Applied Data Science Using R - INF 2167H - Assignment #2

Faria Khandaker

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Use RStudio for this assignment. Edit the file `assignment-02.Rmd` and insert your R code where wherever you see the string "INSERT YOUR ANSWER HERE"

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

Sample Question and Solution

Use `seq()` to create the vector (2,4,6, … ,20).

```
#Insert your code here.
seq(2,20,by = 2)

##  [1]  2  4  6  8 10 12 14 16 18 20

library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE)
knitr::opts_chunk$set(fig.pos = "!H")
library(tidyverse)
library(dplyr)
library(lmtest)
library(ggplot2)
library(rcompanion)
library(MASS)
library(corrplot)
```

Question 1 Visualizations and missing variables

1. Read the csv files "USDA_Macronutrients.csv" and "USDA_Micronutrients.csv" attached with the assignment.

```
micro<-read.csv(file.choose(), header = T)
macro<-read.csv(file.choose(), header = T)
```

2. Merge the two data frames using the variable "ID". Name the Merged Data Frame "USDA". You should get a dataframe of 7057 cases and 15 variables.

Hint: explore (merge) function

```
usda<-merge(micro,macro, by.micro=micro$ID)
```

3. Delete the commas in the Sodium and Potasium records and display the first 21 lines of these two columns after removing the commas. Assign Sodium and Potasium as numeric data types. Check the datatypes of the attributes.

```
usda$Sodium<-gsub(",","",usda$Sodium)
usda$Potassium<-gsub(",","",usda$Potassium)
head(usda$Sodium,21)

##  [1] "714"  "827"  "2"    "1395" "560"  "629"  "842"  "690"  "621"  "700"
## [11] "604"  "364"  "344"  "330"  "330"  "406"  "321"  "965"  "1116" "800"
## [21] "600"

head(usda$Potassium,21)

##  [1] "24"   "26"   "5"    "256"  "136"  "152"  "187"  "93"   "98"   "95"
## [11] "127"  "104"  "90"   "137"  "84"   "86"   "138"  "188"  "62"   "64"
## [21] "1409"

usda$Sodium<-as.numeric(usda$Sodium)
usda$Potassium<-as.numeric(usda$Potassium)
```

4. Are there missing values in USDA? If yes, how many?

There are 8827 missing values in the USDA Dataset.

```
sum(is.na(usda))

## [1] 8827
```

5. Remove records (rows) with missing values in more than 4 attributes (columns). Keep using the same name USDA. How many records remain in the new USDA data frame?

After removing the rows containing more than 4 missing values, the new dataset has 6949 missing values.

```
x=nrow(usda)
y<-0
for (i in 1:x){
  if (sum(is.na(usda[i,]))>4){
    usda<- usda[-c(i),]
  }
}
nrow(usda)

## [1] 6949
```

6. For records with missing values for Sugar, Vitamin E and Vitamin D, replace missing values with mean value for the respective variable.

```
msug<-round(mean(na.omit(usda$Sugar)),digits=2)
mvite<-round(mean(na.omit(usda$VitaminE)),digits=2)
mvitd<-round(mean(na.omit(usda$VitaminD)),digits=2)

usda$Sugar<-replace(usda$Sugar, is.na(usda$Sugar), msug)
```

```r
usda$VitaminE<-replace(usda$VitaminE, is.na(usda$VitaminE), mvite)

usda$VitaminD<-replace(usda$VitaminD, is.na(usda$VitaminD), mvitd)
```

7. With a single line of code, remove all remaining records with missing values. Name the new dataframe “USDAclean” and continue the rest of the exercises using this dataframe. How many records are in the USDAclean data frame?

The USDAclean dataset has 6310 records.

```r
USDAclean<-na.omit(usda)
nrow(USDAclean)

## [1] 6310
```

8. Which food has the highest sodium level in USDAclean?

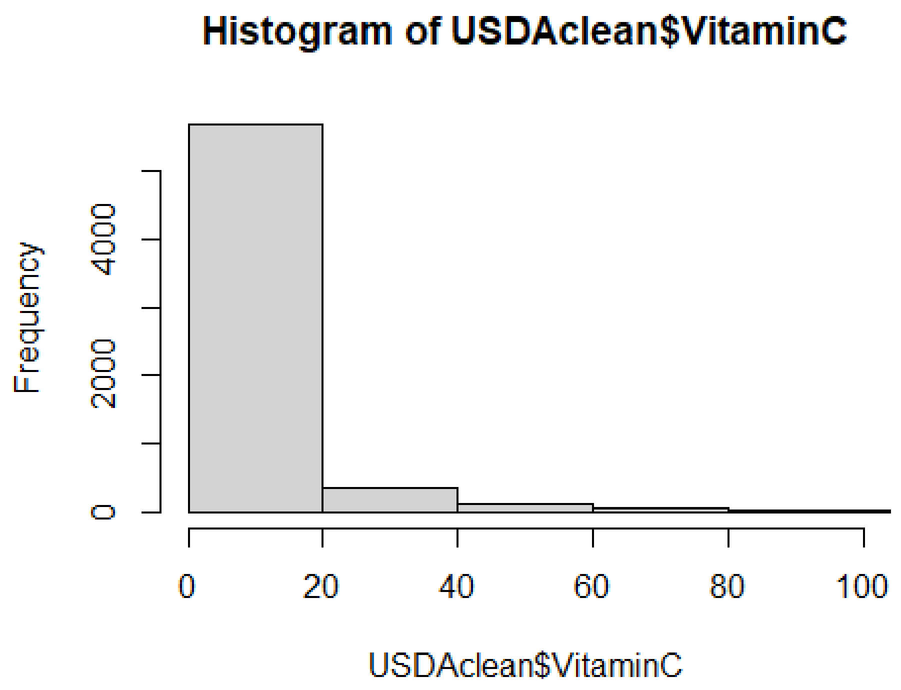
Table salt has the highest sodium level.

```r
USDAclean[which(USDAclean$Sodium==max(USDAclean$Sodium)),"Description"]

## [1] "SALT,TABLE"
```

9. Create a histogram of Vitamin C distribution in foods from USDAclean, with a limit of 0 to 100 on the x-axis and breaks of 100.

This is the histogram.

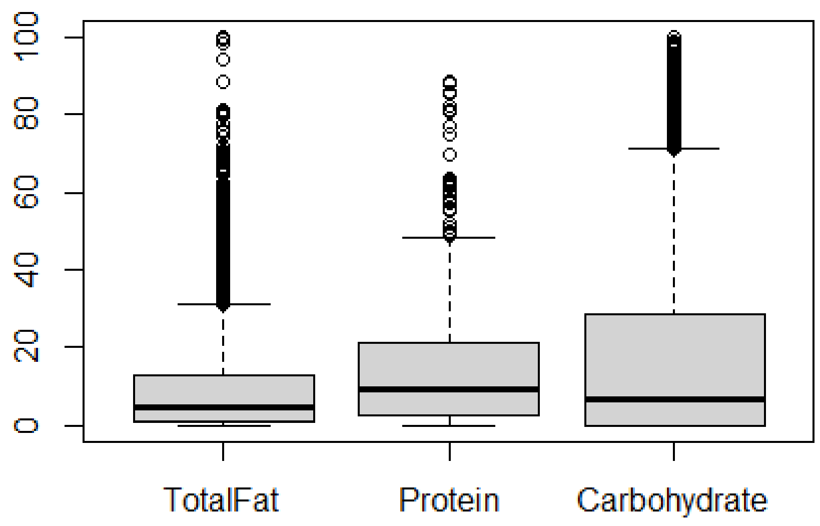
```r
hist(USDAclean$VitaminC, breaks = 100, xlim=range(0,100))
```



10. Create a boxplot to illustrate the distribution of values for TotalFat, Protein and Carbohydrate in USDAclean.

This is the boxplot

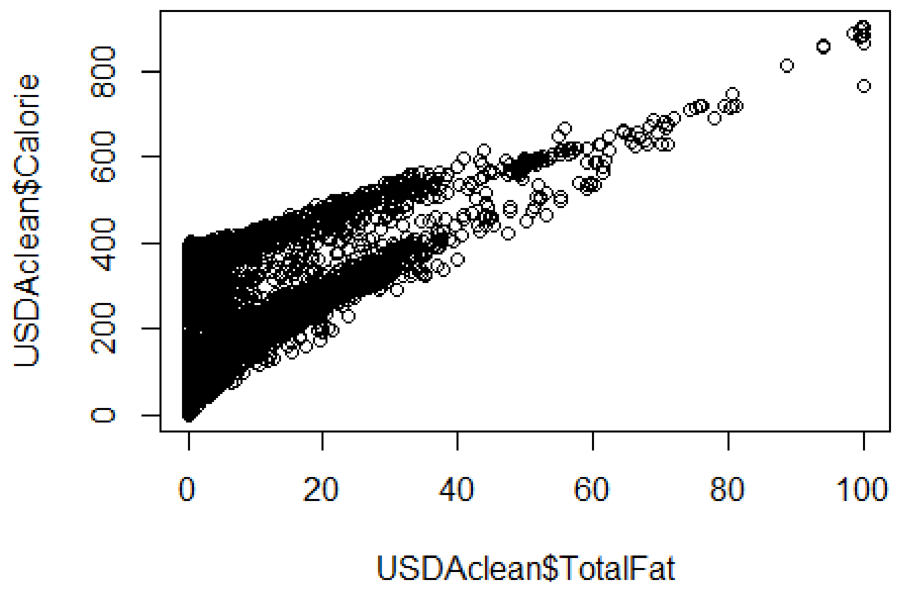
```r
boxplot(USDAclean$TotalFat, USDAclean$Protein,USDAclean$Carbohydrate, names =
c("TotalFat", "Protein", "Carbohydrate"), ylim=c(0,100))
```



11. Create a scatterplot to illustrate the relationship between a food's TotalFat content and its calorie content in USDAclean.

This is the scatterplot.

```
plot(x=USDAclean$TotalFat, y=USDAclean$Calorie)
```



Question 2 Skewness and transformations

Question 2 uses the dataset Carseats in library ISLR

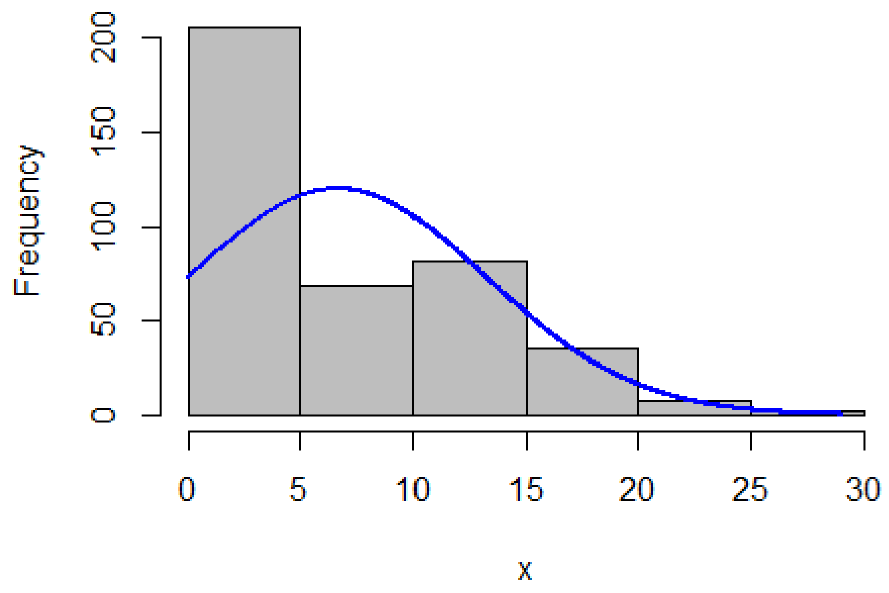
1. Install the ISLR package if it's not already installed and read the Carseats dataset

```
#install.packages("ISLR")
library(ISLR)
data("Carseats")
```

2. Draw a normal histogram of the Advertising column, is the Advertising distribution skewed? If yes, to which direction? Run an appropriate test to confirm your answer about skewness. Interpret the result of the test.

The distribution is right skewed

```
plotNormalHistogram(Carseats$Advertising)
```



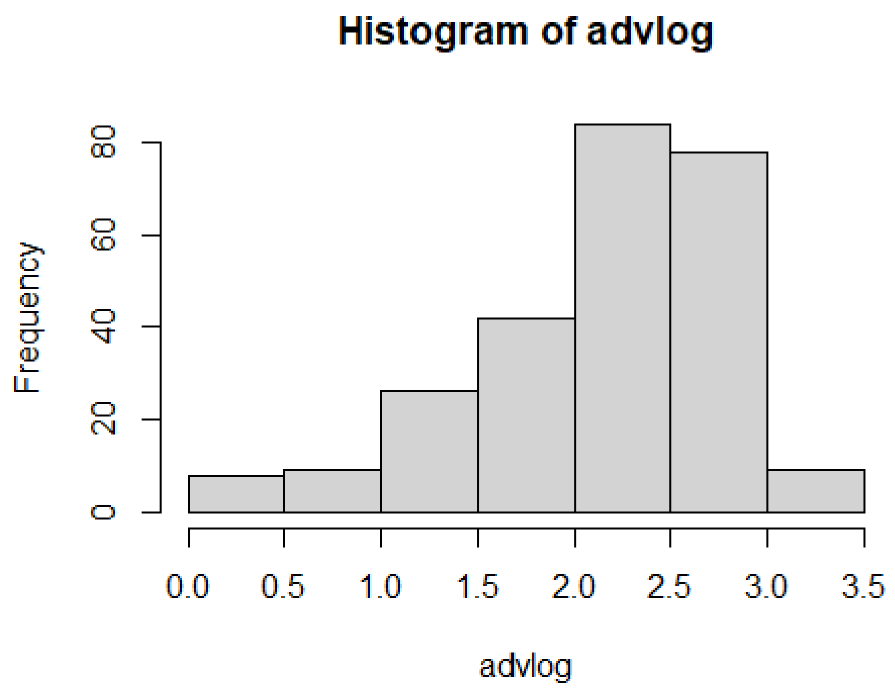
The Shapiro0-Wilkes test was run to test the normality assumption. The p-value is less than 0.05 which means the normality assumption is violated(null hypothesis is rejected). This confirms that the values in the Advertising column are skewed.

```
shapiro.test(Carseats$Advertising)

## 
##  Shapiro-Wilk normality test
## 
## data:  Carseats$Advertising
## W = 0.87354, p-value < 2.2e-16
```

3. If the Advertising variable in the previous question is skewed, find an adequate transformation to bring it close to normality. What is the best lambda you get? Based on the value of lambda, would transformation make a big difference?

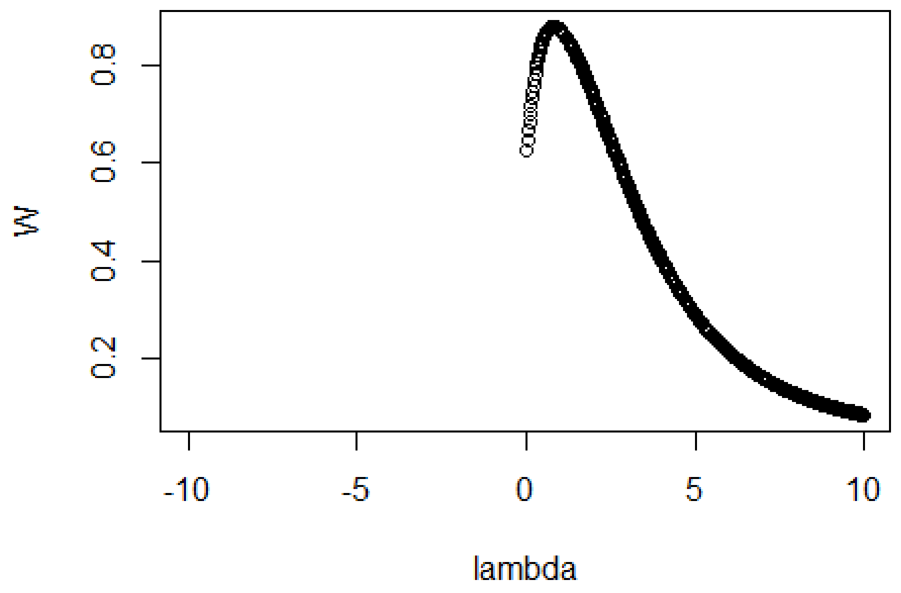
I tried to apply the log function to reduce skewness but it seemed to make the values become left skewed.

```
advlog <-log(Carseats$Advertising)
hist(advlog)
```

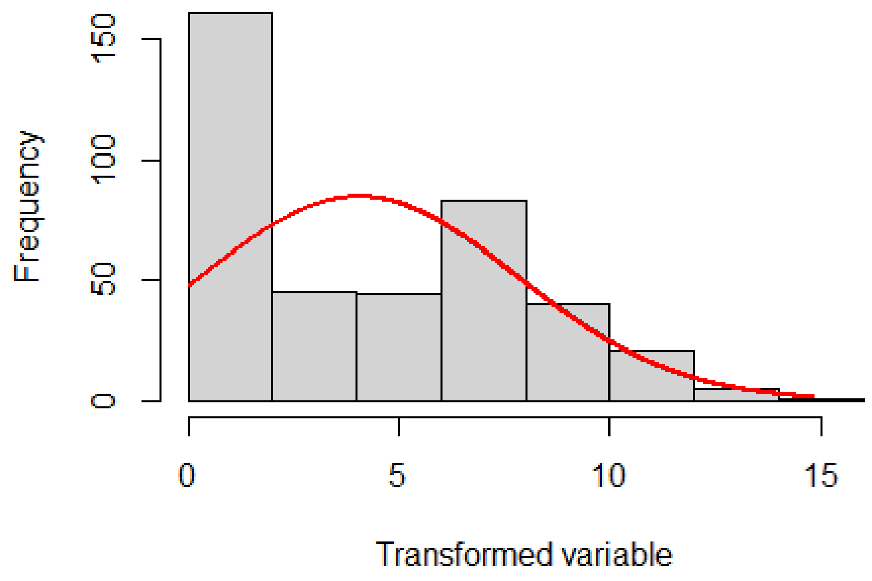


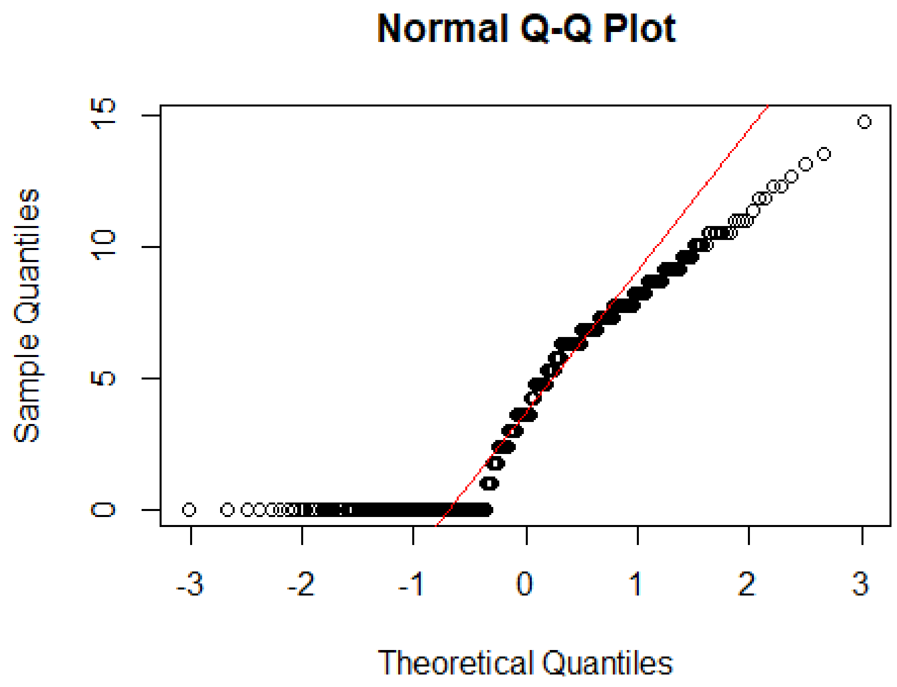
Then I moved on to trying Tukey transformation The value of lambda is 0.8 and the distribution is less skewed after its application. A lambda of 0.8 is very close to 1 and this indicates that performing any sort of transformation on this dataset will not help improve skewness by much.

```
advtuk <- transformTukey(Carseats$Advertising)
```



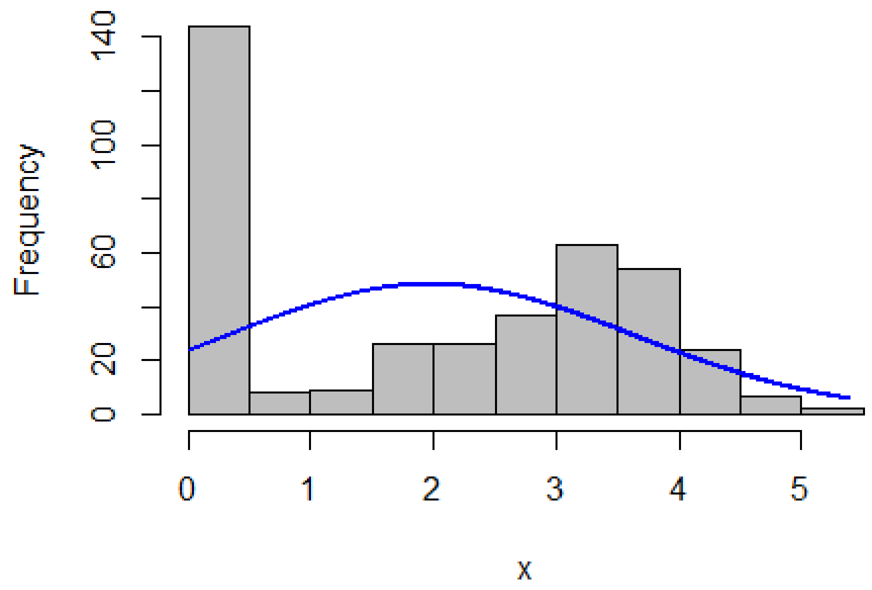
```
##
##     lambda      W Shapiro.p.value
## 433    0.8 0.8794       3.875e-17
##
## if (lambda >  0){TRANS = x ^ lambda}
## if (lambda == 0){TRANS = log(x)}
## if (lambda <  0){TRANS = -1 * x ^ lambda}
```



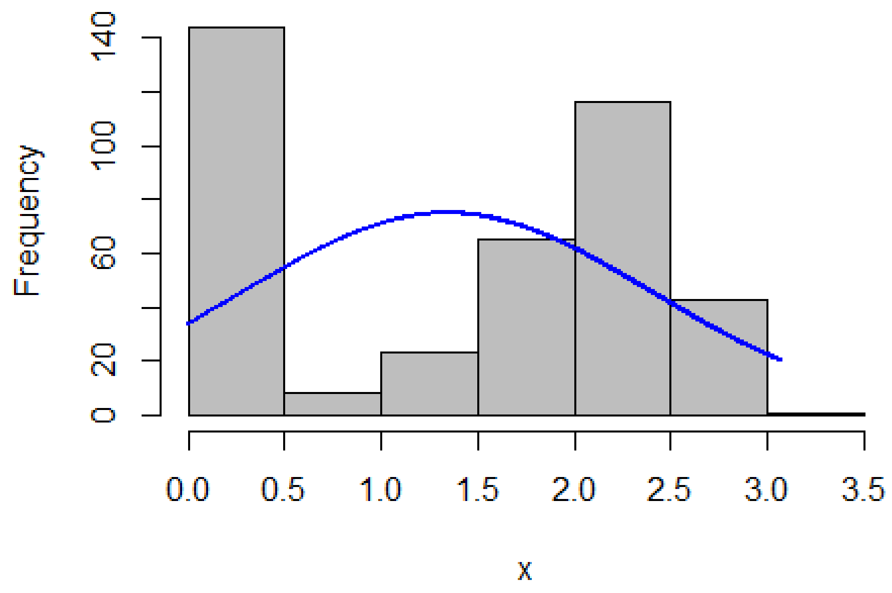


I performed the square-root function but this did not improve the skewness by much. Square-rooting had a better effect than cube root as the lambda for square-root is 0.5 and the lambda for cube-root is 0.3 and 0.5 is closer to the 0.8 lambda suggested by the Tukey transformation function.

```
#square-root
tvalues<-sqrt(Carseats$Advertising)
plotNormalHistogram(tvalues)
```



```
#cube-root
T_cub <- sign(Carseats$Advertising) * abs(Carseats$Advertising)^(1/3)
plotNormalHistogram(T_cub)
```



Question 3

The following question make use of data that is provided by the `mosaic` package. (install mosaic package and load KidsFeet using data(KidsFeet)).

```
#install.packages("mosaic")
library(mosaic)
data("KidsFeet")
```

1. Display the first 6 rows of the KidsFeet dataset.

```
head(KidsFeet)

##     name birthmonth birthyear length width sex biggerfoot domhand
## 1  David          5        88   24.4   8.4   B          L       R
## 2   Lars         10        87   25.4   8.8   B          L       L
## 3   Zach         12        87   24.5   9.7   B          R       R
## 4   Josh          1        88   25.2   9.8   B          L       R
## 5   Lang          2        88   25.1   8.9   B          L       R
## 6 Scotty          3        88   25.7   9.7   B          R       R
```

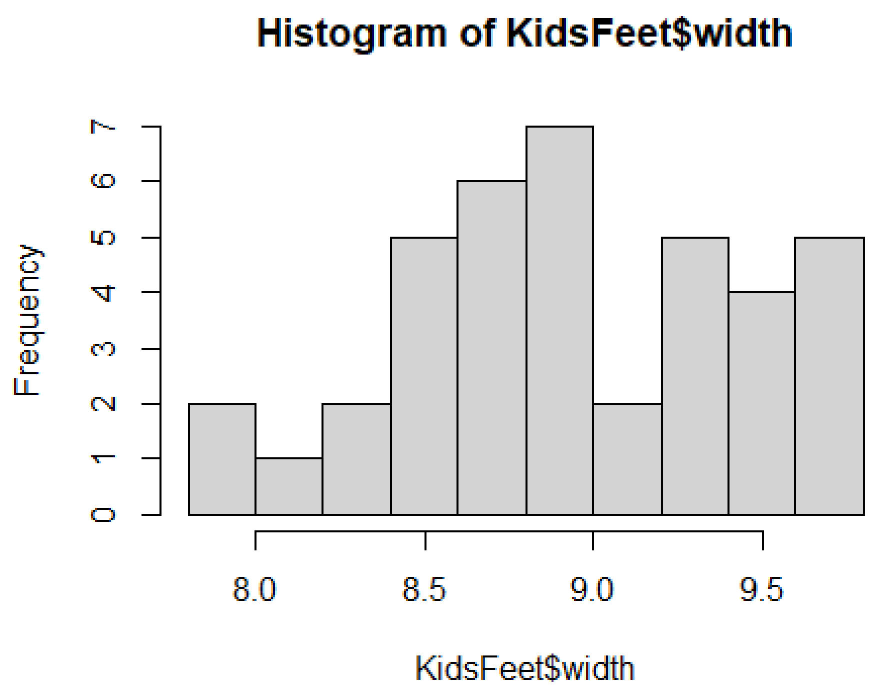
2. Display the type of each column of the KidsFeet dataset, use only one function in R to do so. Do not use (str)

```
sapply(KidsFeet, class)

##       name birthmonth  birthyear     length      width        sex 
biggerfoot 
##   "factor"  "integer"  "integer"  "numeric"  "numeric"   "factor"
```

```
"factor"
##    domhand
##   "factor"
```

3. Plot a histogram of the width variable.

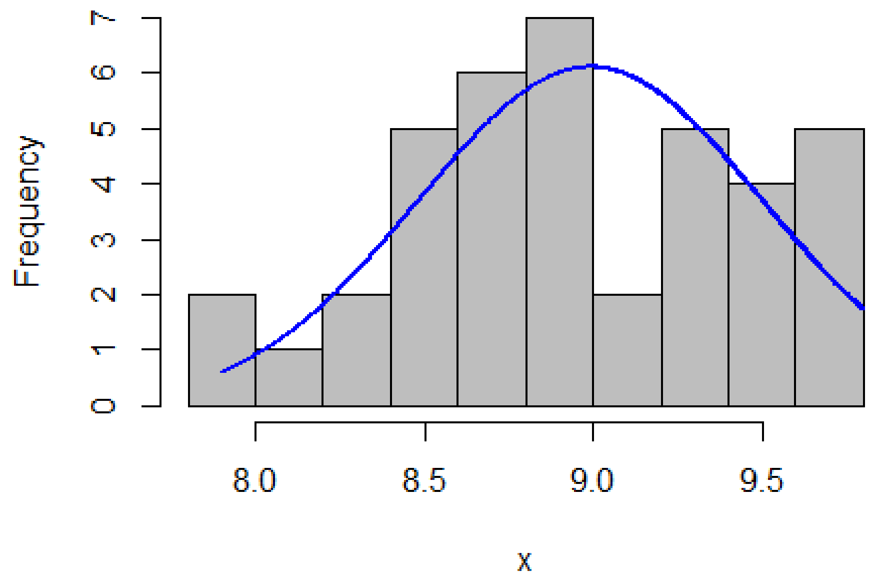
```
hist(KidsFeet$width)
```



4. Draw a normal histogram to show whether width is normally distributedor not. Use an appropriate test to check if width is normally distributed. Interpret your result.

The p-value of width, according to the Shapiro-Wilkes test, is 0.2811,which is higher than 0.05. This means we cannot reject the assumption of normality (null hypothesis is not rejected). This confirms that 'width' is normally distributed

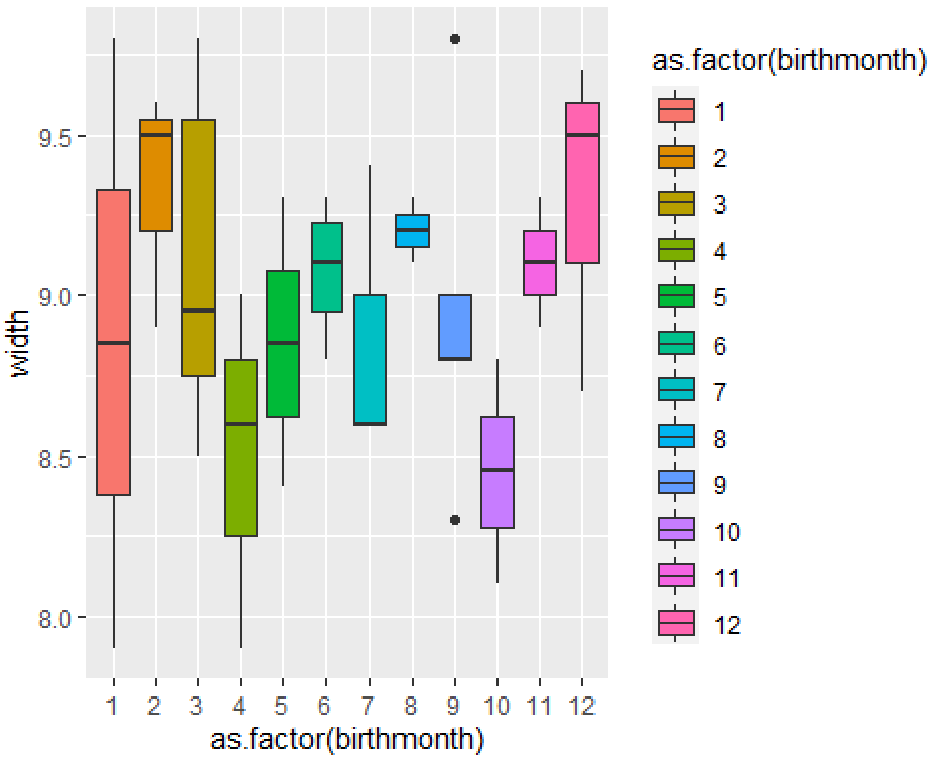
```
plotNormalHistogram(KidsFeet$width)
```



```
shapiro.test(KidsFeet$width)

##
##  Shapiro-Wilk normality test
##
## data:  KidsFeet$width
## W = 0.96601, p-value = 0.2811
```

5. Create a boxplot which shows the distribution of `width` in each `birth month`. Use different colors for each `birth month`.

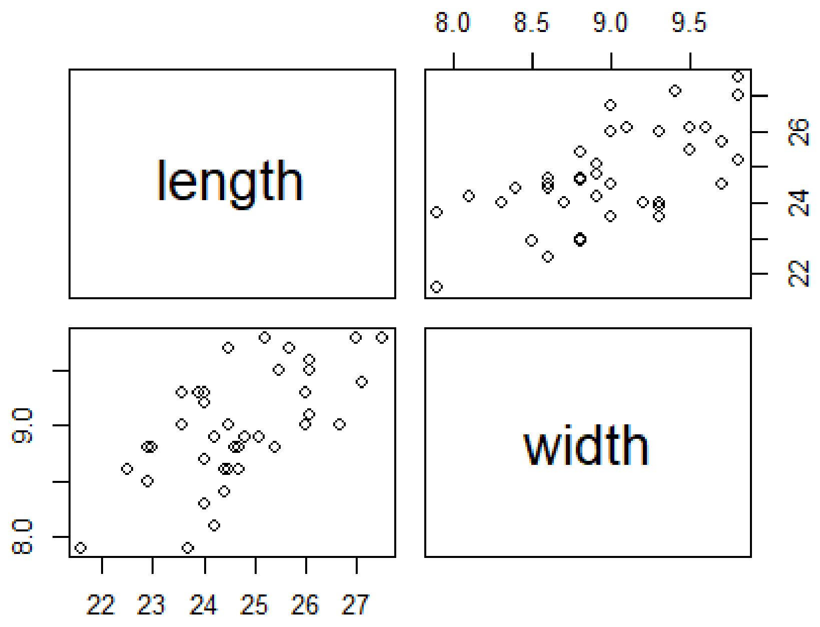
```
ggplot(KidsFeet,aes(x=as.factor(birthmonth),y=width,
fill=as.factor(birthmonth)))+ geom_boxplot()
```



6. Create one scatter plot matrix of the numeric variables (length, width) within the KidsFeet dataset.

(Hint investigate pairs())

```
pairs(~length+width,data=KidsFeet)
```



Question 4 Model Assumptions

This question makes use of package "datarium". Load marketing dataset from the package

```
#install.packages("datarium")
library(datarium)
data("marketing")
```

1. build a model to predict sales on the basis of advertising budget spent in youtube. Show the results of the model and interpret the coefficient of youtube (beta value and significance)

Investment in YouTube has significant impacts on sales. The p-value of YouTube is less than 0.05 which means it significantly impacts sales. The estimated coefficient(BETA) of youtube is 0.0475. The equation of the regression line becomes y=0.0475x+8.439. This means for every 1 unit increase in YouTube investment, sales goes up by 0.0475 units.

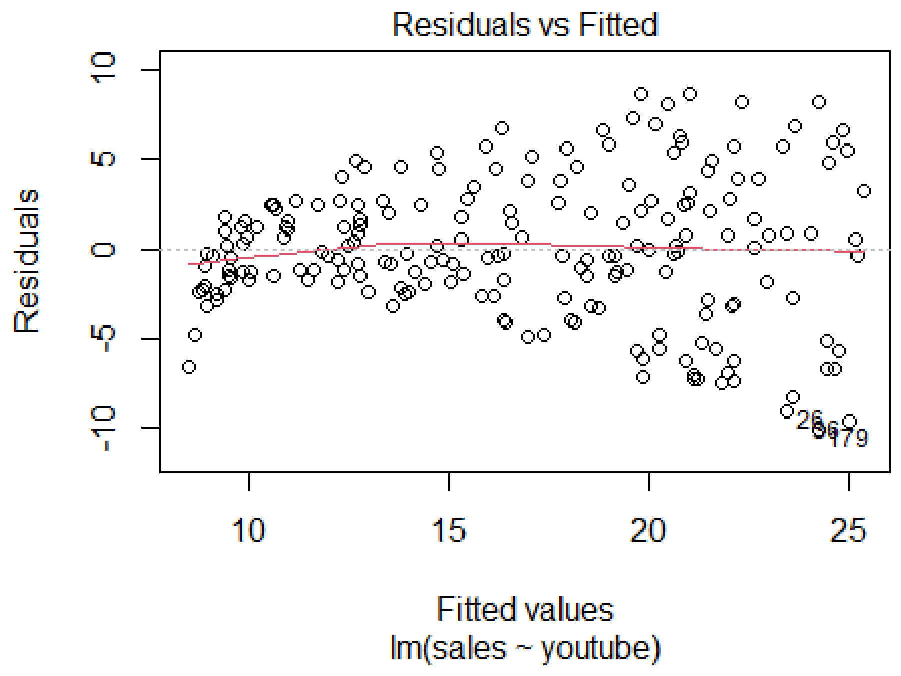
```
sales_model<- lm(sales~youtube, data=marketing)
summary(sales_model)

##
## Call:
## lm(formula = sales ~ youtube, data = marketing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.0632  -2.3454  -0.2295   2.4805   8.6548
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 8.439112   0.549412   15.36   <2e-16 ***
## youtube     0.047537   0.002691   17.67   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.91 on 198 degrees of freedom
## Multiple R-squared:  0.6119, Adjusted R-squared:  0.6099
## F-statistic: 312.1 on 1 and 198 DF,  p-value: < 2.2e-16
```

2. Plot the appropriate model plot to check for linearity, which plot tells about linearity? Interpret the appropriate plot

Looking at the residuals vs fitted plot, the linearity assumption appears to hold as the red trend line appears to closely follow the horizontal dotted line except in a few places. The red line does not have a strong curve and therefore we can assume that the linearity assumption holds.

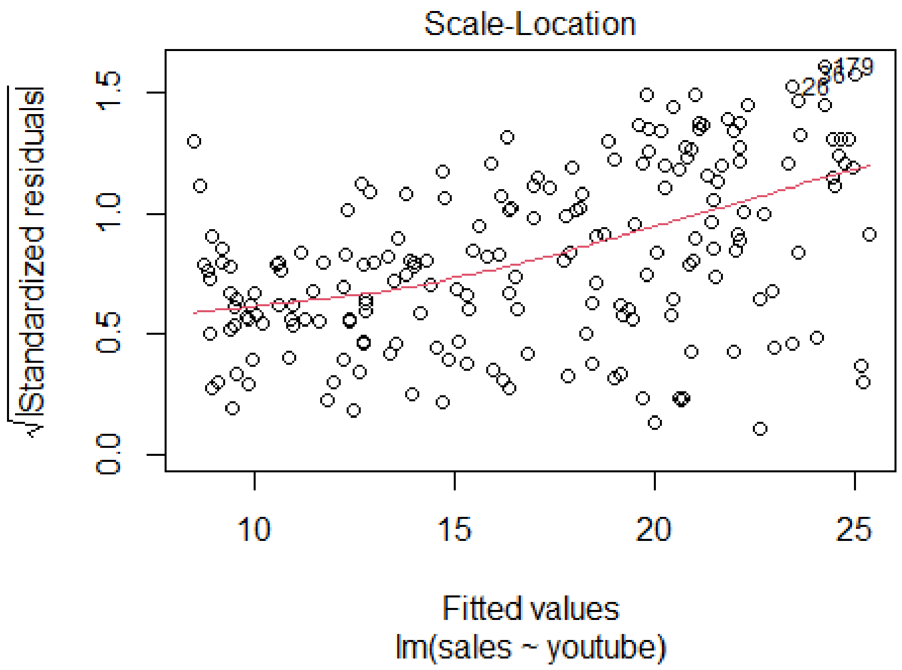
```
plot(sales_model, which=c(1))
```



3. Use two methods to check the assumption of constant variance (homoscedasticity) of the model. Interpret your results.

The first method used to check for homoscedasticity is the scale-location plot. The redline on the graph is not horizontal and follows an upward trend. This indicates non-constant variance.

```r
plot(sales_model, which=c(3))
```



The second method used to check for homoscedasticity is the Breusch-Pagan test. The p-value found was found to be less than 0.05. This mean the null hypothesis of constant variance is rejected. The variance in the model is non-constant.

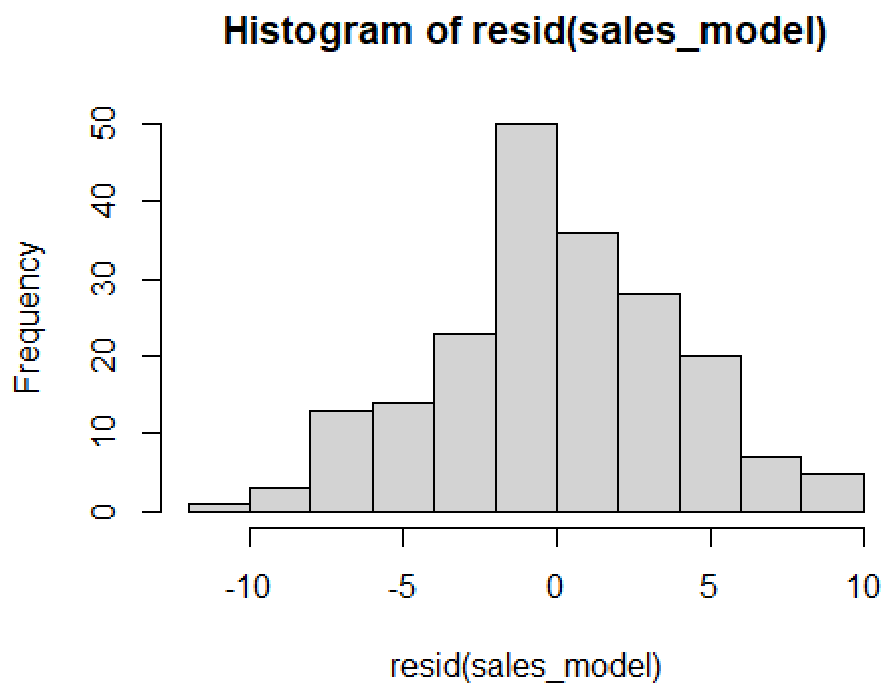
```r
bptest(sales_model)

##
##  studentized Breusch-Pagan test
##
## data:  sales_model
## BP = 48.038, df = 1, p-value = 4.18e-12
```

4. Use three methods to check the assumption of normality of the model, write one sentence after each method to write the name of the method and interpret its result.

There are three methods to check for normality: Histograms, Q-Q plots, and Shapiro-Wilkes Test.

Method 1: Histogram: the histogram displayed appears to have a normal distribution judging by how the bars are distributed around the center

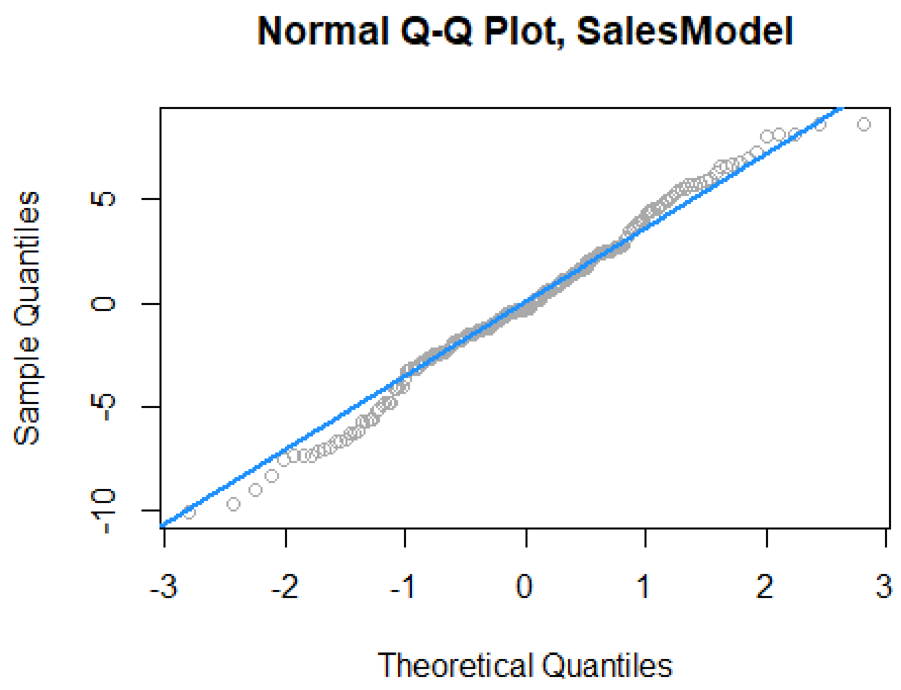
```r
hist(resid(sales_model))
```



Method 2: QQplot:

The Residuals that are plotted appear follow the line pretty closely. This indicates that the error are very close to a normal distribution

```
qqnorm(resid(sales_model), main = "Normal Q-Q Plot, SalesModel", col =
"darkgrey")
qqline(resid(sales_model), col = "dodgerblue", lwd = 2)
```



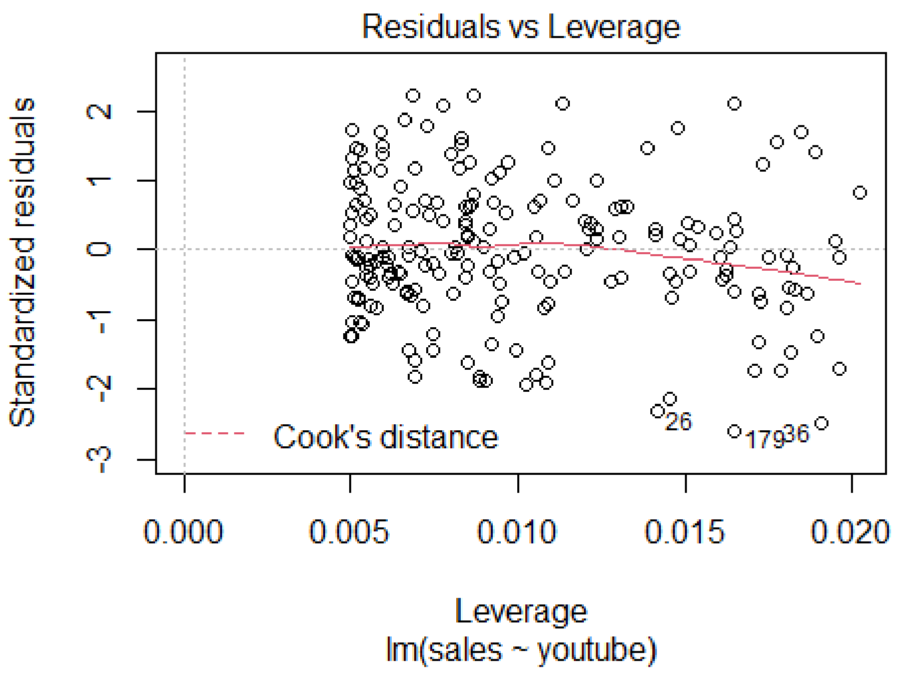
Method 3: Shapiro-Wilkes test: a p-value of 0.2133 is greater than 0.05. This means the null hypothesis is not rejected and so the errors are regarded as following a normal distribution

```
shapiro.test(resid(sales_model))

##
##  Shapiro-Wilk normality test
##
## data:  resid(sales_model)
## W = 0.99053, p-value = 0.2133
```

5. Are there any outliers or influential points in the model? Draw the plot that makes you decide? Interpret your answer

According to the residuals vs leverage plot, there are three points which can be considered outliers: 26, 179, and 36. These points are outliers because they lie close to 3 standard deviations to the left of the distribution (as indicated by the negative side). The points are not influential points as there aren't any dotted Cook's distance lines given to show which points are beyond that distance.

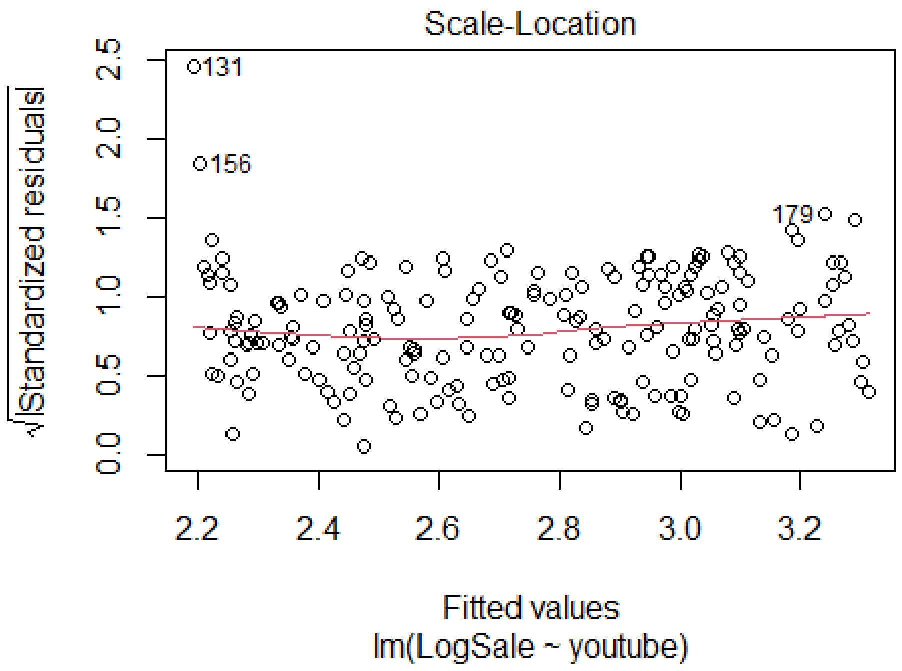
```
plot(sales_model,which=c(5))
```



6. Fix the homoscedasticity of the model through creating a new linear model using an appropriate transformation. Draw the homoscedasticity plot of the new model showing it is fixed

The red line in the Scale-Location Plot,which is used to show homoscedasticity, is much more straight after applying the log transformation. The errors follow a linear pattern.

```
loggedsales<- log(marketing$sales)
marketing<- marketing %>%mutate(LogSale=loggedsales)

sales_model2<-lm(LogSale~youtube, data=marketing)
plot(sales_model2, which=c(3))
```



Question 5 - Regression Analysis

This question makes use of package “plm”, load Crime dataset from this package

```
#install.packages("plm")
library(plm)
data(Crime)
```

1. Display the first 10 rows of crime and the structure of all the variables.

```
head(Crime, 10)

##     county year    crmrte    prbarr   prbconv   prbpris avgsen     polpc
density
## 1        1   81 0.0398849 0.289696 0.402062 0.472222   5.61 0.0017868
2.307159
## 2        1   82 0.0383449 0.338111 0.433005 0.506993   5.59 0.0017666
2.330254
## 3        1   83 0.0303048 0.330449 0.525703 0.479705   5.80 0.0018358
2.341801
## 4        1   84 0.0347259 0.362525 0.604706 0.520104   6.89 0.0018859
2.346420
## 5        1   85 0.0365730 0.325395 0.578723 0.497059   6.55 0.0019244
2.364896
## 6        1   86 0.0347524 0.326062 0.512324 0.439863   6.90 0.0018952
2.385681
## 7        1   87 0.0356036 0.298270 0.527596 0.436170   6.71 0.0018279
2.422633
```

```
## 8       3   81 0.0163921 0.202899 0.869048 0.465753   8.45 0.0005939
0.976834
## 9       3   82 0.0190651 0.162218 0.772152 0.377049   5.71 0.0007047
0.992278
## 10      3   83 0.0151492 0.181586 1.028170 0.438356   8.69 0.0006587
1.003861
##       taxpc  region smsa   pctmin     wcon      wtuc     wtrd     wfir
wser
## 1  25.69763 central   no 20.21870 206.4803  333.6209 182.3330 272.4492
215.7335
## 2  24.87425 central   no 20.21870 212.7542  369.2964 189.5414 300.8788
231.5767
## 3  26.45144 central   no 20.21870 219.7802 1394.8030 196.6395 309.9696
240.1568
## 4  26.84235 central   no 20.21870 223.4238  398.8604 200.5629 350.0863
252.4477
## 5  28.14034 central   no 20.21870 243.7562  358.7830 206.8827 383.0707
261.0861
## 6  29.74098 central   no 20.21870 257.9139  369.5465 218.5165 409.8842
269.6129
## 7  30.99368 central   no 20.21870 281.4259  408.7245 221.2701 453.1722
274.1775
## 8  14.56088 central   no  7.91632 188.7683  292.6422 151.4234 202.4292
191.3742
## 9  35.64073 central   no  7.91632 186.9658  345.7217 156.8826 225.0409
208.8190
## 10 19.26188 central   no  7.91632 193.5983  604.9115 157.1295 248.1390
219.0847
##      wmfg   wfed   wsta   wloc       mix   pctymle   lcrmrte   lprbarr
## 1  229.12 409.37 236.24 231.47 0.0999179 0.0876968 -3.221757 -1.238923
## 2  240.33 419.70 253.88 236.79 0.1030491 0.0863767 -3.261134 -1.084381
## 3  269.70 438.85 250.36 248.58 0.0806787 0.0850909 -3.496449 -1.107303
## 4  281.74 459.17 261.93 264.38 0.0785035 0.0838333 -3.360270 -1.014662
## 5  298.88 490.43 281.44 288.58 0.0932486 0.0823065 -3.308445 -1.122715
## 6  322.65 478.67 286.91 306.70 0.0973228 0.0800806 -3.359507 -1.120668
## 7  334.54 477.58 292.09 311.91 0.0801688 0.0778710 -3.335309 -1.209756
## 8  210.75 381.72 247.38 213.17 0.0561224 0.0870046 -4.110956 -1.595047
## 9  217.77 386.42 374.07 219.18 0.0473118 0.0864722 -3.959896 -1.818814
## 10 236.64 382.65 268.90 223.06 0.0596206 0.0859426 -4.189807 -1.706026
##      lprbconv   lprbpris  lavgsen    lpolpc   ldensity    lwcon    lwtuc
## 1  -0.9111490 -0.7503061 1.724551 -6.327340  0.8360171 5.330205 5.810005
## 2  -0.8370060 -0.6792581 1.720979 -6.338704  0.8459773 5.360137 5.911600
## 3  -0.6430188 -0.7345839 1.757858 -6.300291  0.8509204 5.392628 7.240509
## 4  -0.5030129 -0.6537265 1.930071 -6.273361  0.8528909 5.409070 5.988612
## 5  -0.5469313 -0.6990466 1.879465 -6.253162  0.8607340 5.496169 5.882718
## 6  -0.6687981 -0.8212920 1.931521 -6.268420  0.8694848 5.552626 5.912277
## 7  -0.6394244 -0.8297232 1.903599 -6.304609  0.8848549 5.639869 6.013041
## 8  -0.1403569 -0.7640998 2.134166 -7.428766 -0.0234386 5.240520 5.678950
## 9  -0.2585739 -0.9753801 1.742219 -7.257781 -0.0077520 5.230926 5.845634
## 10  0.0277805 -0.8247239 2.162173 -7.325303  0.0038535 5.265786 6.405082
##       lwtrd    lwfir    lwser    lwmfg    lwfed    lwsta    lwloc
lpctymle
## 1  5.205835 5.607452 5.374044 5.434246 6.014619 5.464848 5.444450 -
2.433870
## 2  5.244607 5.706707 5.444911 5.482013 6.039540 5.536862 5.467174 -
2.449038
## 3  5.281372 5.736475 5.481292 5.597310 6.084157 5.522900 5.515765 -
2.464036
## 4  5.301128 5.858180 5.531204 5.640985 6.129421 5.568077 5.577387 -
2.478925
## 5  5.332152 5.948220 5.564850 5.700042 6.195282 5.639919 5.664972 -
2.497306
## 6  5.386862 6.015875 5.596987 5.776568 6.171011 5.659169 5.725870 -
2.524721
## 7  5.399384 6.116272 5.613776 5.812757 6.168732 5.677062 5.742715 -
2.552702
## 8  5.020080 5.310390 5.254230 5.350673 5.944687 5.510926 5.362090 -
2.441794
## 9  5.055498 5.416282 5.341468 5.383440 5.956925 5.924443 5.389893 -
2.447933
## 10 5.057070 5.513989 5.389459 5.466540 5.947121 5.594339 5.407441 -
2.454076
##     lpctmin   ltaxpc      lmix
## 1  3.006608 3.246399 -2.303407
## 2  3.006608 3.213833 -2.272549
## 3  3.006608 3.275311 -2.517281
## 4  3.006608 3.289981 -2.544612
## 5  3.006608 3.337204 -2.372487
## 6  3.006608 3.392526 -2.329722
## 7  3.006608 3.433783 -2.523621
## 8  2.068926 2.678338 -2.880219
## 9  2.068926 3.573489 -3.050995
## 10 2.068926 2.958128 -2.819754
```

```
str(Crime)
```

```
## 'data.frame':    630 obs. of  44 variables:
##  $ county  : int  1 1 1 1 1 1 1 3 3 3 ...
##  $ year    : int  81 82 83 84 85 86 87 81 82 83 ...
##  $ crmrte  : num  0.0399 0.0383 0.0303 0.0347 0.0366 ...
##  $ prbarr  : num  0.29 0.338 0.33 0.363 0.325 ...
##  $ prbconv : num  0.402 0.433 0.526 0.605 0.579 ...
##  $ prbpris : num  0.472 0.507 0.48 0.52 0.497 ...
##  $ avgsen  : num  5.61 5.59 5.8 6.89 6.55 6.9 6.71 8.45 5.71 8.69 ...
##  $ polpc   : num  0.00179 0.00177 0.00184 0.00189 0.00192 ...
##  $ density : num  2.31 2.33 2.34 2.35 2.36 ...
##  $ taxpc   : num  25.7 24.9 26.5 26.8 28.1 ...
##  $ region  : Factor w/ 3 levels "other","west",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ smsa    : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ pctmin  : num  20.2 20.2 20.2 20.2 20.2 ...
##  $ wcon    : num  206 213 220 223 244 ...
##  $ wtuc    : num  334 369 1395 399 359 ...
##  $ wtrd    : num  182 190 197 201 207 ...
##  $ wfir    : num  272 301 310 350 383 ...
##  $ wser    : num  216 232 240 252 261 ...
##  $ wmfg    : num  229 240 270 282 299 ...
##  $ wfed    : num  409 420 439 459 490 ...
##  $ wsta    : num  236 254 250 262 281 ...
##  $ wloc    : num  231 237 249 264 289 ...
##  $ mix     : num  0.0999 0.103 0.0807 0.0785 0.0932 ...
##  $ pctymle : num  0.0877 0.0864 0.0851 0.0838 0.0823 ...
##  $ lcrmrte : num  -3.22 -3.26 -3.5 -3.36 -3.31 ...
##  $ lprbarr : num  -1.24 -1.08 -1.11 -1.01 -1.12 ...
##  $ lprbconv: num  -0.911 -0.837 -0.643 -0.503 -0.547 ...
##  $ lprbpris: num  -0.75 -0.679 -0.735 -0.654 -0.699 ...
##  $ lavgsen : num  1.72 1.72 1.76 1.93 1.88 ...
##  $ lpolpc  : num  -6.33 -6.34 -6.3 -6.27 -6.25 ...
##  $ ldensity: num  0.836 0.846 0.851 0.853 0.861 ...
##  $ lwcon   : num  5.33 5.36 5.39 5.41 5.5 ...
##  $ lwtuc   : num  5.81 5.91 7.24 5.99 5.88 ...
##  $ lwtrd   : num  5.21 5.24 5.28 5.3 5.33 ...
##  $ lwfir   : num  5.61 5.71 5.74 5.86 5.95 ...
##  $ lwser   : num  5.37 5.44 5.48 5.53 5.56 ...
##  $ lwmfg   : num  5.43 5.48 5.6 5.64 5.7 ...
##  $ lwfed   : num  6.01 6.04 6.08 6.13 6.2 ...
##  $ lwsta   : num  5.46 5.54 5.52 5.57 5.64 ...
##  $ lwloc   : num  5.44 5.47 5.52 5.58 5.66 ...
##  $ lpctymle: num  -2.43 -2.45 -2.46 -2.48 -2.5 ...
##  $ lpctmin : num  3.01 3.01 3.01 3.01 3.01 ...
##  $ ltaxpc  : num  3.25 3.21 3.28 3.29 3.34 ...
##  $ lmix    : num  -2.3 -2.27 -2.52 -2.54 -2.37 ...
```

2. Calculate the mean,variance and standard deviation of tax revenue per capita (taxpc) by omitting the missing values, if any.

The mean, variance and standard deviation of the 'tax revenue per capita' variable are 30.24, 131.21, and 11.46 respectively.

```
taxmean<-mean(na.omit(Crime$taxpc))
taxvariance<-var(na.omit(Crime$taxpc))
taxsd<-sd(na.omit(Crime$taxpc))
taxmean

## [1] 30.23919

taxvariance

## [1] 131.21

taxsd
```

```
## [1] 11.4547
```

3. Use `density` and `smsa` to predict tax per capita and build a multivariate linear regression model, display a summary of your model indicating Residuals, Coefficients..etc. Interpret adjusted R squared, the intercept, coefficients and significance of the predictors in your model?

The Adjusted R squared is 0.063. This means only 6 percent of the variance can be explained by the variables 'density' and 'smsa' and 94% of the variation can be explained by other variables not considered by the model.

The intercept for this model is 29.5615 which means if all other variables are at zero, the tax revenue per capita would be 29.5615 units.

The coefficient for 'density' is -0.2345. This means that keeping all other variables constant, tax per capita decreases by 0.2345 for every one unit increase in density.

The coefficient for 'smsayes'(smsa is a categorical variable with values yes or no filling the columns) is 11.2808. Compared to the reference category(smsano), smsayes impacts tax per capita by 11.2808 units.

The variable of 'density' is not considered statistically significant as the p-value is greater than 0.05. The p-value for 'smsayes' is less than 0.05 but smsa is a categorical variable and the significance for it can only be determined by performing a chi-square test.

```
taxmodel<-lm(taxpc~density+smsa, data=Crime)
summary(taxmodel)

## 
## Call:
## lm(formula = taxpc ~ density + smsa, data = Crime)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -16.960  -6.693  -2.083   3.173  90.320 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  29.5615     0.7134  41.436  < 2e-16 ***
## density      -0.2345     0.5329  -0.440     0.66    
## smsayes      11.2808     2.6939   4.188 3.22e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 11.09 on 627 degrees of freedom
## Multiple R-squared:  0.06603,    Adjusted R-squared:  0.06305 
## F-statistic: 22.16 on 2 and 627 DF,  p-value: 5.011e-10
```

4. Based on the output of your model, write the equations based on the intercept and factors of `smsa` when `density` is set to 2.4, and compare the result with `predict()` function. Hint: Explore `predict()` function

Hint: there will be two equations, one for smsa (no) and one for smsa(yes)

Hint: create a new dataframe on which to run the prediction having density of 2.4 and smsa a vector of no and yes

The results of the predict function are very closely in line with the results of running the equations separately.

```
#Eqn for smsayes(1)
smyes <- -0.2345*(2.4)+11.2808*(1)+29.5615
#Eqn for smsano(0)
smno <- -0.2345*(2.4)+11.2808*(0)+29.5615
new.case<-data.frame(density=c(2.4),smsa=c("yes","no"))
taxpred<-predict(taxmodel,new.case)
smyes

## [1] 40.2795

smno

## [1] 28.9987

taxpred

##        1        2
## 40.27948 28.99864
```

5. Which independent variable has the strongest positive predictive power in the model? (Hint: Look at the coefitients calculated for each independent variable)

The independent variable ‘crmrte’, which stands for “crimes committed per person” has the strongest positive predictive power in the model

```
full <- lm(taxpc~.,data=Crime)
stepB <- stepAIC(full, direction= "backward", trace=FALSE)
summary(stepB)

##
## Call:
## lm(formula = taxpc ~ year + crmrte + prbconv + polpc + region +
##     pctmin + wfir + wloc + pctyml e + ldensity + lwsta + lwloc +
##     lpctymle + lpctmin + ltaxpc, data = Crime)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.744 -1.561 -0.499  0.952 31.936
##
## Coefficients:
```

```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.350e+01  3.679e+01   0.367 0.713806
## year          -6.402e-01  1.325e-01  -4.831 1.72e-06 ***
## crmrte         5.888e+01  1.138e+01   5.175 3.09e-07 ***
## prbconv        1.557e-01  8.186e-02   1.901 0.057712 .
## polpc         -2.031e+02  5.643e+01  -3.600 0.000344 ***
## regionwest    -1.973e+00  5.317e-01  -3.711 0.000225 ***
## regioncentral -7.667e-01  3.352e-01  -2.287 0.022526 *
## pctmin        -4.230e-02  1.956e-02  -2.163 0.030943 *
## wfir           5.132e-03  3.186e-03   1.611 0.107760
## wloc           6.371e-02  3.054e-02   2.086 0.037421 *
## pctymle        5.691e+01  2.312e+01   2.461 0.014119 *
## ldensity      -1.378e+00  2.881e-01  -4.782 2.17e-06 ***
## lwsta         -2.134e+00  1.022e+00  -2.088 0.037234 *
## lwloc         -1.447e+01  7.875e+00  -1.838 0.066579 .
## lpctymle      -7.244e+00  2.933e+00  -2.469 0.013804 *
## lpctmin       -8.796e-01  3.896e-01  -2.258 0.024315 *
## ltaxpc         3.725e+01  4.806e-01  77.507  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.942 on 613 degrees of freedom
## Multiple R-squared:  0.9357, Adjusted R-squared:  0.934
## F-statistic: 557.8 on 16 and 613 DF,  p-value: < 2.2e-16
```

6. Find Pearson correlation between tax per capita and density. Interpret the result with a sentence.

The variables ‘density’ and ‘taxpc’ have a correlation coefficient of 0.1998. A strong correlation relationship does not exist between these two variables. The population density of an area is not strongly correlated to the tax revenue per capita.

```
numdat<- Crime[,c(9,10)]
correlationMatrix <- cor(numdat, method = "pearson")
correlationMatrix
```

```
##           density     taxpc
## density 1.0000000 0.1997634
## taxpc   0.1997634 1.0000000
```

7. Write the correlation matrix of the variables: avgsen, polpc, density, taxpc. Hint: Explore the variables using ?Crime. Comment on the result with a sentence.

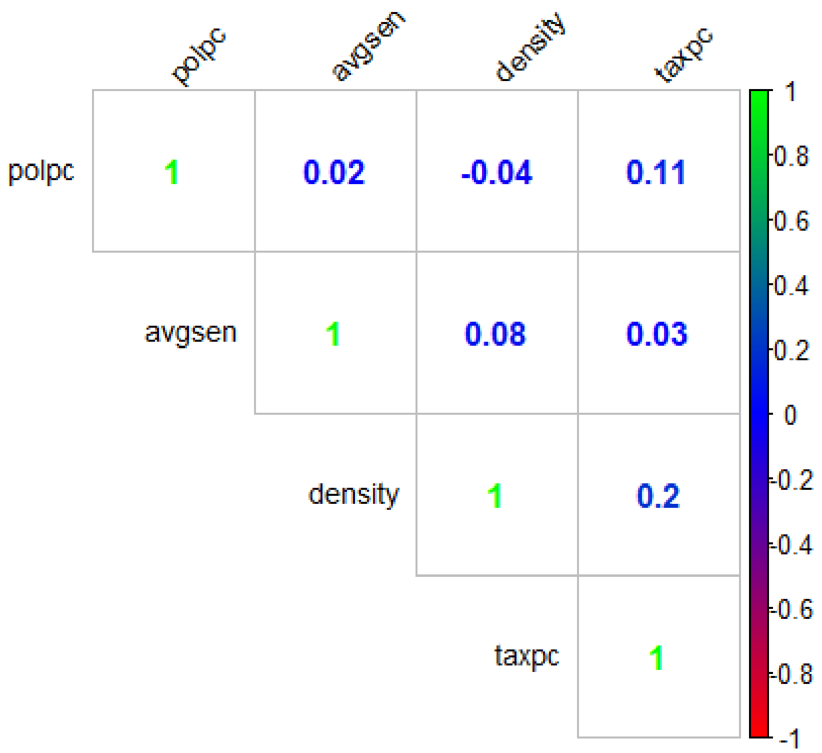
There aren’t any variables that are strongly correlated with each other. Most correlations are below 10 percent. the correlations that show the strongest relationships are tax revenue per capital to police per capita with a correlation coefficient of 0.1083 and tax revenue per capital to density which is 0.1998. These are the highest coefficients but the relationship is considered weak.

```
numdat2<- Crime[,c(7,8,9,10)]
correlationMatrix2 <- cor(numdat2, method = "pearson")
correlationMatrix2

##             avgsen       polpc     density      taxpc
## avgsen  1.00000000  0.01712970  0.07807510 0.02818939
## polpc   0.01712970  1.00000000 -0.03969574 0.10828664
## density 0.07807510 -0.03969574  1.00000000 0.19976339
## taxpc   0.02818939  0.10828664  0.19976339 1.00000000
```

8. Draw the correlation plot of the variables: avgsen, polpc, density, taxpc.

This is the Correlation Plot

```
pal <- colorRampPalette(c("red", "blue", "green"))

corrplot(correlationMatrix2, method="number", col=pal(200),
         type="upper", order="hclust",
         tl.col="black", tl.srt=45, tl.cex= 0.9, diag=TRUE)
```



Question 6 - Visualization

1. Generate normally distributed random numbers for three categories: A (n = 200, mean = 100, sd = 20), B (n = 200, mean = 120, sd = 20), and C (n = 200, mean = 80, sd = 20)

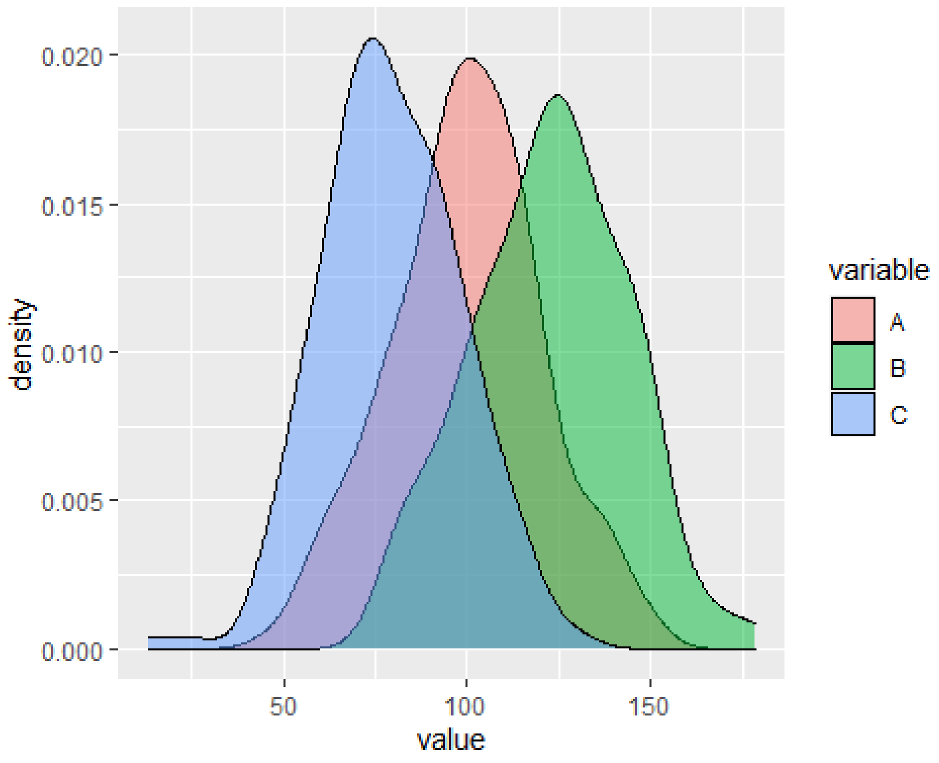
Hint: explore rnorm

```r
A <- rnorm(n = 200, mean = 100, sd = 20)
B <- rnorm(n = 200, mean = 120, sd = 20)
C<- rnorm(n = 200, mean = 80, sd = 20)
```

2. Combine all the three categories in one dataframe so that A is the first column of the dataframe, B is the second column, and C is the third column. Then generate a density plot of the data colored by category.

hint: explore the function (melt) from the library reshape2 on the created dataframe then use geom_density in ggplot2

```r
library(reshape2)

## Warning: package 'reshape2' was built under R version 4.0.3

##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths

randomdataset<-data.frame(A,B,C)
randomdataset2<-melt(randomdataset)

## No id variables; using all as measure variables

ggplot(randomdataset2,aes(x=value, fill=variable))+geom_density(alpha=0.5)
```



END of Assignment #2.