---

title: "624\_Week1"

output: html\_document

---

```{r}

library(readxl)

library(ggplot2)

library(forecast)

library(fpp2)

```

#2.3: Download some monthly Australian retail data from the book website. These represent retail sales in various categories for different Australian states, and are stored in a MS-Excel file.

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

```{r}

retaildata <- read.csv("https://raw.githubusercontent.com/PLombardo811/624\_SPS/Homework/Final%20Code/retail.csv")

names(retaildata) <- as.matrix(retaildata[1, ])

retaildata <- retaildata[-1, ]

retaildata[] <- lapply(retaildata, function(x) type.convert(as.character(x)))

```

Reviewing data:

```{r}

head(retaildata)

```

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

```{r}

myts <- ts(retaildata[,"A3349503T"],

frequency=12, start=c(1982,4))

```

##c. Explore your chosen retail time series using the following functions:

```{r}

autoplot(myts)

ggseasonplot(myts)

ggsubseriesplot(myts)

gglagplot(myts)

ggAcf(myts)

```

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

Analysis: The autoplot and ggseason plot show the clearest trends. It appears as though the trend is cyclical, peaking in the winter months. However, there is also an upward trend as the data moves closer to more recent years.

#2.7: The arrivals data set comprises quarterly international arrivals (in thousands) to Australia from Japan, New Zealand, UK and the US.

Use autoplot(), ggseasonplot() and ggsubseriesplot() to compare the differences between the arrivals from these four countries.

```{r}

arrivals <- ts(arrivals, frequency=4, start=c(1981,1))

head(arrivals)

```

```{r}

autoplot(arrivals)

```

```{r}

ggseasonplot(arrivals[,1])

ggseasonplot(arrivals[,2])

ggseasonplot(arrivals[,3])

ggseasonplot(arrivals[,4])

```

```{r}

ggsubseriesplot(arrivals[,1])

ggsubseriesplot(arrivals[,2])

ggsubseriesplot(arrivals[,3])

ggsubseriesplot(arrivals[,4])

```

```{r}

gglagplot(arrivals[,1])

gglagplot(arrivals[,2])

gglagplot(arrivals[,3])

gglagplot(arrivals[,4])

```

```{r}

ggAcf(arrivals)

```

##Can you identify any unusual observations?

It is interesting and perhaps unusual that acrioss all countries, there appear to be similar seasonal trends - international arrivals peaking towards the end of the year and declining to troughs during the second and third quarters. Given that some of these countries have different seasons, you might expect some more variation across countries.

#2.10: dj contains 292 consecutive trading days of the Dow Jones Index. Use ddj <- diff(dj) to compute the daily changes in the index. Plot ddj and its ACF. Do the changes in the Dow Jones Index look like white noise?

The difference plot looks random except for a huge drop around the 100th day but ACF shows small correaltion for lag 35 and 48, so the change is very close to white noise but not completely.

```{r}

ddj <- diff(dj)

autoplot(ddj)

```

#3.1: For the following series, find an appropriate Box-Cox transformation in order to stabilise the variance.

\* usnetelec

\* usgdp

\* mcopper

\* enplanements

```{r}

plot(usnetelec); plot(BoxCox(usnetelec,lambda=1/2))

plot(usgdp); plot(BoxCox(usgdp,lambda=0))

plot(mcopper); plot(BoxCox(mcopper,lambda=1/3))

plot(enplanements); plot(BoxCox(enplanements,lambda=0))

```

#3.8: For your retail time series (from Exercise 3 in Section 2.10):

##a. Split the data into two parts using

```{r}

myts.train <- window(myts, end=c(2010,12))

myts.test <- window(myts, start=2011)

```

##b. Check that your data have been split appropriately by producing the following plot.

```{r}

autoplot(myts) +

autolayer(myts.train, series="Training") +

autolayer(myts.test, series="Test")

```

##c. Calculate forecasts using snaive applied to myts.train.

```{r}

fc <- snaive(myts.train)

```

##d. Compare the accuracy of your forecasts against the actual values stored in myts.test.

```{r}

accuracy(fc,myts.test)

```

##e. Check the residuals.

```{r}

checkresiduals(fc)

```

##Do the residuals appear to be uncorrelated and normally distributed?

From the ACF plot we see the residuals are not uncorrelated and from the distribution also shows that the residuals are right skewed

the accuracy for training set is much better than the test set among all error measures

##f. How sensitive are the accuracy measures to the training/test split?

Analysis: we tried varying the years of the training set and test set to assess how much the accuracy changed depending on the selection. As you will see, there was wide variation in the accuracy measures of the projections - especially on the test sets - depending on the training/test split (across all measures.)

```{r}

for(i in 2008:2012) {

myts\_new.test <- window(myts, start=i+1)

myts\_new.train <- window(myts, end=c(i,12))

fc\_new <- snaive(myts\_new.train)

accuracy\_new <- accuracy(fc\_new,myts\_new.test)

print(accuracy\_new)

}

```

---

title: "624\_HW2\_Group3""

output: html\_document

---

```{r}

library(readxl)

library(ggplot2)

library(forecast)

library(fpp2)

```

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

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

```{r}

autoplot(plastics)

```

##The plot of the data shows that there are seasonal fluctuations and upward trend.

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

```{r}

decompose\_plastics <- decompose(plastics,

type = "multiplicative")

autoplot(decompose\_plastics)

```

##C Do the results support the graphical interpretation from part a?

Yes, the plot does show an upward trend and presence of the seasonal fluctuations.

##D Compute and plot the seasonally adjusted data.

```{r}

autoplot(plastics, series="Data") +

autolayer(trendcycle(decompose\_plastics), series="Trend") +

autolayer(seasadj(decompose\_plastics), series="Seasonal Data") +

xlab("Year") + ylab("Monthly Sales") +

ggtitle("Sales of plastic products") +

scale\_colour\_manual(values=c("gray","blue","black"),

breaks=c("Data","Seasonal Data","Trend"))

```

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

```{r}

plastics\_outlier <- plastics

plastics\_outlier [50] <- plastics\_outlier [50] + 500

decompose\_plastics\_outlier<- decompose(

plastics\_outlier,

type = "multiplicative"

)

autoplot(plastics\_outlier, series="Data") +

autolayer(trendcycle(decompose\_plastics\_outlier), series="Trend") +

autolayer(seasadj(decompose\_plastics\_outlier), series="Seasonal Data") +

xlab("Year") + ylab("Monthly Sales") +

ggtitle("Sales of plastic products") +

scale\_colour\_manual(values=c("gray","blue","black"),

breaks=c("Data","Seasonal Data","Trend"))

```

##The outlier affects trend little, but affects the seasonally adjusted data severely. Seasonally adjusted data have errors like the original data have.

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

```{r}

plastics\_outlier2 <- plastics

plastics\_outlier2 [10] <- plastics\_outlier2 [10] + 500

decompose\_plastics\_outlier2<- decompose(

plastics\_outlier2,

type = "multiplicative"

)

autoplot(plastics\_outlier2, series="Data") +

autolayer(trendcycle(decompose\_plastics\_outlier2), series="Trend") +

autolayer(seasadj(decompose\_plastics\_outlier2), series="Seasonal Data") +

xlab("Year") + ylab("Monthly Sales") +

ggtitle("Sales of plastic products") +

scale\_colour\_manual(values=c("gray","blue","black"),

breaks=c("Data","Seasonal Data","Trend"))

```

##If an outlier is near the end, the effect to trend decreases.

#6.6 We will use the bricksq data (Australian quarterly clay brick production. 1956â“1994) for this exercise.

##A Use an STL decomposition to calculate the trend-cycle and seasonal indices. (Experiment with having fixed or changing seasonality.)

```{r}

# STL decomposition fixed seasonality

fixed\_stl <- stl(bricksq,

s.window = "periodic",

robust = TRUE)

# STL decomposition changing seasonality

changing\_stl <- stl(bricksq,

s.window = 8,

robust = TRUE)

```

```{r}

#Plot Data

autoplot(fixed\_stl) +

ggtitle('Brick STL Fixed')

autoplot(changing\_stl) +

ggtitle('Brick STL Changing')

```

##B Compute and plot the seasonally adjusted data.

```{r}

# fixed seasonality

autoplot(bricksq, series = 'Data') +

autolayer(seasadj(fixed\_stl),

series = 'Seasonal Data') +

autolayer(trendcycle(fixed\_stl),

series = 'Trend cycle') +

ggtitle('Australian quarterly clay brick production. 1956-1994',

subtitle = 'decomposed using STL with fixed seasonality') +

scale\_color\_manual(values = c('gray', 'blue', ' black'),

breaks = c('Data', 'Seasonal Data','Trend cycle'))

# changing seasonality

autoplot(bricksq, series = 'Data') +

autolayer(seasadj(changing\_stl),

series = 'Seasonal Data') +

autolayer(trendcycle(changing\_stl),

series = 'Trend cycle') +

ggtitle('Australian quarterly clay brick production. 1956-1994',

subtitle = 'decomposed using STL with changing seasonality') +

scale\_color\_manual(values = c('gray', 'blue', ' black'),

breaks = c('Data', 'Seasonal Data','Trend cycle'))

```

##C Use a naÃ¯ve method to produce forecasts of the seasonally adjusted data.

```{r}

fixed\_stl %>% seasadj() %>% naive() %>% autoplot() +

ggtitle(label = "Naive forecast for seasonally adjusted brick data",

subtitle = "decomposed using STL with fixed seasonality")

changing\_stl %>% seasadj() %>% naive() %>% autoplot() +

ggtitle(label = "Naive forecast for seasonally adjusted brick data",

subtitle = "decomposed using STL with changing seasonality")

```

#the prediction intervals of seasonally adjusted data decomposed by STL have smaller ranges than the fixed seasonality. This is because the variance of the remainder component decreased when the seasonality is changed.

##D Use stlf() to reseasonalise the results, giving forecasts for the original data.

```{r}

stlf\_brick <- stlf(bricksq)

autoplot(stlf\_brick)

```

##E Do the residuals look uncorrelated?

```{r}

checkresiduals(stlf\_brick)

```

##The residuals appear to be correlated with each other.

##F Repeat with a robust STL decomposition. Does it make much difference?

```{r}

stlf\_brick\_robust <- stlf(bricksq, robust = TRUE)

autoplot(stlf\_brick\_robust)

checkresiduals(stlf\_brick\_robust)

```

##The autocorrelation lowered in general, but there are still some high value outliers.

##G Compare forecasts from stlf() with those from snaive(), using a test set comprising the last 2 years of data. Which is better?

```{r}

train\_data <- subset(bricksq,

end = length(bricksq) - 7)

test\_data <- subset(bricksq,

start = length(bricksq) - 9)

snaive\_brick <- snaive(train\_data)

stlf\_brick\_part <- stlf(train\_data, robust = TRUE)

autoplot(bricksq, series = "Original") +

geom\_line(size = 1) +

autolayer(stlf\_brick\_part, PI = FALSE, size = 1,

series = "stlf forecast") +

autolayer(snaive\_brick, PI = FALSE, size = 1,

series = "snaive forecast") +

scale\_color\_manual(values = c("gray", "blue", "green"),

breaks = c("Original", "stlf forecast", "snaive forecast")) +

scale\_x\_continuous(limits = c(1990, 1994.5)) +

scale\_y\_continuous(limits = c(350, 550)) +

guides(colour = guide\_legend(title = "Data")) +

ggtitle("Forecast comparison between stlf and snaive functions") +

annotate(

"rect",

xmin=1993,xmax=1994.5,ymin=-Inf,ymax=Inf,

fill="light blue",alpha = 0.5

)

```

##The forecasts from stlf function are more similar to the original data than the forecasts from snaive function. The stlf function trends, and its seasonality tend to change over a period time. Therefore stlf function was better than snaive function to predict brick production amounts for near future.

---

title: "624\_HW3\_Group3"

author: "Group 3"

output: html\_document

---

# Exercise 3.1.

The UC Irvine Machine Learning Repository 6 contains a data set related

to glass identification. The data consist of 214 glass samples labeled as one

of seven class categories. There are nine predictors, including the refractive

index and percentages of eight elements: Na, Mg, Al, Si, K, Ca, Ba, and Fe.

## 3.1 (a) Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors.

##### Accessing the data:

The data was accessed using the library 'mlbench' that loads the glass data. So as the first step the library was loaded first, the 'suppressMessages' and 'suppressWarnings' functions were used to avoid warning and messages:

```{r}

suppressMessages(suppressWarnings(library(mlbench)))

```

load other libraries:

```{r}

suppressMessages(suppressWarnings(library(caret)))

suppressMessages(suppressWarnings(library(stats)))

suppressMessages(suppressWarnings(library(corrplot)))

suppressMessages(suppressWarnings(library(ggplot2)))

suppressMessages(suppressWarnings(library(dplyr)))

```

Access and load the appropriate data:

```{r}

Glass

```

##### visualization of the predictor variables:

histograms and density plots were produced by looping through the variables in Glass dataset. Since the varible 'Type' is not a numeric variable, it was not considered in the loop. The 'mar' and 'mfrow' parameters of par function were set to arrange all the plots in a grid.

A Correlation matrix and a correlation plot were also produced to better understand the data and the relationship between predictor variables

```{r}

par(mar=c(3,1,1,1))

par(mfrow=c(4,3))

for (i in 1:(length(colnames(Glass))-1)){

hist(Glass[,i], main = colnames(Glass)[i], col="gray", prob=TRUE )

lines(density(Glass[,i]), col="red", lw=2)

grid()

}

```

correlation matrix of the numeric variables:

```{r}

numeric\_glass <- Glass[,-10]

corrrelations <- cor(numeric\_glass)

corrrelations

```

correlation plot:

```{r }

corrplot.mixed(corrrelations, lower = "number", upper = "circle")

```

Finally a scatterplot matrix was produced to observe the relationships and patterns of the variables within the total context of the data:

```{r}

pairs(Glass)

```

#### Answer:

Histogram and density plots show that the variables RI, Al, Na and Si have near normal distributions. The variables K, Ca, Ba, Fe are heavily right skewed while Mg has left skewed distribution. Ca, Ba, and Fe have unimodal and Mg and K have bimodal distributions. Both the correlation and scatterplot matrix plots show a strong positive correlation between RI and Ca. Moderately strong but negative Correlations exist between RI and Si, Mg and AI, Mg and Ba, and Mg and Ca. A moderately strong correlation is also present between AI and Ba.

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

While the histogram amd density plots show the skewness of the variables, boxplots of the variable were produced to examine the presence of any outliers:

```{r}

par(mar=c(1,1,1,1))

par(mfrow=c(2,5))

for (i in 1:(length(colnames(Glass))-1)){

boxplot(Glass[,i], main = colnames(Glass)[i], col="gray", prob=TRUE )

}

```

#### Answer:

The boxplots indicates significant outliers (i.e. values fall outside of the whisker) in all variables except Na and Mg. The combination of long and short whiskers in the boxplots for Mg, K, Fe, Ba and Ca prove that their distributions are skewed, which match with the findings in histograms and density plots as discussed earlier.

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

As a first step to determine the relevant transformations the skewness of distributions were computed for those variables who seem to have skewed distributions as determined earlier (i.e. the variables K, Ca, Ba, Fe and Mg ). The library e1071 was loaded to compute skewness.

```{r}

suppressMessages(suppressWarnings(library(e1071)))

```

Skewness of variables:

```{r}

paste( "skewness of k before transformation: ",round(skewness(Glass$K),3))

paste( "skewness of Ca before transformation: ",round(skewness(Glass$Ca),3))

paste( "skewness of Ba before transformation: ",round(skewness(Glass$Ba),3))

paste( "skewness of Fe before transformation: ",round(skewness(Glass$Fe),3))

paste( "skewness of Mg before transformation: ",round(skewness(Glass$Mg),3))

```

#### Answer:

SO except Mg all the above variables have positive skewness. As per Turkey's Ladder of Powers or Bulging Rule, going up with the power or postive exponents reduces negative skew and going down with the power or negative exponent reduces positive skew. After trial and error it seems for k and Ba Reciprocal transformation and for Ca and Fe Reciprocal Squre transformation brings their skewness to some acceptable values. A cube transformation on Mg was found to be acceptable.

Skewness of the variables after transformation:

```{r}

paste( "skewness of k after transformation: ",round(skewness(Glass$K)^(-1),3))

paste( "skewness of Ca after transformation: ",round(skewness(Glass$Ca)^(-2),3))

paste( "skewness of Ba after transformation: ",round(skewness(Glass$Ba)^(-1),3))

paste( "skewness of Fe after transformation: ",round(skewness(Glass$Fe)^(-2),3))

paste( "skewness of Mg after transformation: ",round(skewness((Glass$Mg)^3),3))

```

## Exercise 3.2.

The soybean data can also be found at the UC Irvine Machine Learning

Repository. Data were collected to predict disease in 683 soybeans. The 35

predictors are mostly categorical and include information on the environmental

conditions (e.g., temperature, precipitation) and plant conditions (e.g., left

spots, mold growth). The outcome labels consist of 19 distinct classes.

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

#### solution:

When there is no variance in variables i.e. when a variable has only one single unique value (a constant with probability of 1) or the existance of variables with near zero varaince indicating high frequency of a single value and very low frequency of some other unique values - the distributions of those variables are considered degenerate. The nearZeroVar function in Caret package can be used to indentify the zero or near-zero variances in variables:

load data:

```{r}

Soybean

str(Soybean)

```

Degenerate distributions in Soybean data:

```{r}

dgenDF <- data.frame (nearZeroVar(Soybean[,1:36], names = TRUE, saveMetrics = TRUE))

dgenDF[dgenDF$zeroVar==TRUE | dgenDF$nzv==TRUE,]

```

The above table indicate three degenerate distributions with near-zero variances.

#### Answer:

The distributions of leaf.mild, mycelium and sclerotia are degenerate because all of these variables represent non-zero variances

#### 3.2(b) Roughly 18% of the data are missing. Are there particular predictors that are more likely to be missing? Is the pattern of missing data related to the classes?

#### solution:

```{r}

data(Soybean)

```

Cheking for missing values:

```{r}

Na\_soy <- as.matrix(colSums(is.na(Soybean)))

colnames(Na\_soy) <- paste("Missing values")

Na\_ratio <- round(Na\_soy[,"Missing values"]/nrow(Soybean),3)

Na\_soy <- cbind(Na\_soy,Na\_ratio)

Na\_soy

```

The above matrix shows certain predictor varaibles (hail, sever, seed.tmt, lodging, ) have the most missing values.

In order to find if the missing values are related to the classes, all the rows with missing values were filtered and then rows were grouped by classes.

```{r}

Na\_data <- filter(Soybean,!complete.cases(Soybean)) %>% group\_by(Class)

Na\_data

```

Finally the count of the classes were considered to find which class causing the most missing values.

```{r}

count(Na\_data,Class,sort=TRUE)

```

#### Answer:

The variables- hail, sever, seed.tmt, lodging have the most missing values. The missing values do seem to be related with classes. It is also apparent that phytophthora-rot class causing the most missing values.

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

#### solution::

Total rows with missing values:

```{r}

nrow(filter(Soybean,!complete.cases(Soybean)))

```

Total rows in the dataset:

```{r}

nrow(Soybean)

```

percentage of missing values

```{r}

(nrow(filter(Soybean,!complete.cases(Soybean))))/nrow(Soybean)

```

#### Answer:

If the total rows with missing values (121)were removed from the original data the remaining data is still significant (562) so based on the context or type of models and research question it is possible to remove all the rows with missing values. On the other hand R packages such as MICE,Amelia, missForest, Hmisc and mi can be used for imputing missing values.

---

title: "624\_HW4\_Group3"

output: html\_document

---

```{r}

suppressMessages(suppressWarnings(library(readxl)))

suppressMessages(suppressWarnings(library(ggplot2)))

suppressMessages(suppressWarnings(library(forecast)))

suppressMessages(suppressWarnings(library(fpp2)))

suppressMessages(suppressWarnings(library(mlbench)))

suppressMessages(suppressWarnings(library(caret)))

suppressMessages(suppressWarnings(library(caTools)))

```

#7.6

##A Now apply Holtas linear method to the paperback and hardback series and compute four-day forecasts in each case.

```{r}

holt\_Paper <- holt(books[,1], initial = "simple", h=4)

summary(holt\_Paper)

plot(holt\_Paper)

```

```{r}

holt\_Hardcover <- holt(books[,2], initial = "simple", h=4)

summary(holt\_Hardcover)

plot(holt\_Hardcover)

```

#We can see the linear trend in the forecasts plotted.

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

```{r}

new\_Paper <- sqrt(mean(holt\_Paper$residuals^2))

new\_Hardcover <- sqrt(mean(holt\_Hardcover$residuals^2))

new\_Paper

new\_Hardcover

```

#For both series, RMSE values became lower when Holt's method was used. If there is an approxiamately linear trend in data, it would be better to use Holt's linear method even if one more parameter is needed than in the SES method. But if there isn't any particular trend in data, the SES method is a more direct and simple model to use.

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

#The forecasts of hardcover sales were better than the ones of paperback sales because RMSE value is lower for the hardcover values. Its also worth noting that we had a difficult time forecasting the pattern in the paperback sales data using Holt's method.

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

```{r}

writeLines("95% PI of paperback sales calculated by holt function")

holt\_Paper$upper[1, "95%"]

holt\_Paper$lower[1, "95%"]

writeLines("95% PI of paperback sales calculated by formula")

holt\_Paper$mean[1] + 1.96\*new\_Paper

holt\_Paper$mean[1] - 1.96\*new\_Paper

writeLines("95% PI of hardcover sales calculated by holt function")

holt\_Hardcover$upper[1, "95%"]

holt\_Hardcover$lower[1, "95%"]

writeLines("95% PI of hardcover sales calculated by formula")

holt\_Hardcover$mean[1] + 1.96\*new\_Hardcover

holt\_Hardcover$mean[1] - 1.96\*new\_Hardcover

```

#The prediction interval for the first forecast for each series was almost same regardless of calculating method. For the SES case, the PI was different when it was calculated by SES function and formula respectively.

#7.10

```{r}

data(ukcars)

head(ukcars)

```

##7.10.a: Plot the data and describe the main features of the series

```{r}

autoplot(ukcars)

```

#### Answer:

The data shows trend and seasonality (quarterly)

##7.10.b: Decompose the series using STL and obtain the seasonally adjusted data.

```{r}

decomposed <- stl(ukcars, s.window="periodic", robust=TRUE)

seasonal <- decomposed$time.series[,1]

cars\_stl <- ukcars - seasonal

cars\_stl

autoplot(cars\_stl)

```

After decomposing the seasonally adjusted data shows smaller variability

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

```{r}

stlf\_decompose\_ukcars <- stlf(cars\_stl, etsmodel="AAN", damped=TRUE)

stlf\_decompose\_ukcars

```

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

```{r}

holt\_model <- holt(cars\_stl, etsmodel="AAN", damped=FALSE)

holt\_model

summary(holt\_model)

```

##7.10.e: Now use ets() to choose a seasonal model for the data.

```{r}

ets\_model <- ets(cars\_stl)

summary(ets\_model)

```

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

accuracy of the first (STL + ETS(A,Ad,N)) model (refer to question 7.c):

```{r}

accuracy(stlf\_decompose\_ukcars)

```

accuracy of the second model (refer to question 7.d):

```{r}

accuracy(holt\_model)

```

accuracy of the ETS model (refer to question 7.e):

```{r}

accuracy(ets\_model )

```

##### Answer:

Comparing the RMSE of the models obtained using STL decompositions (stlf\_decompose\_ukcars and holt\_model) with the RMSE of the ETS model (ets\_model), the best model was "stlf\_decompose\_ukcars" created in 7.c (STL + ETS(A,Ad,N))

##7.10.g: Compare the forecasts from the three approaches? Which seems most reasonable?

```{r}

autoplot(stlf\_decompose\_ukcars); autoplot(holt\_model) ; autoplot(forecast(ets\_model,h=8))

```

forecast of ETS model:

```{r}

forecast(ets\_model,h=8)

```

#### Answer:

The forecast of the STL + ETS(A,Ad,N) model seems reasonable.

##7.h: Check the residuals of your preferred model

```{r}

checkresiduals(stlf\_decompose\_ukcars)

```

The residuals seem to still have some autocorrelation.

---

title: "624\_HW5\_Group3"

output: html\_document

---

```{r}

suppressMessages(suppressWarnings(library(readxl)))

suppressMessages(suppressWarnings(library(ggplot2)))

suppressMessages(suppressWarnings(library(forecast)))

suppressMessages(suppressWarnings(library(fpp2)))

suppressMessages(suppressWarnings(library(mlbench)))

suppressMessages(suppressWarnings(library(caret)))

suppressMessages(suppressWarnings(library(caTools)))

suppressMessages(suppressWarnings(library(urca)))

```

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

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

Yes because because there is no correlation from one point to the next and all the values fall within the bounds of Â± 2/âˆšT.

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

It can be assumed that some observations can break the critical values while stil being white noise. The more numbers in a series of data the less fluctuation we are likely to observe in the variation.

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

```{r}

data(ibmclose)

tsdisplay(ibmclose)

```

As you will see from the scatterplot of IBM closing prices, there are clear trends in the data over periods of time. This suggests it is non-stationary, since stationary data would only show random fluctuations.

In the ACF plot, we again see a clear trend in the data, with large spikes in autocorrelations way above the bounds of the critical values within which you'd expect nearly all values if it were white noise.

In the PACF plot, we see a significant spike at lag of 1, well outside the critical values.

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

```{r}

data(wmurders)

```

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

```{r}

tsdisplay(wmurders)

wmurders %>% diff() %>% ur.kpss() %>% summary()

wmurders %>% diff(differences=2) %>% ur.kpss() %>% summary()

ndiffs(wmurders)

```

These queries suggest an ARIMA(0,2,2) model since the PACF decays exponentially over time and the ACF spikes at lag of 2 and the queries above suggest a d of 2.

##B. Should you include a constant in the model? Explain.

No, there is no drastic change in the average.

##C. Write this model in terms of the backshift operator.

(1 - B)^2 yt = (1 + theta1\*B + theta2\*B^2)\* et

##D. Fit the model using R and examine the residuals. Is the model satisfactory?

```{r}

fit <- Arima(wmurders, order = c(0,2,2))

plot(fit$residuals)

```

Yes, the model seems satisfactory. The residual are somewhat normally distributed for both the model we selected and all the values in the ACF plot fall within Â± 2/âˆšT.

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

```{r}

forecast <- forecast(fit, h=3)

forecast

```

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

```{r}

plot(forecast)

```

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

```{r}

fit\_g <- auto.arima(wmurders)

fit\_g

```

Using the auto.arima models gives us an ARIMA (1,2,1) model. The ACF actually returns better results than the model we selected. For example, the AIC value is slightly lower.

#8.12: For the mcopper data:

```{r}

data(mcopper)

plot(mcopper)

```

##A. if necessary, find a suitable Box-Cox transformation for the data;

By reviewing the scatterplot it appears there is not a huge change in variance for this data set, so it does not require a transformation. Below we show a Box-Cox transformation to see what it would look like.

```{r}

lamda <- BoxCox.lambda(mcopper)

mcopper\_bcx <- BoxCox(mcopper, lambda = lamda)

tsdisplay(mcopper\_bcx)

```

##B. fit a suitable ARIMA model to the transformed data using auto.arima();

```{r}

fit <- auto.arima(mcopper, trace = TRUE, ic ="aic", lambda = lamda)

fit

```

The auto.arima function returns an ARIMA (0,1,1) model.

##C. try some other plausible models by experimenting with the orders chosen;

```{r}

fit\_2 <- Arima(mcopper, order = c(2,1,2), lambda = lamda)

fit\_3 <- Arima(mcopper, order = c(0,1,1), lambda = lamda)

accuracy(fit)

accuracy(fit\_2)

accuracy(fit\_3)

```

Above we try a few different models. We select for final analysis a Arima(2,1,2) model - aka fit\_2.

##D. choose what you think is the best model and check the residual diagnostics;

Here we test the residuals for our best model fit\_2. The residuals are very normally distributed, and all the values fall within the critical value bounds on the ACF plot.

```{r}

plot(fit\_2$residuals)

```

##E. produce forecasts of your fitted model. Do the forecasts look reasonable?

```{r}

forecast <- forecast(fit\_2, h=10)

plot(forecast)

```

The forecast looks reasonable.

forecasts by manual calculation:

formula for ARIMA(0,2,2):

(1 - B)^2\*yt = (1 + theta1\*B + theta2\*B^2)\*et (re: chapter 8.8)

or yt - 2yt-1 + yt-2 = (1 + theta1\*B + theta2\*B^2)\*et (re: chapter 8.2)

or, yt = 2yt-1 - yt-2 + et+theta1\*et-1 + theta2\*et-2

```{r}

wmurder\_fit$model$theta

```

here, theta1 = -1.0181

theta2 = 0.1467

```{r}

e <- wmurder\_fit$residuals

T <- length(wmurders)

```

So forecast1 = 2yT - yT-1 + eT+1 + theta1\*eT + theta2\*eT-1 (replacing t with T+1)

Since we do not know the value of eT+1, it will be replaced with zero (re: chapter 8.2 )

By replacing values for theta1, theta2, e and T we get

```{r}

forecast1 <- 2\*wmurders[T] - wmurders[T-1] - 1.0181\*e[T] + 0.1467\*e[T-1]

forecast2 <- 2\*forecast1 - wmurders[T] + 0.1467\*e[T-1]

forecast3 <- 2\*forecast2 - forecast1

```

So the forecasts are

```{r}

c(forecast1,forecast2,forecast3)

```

##F. compare the results with what you would obtain using ets() (with no transformation).

```{r}

fit\_4 <- ets(mcopper); fit\_4

forecast2 <- forecast(fit\_4, h=10)

plot(forecast2)

```

Unlike our Arima models (both the model we selected and the Auto Arima model) which forecast values staying level, the ETS model forecasts falling values. While the upper bounds of each condidence interval appear similar for each model, the ETS model has lower bounds within the confidence intervals.