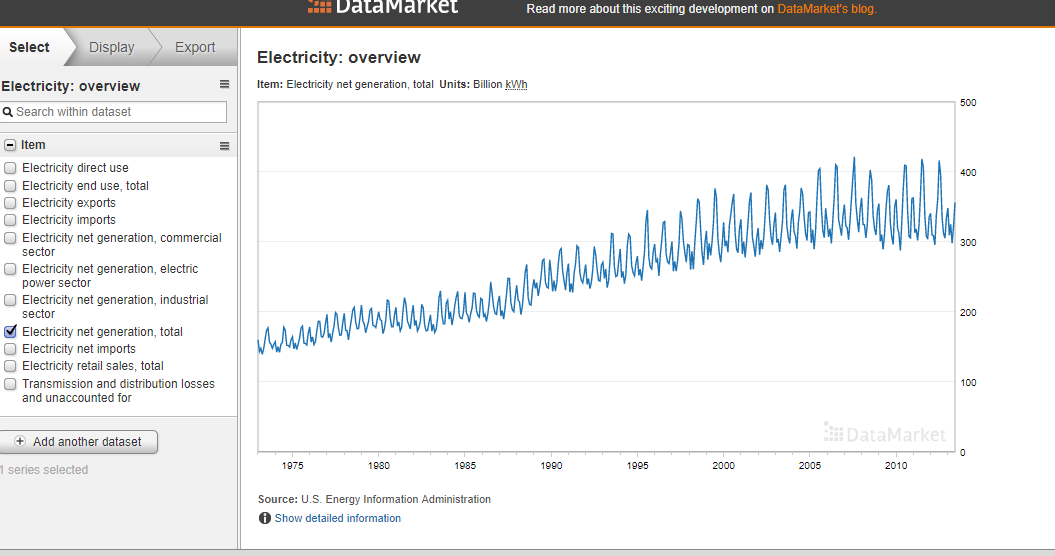


Since all the spikes are within significance limit the residuals do appear to be white noise.

1. Forecast the next 15 years of generation of electricity by the U.S. electric industry. Get the latest figures from <http://data.is/zgRWCO> to check on the accuracy of your forecasts.

plot(forecast(fit, h = (15\*12)))





"Electricity: Overview." DataMarket, datamarket.com/data/set/1d9r/electricity-overview#!display=line&ds=1d9r!x0z=8.

1. How many years of forecasts do you think are sufficiently accurate to be usable?

It depends on the use case, if this is just to have a high level understanding of the tragejectory it does that job well for a number of years. However, if accuracy is extremely important than the model has dimishing accuracy even after a few years. However, as best I can tell I believe its accuracte up to year 2000 so ~4 years is about the usuable forecast period for this model.

# Appendix with all code

All code used previous in below section

library(fma)  
library(ggfortify)  
library(tidyverse)  
library(psych)  
library(knitr)  
library(mlbench)  
library(Hmisc)  
library(fpp)  
library(psych)  
library(missForest)  
library(forecast)  
library(caret)  
library(reshape)  
library(reshape)  
library(caret)  
library(forecast)  
library(missForest)  
library(fma)  
library(ggfortify)  
library(tidyverse)  
library(psych)  
library(knitr)  
library(mlbench)  
library(Hmisc)  
library(fpp)  
library(psych)  
library(fma)  
library(ggfortify)  
data(dole)  
autoplot(dole, xlab = "Year",   
 ylab = "# of Unemployed \n",   
 main = "Monthly total of people on unemployed benefits in Australia \n (January 1956-July 1992)")  
autoplot(BoxCox(dole, BoxCox.lambda(dole)),   
 xlab = "Year",   
 ylab = paste0("# of Unemployed \n", "lambda = ", round(BoxCox.lambda(dole),2), " \n"),  
 main = "BoxCox Transformation \nMonthly total of people on unemployed benefits in Australia \n(January 1956-July 1992)")  
data(usdeaths)  
autoplot(usdeaths,  
 ylab="Total Deaths Per Month",  
 xlab="Year",  
 main="Monthly total of accidental deaths in the United States \n(January 1973-December 1978)")  
monthdays <- rep(c(31,28,31,30,31,30,31,31,30,31,30,31),6)  
monthdays[38] <- 29 #Leap year adjustment  
  
autoplot(usdeaths/monthdays, ylab="Average Deaths Per Day",  
 xlab="Year",  
 main="Average accidental deaths per day by Month in the United States \n(January 1973-December 1978)")  
data(bricksq)  
autoplot(bricksq,  
 xlab = "Year",   
 ylab = paste0("# of bricks (in millions of units)"),  
 main = "Quarterly production of bricks (in millions of units) \nPortland, Australia (March 1956-September 1994)")  
autoplot(BoxCox(bricksq, BoxCox.lambda(bricksq)),   
 xlab = "Year",   
 ylab = paste0("# of bricks (in millions of units) \n", "lambda = ", round(BoxCox.lambda(bricksq),2), " \n"),  
 main = "BoxCox Transformation \nQuarterly production of bricks (in millions of units) \nPortland, Australia (March 1956-September 1994)")  
library(psych)  
library(knitr)  
data(ibmclose)  
summary(ibmclose)  
autoplot(ibmclose,   
 xlab = "\nDay",   
 ylab = "Close\n",  
 main = "Daily closing IBM stock prices")  
lag.plot(ibmclose, lags= 10, do.lines=FALSE)  
ibmclose\_train <- window(ibmclose, start=1, end=300)  
ibmclose\_test <- window(ibmclose, start=301)  
ibmclosefit1 <- meanf(ibmclose\_train, h = 69)  
ibmclosefit2 <- rwf(ibmclose\_train, h = 69)  
ibmclosefit3 <- snaive(ibmclose\_train, h = 69)  
  
plot(ibmclosefit1, col = 1, main="Forecasts for IBM Close")  
lines(ibmclosefit2$mean, col=2)  
lines(ibmclosefit3$mean, col=3)  
lines(ibmclose)  
legend("topright",   
 cex = .75,  
 lty=1,   
 col=c(4,2,3),   
 legend=c("Mean method",   
 "Naive method",   
 "Seasonal naive method"))  
kable(accuracy(ibmclosefit1, ibmclose\_test))  
kable(accuracy(ibmclosefit2, ibmclose\_test))  
kable(accuracy(ibmclosefit3, ibmclose\_test))  
library(fma)  
plastics  
library(ggfortify)  
autoplot(plastics, xlab = "Year", ylab = "monthly sales (in thousands) \n")  
fit <- decompose(plastics, type="multiplicative")  
plot(fit)  
fit <- stl(plastics, t.window = 7, s.window="periodic", robust=TRUE)  
autoplot(seasadj(fit), xlab = "Year", ylab = "seasonally adjusted data monthly sales(in thousands) \n")  
outlier\_plastics <- plastics  
outlier\_plastics[34] <- outlier\_plastics[30] + 600  
fit <- stl(outlier\_plastics, t.window = 7, s.window="periodic", robust=TRUE)  
autoplot(seasadj(fit), xlab = "Year", ylab = "seasonally adjusted data monthly sales \n with outlier (in thousands) \n")  
outlier\_plastics <- plastics  
outlier\_plastics[2] <- outlier\_plastics[2] + 600  
fit <- stl(outlier\_plastics, t.window = 7, s.window="periodic", robust=TRUE)  
autoplot(seasadj(fit), xlab = "Year", ylab = "seasonally adjusted data monthly sales \n with outlier (in thousands) \n")  
fc <- stlf(plastics, method="naive")  
plot(fc)  
fit <- stl(plastics, s.window="periodic", robust=TRUE)  
fitadj <- seasadj(fit)  
plot(naive(fitadj))  
library(mlbench)  
library(Hmisc)  
data(Glass)  
describe(Glass)  
library(corrplot)  
library(PerformanceAnalytics)  
Glass %>% dplyr::select(-(Type)) %>% cor() %>% chart.Correlation(histogram=TRUE, pch=19)  
library(ggplot2)  
ggplot(stack(Glass), aes(x = ind, y = values)) +  
 facet\_wrap(~ind, scales = "free", nrow = 1) +   
 geom\_boxplot()  
library(ggplot2)  
ggplot(stack(Glass), aes(values)) +  
 facet\_wrap(~ind, scales = "free") +   
 geom\_histogram()  
apply(Glass %>% dplyr::select(-(Type)), 2, BoxCoxTrans)  
library(mlbench)  
data(Soybean)  
t <- melt(apply(Soybean %>% dplyr::select(-plant.stand, -precip, -temp), 2, FUN = table)) %>%   
 rename(c("Var.1" = "Value", "value" = "Count", "L1" = "Variable"))  
  
ggplot(t %>% dplyr::filter(Variable == "Class"), aes(x = Value, y = Count, group = Variable)) +  
 geom\_col() +  
 theme(axis.text.x = element\_text(angle = 75, hjust = 1, vjust = 1))  
ggplot(t %>% dplyr::filter(Variable != "Class"), aes(x = Value, y = Count, group = Variable)) +  
 facet\_wrap(~Variable, scales = "free") +  
 geom\_col()  
Soybean %>%   
 summarise\_each(funs(sum(is.na(.))/nrow(Soybean))) %>%   
 melt() %>%   
 ggplot(aes(x = variable, y = value)) +  
 geom\_col() +  
 ggtitle("Missing Values") +  
 theme(axis.text.x = element\_text(angle = 75, hjust = 1, vjust = 1))  
Soybean %>%   
 group\_by(Class) %>%   
 summarise\_all(funs(sum(is.na(.))/nrow(Soybean))) %>%   
 as.data.frame() %>%  
 melt() %>%  
 ggplot(aes(x = variable, y = value, fill = Class)) +  
 geom\_col() +  
 #facet\_grid(~variable, scales = "free") +  
 ggtitle("Missing Values by Class") +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1, vjust = .1))  
library(missForest)  
Soybean.imp <- missForest(Soybean)  
t <- melt(apply(Soybean.imp$ximp %>%   
 dplyr::select(-plant.stand, -precip, -temp), 2, FUN = table)) %>%   
 rename(c("Var.1" = "Value", "value" = "Count", "L1" = "Variable"))  
  
ggplot(t %>% dplyr::filter(Variable != "Class"), aes(x = Value, y = Count, group = Variable)) +  
 facet\_wrap(~Variable, scales = "free") +  
 geom\_col()  
library(fma)  
library(tidyverse)  
library(ggfortify)  
books <- fma::books  
plot(books)  
autoplot(books)  
paperback <- books[,1]  
  
fit1 <- ses(paperback, alpha=0.4, initial="simple", h=3)  
fit2 <- ses(paperback, alpha=0.6, initial="simple", h=3)  
fit3 <- ses(paperback, alpha=0.8, h=3)  
  
plot(fit1, ylab="Paperback Books Sold",  
 xlab="Time Series", main="", fcol="white", type="o")  
lines(fitted(fit1), col="blue", type="o")  
lines(fitted(fit2), col="red", type="o")  
lines(fitted(fit3), col="green", type="o")  
lines(fit1$mean, col="blue", type="o")  
lines(fit2$mean, col="red", type="o")  
lines(fit3$mean, col="green", type="o")  
legend("topleft",lty=1, col=c(1,"blue","red","green"),   
 c("data", expression(alpha == 0.4), expression(alpha == 0.6),  
 expression(alpha == 0.8)),pch=1)  
fit\_opt <- ses(paperback, initial="simple", h=4)  
plot(fit\_opt, ylab="Paperback Books Sold",  
 xlab="Time Series", main="", fcol="white", type="o")  
lines(fitted(fit\_opt), col="blue", type="o")  
lines(fit\_opt$mean, col="blue", type="o")  
summary(fit\_opt)  
fit\_opt <- ses(paperback, initial="optimal", h=4)  
plot(fit\_opt, ylab="Paperback Books Sold",  
 xlab="Time Series", main="", fcol="white", type="o")  
lines(fitted(fit\_opt), col="blue", type="o")  
lines(fit\_opt$mean, col="blue", type="o")  
summary(fit\_opt)  
hardcover <- books[,2]  
  
fit1 <- ses(hardcover, alpha=0.4, initial="simple", h=3)  
fit2 <- ses(hardcover, alpha=0.6, initial="simple", h=3)  
fit3 <- ses(hardcover, alpha=0.8, h=3)  
  
plot(fit1, ylab="Hardcover Books Sold",  
 xlab="Time Series", main="", fcol="white", type="o")  
lines(fitted(fit1), col="blue", type="o")  
lines(fitted(fit2), col="red", type="o")  
lines(fitted(fit3), col="green", type="o")  
lines(fit1$mean, col="blue", type="o")  
lines(fit2$mean, col="red", type="o")  
lines(fit3$mean, col="green", type="o")  
legend("topleft",lty=1, col=c(1,"blue","red","green"),   
 c("data", expression(alpha == 0.4), expression(alpha == 0.6),  
 expression(alpha == 0.8)),pch=1)  
fit\_opt <- ses(hardcover, initial="simple", h=4)  
plot(fit\_opt, ylab="Hardcover Books Sold",  
 xlab="Time Series", main="", fcol="white", type="o")  
lines(fitted(fit\_opt), col="blue", type="o")  
lines(fit\_opt$mean, col="blue", type="o")  
summary(fit\_opt)  
fit\_opt <- ses(hardcover, initial="optimal", h=4)  
plot(fit\_opt, ylab="Paperback Books Sold",  
 xlab="Time Series", main="", fcol="white", type="o")  
lines(fitted(fit\_opt), col="blue", type="o")  
lines(fit\_opt$mean, col="blue", type="o")  
summary(fit\_opt)  
library(expsmooth)  
ukcars <- expsmooth::ukcars  
autoplot(ukcars)  
fit\_stl <- stl(ukcars, s.window = "periodic")  
plot(fit\_stl)  
sadj <- seasadj(fit\_stl)  
sfactors <- fit\_stl$time.series[2:11, "seasonal"]   
fit\_dpd\_sadj <- holt(sadj, damped = TRUE)  
paste0("Additive damped RMSE : ",round(accuracy(fit\_dpd\_sadj)[2], 3))  
rsfit\_dpd\_sadj <- fit\_dpd\_sadj$mean + sfactors   
rsfit\_dpd\_sadj  
plot(ukcars,   
 type = "o",  
 xlab = "Years",   
 ylab = "UK Cars",   
 xlim = c(1997, 2008))  
  
lines(rsfit\_dpd\_sadj, type = "o", col = "blue")  
fit\_hl <- holt(sadj)  
paste0("Holt Linear Method with seasonal adjustment RMSE: ",round(accuracy(fit\_hl)[2],3))  
autoplot(fit\_hl, xlab = "Years", ylab = "UK Car Production")  
accuracy(fit\_hl)  
rs\_l <- fit\_hl$mean + sfactors  
plot(ukcars, ylab = "UK Care Production")  
lines(rs\_l, type = "o", col = "red")  
fit\_ets <- ets(ukcars, model = "ANN")  
summary(fit\_ets)  
autoplot(forecast(fit\_ets), xlab = "Years", ylab = "UK Car Production")  
paste0("RMSE from Simple Exp Smoothing Method: ",round(accuracy(fit\_ets)[2], 3))  
paste0("Additive Damped - RMSE : ",round(accuracy(fit\_dpd\_sadj)[2], 3))  
paste0("Holt - RMSE: ",round(accuracy(fit\_hl)[2],3))  
paste0("RMSE from Simple Exp Smoothing Method: ",round(accuracy(fit\_ets)[2], 3))  
autoplot(forecast(fit\_dpd\_sadj))  
autoplot(forecast(fit\_hl))  
autoplot(forecast(fit\_ets))  
library(fma)  
data(ibmclose)  
par(mfrow=c(1, 2))  
plot(ibmclose)  
plot(diff(ibmclose,1))  
par(mfrow=c(1, 3))  
acf(ibmclose)  
acf(diff(ibmclose,1))  
par(mfrow=c(1, 2))  
pacf(ibmclose)  
pacf(diff(ibmclose,1))  
data(wmurders)  
plot(wmurders)  
plot(diff(wmurders, differences = 2))  
par(mfrow=c(1, 2))  
acf(diff(wmurders, differences = 2))  
pacf(diff(wmurders, differences = 2))  
arima(wmurders, order = c(1,2,0))  
arima\_1\_2 <- arima(wmurders, order = c(1,2,0))  
tsdiag(arima\_1\_2)  
forecast(arima\_1\_2, h = 3)  
tail(wmurders)  
plot(forecast(arima\_1\_2, h = 3))  
auto.arima(wmurders)  
data(usmelec)  
usmelec <- window(usmelec, start = 1985, end = 1996)  
plot(usmelec)  
plot(ma(usmelec, order=12))  
library(forecast)  
BoxCox.lambda(usmelec)  
kpss.test(usmelec)  
paste(nsdiffs(usmelec), "order of seasonal differences")  
paste(ndiffs(nsdiffs(usmelec)), "order of differences")  
par(mfrow=c(1, 2))  
acf(diff(usmelec, lag = 12, differences = 1))  
pacf(diff(usmelec, lag = 12, differences = 1))  
arima(usmelec, order = c(1,0,2), seasonal = c(1,1,1))  
arima(usmelec, order = c(1,0,2), seasonal = c(1,1,0))  
arima(usmelec, order = c(1,0,2), seasonal = c(0,1,1))  
fit <- arima(usmelec, order = c(1,0,2), seasonal = c(0,1,1))  
summary(fit)  
tsdisplay(fit$residuals)  
plot(forecast(fit, h = (15\*12)))  
## NA