Introduction to Big Data and Analytics

Group 4 - Project 2

The Dataset

Physical measurements of abalone, a type of sea snail, are contained in the Abalone dataset. There are 4177 instances in the dataset, each with eight attributes and one target variable.

The characteristics are as follows:

- Sex is a categorical variable with three values: "M" for a male, "F" for a female, and "I" for an infant.
- Length: a continuous variable that represents the abalone's length in millimeters.
- Diameter: a continuous variable that represents the abalone's diameter in millimeters.
- Height: a continuous variable that represents the abalone's height in millimeters.
- Whole weight: a continuous variable that represents the weight in grams of the entire abalone.
- Shucked weight: a continuous variable that represents the weight of abalone meat in grams.
- Viscera weight: a continuous variable representing the weight of an abalone's viscera (internal organs) in grams.
- Shell weight: a continuous variable that represents the weight in grams of the abalone shell.

The desired variable is:

- Rings: an integer variable representing the number of rings on the shell of an abalone. The number of rings on the abalone can be used to estimate its age.
- The abalone.names file contains more information about the dataset, such as the data collection process, missing values, and potential analysis challenges.

The objective of the project

The project aims to estimate the age of each abalone specimen using the number of rings as a proxy. We want to create a model that can accurately predict the number of rings (and thus the age) of a new abalone specimen based on its physical characteristics.

First, we start by observing the whole dataset:

```
> # read the data set file and convert it to data frame
> abalone <- read.table("C:/Masters/gwu/big data/abalone.data", sep = ",")
>
> # checking the column names
> colnames(abalone)
[1] "V1" "V2" "V3" "V4" "V5" "V6" "V7" "V8" "V9"
>
> # checking number of rows and columns in the dataframe
> dim(abalone)
[1] 4177    9
>
> # displaying all the columns and 10 rows from the dataset
> abalone[1:10, ]
   V1    V2    V3    V4     V5     V6     V7    V8 V9
1   M 0.455 0.365 0.095 0.5140 0.2245 0.1010 0.150 15
2   M 0.350 0.265 0.090 0.2255 0.0995 0.0485 0.070  7
3   F 0.530 0.420 0.135 0.6770 0.2565 0.1415 0.210  9
4   M 0.440 0.365 0.125 0.5160 0.2155 0.1140 0.155 10
5   I 0.330 0.255 0.080 0.2050 0.0895 0.0395 0.055  7
6   I 0.425 0.300 0.095 0.3515 0.1410 0.0775 0.120  8
7   F 0.530 0.415 0.150 0.7775 0.2370 0.1415 0.330 20
8   F 0.545 0.425 0.125 0.7680 0.2940 0.1495 0.260 16
9   M 0.475 0.370 0.125 0.5095 0.2165 0.1125 0.165  9
10  F 0.550 0.440 0.150 0.8945 0.3145 0.1510 0.320 19
>
```

Assigning the column names:

Note "rings" is added to the end of the list of column names, since it represents the target variable (the age of the abalone is based on the number of rings).

Also by running str(), the structure of the dataset is visible. Only the variable “sex” is a character.

```
> # assigning the column names
> abalone.names = c("sex", "length", "diameter", "height", "whole_weight", "shucked_weight", "viscera_weight", "shell_weight", "rings")
>
> # checking the column name assignment
> abalone.names
[1] "sex"            "length"         "diameter"       "height"         "whole_weight"   "shucked_weight" "viscera_weight"
[8] "shell_weight"   "rings"
>
> # using str to view the structure of the abalone object
> str(abalone)
'data.frame':   4177 obs. of  9 variables:
 $ V1: chr  "M" "M" "F" "M" ...
 $ V2: num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 0.475 0.55 ...
 $ V3: num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 0.37 0.44 ...
 $ V4: num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125 0.125 0.15 ...
 $ V5: num  0.514 0.226 0.677 0.516 0.205 ...
 $ V6: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
 $ V7: num  0.101 0.0485 0.1415 0.114 0.0395 ...
 $ V8: num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 0.165 0.32 ...
 $ V9: int  15 7 9 10 7 8 20 16 9 19 ...
>
```

```
> # associating column names with the columns in the dataset
> names(abalone) <- abalone.names
> abalone[1:10, ]
   sex length diameter height whole_weight shucked_weight viscera_weight shell_weight rings
1    M  0.455    0.365  0.095       0.5140         0.2245         0.1010        0.150    15
2    M  0.350    0.265  0.090       0.2255         0.0995         0.0485        0.070     7
3    F  0.530    0.420  0.135       0.6770         0.2565         0.1415        0.210     9
4    M  0.440    0.365  0.125       0.5160         0.2155         0.1140        0.155    10
5    I  0.330    0.255  0.080       0.2050         0.0895         0.0395        0.055     7
6    I  0.425    0.300  0.095       0.3515         0.1410         0.0775        0.120     8
7    F  0.530    0.415  0.150       0.7775         0.2370         0.1415        0.330    20
8    F  0.545    0.425  0.125       0.7680         0.2940         0.1495        0.260    16
9    M  0.475    0.370  0.125       0.5095         0.2165         0.1125        0.165     9
10   F  0.550    0.440  0.150       0.8945         0.3145         0.1510        0.320    19
>
```

The “sex” variable has chr values which we will transform into numeric values for future analysis of the dataset. The “rings” variable represents the number of rings in the shell of the abalone, and since it's a count variable, it's appropriate to keep it as an integer. Value conversion:

```
> # converting the chr value of column sex to numeric values
> abalone$sex <- as.numeric(factor(abalone$sex))
> abalone[1:10, ]
   sex length diameter height whole_weight shucked_weight viscera_weight shell_weight rings
1    3  0.455    0.365  0.095       0.5140         0.2245         0.1010        0.150    15
2    3  0.350    0.265  0.090       0.2255         0.0995         0.0485        0.070     7
3    1  0.530    0.420  0.135       0.6770         0.2565         0.1415        0.210     9
4    3  0.440    0.365  0.125       0.5160         0.2155         0.1140        0.155    10
5    2  0.330    0.255  0.080       0.2050         0.0895         0.0395        0.055     7
6    2  0.425    0.300  0.095       0.3515         0.1410         0.0775        0.120     8
7    1  0.530    0.415  0.150       0.7775         0.2370         0.1415        0.330    20
8    1  0.545    0.425  0.125       0.7680         0.2940         0.1495        0.260    16
9    3  0.475    0.370  0.125       0.5095         0.2165         0.1125        0.165     9
10   1  0.550    0.440  0.150       0.8945         0.3145         0.1510        0.320    19
>
```

```
> str(abalone)
'data.frame':   4177 obs. of  9 variables:
 $ sex           : num  3 3 1 3 2 2 1 1 3 1 ...
 $ length        : num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 0.475 0.55 ...
 $ diameter      : num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 0.37 0.44 ...
 $ height        : num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125 0.125 0.15 ...
 $ whole_weight  : num  0.514 0.226 0.677 0.516 0.205 ...
 $ shucked_weight: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
 $ viscera_weight: num  0.101 0.0485 0.1415 0.114 0.0395 ...
 $ shell_weight  : num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 0.165 0.32 ...
 $ rings         : int  15 7 9 10 7 8 20 16 9 19 ...
>
>
```

Also, there are no NA values in the dataset, so no need to remove such rows from the dataset:

```
> # there are no more chr values in the dataset so we don't need to convert anything
> # summary of the dataset
> summary(abalone)
      sex           length         diameter          height        whole_weight    shucked_weight   viscera_weight
 Min.   :1.000   Min.   :0.075   Min.   :0.0550   Min.   :0.0000   Min.   :0.0020   Min.   :0.0010   Min.   :0.0005
 1st Qu.:1.000   1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150   1st Qu.:0.4415   1st Qu.:0.1860   1st Qu.:0.0935
 Median :2.000   Median :0.545   Median :0.4250   Median :0.1400   Median :0.7995   Median :0.3360   Median :0.1710
 Mean   :2.053   Mean   :0.524   Mean   :0.4079   Mean   :0.1395   Mean   :0.8287   Mean   :0.3594   Mean   :0.1806
 3rd Qu.:3.000   3rd Qu.:0.615   3rd Qu.:0.4800   3rd Qu.:0.1650   3rd Qu.:1.1530   3rd Qu.:0.5020   3rd Qu.:0.2530
 Max.   :3.000   Max.   :0.815   Max.   :0.6500   Max.   :1.1300   Max.   :2.8255   Max.   :1.4880   Max.   :0.7600
  shell_weight        rings
 Min.   :0.0015   Min.   : 1.000
 1st Qu.:0.1300   1st Qu.: 8.000
 Median :0.2340   Median : 9.000
 Mean   :0.2388   Mean   : 9.934
 3rd Qu.:0.3290   3rd Qu.:11.000
 Max.   :1.0050   Max.   :29.000
>
> # checking if there are any NA values in the dataset
> sum(is.na(abalone))
[1] 0
>
```

Describing the statistics of the dataset using “describe” function of the psych package:

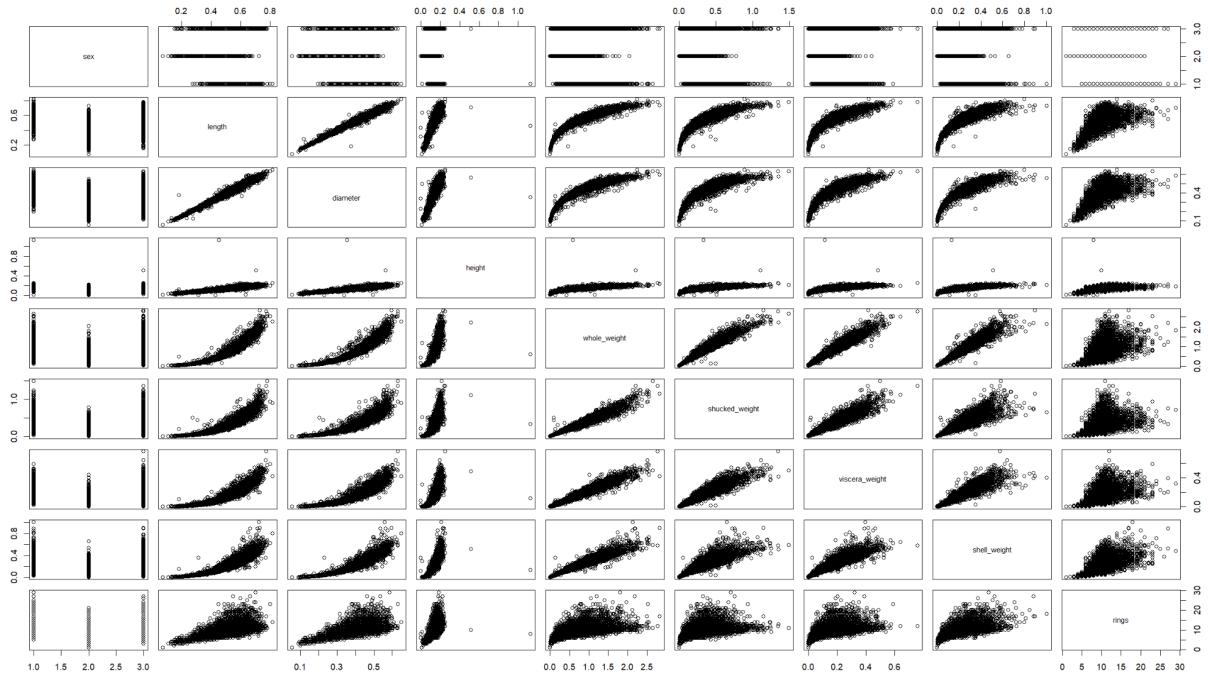
```
> # describing the statistics of the dataset
> library(psych)
> describe(abalone)
               vars    n mean   sd median trimmed  mad  min   max range  skew kurtosis   se
sex               1 4177 2.05 0.82   2.00    2.07 1.48 1.00  3.00  2.00 -0.10    -1.51 0.01
length            2 4177 0.52 0.12   0.54    0.53 0.12 0.07  0.81  0.74 -0.64     0.06 0.00
diameter          3 4177 0.41 0.10   0.42    0.41 0.10 0.06  0.65  0.60 -0.61    -0.05 0.00
height            4 4177 0.14 0.04   0.14    0.14 0.04 0.00  1.13  1.13  3.13    75.90 0.00
whole_weight      5 4177 0.83 0.49   0.80    0.80 0.53 0.00  2.83  2.82  0.53    -0.03 0.01
shucked_weight    6 4177 0.36 0.22   0.34    0.34 0.23 0.00  1.49  1.49  0.72     0.59 0.00
viscera_weight    7 4177 0.18 0.11   0.17    0.17 0.12 0.00  0.76  0.76  0.59     0.08 0.00
shell_weight      8 4177 0.24 0.14   0.23    0.23 0.15 0.00  1.00  1.00  0.62     0.53 0.00
rings             9 4177 9.93 3.22   9.00    9.64 2.97 1.00 29.00 28.00  1.11     2.32 0.05
>
```

To create pairwise scatterplots for the seven variables in the data frame, the pairs() function is used.

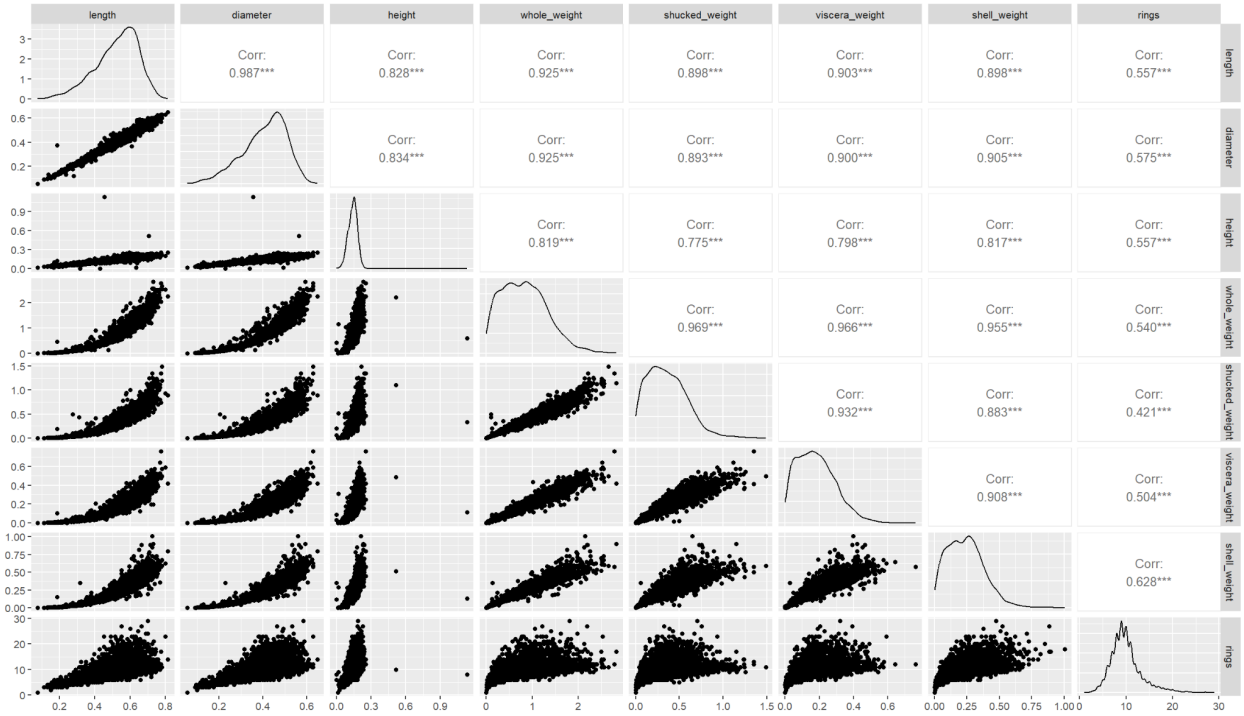
The "sex" column of the abalone dataset can be removed because it is a categorical variable that is irrelevant for predicting the specimen's age. Plus it's also seen from the pairwise plot that "sex" is not related to any other variable:

```
> # plotting the variables in a pairwise manner
> pairs(abalone)
>
> # "sex" column can be stripped off as it is a categorical variable that is not relevant for predicting the age of the specimen
> # it's also seen from pairwise plot that "sex" is not related with any other variable
> abalone.df <- abalone[, -1]
> abalone.df[1:10, ]
   length diameter height whole_weight shucked_weight viscera_weight shell_weight rings
1   0.455    0.365  0.095       0.5140         0.2245         0.1010        0.150    15
2   0.350    0.265  0.090       0.2255         0.0995         0.0485        0.070     7
3   0.530    0.420  0.135       0.6770         0.2565         0.1415        0.210     9
4   0.440    0.365  0.125       0.5160         0.2155         0.1140        0.155    10
5   0.330    0.255  0.080       0.2050         0.0895         0.0395        0.055     7
6   0.425    0.300  0.095       0.3515         0.1410         0.0775        0.120     8
7   0.530    0.415  0.150       0.7775         0.2370         0.1415        0.330    20
8   0.545    0.425  0.125       0.7680         0.2940         0.1495        0.260    16
9   0.475    0.370  0.125       0.5095         0.2165         0.1125        0.165     9
10  0.550    0.440  0.150       0.8945         0.3145         0.1510        0.320    19
>
> # checking the structure again
> str(abalone.df)
'data.frame':   4177 obs. of  8 variables:
 $ length        : num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 0.475 0.55 ...
 $ diameter      : num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 0.37 0.44 ...
 $ height        : num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125 0.125 0.15 ...
 $ whole_weight  : num  0.514 0.226 0.677 0.516 0.205 ...
 $ shucked_weight: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
 $ viscera_weight: num  0.101 0.0485 0.1415 0.114 0.0395 ...
 $ shell_weight  : num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 0.165 0.32 ...
 $ rings         : int  15 7 9 10 7 8 20 16 9 19 ...
>
```

Pairwise Plot:



To plot the data using several plotting functions, the ggplot2 and GGally packages can be used. We can use the ggpairs function from GGally to plot the pairwise relationships between the variables. This will create a matrix of pairwise scatterplots, histograms, and density plots. Here's the output:



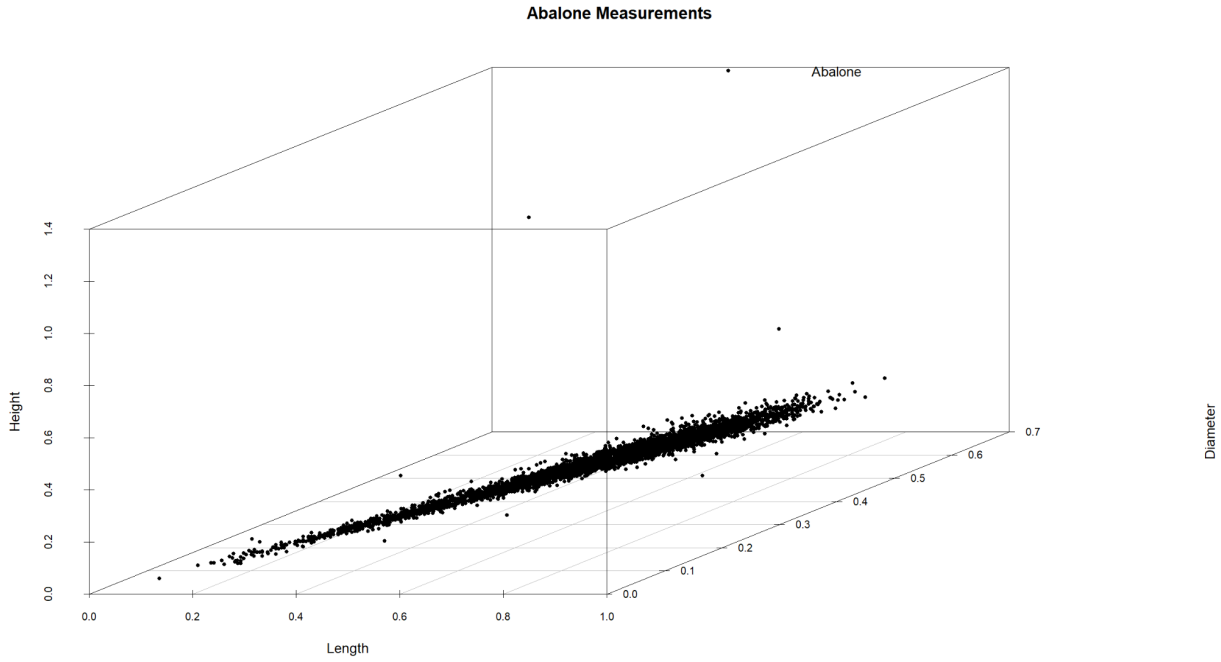
The plot shows that the following pairs of variables appear to be correlated:

- Length and diameter: These two variables have a strong linear relationship.
- Length and height: these two variables have a moderate linear relationship.
- Diameter and height: these two variables have a weak linear relationship.
- All other variables and shucked weight: There is a positive correlation between shucked weight and all other variables.
- Visceral weight and all other variables: Visceral weight and all other variables have a positive correlation.
- All other variables and shell weight: There is a positive correlation between shell weight and all other variables.

Overall, we can conclude that most of the variables are positively correlated with each other, although the strength of the correlation varies.

3D Plot using scatterplot3d:

```
> # Creating a new 3D plot with length on the x-axis, diameter on the y-axis, and height on the z-axis
> s3d <- scatterplot3d(abalone.df[, c("length", "diameter", "height")], pch = 20, main = "Abalone Measurements", xlab = "Length", ylab =
"Diameter", zlab = "Height")
> # Add a legend to the plot
> legend("topright", legend = c("Abalone"), pch = 20, col = "black", bty = "n")
>
```



Normalization:

The normalize function takes a vector of numerical values x and returns a normalized version of the input in which the minimum and maximum values of x are mapped to 0 and 1, respectively.

The code applies the normalize() function to each column of the abalone.df data frame using the lapply() function. The normalize() function takes a vector x as input, subtracts the vector's minimum value from each element, and then divides the result by the vector's range (i.e., the difference between the maximum and minimum values of the vector). As a result, each vector element is scaled to fall within the range of 0 to 1.

The resulting abalone.norm data frame has the same columns as abalone.df, but each column has been scaled to be between 0 and 1, making it easier to compare the different variables in the clustering analysis.

Finally, the code uses the min and max functions to extract the minimum and maximum values of each column in the original abalone.df data frame and stores them in separate variables. If necessary, these values are used in the formula x = norm(x) * (max(x)-min(x)) + min(x) to return the normalized values to the original scale.

The final few lines of code show how to convert the normalized value of 0.5135135 for the length column back to the original scale of 0.455.

```
> # normalization
> normalize <- function(x) {((x-min(x)) / (max(x)-min(x)))}
> normalize
function(x) {((x-min(x)) / (max(x)-min(x)))}
>
> # apply normalization on the data
> abalone.norm <- as.data.frame(lapply(abalone.df, normalize))
> abalone.norm[1:10, ]
      length  diameter     height whole_weight shucked_weight viscera_weight shell_weight     rings
1  0.5135135 0.5210084 0.08407080   0.18133522     0.15030262     0.13232390   0.14798206 0.5000000
2  0.3716216 0.3529412 0.07964602   0.07915707     0.06624075     0.06319947   0.06826109 0.2142857
3  0.6148649 0.6134454 0.11946903   0.23906499     0.17182246     0.18564845   0.20777280 0.2857143
4  0.4932432 0.5210084 0.11061947   0.18204356     0.14425017     0.14944042   0.15296462 0.3214286
5  0.3445946 0.3361345 0.07079646   0.07189658     0.05951580     0.05134957   0.05331340 0.2142857
6  0.4729730 0.4117647 0.08407080   0.12378254     0.09414929     0.10138249   0.11808670 0.2500000
7  0.6148649 0.6050420 0.13274336   0.27465911     0.15870881     0.18564845   0.32735426 0.6785714
8  0.6351351 0.6218487 0.11061947   0.27129449     0.19704102     0.19618170   0.25759841 0.5357143
9  0.5405405 0.5294118 0.11061947   0.17974146     0.14492266     0.14746544   0.16292975 0.2857143
10 0.6418919 0.6470588 0.13274336   0.31609704     0.21082717     0.19815668   0.31738914 0.6428571
>
```

```
> # finding the smallest value in length column
> abalone.df.length<- min(abalone.df$length)
> abalone.df.length
[1] 0.075
> # finding the smallest value in shucked_weight column
> abalone.df.shucked_weight<- min(abalone.df$shucked_weight)
> abalone.df.shucked_weight
[1] 0.001
> # finding the smallest value in viscera_weight
> abalone.df.viscera_weight<- min(abalone.df$viscera_weight)
> abalone.df.viscera_weight
[1] 5e-04
> # finding the largest value in length column
> abalone.df.max.length<- max(abalone.df$length)
> abalone.df.max.length
[1] 0.815
> # finding the largest value in shucked_weight column
> abalone.df.max.shucked_weight<- max(abalone.df$shucked_weight)
> abalone.df.max.shucked_weight
[1] 1.488
> # finding the largest value in viscera_weight
> abalone.df.max.viscera_weight<- max(abalone.df$viscera_weight)
> abalone.df.max.viscera_weight
[1] 0.76
> # to get back to the original values, later:
> # x = norm(x) * (max(x)-min(x) ) + min(x)
> # it is working correctly
> length<- 0.5135135*(abalone.df.max.length-abalone.df.length)+ abalone.df.length
> length
[1] 0.455
>
```

Z-Score Normalization:

Zscore is a function that takes a numeric vector as input and returns the vector's z-score normalized values. The formula for z-score normalization is (x - mean(x)) / sd(x), where x is the input vector, mean(x) is the input vector's mean, and sd(x) is the input vector's standard deviation.

The scale function is then applied to the columns of the abalone.df data frame using the lapply function, which applies the scale function to each column individually. The z-score scale function normalizes each column of the data frame so that the mean of each column is zero and the standard deviation is one. The resulting normalized data frame is stored in abalone.znorm, and the first 10 rows are displayed using the [] indexing syntax.

```
> # Z-Score Normalization
> zscore<- function(x){(x-mean(x))/sd(x)}
> zscore(c(10,20,30,40,50))
[1] -1.2649111 -0.6324555  0.0000000  0.6324555  1.2649111
>
> abalone.znorm<- as.data.frame(lapply(abalone.df, scale))
> abalone.znorm[1:10,]
       length     diameter     height whole_weight shucked_weight viscera_weight shell_weight       rings
1  -0.5744894 -0.43209706 -1.0642967   -0.6418214     -0.6076126     -0.7261246   -0.6381405  1.57135544
2  -1.4488124 -1.43975662 -1.1838366   -1.2301298     -1.1707697     -1.2050770   -1.2128421 -0.90990405
3   0.0500271  0.12211570 -0.1079779   -0.3094322     -0.4634444     -0.3566471   -0.2071143 -0.28958918
4  -0.6993926 -0.43209706 -0.3470576   -0.6377430     -0.6481599     -0.6075269   -0.6022216  0.02056826
5  -1.6153501 -1.54052258 -1.4229163   -1.2719334     -1.2158222     -1.2871831   -1.3205987 -0.90990405
6  -0.8242959 -1.08707578 -1.0642967   -0.9731910     -0.9838015     -0.9405128   -0.8536536 -0.59974661
7   0.0500271  0.07173272  0.2506416   -0.1044929     -0.5512969     -0.3566471    0.6549382  3.12214262
8   0.1749304  0.17249868 -0.3470576   -0.1238653     -0.2944973     -0.2836639    0.1520742  1.88151287
9  -0.4079516 -0.38171408 -0.3470576   -0.6509978     -0.6436547     -0.6212113   -0.5303839 -0.28958918
10  0.2165648  0.32364761  0.2506416    0.1340932     -0.2021395     -0.2699796    0.5831005  2.81198518
> .
```

Correlation Plot:

The Correlation is plotted by using the "corrplot" package. First the correlation matrix of the columns in the abalone.df data frame is computed using the cor() function

Then, three different correlation plots are created using the corrplot() function. The first plot is a basic scatter plot of the correlation matrix, which shows the strength and direction of the correlations between pairs of variables.

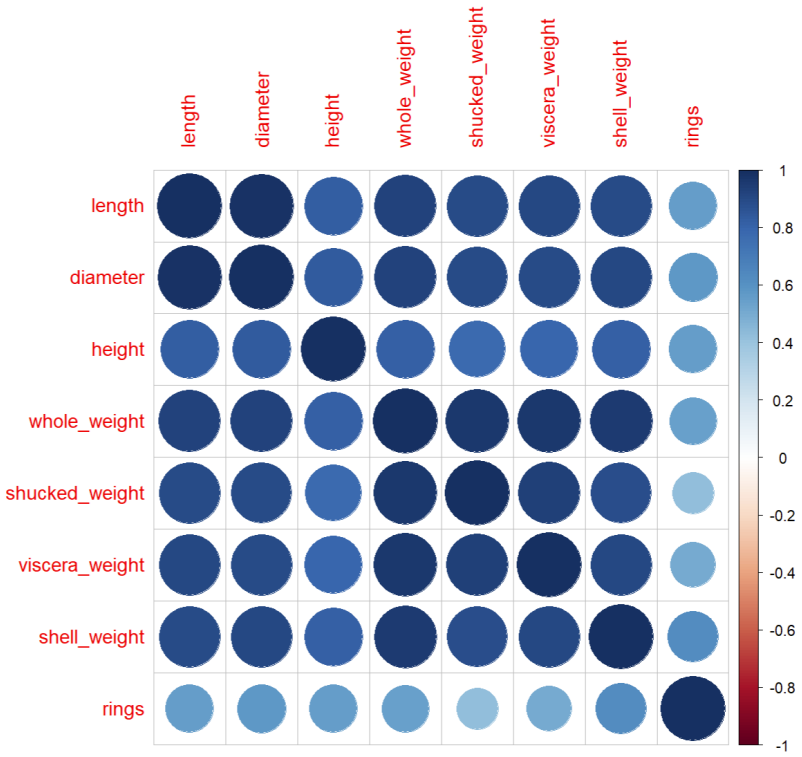
The second plot is a color-coded plot that only displays the upper triangle of the matrix, making it easier to read when there are many variables.

The third plot is similar to the second one, but it only displays the lower triangle of the matrix.

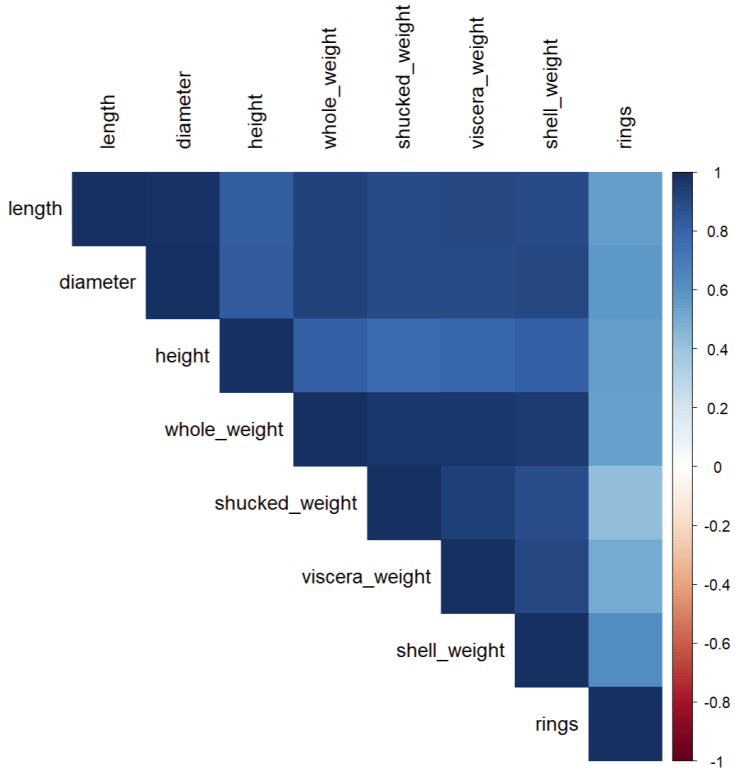
Overall, these plots can be useful for exploring the relationships between variables and identifying which variables may be most important for predicting the target variable in a regression or classification model.

```
>
> # correlation
> library(corrplot)
> abalone.cor <- cor(abalone.df)
> corrplot(abalone.cor)
> corrplot(abalone.cor, type = "upper", method = "color", tl.col = "black")
> corrplot(abalone.cor, type = "lower", method = "color", tl.col = "black")
>
>
```

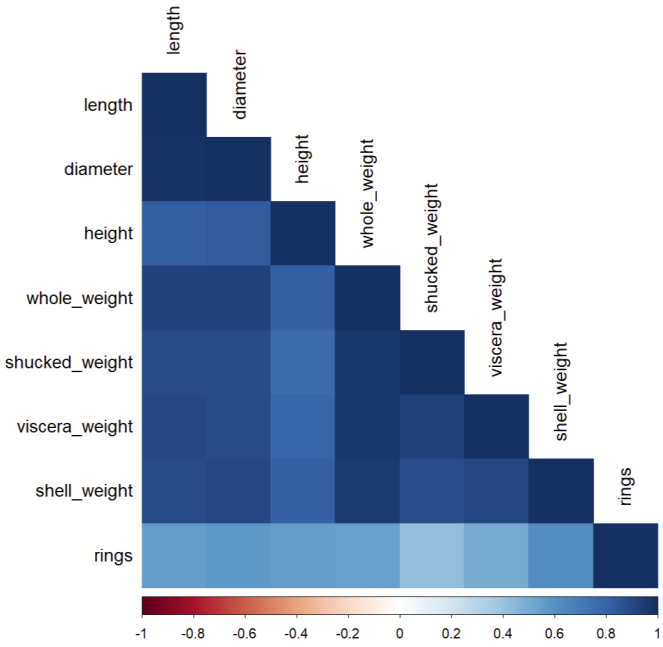
First Plot:


length
diameter
height
whole_weight
shucked_weight
viscera_weight
shell_weight
rings
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

Second Plot:


length
diameter
height
whole_weight
shucked_weight
viscera_weight
shell_weight
rings
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

Third Plot:



K-Means:

K-means clustering is performed on a dataset using the kmeans() function. We want to cluster the data into two groups by using the norm dataset with centers = 2.
The cluster means, sizes, and assignments are all included in the abalone.kmeans2 object. The cluster component assigns a cluster to each observation in the dataset. The mean values for each variable in each of the two clusters are given by the centers component. The value of **between_ss/total_ss** in the output is important as it tells us how well the clustering algorithm can group the data.

```
> # k-means clustering
> library(ggplot2)
> library(factoextra)
> # with k=2
> abalone.kmeans2 <- kmeans(abalone.znorm, centers = 2, nstart = 10)
> str(abalone.kmeans2)
List of 9
 $ cluster     : int [1:4177] 2 2 2 2 2 2 1 1 2 1 ...
 $ centers     : num [1:2, 1:8] 0.727 -0.865 0.731 -0.87 0.652 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "1" "2"
  .. ..$ : chr [1:8] "length" "diameter" "height" "whole_weight" ...
 $ totss       : num 33408
 $ withinss    : num [1:2] 9308 5301
 $ tot.withinss: num 14609
 $ betweenss   : num 18799
 $ size        : int [1:2] 2270 1907
 $ iter        : int 1
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
>
```

```
> abalone.kmeans2
K-means clustering with 2 clusters of sizes 2270, 1907

Cluster means:
      length   diameter     height whole_weight shucked_weight viscera_weight shell_weight      rings
1  0.7270910  0.7311727  0.6517584    0.7355855      0.7030945      0.7191419    0.7219055  0.4680421
2 -0.8654937 -0.8703524 -0.7758214   -0.8756052     -0.8369295     -0.8560315   -0.8593211 -0.5571345

Clustering vector:
   [1] 2 2 2 2 2 2 1 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2
  [64] 2 2 2 1 1 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 2 2 2 2 1 1 2 1 1 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2
 [127] 2 2 1 1 1 2 2 2 2 2 2 2 2 2 1 1 1 1 2 2 2 2 2 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
 [190] 1 1 1 1 2 1 1 2 1 1 1 2 1 2 1 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1
 [253] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 1 2 2 2 1 1 1 1 1 1 1 1 1 2 1 2 2 2 1 1 1 1 2 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2 1 1 1 1 1 1 1 1
 [316] 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 2 2 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [379] 1 1 1 1 2 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 2 2 2 2 2 1 2
 [442] 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 2 2 2 1 1 1 1 1 2 2 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [505] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2 2 1 1 1 2 2 2 2 1 2 2 2
 [568] 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 2 2 2 1 1 2 2 1 2 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2
 [631] 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 1 1 1 1 1 1 2 1 2 2 2 2 1 2 1 1 1 1 1 2 2 2
 [694] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 2 2 2 2 2 1 1 1 2 1 2 2 2 1 2 1 2 1 1 1 1 1 2 1 1 2 2 1 1 1 1
 [757] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 2 2 2 1 1 2 2 2 1 2 1 2 2 1 2 2 2 1 1 2 2 1 1 1 1 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2
 [820] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [883] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [946] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [ reached getOption("max.print") -- omitted 3177 entries ]

Within cluster sum of squares by cluster:
[1] 9308.050 5301.108
 (between_SS / total_SS =  56.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
>
```

```
>
> # k-means cluster plot
> fviz_cluster(abalone.kmeans2, data = abalone.znorm, geom = "point")
> |
```

K-means cluster with k=3

```
> abalone.kmeans3
K-means clustering with 3 clusters of sizes 1177, 1229, 1771

Cluster means:
      length   diameter      height whole_weight shucked_weight viscera_weight shell_weight      rings
1 -1.2786312 -1.2797896 -1.12228929   -1.1384197    -1.08490186    -1.11668868  -1.12936206 -0.8593091
2  1.0338734  1.0398320  0.97838542    1.2176645     1.17368659     1.19468778   1.17285936  0.6196844
3  0.1323086  0.1289434  0.06691068   -0.0884188    -0.09346772    -0.08691626  -0.06334557  0.1410586

Clustering vector:
   [1] 3 1 3 3 1 1 3 3 3 3 3 1 3 3 1 3 1 1 1 1 1 1 3 3 2 3 3 3 3 3 2 2 2 2 1 2 1 3 1 1 3 1 1 1 1 3 1 1 3 3 1 3 3 1 3 1 3 1 3 1 3 3
  [64] 1 3 1 3 2 3 1 3 1 2 2 2 3 3 3 3 2 3 2 3 2 3 2 2 3 3 3 3 3 2 2 2 2 3 3 3 1 3 3 3 2 3 3 3 3 1 3 1 1 1 3 3 3 1 3 1 1 1 3 1 1 1
 [127] 1 1 2 2 2 1 1 1 1 1 1 1 1 1 3 3 2 3 3 3 3 1 1 1 2 2 2 3 3 3 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 3 3 3 1 1 1 1 1 3 3 2 3 3 2 2 2 2 2
 [190] 3 2 2 3 1 3 3 3 2 3 3 1 3 3 3 1 3 1 1 3 1 3 1 1 3 3 3 1 1 3 1 1 3 3 3 1 3 1 1 3 3 3 3 2 1 3 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 2
 [253] 3 2 2 3 2 3 2 2 3 3 3 1 1 3 3 1 3 3 2 2 2 2 2 2 2 2 2 2 3 3 1 1 3 3 3 3 3 1 3 3 2 2 2 2 1 1 1 3 1 1 3 1 1 3 1 1 2 3 3 2 3 2 2 2
 [316] 1 2 1 3 1 1 1 1 1 1 1 1 3 1 1 3 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 1 1 2 2 3 3 3 2 2 2 2 2 2 2 3 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 3
 [379] 3 2 2 3 3 3 3 1 3 3 3 3 1 3 3 1 1 1 3 3 3 3 3 3 1 3 1 3 1 3 3 3 2 3 3 3 2 3 2 2 2 3 2 3 3 1 1 3 2 2 3 3 3 2 3 3 1 3 1 1 1 3 1
 [442] 1 3 1 1 3 3 3 2 3 2 2 2 3 2 3 2 1 1 3 1 3 1 1 1 1 2 2 2 2 3 3 1 3 3 3 1 2 2 2 2 3 3 3 3 3 3 2 3 3 3 3 2 2 2 3 2 2 2 3 3 2 2 2
 [505] 2 2 3 2 3 3 2 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 1 3 3 3 1 3 3 3 1 1 1 3 1 1 1 1 1 1 1 3 3 2 3 3 3 3 3 3 2 2 3 1 3 3 3 3 1 3
 [568] 1 1 1 3 3 2 3 3 2 3 3 3 2 2 3 2 3 1 3 3 3 3 1 3 1 3 2 3 3 3 3 3 3 3 1 1 3 3 3 1 3 1 1 3 1 1 3 3 3 3 1 1 1 1 3 3 3 3 3 1 1 3 1
 [631] 3 1 3 3 1 1 1 1 3 3 1 2 3 1 1 1 1 3 3 1 1 1 3 1 1 1 3 2 2 3 2 3 1 1 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 3 1 3 3 3 3 3 3 3 3 1 1
 [694] 1 1 1 1 1 3 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 2 3 1 3 1 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 3 3 3 3 2 3 3 2 3 3 2 3 2 2
 [757] 2 2 3 3 3 3 3 2 2 3 3 1 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 3 1 1 1 3 1 1 3 1 3 3 3 1 1 1 1 1 1 1
 [820] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [883] 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [946] 3 3 3 1 3 3 1 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3
 [ reached getOption("max.print") -- omitted 3177 entries ]

Within cluster sum of squares by cluster:
[1] 2105.546 4569.999 3244.856
 (between_SS / total_SS =  70.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"        
[9] "ifault"      
>
```

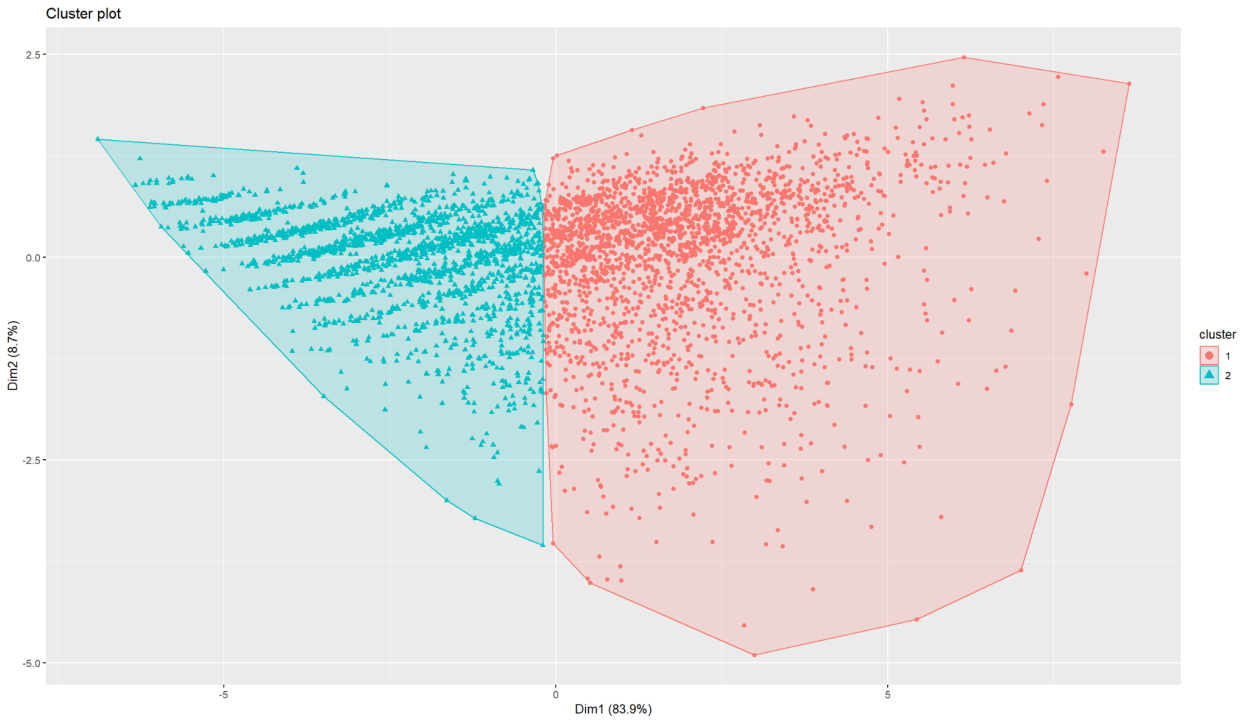
K-means cluster with k=4

```
> abalone.kmeans4
K-means clustering with 4 clusters of sizes 805, 1478, 618, 1276

Cluster means:
      length   diameter     height whole_weight shucked_weight viscera_weight shell_weight      rings
1 -1.5698367 -1.5711279 -1.3462920   -1.2833491     -1.2263130     -1.2555476   -1.2771926 -1.0312700
2  0.5767674  0.5769260  0.4669942    0.4524577      0.4382074      0.4480477    0.4414980  0.3485629
3  1.2630173  1.2688584  1.2531902    1.6585186      1.5846794      1.6024148    1.6154536  0.8325986
4 -0.2894110 -0.2916091 -0.2985297   -0.5177124     -0.5014266     -0.5029710   -0.4880442 -0.1563868

Clustering vector:
   [1] 4 1 4 4 1 1 2 2 4 2 4 4 4 4 4 1 4 1 4 1 1 2 4 2 2 2 2 2 2 2 3 3 3 3 4 2 4 2 1 4 2 1 1 1 1 4 4 1 2 4 1 4 4 1 4 4 4 1 4 4 4 4
  [64] 1 4 4 2 2 4 1 2 1 2 2 2 2 2 2 2 2 2 3 2 3 2 2 2 2 4 2 2 2 2 2 2 3 3 4 4 4 4 1 2 2 4 2 4 2 4 4 4 4 4 1 4 4 4 4 4 2 1 4 1 2 1 1 1
 [127] 1 1 3 3 2 4 1 1 1 4 1 1 4 1 2 4 3 2 4 4 4 1 1 1 2 2 3 2 2 4 2 3 2 2 2 2 2 3 3 3 3 3 3 3 3 4 2 4 1 1 1 1 1 2 2 3 2 2 3 2 2 3 2
 [190] 2 2 2 2 1 4 4 4 3 2 2 4 2 4 2 4 4 1 4 2 1 4 1 4 4 4 4 4 1 4 4 4 4 4 4 4 4 1 1 2 2 2 2 3 1 4 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2
 [253] 2 2 2 2 2 2 3 2 2 2 2 1 1 4 2 1 4 4 3 2 2 3 2 3 3 3 3 4 2 1 4 4 2 2 4 4 4 2 2 2 2 2 2 2 1 1 1 4 1 1 2 1 1 4 1 1 3 4 2 3 2 3 3 3
 [316] 4 3 4 2 1 1 1 1 1 1 1 1 4 1 1 4 1 1 1 3 2 2 2 3 2 2 3 2 4 4 4 4 1 1 2 2 2 2 2 2 3 3 3 3 3 2 2 3 2 2 2 2 2 2 3 3 3 3 3 2 3 2 2
 [379] 2 2 2 4 4 4 2 4 4 4 4 4 4 4 4 1 1 1 4 4 4 2 4 4 4 4 4 2 4 2 2 4 2 2 2 2 2 2 3 3 2 4 3 4 4 1 1 2 2 2 2 2 2 2 2 2 2 4 4 1 4 1 4 1
 [442] 1 2 1 4 2 2 2 2 2 2 3 3 3 2 3 2 2 1 4 4 1 2 1 1 1 1 3 3 3 3 4 4 4 4 4 2 4 3 3 3 3 2 2 2 2 2 2 2 4 2 2 2 2 3 3 2 2 2 2 2 2 3 2 2
 [505] 2 2 2 2 2 2 2 4 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 4 4 4 4 4 4 1 1 1 4 4 4 4 1 1 1 1 2 2 2 2 2 4 4 4 2 2 3 4 4 4 4 4 4 1 4
 [568] 4 1 4 4 4 2 2 2 2 2 2 2 3 2 2 3 4 4 4 4 2 4 4 4 1 2 3 4 2 4 2 2 2 2 1 4 4 4 4 1 4 1 1 4 1 1 4 4 4 4 1 1 1 1 4 4 4 4 4 4 1 2 1
 [631] 4 4 4 4 1 1 1 1 4 4 1 2 2 1 4 4 1 4 4 4 1 1 4 1 1 1 2 2 3 2 3 2 4 1 4 1 4 4 2 4 4 4 2 4 2 2 4 4 2 1 1 4 4 2 4 2 2 2 4 2 4 1 4
 [694] 1 1 1 1 1 4 4 1 4 4 4 4 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 4 4 1 4 4 4 4 2 4 4 2 4 4 4 2 4 2 1 2 2 2 2 3 4 2 3 4 4 2 2 3 3
 [757] 2 3 2 2 2 2 2 3 2 2 2 1 2 2 2 4 1 4 4 4 4 4 4 4 2 4 2 1 1 2 4 4 4 2 2 4 4 2 2 4 4 4 4 4 4 4 4 1 4 1 1 4 4 4 4 4 1 1 1 1 1 1 1
 [820] 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 3 3 2
 [883] 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 [946] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [ reached getOption("max.print") -- omitted 3177 entries ]

Within cluster sum of squares by cluster:
[1] 1074.182 2608.099 2599.581 1583.616
 (between_SS / total_SS =  76.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"        
[9] "ifault"      
> |
```

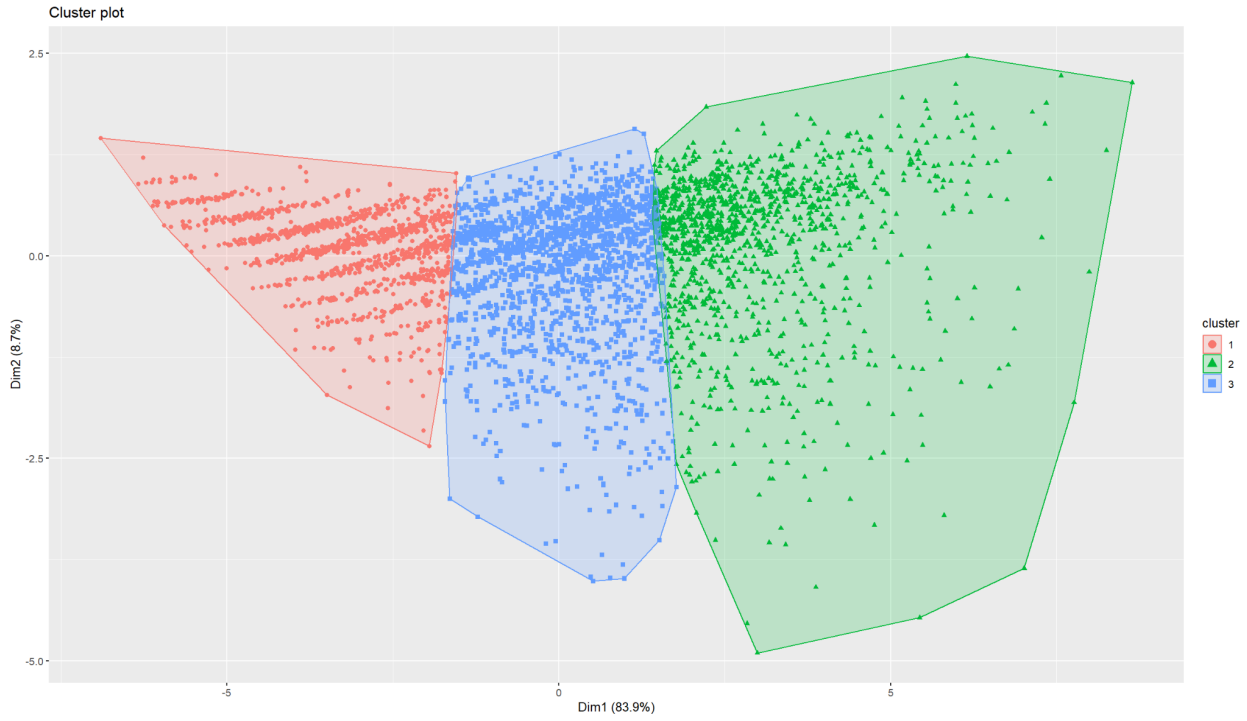
K-means cluster with k=5

```
> abalone.kmeans5
K-means clustering with 5 clusters of sizes 522, 801, 370, 1222, 1262

Cluster means:
      length   diameter     height whole_weight shucked_weight viscera_weight shell_weight        rings
1  1.3156500  1.3159606  1.2994778    1.7593433      1.7094874      1.7132990    1.6892428  0.697923577
2 -1.5737677 -1.5748660 -1.3493418   -1.2850494     -1.2281793     -1.2569499   -1.2787113 -1.031876076
3  0.4030196  0.4475625  0.5572452    0.3523700      0.0767541      0.2790835    0.5908279  1.961986153
4  0.6441865  0.6360056  0.4742653    0.5191928      0.5656417      0.5290657    0.4438164 -0.004559064
5 -0.2872382 -0.2918072 -0.3036747   -0.5181324     -0.4977765     -0.5049955   -0.4900848 -0.204553939

Clustering vector:
   [1] 5 2 5 5 2 2 3 3 5 3 5 5 5 5 5 5 2 5 2 5 2 2 4 5 4 4 4 4 3 4 4 1 3 1 1 5 3 5 4 2 5 3 2 2 2 2 5 5 2 4 5 2 5 5 2 5 5 5 2 5 5 5 5
  [64] 2 5 5 4 3 5 2 3 2 3 4 4 3 4 4 5 3 4 1 3 3 3 4 4 4 5 4 3 5 3 4 1 1 5 5 5 5 2 3 3 5 4 5 4 5 5 5 5 5 5 2 5 5 5 5 5 4 2 5 2 3 2 2 2
 [127] 2 2 1 1 3 5 2 2 2 5 2 2 5 2 4 5 1 4 5 5 5 2 2 2 3 4 1 4 4 5 3 1 3 3 4 4 4 1 1 1 1 1 1 1 5 4 5 2 2 2 2 2 4 3 3 4 4 1 4 4 4 4
 [190] 4 4 4 3 2 5 5 5 3 3 4 5 3 5 3 5 5 2 5 3 2 5 2 5 5 5 5 5 2 5 5 5 5 5 5 5 5 5 2 2 4 3 3 3 3 2 5 2 2 2 2 2 3 2 2 2 2 2 2 2 2 4
 [253] 4 3 3 3 3 3 4 1 4 4 4 4 2 2 5 3 2 5 5 3 3 3 1 4 1 1 1 3 5 3 2 5 5 3 4 5 3 5 4 3 4 3 3 3 2 2 2 5 2 2 3 2 2 5 2 2 1 5 4 3 3 3 3 3
 [316] 5 3 5 3 2 2 2 2 2 2 2 5 2 2 5 2 2 2 1 4 3 3 4 3 3 1 4 5 5 5 5 2 2 3 4 3 4 4 3 1 1 1 1 1 3 4 3 3 4 3 4 4 4 1 1 1 1 1 4 1 4 4
 [379] 3 4 3 5 5 5 4 5 5 5 5 5 5 5 2 2 5 5 5 4 5 5 5 5 5 4 5 3 4 5 3 3 4 5 3 3 3 3 3 5 1 5 5 2 2 3 3 3 3 3 3 3 3 3 5 5 2 5 2 5 2 2
 [442] 2 3 2 5 4 3 3 4 3 1 1 1 4 4 4 3 2 5 5 2 4 2 2 2 2 1 1 1 3 5 5 5 5 5 3 5 3 1 3 3 3 3 3 4 3 4 3 5 3 3 3 4 3 3 3 4 3 3 4 4 4 4 3 3 4
 [505] 3 3 4 3 4 3 4 5 5 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 4 2 3 5 5 5 5 5 5 2 2 2 5 5 5 5 2 2 2 2 4 4 3 4 4 5 5 5 3 4 4 5 5 5 5 5 5 2 5
 [568] 5 2 5 5 5 3 3 4 4 4 4 4 3 4 3 1 5 5 5 5 3 5 5 5 2 3 3 5 3 5 3 4 3 3 2 5 5 5 5 2 5 2 2 5 2 2 3 5 5 5 2 2 2 2 3 5 5 5 5 5 2 3 2
 [631] 5 5 5 5 2 2 2 2 5 5 2 4 3 2 5 5 2 5 5 5 5 2 2 5 2 2 2 3 3 3 4 1 4 5 2 3 2 5 5 3 5 5 3 3 3 3 3 5 5 3 2 2 5 5 3 5 3 3 3 5 4 5 2 5
 [694] 2 2 2 2 2 5 5 2 5 5 5 5 5 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 5 5 2 5 5 3 5 3 3 5 3 5 5 5 3 5 3 2 3 3 3 3 1 5 3 1 5 5 4 4 3 1
 [757] 3 3 4 3 3 3 3 1 3 4 4 2 4 3 4 5 2 5 5 5 3 5 5 5 3 5 4 2 2 3 5 5 5 4 4 5 5 4 4 3 5 5 5 5 5 5 5 5 2 5 2 2 2 3 5 5 5 3 2 2 2 2 2 2 2
 [820] 2 2 2 2 2 2 2 2 2 2 2 2 2 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 4 4 4 4 4 4 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 1 4
 [883] 1 3 1 4 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 5 5 5 2 5 5 5 5 5 5 5 5 5 5 5
 [946] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 5 5 5 4 4 4 4 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 [ reached getOption("max.print") -- omitted 3177 entries ]

Within cluster sum of squares by cluster:
[1] 2063.0186 1065.5877  838.6202 1392.2357 1438.1221
 (between_SS / total_SS =  79.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
> 
```

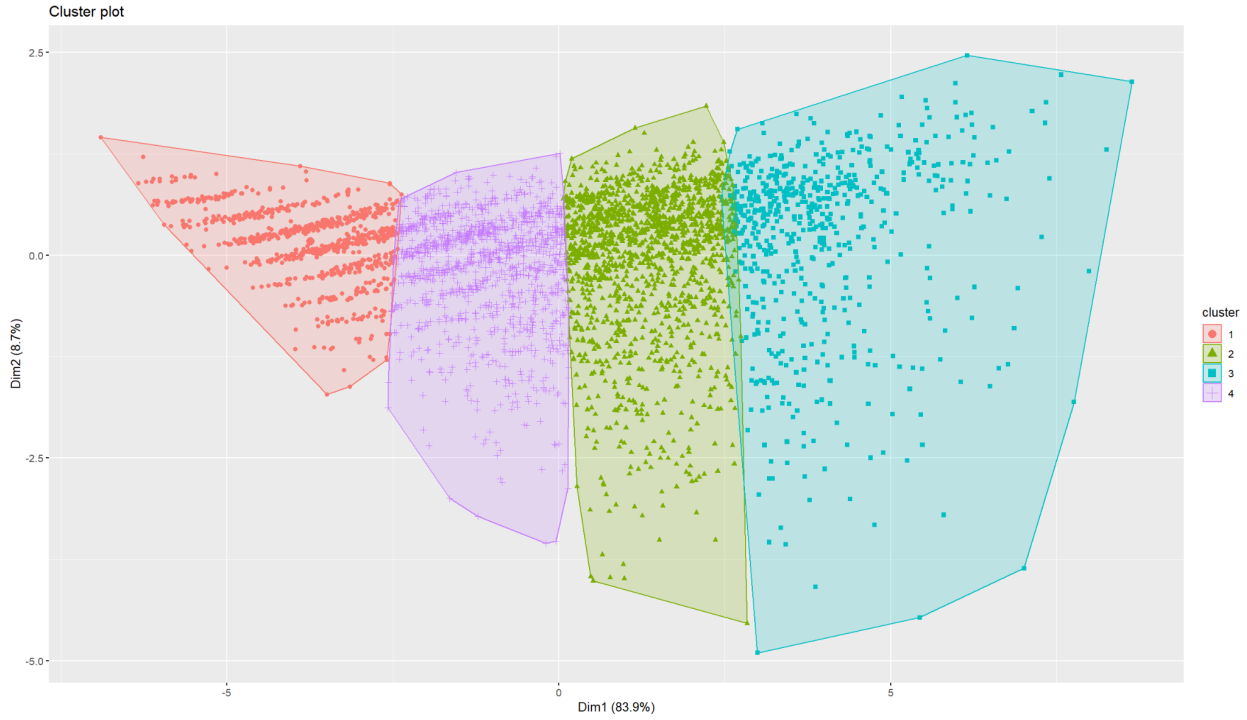
Cluster Plot with k=2:



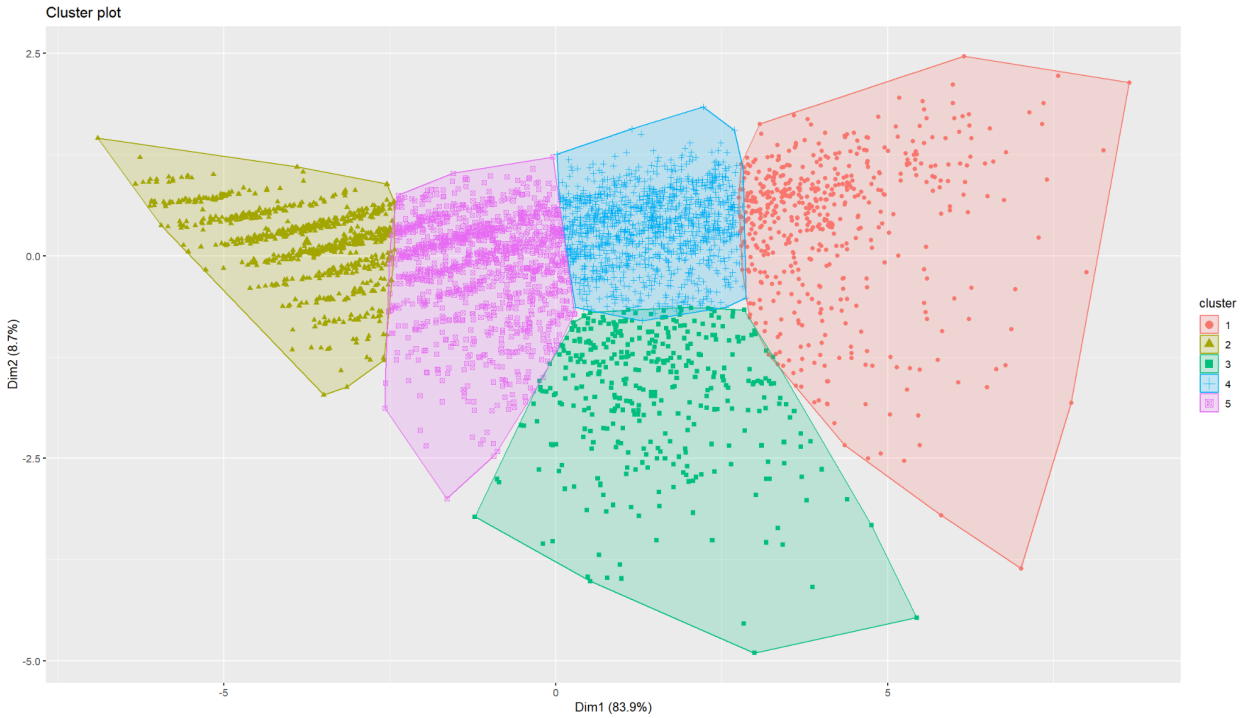
Cluster Plot with k=3



Cluster Plot with k=4



Cluster Plot with k=5



Finding the best number of clusters (k):

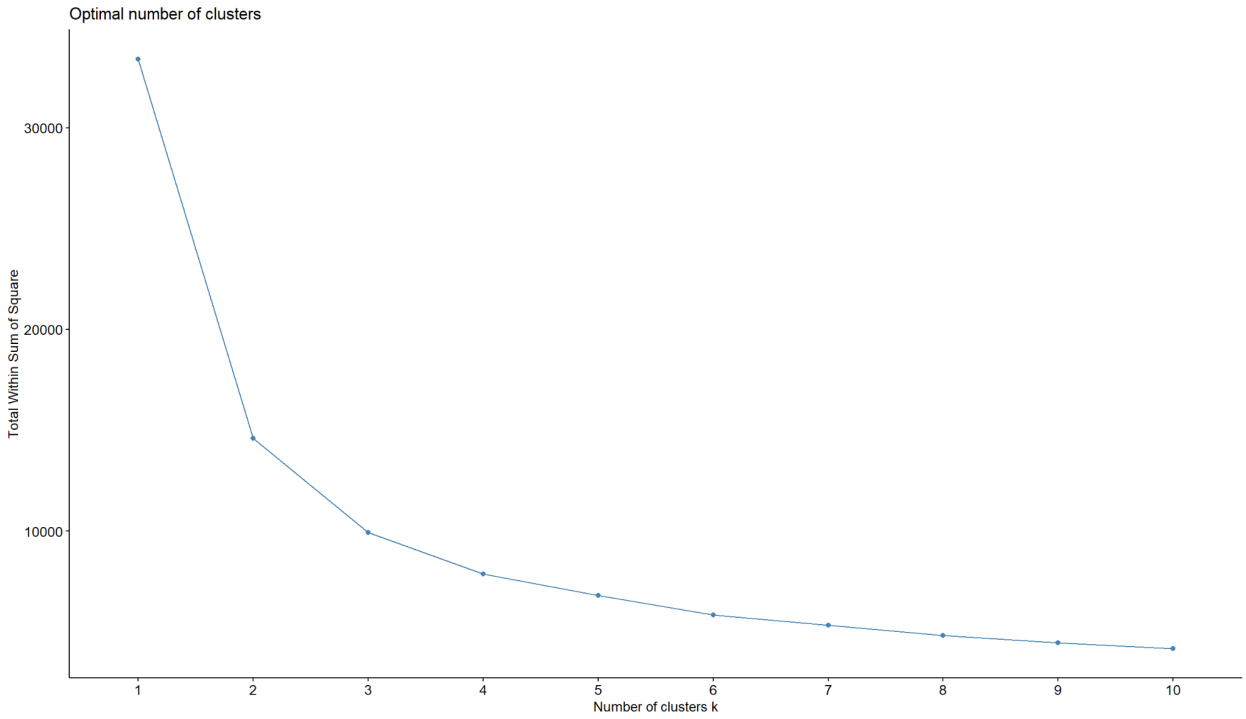
For different values of k, the wssplot function is used to plot the within-groups sum of squares (number of clusters). The function accepts three arguments: **data**, which is the data frame to be clustered, **nc**, which is the maximum number of clusters to be tested, and **seed**, which is the random seed used by the k-means algorithm.

Using the elbow method, the fviz nbclust function determines the optimal number of clusters. It considers four arguments: data denotes the data frame to be clustered, FUNcluster denotes the clustering function to be used, method denotes the method for determining the optimal number of clusters, and k.max denotes the maximum number of clusters to be tested. The function returns a plot of the within-cluster sum of squares for each k value, along with a vertical line at the optimal number of clusters suggested by the method.

The elbow point in this case is observed to be around **4 to 6 clusters**, as suggested by the fviz nbclust function.

```
> # calculate appropriate number of clusters
> wssplot <- function(data, nc=15, seed=1234) {
+     wss <- (nrow(data)-1)*sum(apply(data, 2, var))
+     for (i in 2:nc) {
+         set.seed(seed)
+         wss[i] <- sum(kmeans(data, centers = i)$withinss)
+     }
+     plot(1:nc, wss, type="b", xlab="Number of clusters", ylab="Within group sum of squares")
+ }
> wssplot(abalone.znorm, 10, 12342)
> # finding number of clusters using fviz_nbclust function
> fviz_nbclust(abalone.znorm, kmeans, method = "wss", k.max = 10, verbose = TRUE)
>
```

Output:



KNN

Creating datasets for KNN
The number of rows in the abalone.znorm data frame is assigned to the abalone.znorm.rows variable. The variable abalone.znorm.sample_70 is set to 0.7, which indicates that 70% of the data will be used for training. The variable abalone.rows_70 is then calculated as the product of abalone.znorm.sample_70 and abalone.znorm.rows, which represents the number of rows to be used for training.

The sample() function is then used to generate a random set of abalone.rows_70 indices from 1:abalone.znorm.rows, which represents the row indices of the abalone.znorm data frame. These chosen indices are then saved in the variable abalone.train.index_70. Finally, the length() function on abalone.train.index_70 is called to confirm the number of indices chosen for training.

a) Dataset for 70-30 split:

```
> # splitting data into 70-30
> abalone.znorm.rows <- nrow(abalone.znorm)
> abalone.znorm.sample_70 <- 0.7
> abalone.rows_70 <- abalone.znorm.rows * abalone.znorm.sample_70
> abalone.rows_70
[1] 2923.9
> abalone.train.index_70 <- sample(abalone.znorm.rows, abalone.rows_70)
> length(abalone.train.index_70)
[1] 2923
```

```
> abalone.train_70 <- abalone.znorm[abalone.train.index_70, ]
> abalone.train_70[1:10, ]
         length   diameter      height whole_weight shucked_weight viscera_weight shell_weight      rings
755   1.0492534  1.0793923  0.84834085    1.3290629      0.7664906      0.8338915   2.05577337  3.1221426
1006  0.6745436  0.6763285  0.37018145    0.6775393      0.9579640      0.5921346   0.25983080 -0.2895892
2585  0.1332960  0.2732646  0.01156191   -0.1921784     -0.1323081     -0.1924349  -0.18556296 -0.2895892
2339  1.0908879  0.7770944  1.32650024    1.6135309      1.2080057      1.6184610   1.76842256  2.1916703
1448 -0.6993926 -0.8855439 -0.46659748   -0.7590752     -0.5220127     -0.8082308  -0.97936958 -0.5997466
3952 -0.4912205 -0.1801822 -0.70567718   -0.3940589     -0.8013386     -0.2152422  -0.09935772  0.9510406
3358 -1.2406402 -1.2382247 -0.94475688   -1.2026007     -1.1459908     -1.1412167  -1.21284211 -0.5997466
3980 -0.2414139 -0.3313311 -0.58613733   -0.4164901     -0.3485604     -0.3429628  -0.60222164 -1.2200615
1134  0.3831025  0.5251795  0.25064161    0.4256984      0.8205537      0.2089728   0.08023654 -0.2895892
3230  0.9659846  0.9786263  1.20696039    0.9130666      0.5592488      1.3447740   1.15780208  0.6408831
>
> abalone.test_70 <- abalone.znorm[-abalone.train.index_70, ]
> abalone.test_70[1:10, ]
         length    diameter     height whole_weight shucked_weight viscera_weight shell_weight       rings
3   0.050027102  0.12211570 -0.1079779  -0.30943221    -0.46344441    -0.35664713  -0.20711427 -0.28958918
6  -0.824295935 -1.08707578 -1.0642967  -0.97319096    -0.98380153    -0.94051282  -0.85365360 -0.59974661
14  0.091661532 -0.02903324  0.1311018  -0.29413823    -0.39136031    -0.08752154  -0.24303312  0.02056826
22 -1.199005807 -1.33899067 -0.9447569  -1.23012983    -1.25862217    -1.20051551  -1.10508556  0.02056826
34  1.299060012  1.43207313  0.8483408   1.97650805     2.05274129     1.93320113   1.55290945  2.81198518
43 -2.364769856 -2.34665023 -2.2596952  -1.54722502    -1.47712711    -1.43314950  -1.57203062 -1.53021892
47 -0.449586062 -0.38171408 -0.4665975  -0.50825396    -0.29900255     0.42336094  -0.70997819 -0.28958918
50  0.008392672  0.17249868  0.4897213   0.01378057    -0.02192928     0.30020177   0.04431769 -0.28958918
53 -0.324682771 -0.48248004 -0.2275178  -0.58574346    -0.44992864    -0.77173915  -0.56630279  0.02056826
55 -0.990833656 -0.98630982 -0.9447569  -0.90487785    -0.83963332    -0.81279220  -0.92549130 -0.90990405
>
> dim(abalone.test_70)
[1] 1254    8
> dim(abalone.train_70)
[1] 2923    8
>
```

b) Dataset for 60-40 split:

```
> # splitting data into 60-40
> abalone.znorm.rows <- nrow(abalone.znorm)
> abalone.znorm.sample_60 <- 0.6
> abalone.rows_60 <- abalone.znorm.rows * abalone.znorm.sample_60
> abalone.rows_60
[1] 2506.2
> abalone.train.index_60 <- sample(abalone.znorm.rows, abalone.rows_60)
> length(abalone.train.index_60)
[1] 2506
> abalone.train_60 <- abalone.znorm[abalone.train.index_60, ]
> abalone.train_60[1:10, ]
         length   diameter     height whole_weight shucked_weight viscera_weight shell_weight      rings
3741  1.0492534  0.9282433  0.7288010    1.1740839      1.5076053      1.2535450    0.6082437  0.3307257
2264  0.7994468  1.0290093  0.2506416    1.2791025      0.9985113      0.9752965    0.5831005  0.9510406
1067 -1.6986190 -1.8428204 -1.3033764   -1.4014224     -1.3149379     -1.3784121   -1.4247633 -1.2200615
3012 -1.2406402 -1.2382247 -1.1838366   -1.2066791     -1.1414855     -1.2871831   -1.1410044 -0.5997466
1983  1.6321355  1.5832221  1.2069604    2.5536009      3.2533921      2.2798714    1.8977304  0.3307257
1736  1.2157912  1.1297753  0.6092612    1.1445155      1.5841946      0.9661736    0.4394251  0.3307257
3617  0.7578124  0.8778604  1.4460401    0.9691445      0.9917534      1.1942461    0.9494727  1.2611980
2586  0.2165648  0.1724987 -0.3470576    0.2758174      0.8475852     -0.1969964   -0.1711954 -0.5997466
1995 -1.9484256 -1.9939694 -1.7815358   -1.4921667     -1.4500956     -1.4468339   -1.5001929 -1.2200615
2750 -0.1581450 -0.2809481 -0.1079779   -0.5918611     -0.4274024     -0.7808620   -0.5303839 -0.2895892
> abalone.test_60 <- abalone.znorm[-abalone.train.index_60, ]
> abalone.test_60[1:10, ]
        length    diameter     height whole_weight shucked_weight viscera_weight shell_weight       rings
4  -0.69939264 -0.43209706 -0.3470576   -0.6377430     -0.6481599    -0.60752692 -0.602221637  0.02056826
5  -1.61535011 -1.54052258 -1.4229163   -1.2719334     -1.2158222    -1.28718308 -1.320598665 -0.90990405
8   0.17493039  0.17249868 -0.3470576   -0.1238653     -0.2944973    -0.28366392  0.152074243  1.88151287
14  0.09166153 -0.02903324  0.1311018   -0.2941382     -0.3913603    -0.08752154 -0.243033123  0.02056826
15 -0.44958606 -0.53286302 -0.9447569   -0.7203305     -0.8644122    -0.91314412 -0.386708528  0.02056826
16 -0.19977948 -0.07941621 -0.2275178   -0.3349222     -0.4566865    -0.43419179  0.008398837  0.64088313
17 -1.40717796 -1.28860769 -1.3033764   -1.0975820     -1.1910433    -1.28718308 -0.889572448 -0.90990405
18 -0.69939264 -0.68401195 -0.9447569   -0.7702908     -0.7720545    -0.85384526 -0.781815894  0.02056826
19 -1.32390910 -1.13745875 -1.4229163   -1.1689539     -1.1820328    -1.25525292 -0.997329002 -0.90990405
22 -1.19900581 -1.33899067 -0.9447569   -1.2301298     -1.2586222    -1.20051551 -1.105085556  0.02056826
> dim(abalone.test_60)
[1] 1671    8
> dim(abalone.train_60)
[1] 2506    8
>
```

c) Dataset for 50-50 split:

```
> # splitting data into 50-50
> abalone.znorm.rows <- nrow(abalone.znorm)
> abalone.znorm.sample_50 <- 0.5
> abalone.rows_50 <- abalone.znorm.rows * abalone.znorm.sample_50
> abalone.rows_50
[1] 2088.5
> abalone.train.index_50 <- sample(abalone.znorm.rows, abalone.rows_50)
> length(abalone.train.index_50)
[1] 2088
> abalone.train_50 <- abalone.znorm[abalone.train.index_50, ]
> abalone.train_50[1:10, ]
         length   diameter     height whole_weight shucked_weight viscera_weight shell_weight      rings
3869 -0.6993926 -0.6840120 -0.1079779  -0.87938788    -0.94100159    -0.78542350  -0.74589704 -0.5997466
3140 -1.5737157 -1.4901396 -1.1838366  -1.31577612    -1.26763268    -1.42858805  -1.24876096  0.3307257
1000  0.5912747  0.6259455  0.3701815   0.59597142     0.41508059    -0.02366123  -0.60222164  0.3307257
3665  0.2581993  0.2228817  0.2506416  -0.09327729    -0.06472922    -0.04646848   0.05868523 -0.2895892
2157  0.7161780  0.9282433  0.6092612   0.90287063     0.59529084     0.04019908   1.80434141  0.6408831
3796  1.2574256  1.3313072  0.8483408   1.47996349     1.37470020     0.88862891   1.96238435  0.6408831
1283 -0.1997795  0.1221157 -0.1079779  -0.31045181    -0.25845525    -0.35664713  -0.23225747 -0.2895892
3519  1.5488666  1.6336050  1.3265002   1.05886923     2.42893021     2.39846910   1.54213379  0.3307257
3043  0.4247370  0.2228817 -0.2275178  -0.18566479    -0.31477095     0.18160405  -0.13527657 -0.5997466
3357 -1.1157369 -1.0870758 -0.9447569  -1.06087645    -0.99506467    -0.97700443  -1.10508556 -1.2200615
> abalone.test_50 <- abalone.znorm[-abalone.train.index_50, ]
> abalone.test_50[1:10, ]
       length    diameter     height whole_weight shucked_weight viscera_weight shell_weight       rings
5  -1.6153501 -1.54052258 -1.4229163   -1.2719334     -1.2158222     -1.2871831   -1.3205987 -0.90990405
6  -0.8242959 -1.08707578 -1.0642967   -0.9731910     -0.9838015     -0.9405128   -0.8536536 -0.59974661
7   0.0500271  0.07173272  0.2506416   -0.1044929     -0.5512969     -0.3566471    0.6549382  3.12214262
8   0.1749304  0.17249868 -0.3470576   -0.1238653     -0.2944973     -0.2836639    0.1520742  1.88151287
9  -0.4079516 -0.38171408 -0.3470576   -0.6509978     -0.6436547     -0.6212113   -0.5303839 -0.28958918
10  0.2165648  0.32364761  0.2506416    0.1340932     -0.2021395     -0.2699796    0.5831005  2.81198518
12 -0.7826615 -0.58324600 -0.7056772   -0.8620547     -0.8644122     -0.9085827   -0.7458970  0.02056826
15 -0.4495861 -0.53286302 -0.9447569   -0.7203305     -0.8644122     -0.9131441   -0.3867085  0.02056826
20 -0.6161238 -0.88554386 -0.9447569   -0.9130346     -0.8508965     -0.9633201   -0.8895724 -0.28958918
21 -1.4071780 -1.28860769 -1.0642967   -1.1893459     -1.1887907     -1.0819178   -1.1769233  0.33072569
> dim(abalone.test_50)
[1] 2089    8
> dim(abalone.train_50)
[1] 2088    8
>
```

Implementation for k=5 with 70-30 split

- Creating training labels with k-means

```
> abalone.train_70_k5 <- kmeans(abalone.train_70, centers = 5)
> abalone.train_70_k5
K-means clustering with 5 clusters of sizes 545, 376, 884, 261, 857

Cluster means:
      length   diameter     height whole_weight shucked_weight viscera_weight shell_weight       rings
1 -1.5920501 -1.5888717 -1.3531664   -1.2899326    -1.23002826     -1.2654471   -1.2820963 -1.031121544
2  1.2947415  1.2992818  1.2285793    1.7204152     1.64638754      1.6862156    1.6958970  0.769565469
3  0.6442597  0.6381993  0.4698431    0.5139652     0.55867799      0.5217263    0.4437199  0.006533984
4  0.3601318  0.4024072  0.4796451    0.2876932     0.02374469      0.2296828    0.5144831  1.907656411
5 -0.3024810 -0.3085206 -0.3114886   -0.5248328    -0.49963629     -0.5146160   -0.4994988 -0.232045384

Clustering vector:
 755 1006 2585 2339 1448 3952 3358 3980 1134 3230 1934 1501 3783 1914 1109 1075 3146 1386 2284 2378 4044 2260  686 3857  847  983  151
   4    3    5    2    5    5    1    5    3    2    2    3    3    3    5    1    4    3    5    5    3    3    4    1    5    3    4
1638 2208 2474 1029  326 3856 2837 1956 4093  985  986 2503 1762 1584 4084 2087 2244 1793 4141 1808  344 2507 2992 3092  195 3236 3124
   3    5    3    3    1    5    3    2    2    3    3    1    2    5    3    3    5    3    3    3    5    1    3    5    5    2    3
2225 2875 2132 4023 3464 1326 3949 2667 3502 2758 3833  712 1452 1828 3501 3069 2446 3061  528 4055 1569  473 1149 2037 2313 2823 1927
   5    5    1    2    2    3    4    3    3    5    5    1    5    1    3    3    5    3    4    3    5    5    3    1    1    5    3
2234 3645 3324  458 3224  831 1078  919 1562 2283 1313  185 3837  413  627 2570 1333 3126 2253 1899  779 1561  564  794 1415 2457 3799
   4    5    5    1    5    1    1    1    5    5    5    2    5    3    5    5    5    3    4    5    5    1    5    3    2    1    2
1370  160 2516 3581 2968 3129 2505  617  357  279  270 2926 2694 2395 3201 2266  618 1905 2746  337 2845 2074  539  981 3591  956 1781
   3    4    5    3    3    3    1    5    2    2    5    3    3    3    5    2    1    3    5    4    3    3    1    3    2    5    3
1690 3625 2772 3462 1522 3294 2719 2909 2705 3008 1445 1081 2708 2211 2286 1697 2626 3588 4158 3297 1413  117 3609  648 1079 1241 2645
   3    2    3    3    2    3    1    3    2    2    5    5    2    2    5    3    2    3    5    4    3    5    5    5    1    1    5
2605 3730 1706 2463 3206 4165 3005  873 2736 3985  757 1965  988 2495 3893 3176 2869 2879  711 1492 1234  349 1425 3809 3330 4123  386
   3    5    2    5    1    1    2    3    5    3    4    2    3    5    4    5    1    5    1    3    1    1    2    5    5    5    5
1165 2072 2993 3071 1987 3538 1154 2213 1727 1612 1401 3853 2218 1469  710 3946 1258 2470 2556 4172 2112 2128 1507 2596 3035 2523 4174
   3    3    3    3    1    1    3    2    2    5    2    4    4    5    1    1    1    4    1    3    5    1    2    3    3    3    3
2813 3415 1347 2527 3632 3522 3183  791 3984  905 1341  807 1246 1311 1758 3657  292 1250 1814 1708  297  860  605 2685 1835 3111 1261
   1    5    3    3    1    1    3    3    3    1    3    1    1    5    2    5    3    1    2    3    1    3    5    3    1    1    5
3981  619 1057   83 3468  866 2325 3281 1023   76 1118 3875 3102 3289 2994 3247 2422 2371 2163 3978 2898 2656 3537 1382  682 4173 2986
   5    1    1    4    2    3    2    2    2    4    5    5    3    4    2    4    4    5    2    5    5    5    5    3    5    3    5
3168 4088 2445 3476 3814 3220  989 1961 2440 1738 2641 1471 1362 2792  243 2154 2059  625 3436 3989 2451  461  141   31 1950 1139 3943
   4    3    5    1    1    2    3    2    1    2    5    5    3    3    1    1    5    5    1    3    5    1    3    3    3    3    5
  94 3786   16 3781 3492  178 3489 3536 1436 3990  524 1461  924 1602 3634  204 3919 2421 4057 3873 2432 1146 3471 3475 2363  259  494
   3    3    5    3    3    1    5    1    1    2    1    5    1    3    1    4    4    1    3    5    4    2    2    1    4    2    4
1889 1784 4086 3120 1355 3172 3064 1132 1325   10  170 3352 3420 2450 1233 3540 1851 1673 3381 3467  108    8  626  261 2589 3378 2374
   5    5    3    5    3    4    3    3    3    4    2    5    3    1    1    5    5    3    1    2    5    4    5    3    3    1    5
1098 2330 1609 3339 1911 1286 1760 3865 1747 1243 2744 3424  667  990 1143  452  856 1948 1603 2670 1960  646 1060 3127 3562 1264  891
```

```
 [ reached getOption("max.print") -- omitted 1923 entries ]

Within cluster sum of squares by cluster:
[1]  751.1423 1087.3456 1002.8202  553.9594  964.8323
 (between_SS / total_SS =  80.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

- Then the cluster vector is passed to KNN

  The output is a vector which contains the predicted values for the test dataset.

```
> abalone.test_70_k5 <- knn(abalone.train_70, abalone.test_70, abalone.train_70_k5$cluster, k = 5)
> abalone.test_70_k5
   [1] 5 1 5 1 2 1 5 3 5 1 5 5 5 5 1 5 5 5 4 3 4 4 3 4 3 5 1 3 5 5 5 5 5 1 5 5 3 1 1 1 1 1 2 2 1 5 5 1 5 3 2 2 2 5 1 1 3 3 3 5 5 5 1 1 5 5
  [67] 5 5 5 5 1 1 4 5 1 1 1 1 4 4 3 1 3 2 4 4 1 1 5 5 1 2 3 4 1 1 1 1 3 3 3 5 2 2 2 4 3 2 3 4 3 5 5 5 5 5 5 5 5 5 5 5 4 4 5 4 4 5 1 3 4 1
 [133] 5 4 1 1 3 4 3 1 2 5 4 4 4 4 4 4 3 4 4 4 3 3 5 1 1 1 1 1 4 5 1 1 1 1 1 3 5 4 5 5 5 5 1 5 1 5 4 3 3 3 4 5 5 1 4 4 5 4 4 5 5 1 5 5 5 5
 [199] 1 1 1 5 1 5 3 4 4 4 4 5 1 1 5 5 5 1 1 1 1 3 4 4 5 5 5 5 4 4 5 5 3 4 4 1 3 4 5 5 1 5 3 5 5 5 1 4 5 1 1 1 1 1 5 5 3 3 3 3 3 3 3 3 3
 [265] 3 3 2 2 1 1 1 1 1 1 1 1 1 1 1 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 2 3 2 2 2 2 3 2 2 2 1 1 1 1 1 1 5 5 5 5 5 5 5
 [331] 5 5 5 3 3 3 3 3 3 3 3 3 2 3 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 3 5 5 3 3 3 5 3 3 5 3
 [397] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 5 5 5 5 5 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1
 [463] 1 1 1 1 1 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 3 3 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 3 2 2 3 3 3 2 2 2 2 2 2 2 2 2
 [529] 2 1 5 5 5 5 3 5 3 3 3 3 3 3 3 3 2 2 2 2 2 1 1 1 1 1 5 1 5 5 5 5 5 5 5 5 3 5 3 5 3 3 3 4 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 3 2 3 2 2 2 2
 [595] 2 1 1 1 1 1 5 5 5 5 5 5 3 3 1 1 5 5 4 5 5 5 5 3 3 2 2 5 5 1 3 4 3 1 5 1 1 1 1 1 1 5 3 5 5 5 1 4 4 3 3 1 3 4 1 1 1 2 4 1 1 2 1 4 5 3 1
 [661] 5 5 1 5 5 5 3 5 3 3 3 4 3 4 3 5 5 1 1 5 4 5 4 5 5 4 5 2 4 4 3 3 1 2 4 4 2 4 3 4 1 5 5 1 4 5 1 4 5 5 1 3 5 5 5 1 5 3 5 2 2 4 4 1 1 1
 [727] 1 5 5 4 1 3 4 4 3 5 4 5 4 4 5 5 5 4 4 5 1 1 1 5 5 5 3 3 3 3 3 2 2 2 2 2 1 1 1 1 1 1 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 2 2 3 1 5
 [793] 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 1 1 1 1 1 1 1 1 1 1 5 5 5 5 5 5 5 5 5 5 5 3 3 5 3 3 3 3 3 3 3 3 2 2 2 1 1 1 5 5 5
 [859] 5 3 3 3 3 3 3 2 2 2 2 1 1 1 1 5 5 5 5 3 3 5 5 5 3 3 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 3 3 3 3 3 3 3 3 2 2 5 5 5 5 5 5 5 5 5 3 5
 [925] 3 3 3 3 2 3 2 1 1 1 1 5 5 5 5 3 5 3 2 1 1 1 5 5 4 1 1 3 4 2 5 3 5 1 4 5 3 5 1 3 3 5 2 1 4 3 1 2 5 1 3 4 3 3 5 1 4 5 5 3 1 2 2 3 5 1
 [991] 5 4 5 5 4 5 4 5 5 5
 [ reached getOption("max.print") -- omitted 254 entries ]
Levels: 1 2 3 4 5
> 
```

- Applying KMeans to generate labels for the test records

```
> abalone.test_70_kmeans_k5 <- kmeans(abalone.test_70, centers = 5)
> abalone.test_70_kmeans_k5
K-means clustering with 5 clusters of sizes 318, 317, 207, 116, 296

Cluster means:
      length   diameter     height whole_weight shucked_weight viscera_weight shell_weight      rings
1  0.1710026  0.1894514  0.1288463   -0.0475749     -0.0910453    -0.05500327  0.007834075  0.3638872
2  0.8077212  0.8023654  0.7027813    0.7936385      0.7937993     0.79857983  0.736837676  0.3150711
3 -1.6760921 -1.6887508 -1.4483257   -1.3367148     -1.2850008    -1.29714335 -1.333473924 -1.1106823
4  1.4121190  1.4099220  1.6387466    2.0047140      1.9545189     1.87755929  1.847041489  0.9162815
5 -0.5092246 -0.5269056 -0.5138480   -0.7291073     -0.6890801    -0.70750899 -0.715099054 -0.4195200

Clustering vector:
   3    6   14   22   34   43   47   50   53   55   57   58   61   62   64   65   66   69   73   74   84   85   86   91   93   98  101
   1    5    1    3    4    3    5    1    5    5    5    5    5    5    5    5    5    5    2    2    2    1    2    1    2    5    3
 105  106  109  110  111  112  113  115  116  119  124  125  126  127  128  129  130  135  145  146  149  156  157  166  168  169  174
   2    1    1    5    5    5    5    1    1    1    3    3    3    3    3    4    4    3    5    5    3    1    2    4    4    4    1
 176  177  188  189  190  196  205  208  210  212  213  219  221  222  225  226  227  228  233  235  236  238  240  248  254  257  263
   3    3    2    2    2    1    5    5    3    3    5    5    5    5    5    5    5    3    4    5    3    3    3    3    2    2    1
 265  273  274  285  293  298  300  301  305  306  308  310  313  320  324  332  334  336  339  342  345  356  360  363  364  367  371
   3    2    4    1    2    3    3    5    5    3    4    1    2    3    3    5    3    2    2    2    1    4    4    4    2    1    4
 375  381  385  389  390  391  397  399  401  402  403  404  405  410  411  412  414  416  417  423  425  426  427  437  438  440  441
   2    2    1    1    5    5    1    1    5    1    5    1    5    1    2    1    1    2    2    1    3    2    2    3    5    1    3
 444  446  447  454  465  468  475  476  478  486  490  495  496  501  505  507  508  509  511  513  514  516  519  520  521  531  534
   5    1    1    1    3    2    1    1    2    1    2    2    1    1    2    1    2    1    2    1    3    3    3    3    3    1    5
 538  540  545  546  547  550  555  557  560  562  563  565  566  568  569  571  573  575  576  578  588  589  591  592  593  594  595
   3    5    5    3    3    1    5    1    1    1    5    1    3    5    3    1    2    2    2    2    1    5    1    5    1    4    1
 596  600  604  608  609  611  616  631  632  636  638  641  645  652  653  662  675  676  679  687  691  692  695  699  700  704  709
   1    1    5    5    3    5    5    5    5    3    3    3    5    3    5    1    1    1    1    1    1    3    3    5    5    5    5
 714  715  717  722  724  726  731  734  736  738  744  746  751  752  759  760  762  768  769  770  775  776  784  792  795  798  800
   3    3    3    1    1    1    1    1    1    1    1    1    1    5    2    1    1    3    1    1    1    1    3    5    2    5    5
 801  806  808  809  818  819  823  824  826  840  844  846  849  852  853  857  858  862  863  864  868  870  880  885  892  895  897
   5    5    1    5    3    3    3    3    3    5    1    1    1    1    2    2    1    2    2    2    2    2    2    2    4    3    3
 898  902  908  909  910  917  918  921  926  928  935  936  937  938  939  940  941  942  944  947  955  962  966  972  973  994  995
   3    3    3    3    3    3    5    5    5    5    5    5    5    5    5    5    5    5    5    5    5    5    1    5    1    2    2
 996  999 1009 1012 1013 1014 1016 1020 1024 1027 1031 1033 1034 1038 1039 1040 1045 1046 1052 1054 1059 1067 1069 1072 1077 1083 1084
   2    2    2    2    2    2    2    2    2    2    2    4    4    4    4    2    4    4    4    3    3    3    3    3    5    5    5
1092 1095 1097 1102 1106 1112 1115 1117 1124 1131 1133 1138 1142 1145 1168 1175 1176 1177 1181 1183 1187 1198 1200 1206 1207 1209 1216
   5    5    5    5    5    5    5    5    1    1    2    2    1    2    2    1    2    2    2    2    4    4    4    4    4    4    3
1218 1219 1220 1223 1224 1225 1228 1232 1237 1238 1239 1240 1248 1254 1259 1262 1263 1266 1267 1275 1280 1282 1285 1287 1289 1292 1293

 [ reached getOption("max.print") -- omitted 254 entries ]

Within cluster sum of squares by cluster:
[1] 523.8909 489.5858 217.9915 980.5023 240.6275
 (between_SS / total_SS =  77.0 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
> 
```

- Obtaining labels for the test data by using K-Means on the test data

```
> abalone.test_70_k5.labels <- abalone.test_70_kmeans_k5$cluster
> length(abalone.test_70_k5.labels)
[1] 1254
> abalone.test_70_k5.labels
   3    6   14   22   34   43   47   50   53   55   57   58   61   62   64   65   66   69   73   74   84   85   86   91   93   98  101
   1    5    1    3    4    3    5    1    5    5    5    5    5    5    5    5    5    5    2    2    2    1    2    1    2    5    3
 105  106  109  110  111  112  113  115  116  119  124  125  126  127  128  129  130  135  145  146  149  156  157  166  168  169  174
   2    1    1    5    5    5    5    1    1    1    3    3    3    3    3    4    4    3    5    5    3    1    2    4    4    4    1
 176  177  188  189  190  196  205  208  210  212  213  219  221  222  225  226  227  228  233  235  236  238  240  248  254  257  263
   3    3    2    2    2    1    5    5    3    3    5    5    5    5    5    5    5    3    4    5    3    3    3    3    2    2    1
 265  273  274  285  293  298  300  301  305  306  308  310  313  320  324  332  334  336  339  342  345  356  360  363  364  367  371
   3    2    4    1    2    3    3    5    5    3    4    1    2    3    3    5    3    2    2    2    1    4    4    4    2    1    4
 375  381  385  389  390  391  397  399  401  402  403  404  405  410  411  412  414  416  417  423  425  426  427  437  438  440  441
   2    2    1    1    5    5    1    1    5    1    5    1    5    1    2    1    1    2    2    1    3    2    2    3    5    1    3
 444  446  447  454  465  468  475  476  478  486  490  495  496  501  505  507  508  509  511  513  514  516  519  520  521  531  534
   5    1    1    1    3    2    1    1    2    1    2    2    1    1    2    1    2    1    2    1    3    3    3    3    3    1    5
 538  540  545  546  547  550  555  557  560  562  563  565  566  568  569  571  573  575  576  578  588  589  591  592  593  594  595
   3    5    5    3    3    1    5    1    1    1    5    1    3    5    3    1    2    2    2    2    1    5    1    5    1    4    1
 596  600  604  608  609  611  616  631  632  636  638  641  645  652  653  662  675  676  679  687  691  692  695  699  700  704  709
   1    1    5    5    3    5    5    5    5    3    3    3    5    3    5    1    1    1    1    1    1    3    3    5    5    5    5
 714  715  717  722  724  726  731  734  736  738  744  746  751  752  759  760  762  768  769  770  775  776  784  792  795  798  800
   3    3    3    1    1    1    1    1    1    1    1    1    1    5    2    1    1    3    1    1    1    1    3    5    2    5    5
 801  806  808  809  818  819  823  824  826  840  844  846  849  852  853  857  858  862  863  864  868  870  880  885  892  895  897
   5    5    1    5    3    3    3    3    3    5    1    1    1    1    2    2    1    2    2    2    2    2    2    2    4    3    3
 898  902  908  909  910  917  918  921  926  928  935  936  937  938  939  940  941  942  944  947  955  962  966  972  973  994  995
   3    3    3    3    3    3    5    5    5    5    5    5    5    5    5    5    5    5    5    5    5    5    1    5    1    2    2
 996  999 1009 1012 1013 1014 1016 1020 1024 1027 1031 1033 1034 1038 1039 1040 1045 1046 1052 1054 1059 1067 1069 1072 1077 1083 1084
   2    2    2    2    2    2    2    2    2    2    2    4    4    4    4    2    4    4    4    3    3    3    3    3    5    5    5
1092 1095 1097 1102 1106 1112 1115 1117 1124 1131 1133 1138 1142 1145 1168 1175 1176 1177 1181 1183 1187 1198 1200 1206 1207 1209 1216
   5    5    5    5    5    5    5    5    1    1    2    2    1    2    2    1    2    2    2    2    4    4    4    4    4    4    3
```

- Linear modeling by using glm function of R.

```
>
> # linear modelling using GLM
> abalone.train_70.glm <- glm(formula = rings ~ length + diameter + height + whole_weight + shucked_weight +  viscera_weight + shell_wei
ght, family = gaussian, data = abalone.train_70)
> summary(abalone.train_70.glm)

Call:
glm(formula = rings ~ length + diameter + height + whole_weight +
    shucked_weight + viscera_weight + shell_weight, family = gaussian,
    data = abalone.train_70)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5111  -0.4220  -0.1166   0.2718   4.3572

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.006158   0.012720   0.484 0.628339
length         -0.044591   0.080235  -0.556 0.578426
diameter        0.315420   0.081512   3.870 0.000111 ***
height          0.345964   0.036348   9.518  < 2e-16 ***
whole_weight    1.346626   0.129331  10.412  < 2e-16 ***
shucked_weight -1.344865   0.066020 -20.371  < 2e-16 ***
viscera_weight -0.358406   0.052756  -6.794 1.32e-11 ***
shell_weight    0.322172   0.056853   5.667 1.60e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.4723417)

    Null deviance: 2951.3  on 2922  degrees of freedom
Residual deviance: 1376.9  on 2915  degrees of freedom
AIC: 6112.7

Number of Fisher Scoring iterations: 2

>
```

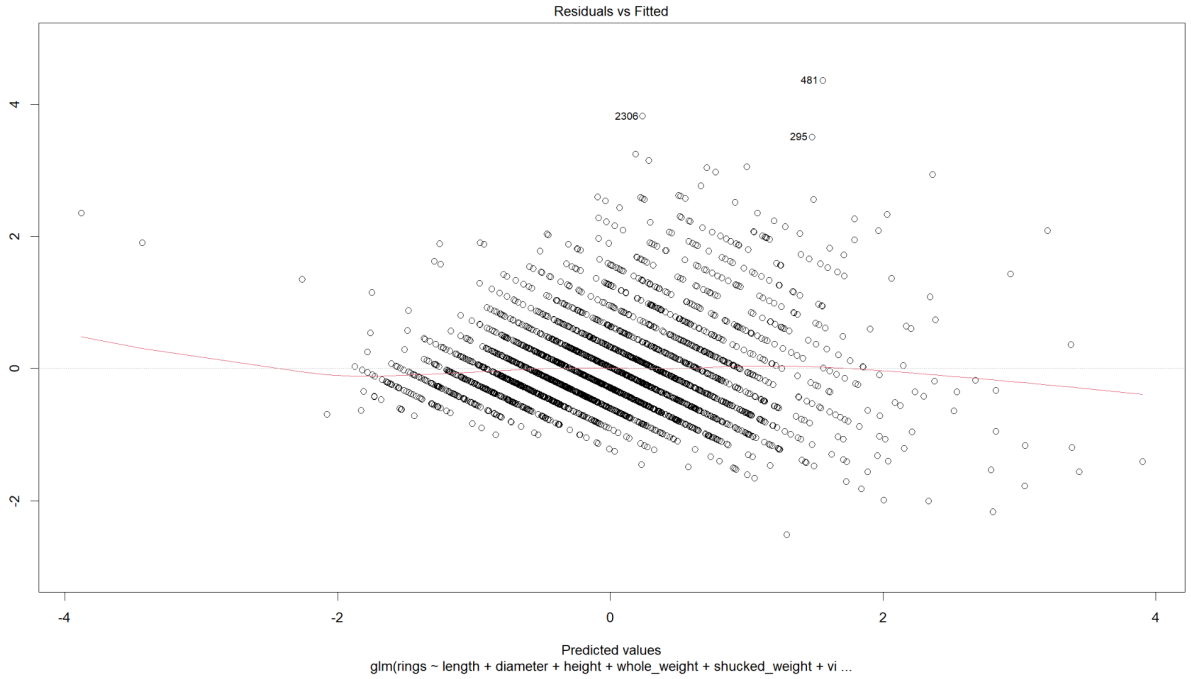
- Applying anova to the training data to get the deviance of the data

```
>
>
> # anova - analysis of variance on the model result
> abalone.train_70.glm.anova <- anova(abalone.train_70.glm, test = "Chisq")
> abalone.train_70.glm.anova
Analysis of Deviance Table

Model: gaussian, link: identity

Response: rings

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                          2922     2951.3
length          1   918.29      2921     2033.0 < 2.2e-16 ***
diameter        1    69.50      2920     1963.5 < 2.2e-16 ***
height          1   132.12      2919     1831.4 < 2.2e-16 ***
whole_weight    1     3.30      2918     1828.1  0.008244 **
shucked_weight  1   398.24      2917     1429.8 < 2.2e-16 ***
viscera_weight  1    37.78      2916     1392.0 < 2.2e-16 ***
shell_weight    1    15.17      2915     1376.9 1.455e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
>
>
```

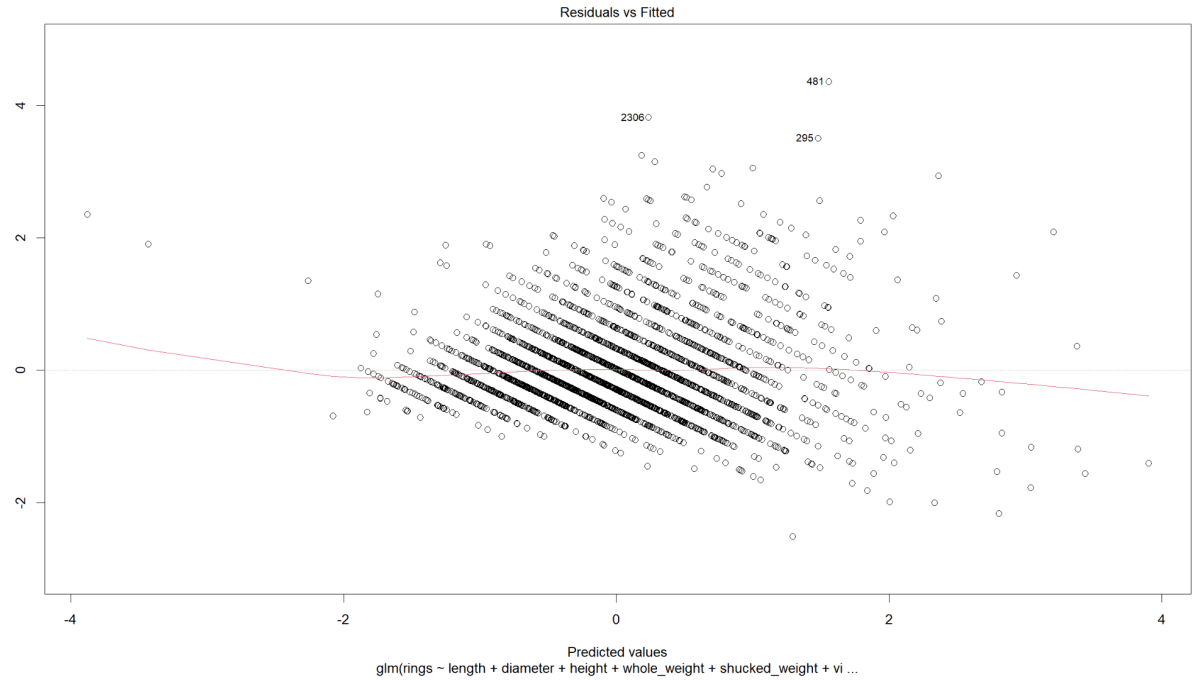
- Plotting graph

```
>
>
> plot(abalone.train_70.glm)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
>
>
>
```

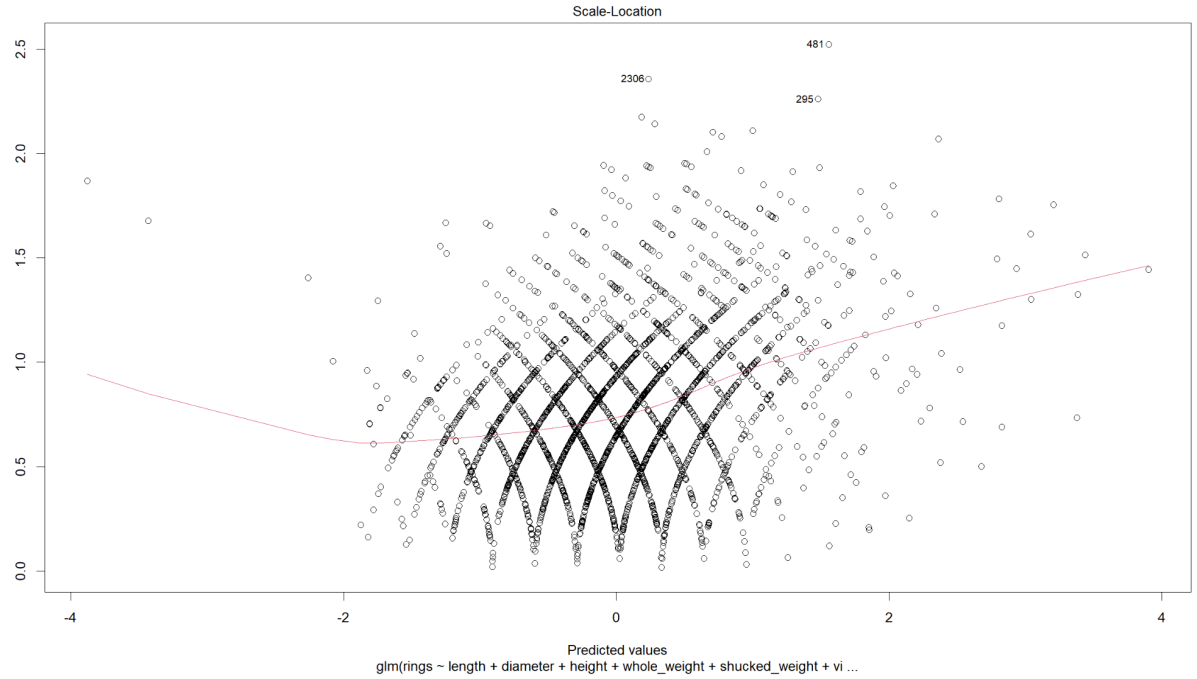
- Residual values vs Fitted values Plot



- Normal Q-Q Plot

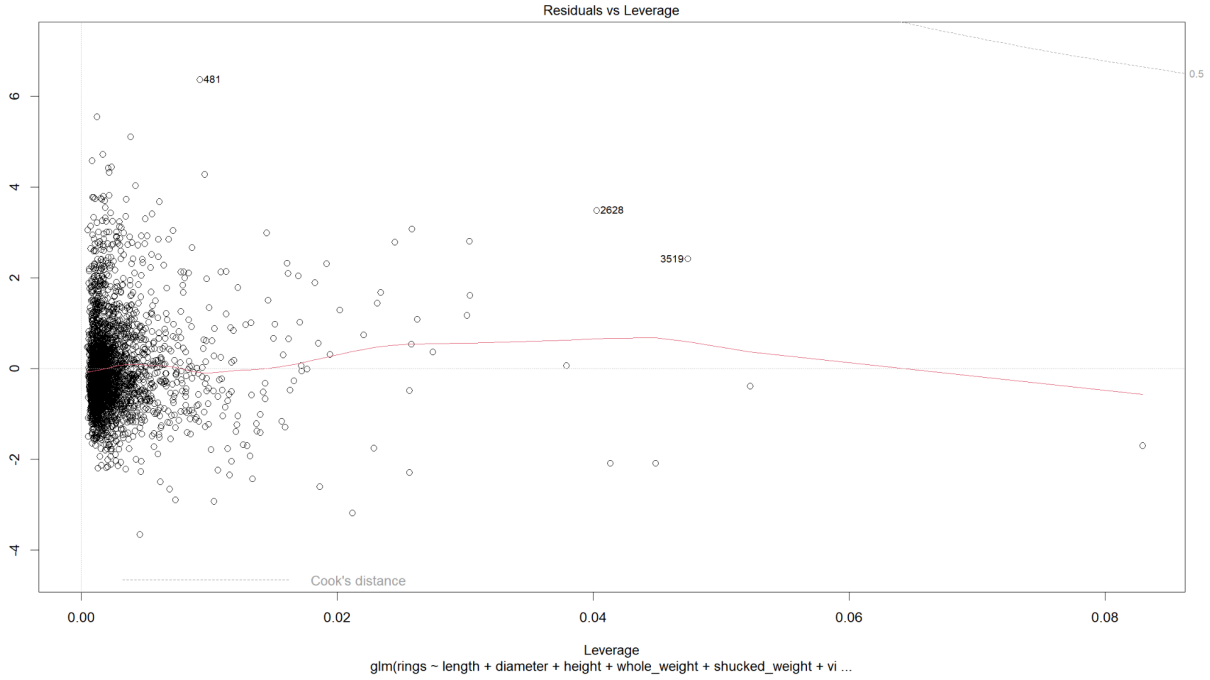


- Scale-Location Plot



-

- Residuals vs Leverage Plot



- Using predict function in R

```
>
> # predict
> abalone.test_70.predict <- predict(abalone.train_70.glm, newdata = abalone.test_70)
> length(abalone.test_70.predict)
[1] 1254
> summary(abalone.test_70.predict)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.840315 -0.494013 -0.037437 -0.005171  0.385751  7.544918
>
```

- Confidence intervals: Indicating the probability that the true parameter values lies within the range.

```
>
>
>
> # confidence intervals
> confint(abalone.train_70.glm)
Waiting for profiling to be done...
                    2.5 %      97.5 %
(Intercept)    -0.01877274  0.03108867
length         -0.20184914  0.11266788
diameter        0.15565992  0.47517991
height          0.27472396  0.41720438
whole_weight    1.09314051  1.60011067
shucked_weight -1.47426182 -1.21546872
viscera_weight -0.46180597 -0.25500544
shell_weight    0.21074271  0.43360187
>
>
>
> |
```

- Comparing the predicted values

```
>
>
> # comparing actual vs prediction
> abalone.test_70.predict.k5 <- kmeans(abalone.test_70.predict, centers = 5)
> abalone.test_70.predict.k5
K-means clustering with 5 clusters of sizes 429, 34, 226, 396, 169

Cluster means:
       [,1]
1  0.2469880
2  2.2482062
3 -0.9909026
4 -0.3351495
5  0.9927889

Clustering vector:
   3    6   14   22   34   43   47   50   53   55   57   58   61   62   64   65   66   69   73   74   84   85   86   91   93   98
   1    4    1    4    1    3    3    1    4    3    4    4    4    4    3    4    4    4    5    5    2    5    2    1    5    4
 101  105  106  109  110  111  112  113  115  116  119  124  125  126  127  128  129  130  135  145  146  149  156  157  166  168
   3    1    1    1    4    4    4    3    4    4    1    3    3    3    3    3    2    2    3    1    4    3    1    5    5    2
 169  174  176  177  188  189  190  196  205  208  210  212  213  219  221  222  225  226  227  228  233  235  236  238  240  248
   2    4    3    3    5    1    1    1    4    4    3    3    4    1    4    1    4    1    4    3    2    4    3    3    3    3
 254  257  263  265  273  274  285  293  298  300  301  305  306  308  310  313  320  324  332  334  336  339  342  345  356  360
   5    5    1    3    2    2    5    5    3    4    3    4    3    2    1    5    3    3    4    3    1    1    2    1    2    5
 363  364  367  371  375  381  385  389  390  391  397  399  401  402  403  404  405  410  411  412  414  416  417  423  425  426
   5    1    1    5    5    5    4    1    4    4    4    1    4    1    4    1    4    4    5    1    1    5    5    1    3    5
 427  437  438  440  441  444  446  447  454  465  468  475  476  478  486  490  495  496  501  505  507  508  509  511  513  514
```

```
3181 3184 3186 3187 3189 3191 3193 3195 3198 3203 3208 3209 3213 3214 3215 3219 3221 3222 3223 3228 3229 3231 3237 3240 3245 3249
   4    1    1    4    2    3    4    1    3    2    1    3    5    1    1    5    1    4    5    4    4    1    4    5    5    1
3254 3256 3259 3261 3267 3268 3272 3273 3275 3276 3277 3278
   4    3    4    1    4    4    5    4    1    4    1    1
 [ reached getOption("max.print") -- omitted 254 entries ]

Within cluster sum of squares by cluster:
[1] 14.02199 31.58632 14.75412 10.98265 11.71965
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
> |
```

- Comparing results of kmeans and knn for 5 clusters using the crosstable functionality provided by gmodels package.

```
Total Observations in Table:  1254


                                 | abalone.test_70_kmeans_k5$cluster
abalone.test_70.predict.k5$cluster |         1 |         2 |         3 |         4 |         5 | Row Total |
-----------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                 1 |         0 |        37 |       174 |        39 |       179 |       429 |
                                   |    70.816 |     0.182 |    39.088 |    38.283 |    45.899 |           |
                                   |     0.000 |     0.086 |     0.406 |     0.091 |     0.417 |     0.342 |
                                   |     0.000 |     0.319 |     0.547 |     0.132 |     0.565 |           |
                                   |     0.000 |     0.030 |     0.139 |     0.031 |     0.143 |           |
-----------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                 2 |         0 |        26 |         1 |         0 |         7 |        34 |
                                   |     5.612 |   166.080 |     6.738 |     8.026 |     0.296 |           |
                                   |     0.000 |     0.765 |     0.029 |     0.000 |     0.206 |     0.027 |
                                   |     0.000 |     0.224 |     0.003 |     0.000 |     0.022 |           |
                                   |     0.000 |     0.021 |     0.001 |     0.000 |     0.006 |           |
-----------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                 3 |       187 |         3 |         4 |        31 |         1 |       226 |
                                   |   600.657 |    15.336 |    49.590 |     9.361 |    55.148 |           |
                                   |     0.827 |     0.013 |     0.018 |     0.137 |     0.004 |     0.180 |
                                   |     0.903 |     0.026 |     0.013 |     0.105 |     0.003 |           |
                                   |     0.149 |     0.002 |     0.003 |     0.025 |     0.001 |           |
-----------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                 4 |        20 |        10 |        91 |       226 |        49 |       396 |
                                   |    31.488 |    19.361 |     0.884 |   187.895 |    26.090 |           |
                                   |     0.051 |     0.025 |     0.230 |     0.571 |     0.124 |     0.316 |
                                   |     0.097 |     0.086 |     0.286 |     0.764 |     0.155 |           |
                                   |     0.016 |     0.008 |     0.073 |     0.180 |     0.039 |           |
-----------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                 5 |         0 |        40 |        48 |         0 |        81 |       169 |
                                   |    27.897 |    37.980 |     0.617 |    39.892 |    34.297 |           |
                                   |     0.000 |     0.237 |     0.284 |     0.000 |     0.479 |     0.135 |
                                   |     0.000 |     0.345 |     0.151 |     0.000 |     0.256 |           |
                                   |     0.000 |     0.032 |     0.038 |     0.000 |     0.065 |           |
-----------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                     Column Total |       207 |       116 |       318 |       296 |       317 |      1254 |
                                   |     0.165 |     0.093 |     0.254 |     0.236 |     0.253 |           |
-----------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
```

- Confusion Matrix

```
>
>
> # confusion matrix
> abalone.test_70.cm_k5 <- abalone.test_70.ct.k5$t
> abalone.test_70.cm_k5
   y
x     1   2   3   4   5
  1   0  37 174  39 179
  2   0  26   1   0   7
  3 187   3   4  31   1
  4  20  10  91 226  49
  5   0  40  48   0  81
> |
```

- Calculating the metrices for k=5

```
>
>
>
>
> # calculate the metrics
> abalone.test_70.accuracy <- sum(diag(abalone.test_70.cm_k5)) / sum(abalone.test_70.cm_k5)
> abalone.test_70.precision <- diag(abalone.test_70.cm_k5) / colSums(abalone.test_70.cm_k5)
> abalone.test_70.recall <- diag(abalone.test_70.cm_k5) / rowSums(abalone.test_70.cm_k5)
> abalone.test_70.specificity <- sapply(1:nrow(abalone.test_70.cm_k5), function(i){
+     TN <- sum(abalone.test_70.cm_k5[-i, -i])
+     FP <- sum(abalone.test_70.cm_k5[-i, i])
+     TN / (TN + FP)
+ })
> abalone.test_70.error <- 1 - abalone.test_70.accuracy
> # print results
> abalone.test_70.accuracy
[1] 0.26874
> abalone.test_70.precision
         1          2          3          4          5
0.00000000 0.22413793 0.01257862 0.76351351 0.25552050
> abalone.test_70.recall
         1          2          3          4          5
0.00000000 0.76470588 0.01769912 0.57070707 0.47928994
> abalone.test_70.specificity
[1] 0.7490909 0.9262295 0.6945525 0.9184149 0.7824885
> abalone.test_70.error
[1] 0.73126
>
```

Implementation for k=5 with 60-40 split

- Creating training labels with k-means

```
> abalone.train_60_k5 <- kmeans(abalone.train_60, centers = 5)
> abalone.train_60_k5
K-means clustering with 5 clusters of sizes 762, 506, 210, 296, 732

Cluster means:
      length   diameter     height whole_weight shucked_weight viscera_weight shell_weight       rings
1 -0.2910802 -0.2913289 -0.2998378   -0.5194334     -0.5023629     -0.5085398   -0.4879177 -0.15486200
2 -1.5422842 -1.5466960 -1.3326708   -1.2720805     -1.2095051     -1.2462021   -1.2655775 -1.03862552
3  0.5173240  0.5726835  0.7037546    0.5077470      0.1976805      0.4238822    0.7569648  2.06760733
4  1.3397098  1.3347114  1.2570379    1.7821849      1.7563228      1.7275968    1.7141681  0.64821793
5  0.6368337  0.6224352  0.4493848    0.5021767      0.5547743      0.5158113    0.4219416  0.00658575

Clustering vector:
1089  870 2407 3354 3280  754 2813 2581 2468 3927 3767 1356   55 1280 3079 1557 3641  603 3951  819 1467 3346  910 3814 3445  986 3760
   1    5    5    1    3    5    2    1    3    3    5    5    2    1    4    2    1    1    1    2    5    1    2    2    1    5    1
1019 2809  444  666 3742  938 2866 1526  162 1962  843 1501 2786 1468 3957 1358  132 2994 3180 1700  125 2385 3082 3325 3386   65 1733
   5    4    2    2    4    1    2    4    5    4    1    5    5    1    1    5    1    4    2    4    2    1    4    2    1    1    4
 333  956 1684 3360 3756 2232 2464 2035 3436  299 4062 2594 2403 3171 3887 3090 3238 3363 2422 1482 3690 2882 3298 1618 3684 1193 3661
   2    1    5    3    1    1    3    3    2    1    5    5    1    3    1    1    3    2    3    5    5    1    3    1    5    4    1
1783  398  217 3832 1357  999 2788 1289 3798 1641 4002 1612  809 1161 3067 3154 2463  905 3176 2298 3801  124  490  482 3042 2490  165
   5    1    1    5    5    5    5    1    4    5    1    1    1    5    5    1    2    2    1    1    4    2    3    3    5    1    4
 362  706 1300  614 1855 1380  212  746  980 3880 3189 1266 1296 3263 2306 4012  528  788 2286 3137 3473  370  777 4006 4000 1768 1256
   5    1    1    1    1    5    2    3    5    3    4    1    1    3    3    5    3    1    1    1    2    4    1    1    2    1    2
 601 1226 3058 1114 1456 1534 3872 3091 3560 2944 1739   79  903 1372 3459 1980 3410 2867 3709 1682  537   18  993  371 1663 3550 2692
   3    2    5    1    1    2    1    1    5    5    4    1    2    5    5    4    2    2    4    5    1    1    5    4    5    1    5
3978  275 4046 3584 3221  164 3272 1652 3056 1607 2081 1776 3234 2024  269 2147 3184 3693   32 3340 3045  772  729  240 2011 1331 4166
   1    3    5    5    1    4    3    5    5    1    5    1    5    1    1    1    5    5    4    5    5    1    1    2    1    5    2
1278  368 3969  372 3417 3921 3088 3302 2533 1794 1863 1344  543 1083  686 3003  761 3471 1168  222  295  636  612 4120 3780  586 4128
   1    5    2    4    5    2    2    4    5    5    1    5    1    1    3    4    3    4    5    1    3    2    2    2    5    1    1
3905 2553  733 1636 2344  331 3846 3889 1164 2359 1565 2817 1219 3014 2808 2780 1026 1736  735 2018 3054 1716 1841 3904 3558 2724 1272
   3    2    3    1    2    1    1    5    5    3    1    2    2    2    4    5    4    5    3    1    5    5    1    3    1    2    1
 583 2138 4056 1725  506 1714 2933 2290 3284  272  408 1921 2438 4009  707 1862  123 1680 2082 3342 3975 1734 1815 3029 2177 2774 1604
   3    5    5    4    3    5    5    5    5    3    3    3    5    2    5    2    1    3    5    5    1    1    5    4    1    3    5
1947 1579 2031 2910 2590 3778  916 3092 2368 3073 3616  842 1516 1721 2894 1345 3314 1447 1675 1014 2471 1309 2990 1243 4172   69 2262
   5    1    5    5    5    5    2    1    3    4    5    1    4    5    5    5    1    2    5    5    5    1    5    2    5    1    5
3002 1198  812 2600 1184 3934 3881 4064 3425 2049 2789 1803 1192 1072 1933 1553 1902 2187 3520  241 3642 1738 1842 1166 3852  224  256
   5    4    3    5    5    2    2    5    5    1    5    5    4    2    5    2    5    1    4    3    1    4    2    5    1    1    3
4134 3321  315 3967 2091 2075 1252  745 4144 1265 1868 4089 1424 2686  796 2428 2125  967 3446 3587 2519 4082 1548 2392 1812 3105 2753
   5    3    3    2    4    5    2    3    4    1    1    5    4    5    3    2    2    1    1    5    1    5    2    2    4    4    1
1785 3940 3301  912 2851  783 1753 4171 3865 3824  926 3753 4069  525 4161 1637 3329 2740 3788 3316  287 1773 3038 3490 3114 3710 3333
   1    3    3    2    5    5    4    1    2    5    2    1    2    2    5    5    5    2    5    2    1    1    5    1    2    4    1
2691  109  845  719 1286  193 2122 3255 2930 3737 3849 1556 1490 1167  474  551 2435 1880 1853 1299 1445 3375 3283  822  263 2268 3982
   5    1    1    2    1    3    1    1    5    5    4    2    5    5    1    3    3    1    1    1    2    1    5    2    5    3    5
2053  152 2744 3187 2730 3617  116  804   27 2239 4067 1417 2117 2741 4005 2100 3074  875  513 1133 3487 3072  882 2596 2810 2752 2722
   1    5    1    1    2    3    1    2    5    1    2    5    2    1    1    1    5    5    1    5    5    5    5    5    4    1    2
2448 2108 3728 1448 1789  374 3028 2674  427 1169 4057  264 2245 2828 1723 1631 2287 1431 3522  527 2981  297 1999 4175 2008 1353 2507
   2    4    1    1    5    4    1    5    3    5    5    2    2    1    5    5    1    2    2    2    1    2    2    5    2    5    2
 714 2865  234 3183 2855 3228 2449 3564 3725  675 1106 3278  701 3015 1069  671 3783 3362 1049 1227 2026 1913 3024 2745 1780 2683  890
   2    2    2    5    4    1    2    5    1    3    1    1    2    2    2    1    5    2    4    2    5    5    1    1    5    5    4
1763 1444 3463 1802 3604 1598  913 3285 3307  462  643  180  846 3040  113 1381 2078 2339  206 3818  779 1537  841 2107 3547 2499  436
   4    1    5    5    2    1    2    5    2    5    3    5    5    5    2    5    5    4    1    1    1    2    1    3    1    3    1
```

```
Within cluster sum of squares by cluster:
[1] 495.6957 962.8783 496.2624 882.2464 924.6583
 (between_SS / total_SS =  81.0 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
> 
```

- Then the cluster vector is passed to KNN

  The output is a vector which contains the predicted values for the test dataset.

```
> abalone.test_60_k5 <- knn(abalone.train_60, abalone.test_60, abalone.train_60_k5$cluster, k = 5)
> abalone.test_60_k5
   [1] 2 1 2 2 2 2 1 2 1 3 3 3 2 2 5 5 5 5 2 3 1 3 3 3 1 1 3 2 3 1 1 2 5 3 2 1 5 5 5 5 2 5 5 5 5 2 5 4 4 2 1 2 2 2 2 1 1 2 3 5 3 3
  [67] 3 1 3 2 5 1 3 5 5 4 2 4 4 2 3 3 3 5 5 5 5 5 5 5 2 2 5 2 2 2 1 1 1 2 3 1 2 1 1 1 1 1 1 1 3 2 4 1 3 3 3 3 3 5 5 3 2 3 1 4 3 1 2 2 2 5 3
 [133] 2 3 4 2 5 2 5 5 3 1 3 3 3 1 2 1 5 4 4 4 5 5 5 2 1 1 1 2 1 3 2 2 2 1 2 1 2 2 5 5 4 3 3 2 5 2 2 1 3 1 2 5 2 4 4 2 5 2 1 1 3 3 3 1 1 4
 [199] 2 2 5 2 2 5 5 5 5 5 5 2 2 2 1 3 3 3 3 3 3 3 1 1 3 3 3 3 2 2 1 1 2 2 2 2 3 1 2 5 5 5 2 1 2 2 3 2 2 2 2 2 5 1 3 1 3 3 3 2 2 2 3 2 3 1
 [265] 1 1 1 3 3 5 4 2 1 1 1 2 2 2 3 1 1 2 2 2 3 3 2 1 1 1 1 3 3 5 1 1 1 2 2 2 2 2 3 4 2 2 4 5 5 2 2 2 2 1 2 2 2 2 3 2 1 1 1 1 2 3 3 3 3 3
 [331] 3 3 3 3 1 1 1 1 1 2 2 2 2 2 5 5 5 5 5 5 5 5 5 5 5 5 4 5 4 4 4 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 2 2 2 1 2 2 2 5
 [397] 2 2 2 5 2 5 5 5 5 5 5 5 5 5 5 5 4 4 4 5 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 2 2 2 2 2 2 2 2 5 2 2 2 2 2
 [463] 2 2 2 5 5 5 5 5 5 5 5 5 4 5 5 4 4 4 4 4 5 4 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3 1 3 3 1 1 1 1 1 1 1 1 1 1 2 2 2 1 1 2 1 2 2 2 2 2 2 2 2 2
 [529] 2 2 2 2 5 5 2 2 5 5 5 5 2 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 5 5 4 4 4 4 4 4 4 3 3 1 3 1 3 3 1 1 1 1 1 1 1 1 1 2 2 2 2 2 5 2 5 5 5 5
 [595] 5 5 5 4 4 5 5 4 5 2 5 5 4 5 4 4 4 4 4 3 3 3 3 3 3 3 3 3 3 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 2 1 2 2 2 2 1 2 2 2 2 2 2
 [661] 2 5 5 2 2 2 5 2 2 2 5 2 2 2 2 5 5 5 5 5 5 5 5 5 5 5 5 5 4 5 5 4 5 5 5 5 5 4 5 5 5 4 4 4 4 4 4 4 4 4 4 3 1 1 1 1 2 2 2 2 2 2 2 2 5 5 5 5 5
 [727] 5 5 4 5 4 4 4 4 4 3 3 3 3 3 3 1 1 1 1 1 1 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 5 5 2 2 2 2 2 5 2 5 5 5 5 5 5 5 5 5 5 5 5 4 5 5 5 5 5 4 5
 [793] 4 4 5 4 4 3 3 3 3 3 3 3 3 3 1 1 1 1 2 2 2 3 3 3 3 3 1 5 1 1 2 1 2 2 2 2 5 4 5 1 2 1 2 1 2 1 5 1 3 3 1 1 3 3 5 2 1 2 5 1 3 3 1 4 1 3 2
 [859] 5 4 5 4 2 3 3 3 3 2 5 2 2 2 2 5 3 4 2 5 1 3 3 4 5 4 5 5 2 2 5 5 5 2 2 3 3 2 2 2 2 5 5 1 1 1 1 1 5 5 5 5 2 1 5 2 2 5 2 4 5 5 5 1 1 1
 [925] 1 3 3 2 2 2 1 2 5 2 1 3 2 1 2 1 1 2 2 1 2 4 4 5 5 2 3 2 2 2 2 5 5 5 2 1 3 1 2 3 3 2 2 3 5 5 5 5 2 2 3 5 5 3 1 1 3 3 2 2 2 2 2 1 2 1
 [991] 1 2 2 1 1 3 3 3 2 2
 [ reached getOption("max.print") -- omitted 671 entries ]
Levels: 1 2 3 4 5
> 
```

- Applying KMeans to generate labels for the test records

```
> abalone.test_60_kmeans_k5 <- kmeans(abalone.test_60, centers = 5)
> abalone.test_60_kmeans_k5
K-means clustering with 5 clusters of sizes 209, 495, 156, 506, 305

Cluster means:
      length   diameter     height whole_weight shucked_weight viscera_weight shell_weight       rings
1  1.3080244  1.3108165  1.3688254    1.7718565     1.71382314      1.7370369    1.6632955  0.68391933
2  0.6604131  0.6599413  0.5184592    0.5525736     0.58620752      0.5586840    0.4905387  0.01806193
3  0.3153132  0.3540066  0.4537061    0.2500726    -0.01052174      0.1844111    0.5074406  1.89741838
4 -0.2567183 -0.2663112 -0.2882327   -0.4931494    -0.47059406     -0.4780755   -0.4724737 -0.25403754
5 -1.6023820 -1.5981739 -1.3562875   -1.2970021    -1.25006218     -1.2666341   -1.2904151 -1.00549356

Clustering vector:
   3    4    7    8   10   11   12   14   15   19   21   22   24   28   29   31   33   37   39   40   41   43   45   46   47   48   49
   4    4    3    3    3    4    4    4    4    5    5    5    4    2    3    2    3    3    2    5    4    5    5    5    4    4    5
  50   52   53   54   56   59   60   62   63   68   70   71   72   73   74   75   76   78   80   84   85   86   91   93   95   96   97
   2    5    4    4    4    5    4    4    4    2    5    3    5    3    2    2    3    2    3    3    3    2    3    2    1    1    4
  99  102  104  106  107  110  111  115  120  131  134  137  138  139  140  141  143  146  150  154  157  158  160  166  168  174  175
   4    3    4    4    2    4    4    4    5    3    5    5    5    4    5    2    1    4    5    2    2    1    3    1    1    4    5
 176  177  184  187  188  189  190  192  195  197  198  202  203  204  205  207  208  209  210  214  216  218  219  220  221  223  225
   5    5    2    2    2    2    2    2    4    4    3    3    4    3    4    5    4    3    5    4    4    5    4    4    4    4    4
 226  228  232  233  235  242  245  247  248  250  254  259  265  267  268  270  276  282  283  285  286  290  294  296  302  304  308
   4    5    3    3    4    5    5    5    5    5    3    1    5    3    5    4    1    5    4    3    2    4    3    5    3    5    1
 310  311  312  313  317  320  323  324  325  329  332  347  349  351  356  357  360  369  377  380  383  384  386  388  389  391  395
   2    3    3    3    2    5    5    5    5    5    5    4    5    2    1    1    1    2    2    2    4    4    4    4    4    4    5
 397  399  400  405  406  407  409  412  416  419  421  424  425  429  430  433  434  435  437  439  446  449  450  452  453  454  455
   4    4    2    4    2    4    2    3    3    3    1    5    5    3    3    3    3    4    5    5    2    2    3    1    1    2    2
 456  459  461  463  464  466  473  477  479  486  487  488  489  492  493  494  495  497  498  499  500  501  509  512  515  516  518
   2    5    5    5    5    5    4    4    1    3    2    3    4    3    2    3    3    2    3    2    2    2    2    4    5    5    5
```

```
Within cluster sum of squares by cluster:
[1] 1161.7788  557.5648  332.0865  547.5317  408.7424
 (between_SS / total_SS =  77.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
> 
```

- Obtaining labels for the test data by using K-Means on the test data

```
> abalone.test_60_k5.labels <- abalone.test_60_kmeans_k5$cluster
> length(abalone.test_60_k5.labels)
[1] 1671
> abalone.test_60_k5.labels
   3    4    7    8   10   11   12   14   15   19   21   22   24   28   29   31   33   37   39   40   41   43   45   46   47   48   49
   4    4    3    3    3    4    4    4    4    5    5    5    4    2    3    2    3    3    2    5    4    5    5    5    4    4    5
  50   52   53   54   56   59   60   62   63   68   70   71   72   73   74   75   76   78   80   84   85   86   91   93   95   96   97
   2    5    4    4    4    5    4    4    4    2    5    3    5    3    2    2    3    2    3    3    3    2    3    2    1    1    4
  99  102  104  106  107  110  111  115  120  131  134  137  138  139  140  141  143  146  150  154  157  158  160  166  168  174  175
   4    3    4    4    2    4    4    4    5    3    5    5    5    4    5    2    1    4    5    2    2    1    3    1    1    4    5
 176  177  184  187  188  189  190  192  195  197  198  202  203  204  205  207  208  209  210  214  216  218  219  220  221  223  225
   5    5    2    2    2    2    2    2    4    4    3    3    4    3    4    5    4    3    5    4    4    5    4    4    4    4    4
 226  228  232  233  235  242  245  247  248  250  254  259  265  267  268  270  276  282  283  285  286  290  294  296  302  304  308
   4    5    3    3    4    5    5    5    5    5    3    1    5    3    5    4    1    5    4    3    2    4    3    5    3    5    1
 310  311  312  313  317  320  323  324  325  329  332  347  349  351  356  357  360  369  377  380  383  384  386  388  389  391  395
   2    3    3    3    2    5    5    5    5    5    5    4    5    2    1    1    1    2    2    2    4    4    4    4    4    4    5
 397  399  400  405  406  407  409  412  416  419  421  424  425  429  430  433  434  435  437  439  446  449  450  452  453  454  455
   4    4    2    4    2    4    2    3    3    3    1    5    5    3    3    3    3    4    5    5    2    2    3    1    1    2    2
 456  459  461  463  464  466  473  477  479  486  487  488  489  492  493  494  495  497  498  499  500  501  509  512  515  516  518
   2    5    5    5    5    5    4    4    1    3    2    3    4    3    2    3    3    2    3    2    2    2    2    4    5    5    5
 519  523  524  530  535  536  538  540  545  548  550  553  554  555  556  557  564  565  566  571  574  576  580  581  582  585  587
   5    5    5    5    4    4    5    5    5    5    2    3    4    4    4    3    4    4    5    4    3    2    3    2    3    4    4
 591  592  593  595  596  597  598  599  602  613  615  618  620  621  622  623  626  628  629  635  645  646  648  649  654  656  658
   3    5    3    4    3    4    3    2    5    5    4    5    5    5    3    4    4    5    3    5    4    4    4    4    5    5    3
 661  662  665  667  670  673  677  679  680  682  683  684  685  687  695  698  702  703  704  709  712  717  718  722  725  726  728
   1    2    3    4    4    3    4    3    5    4    4    3    4    3    5    5    4    4    4    5    5    5    5    3    4    3    4
 731  734  736  737  741  742  747  748  751  755  756  757  762  766  769  770  774  775  776  778  780  784  791  793  801  803  806
   4    4    4    4    3    5    1    4    4    3    1    3    3    2    2    3    4    4    4    4    4    5    2    4    4    4    5
 811  813  814  815  818  820  823  825  826  827  832  833  835  836  839  847  848  850  853  854  857  859  860  865  866  868  869
   4    5    5    5    5    5    5    5    5    5    5    4    4    4    4    4    2    2    2    2    2    2    2    2    2    2    2
```

- Linear modeling by using glm function of R.

```
> abalone.train_60.glm <- glm(formula = rings ~ length + diameter + height + whole_weight + shucked_weight +  viscera_weight + shell_weight, fa
mily = gaussian, data = abalone.train_70)
> summary(abalone.train_60.glm)

Call:
glm(formula = rings ~ length + diameter + height + whole_weight +
    shucked_weight + viscera_weight + shell_weight, family = gaussian,
    data = abalone.train_70)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5111  -0.4220  -0.1166   0.2718   4.3572

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.006158   0.012720   0.484 0.628339
length         -0.044591   0.080235  -0.556 0.578426
diameter        0.315420   0.081512   3.870 0.000111 ***
height          0.345964   0.036348   9.518  < 2e-16 ***
whole_weight    1.346626   0.129331  10.412  < 2e-16 ***
shucked_weight -1.344865   0.066020 -20.371  < 2e-16 ***
viscera_weight -0.358406   0.052756  -6.794 1.32e-11 ***
shell_weight    0.322172   0.056853   5.667 1.60e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.4723417)

    Null deviance: 2951.3  on 2922  degrees of freedom
Residual deviance: 1376.9  on 2915  degrees of freedom
AIC: 6112.7

Number of Fisher Scoring iterations: 2
```

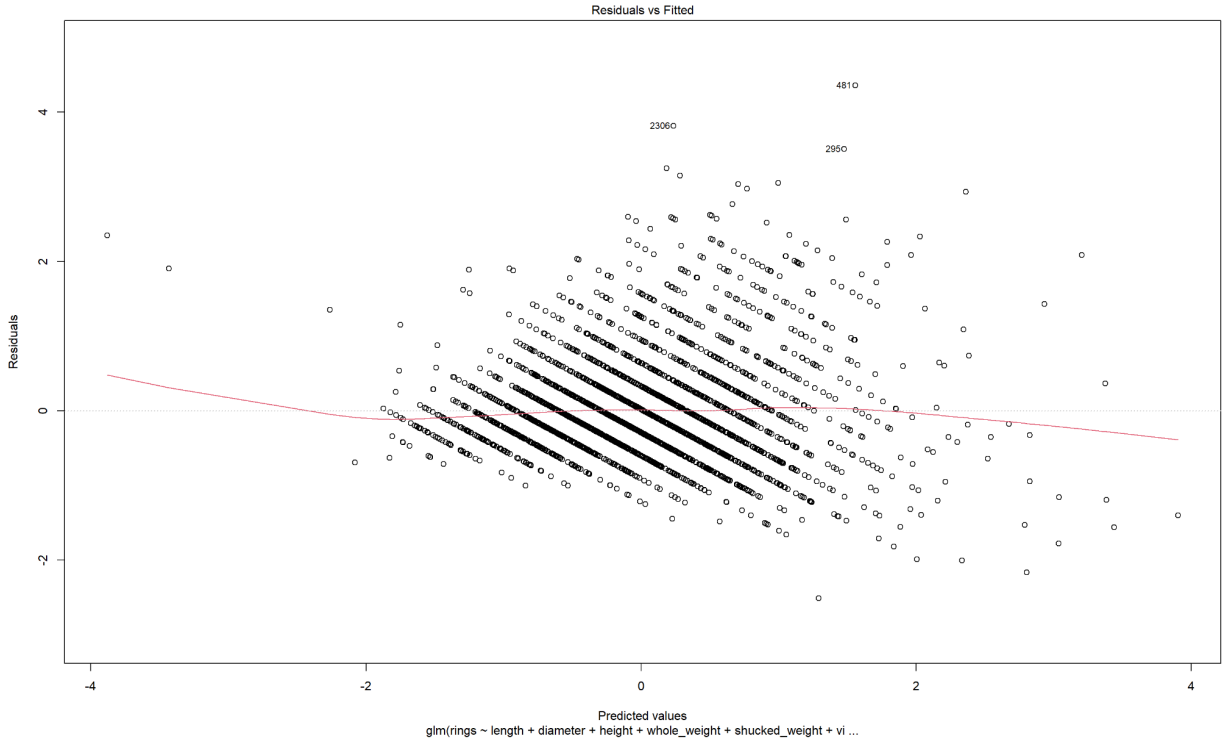
- Applying anova to the training data

```
> # anova - analysis of variance on the model result
> abalone.train_60.glm.anova <- anova(abalone.train_60.glm, test = "Chisq")
> abalone.train_60.glm.anova
Analysis of Deviance Table

Model: gaussian, link: identity

Response: rings

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                          2922     2951.3
length          1   918.29      2921     2033.0 < 2.2e-16 ***
diameter        1    69.50      2920     1963.5 < 2.2e-16 ***
height          1   132.12      2919     1831.4 < 2.2e-16 ***
whole_weight    1     3.30      2918     1828.1  0.008244 **
shucked_weight  1   398.24      2917     1429.8 < 2.2e-16 ***
viscera_weight  1    37.78      2916     1392.0 < 2.2e-16 ***
shell_weight    1    15.17      2915     1376.9 1.455e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

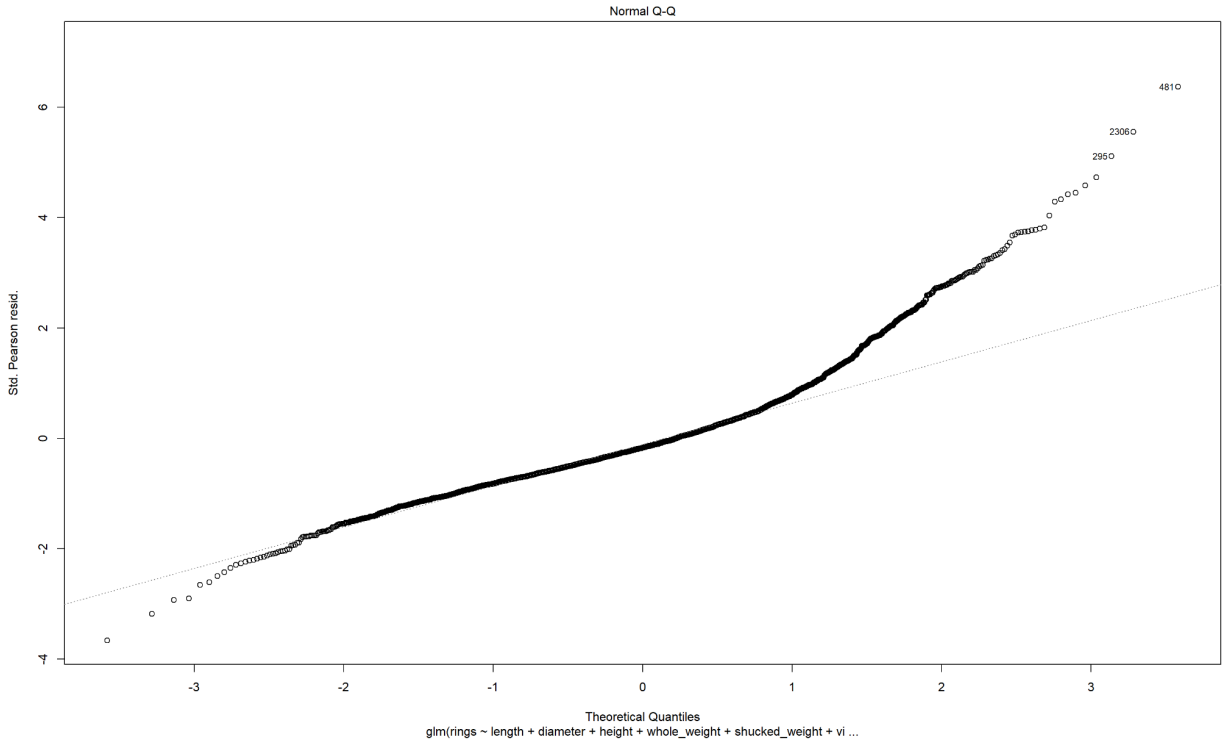
- Plotting the graphs

```
> # plotting graphs
> plot(abalone.train_60.glm)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
>
```

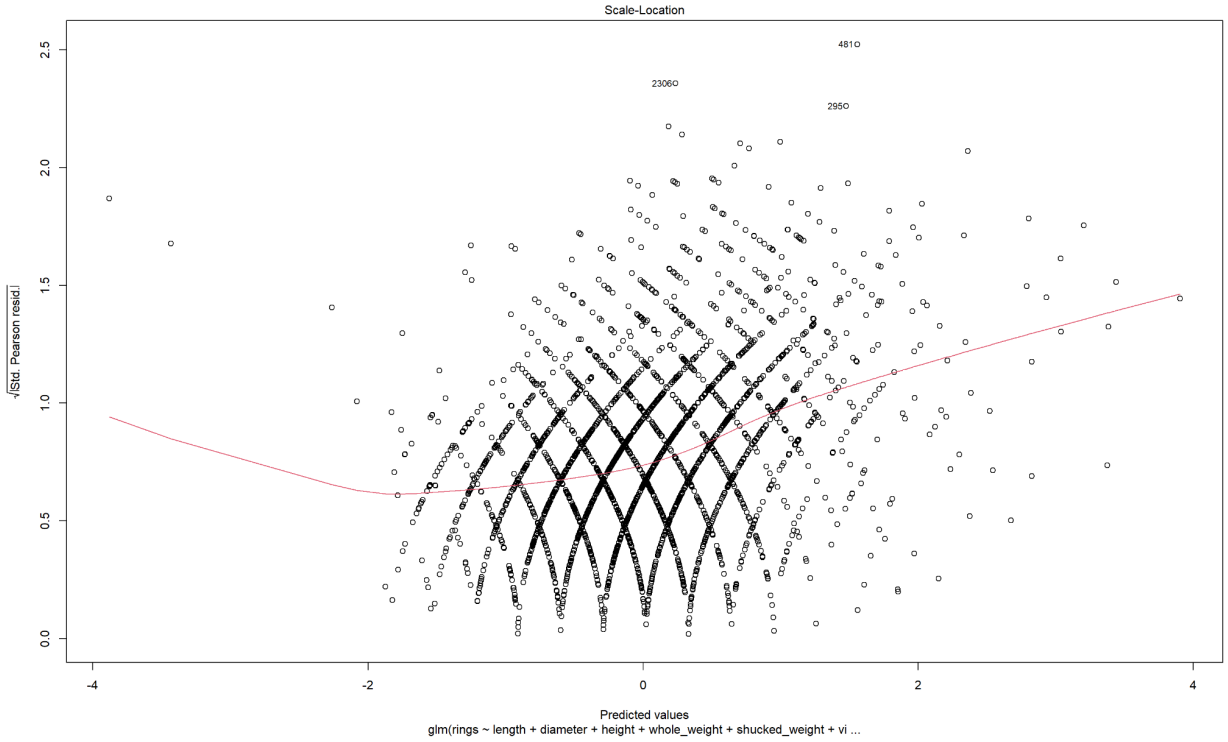
- Residual vs Fitted



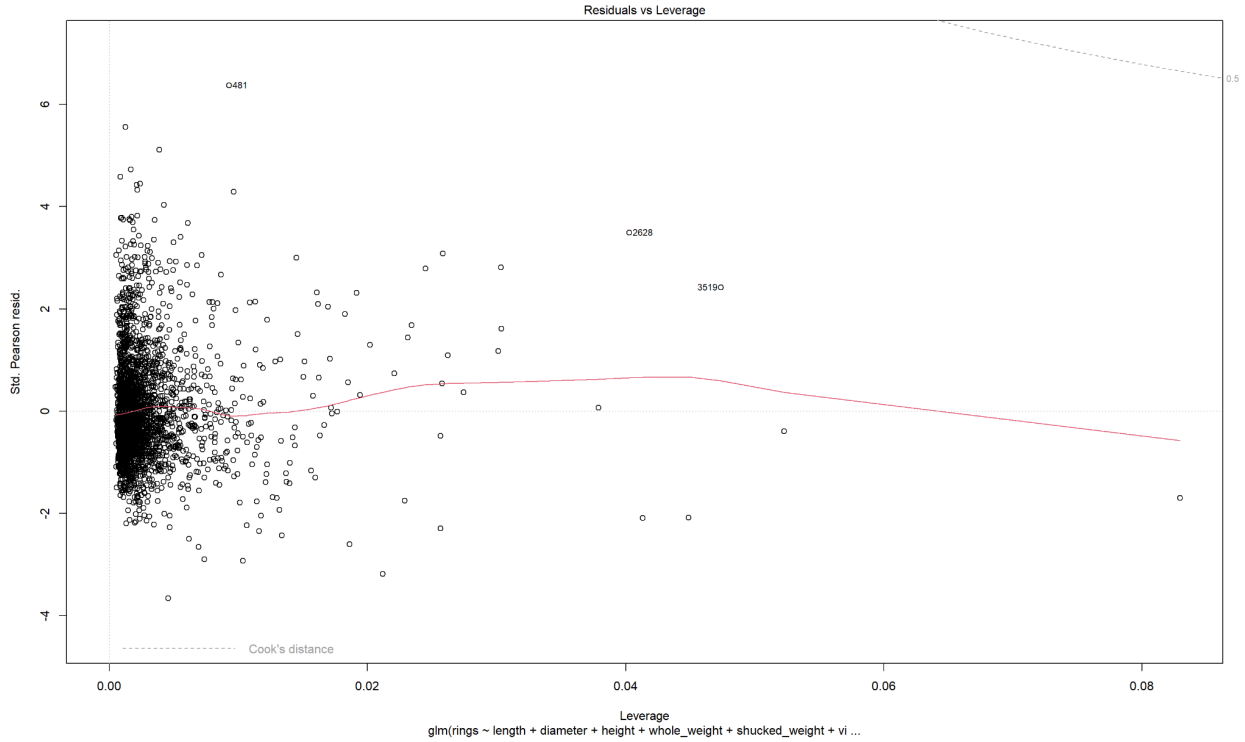
Normal Q-Q Plot



- Scale-Location Plot



Residuals vs Leverage Plot



- Using predict function in R
- Confidence intervals

```
> # predict
> abalone.test_60.predict <- predict(abalone.train_60.glm, newdata = abalone.test_60)
> length(abalone.test_60.predict)
[1] 1671
> summary(abalone.test_60.predict)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.873713 -0.434182 -0.000945  0.029797  0.461208  7.544918
> # confidence intervals
> confint(abalone.train_60.glm)
Waiting for profiling to be done...
                     2.5 %      97.5 %
(Intercept)    -0.01877274  0.03108867
length         -0.20184914  0.11266788
diameter        0.15565992  0.47517991
height          0.27472396  0.41720438
whole_weight    1.09314051  1.60011067
shucked_weight -1.47426182 -1.21546872
viscera_weight -0.46180597 -0.25500544
shell_weight    0.21074271  0.43360187
```

- Comparing the predicted values

```
> # comparing actual vs prediction
> abalone.test_60.predict.k5 <- kmeans(abalone.test_60.predict, centers = 5)
> abalone.test_60.predict.k5
K-means clustering with 5 clusters of sizes 284, 563, 38, 550, 236

Cluster means:
       [,1]
1 -0.9940505
2 -0.2862743
3  2.2717613
4  0.3055759
5  1.0122046

Clustering vector:
   3    4    7    8   10   11   12   14   15   19   21   22   24   28   29   31   33   37   39   40   41   43   45   46   47   48   49
   4    2    5    4    5    4    2    4    2    1    1    2    2    4    4    5    4    4    4    1    2    1    1    1    1    2    1
  50   52   53   54   56   59   60   62   63   68   70   71   72   73   74   75   76   78   80   84   85   86   91   93   95   96   97
   4    2    2    2    4    1    2    2    2    3    1    4    2    5    5    4    4    4    4    3    5    3    4    5    5    5    4
  99  102  104  106  107  110  111  115  120  131  134  137  138  139  140  141  143  146  150  154  157  158  160  166  168  174  175
   2    4    2    4    5    2    2    2    1    5    1    1    1    2    2    4    5    2    1    5    5    3    3    5    3    2    1
 176  177  184  187  188  189  190  192  195  197  198  202  203  204  205  207  208  209  210  214  216  218  219  220  221  223  225
   1    1    4    4    5    4    4    5    1    4    3    5    4    5    2    2    2    5    1    4    5    2    4    2    2    4    2
 226  228  232  233  235  242  245  247  248  250  254  259  265  267  268  270  276  282  283  285  286  290  294  296  302  304  308
   2    1    5    3    2    1    1    1    1    2    5    4    1    4    2    2    3    2    2    5    5    4    5    1    5    1    3
 310  311  312  313  317  320  323  324  325  329  332  347  349  351  356  357  360  369  377  380  383  384  386  388  389  391  395
   4    5    2    5    5    1    2    1    1    2    2    4    2    4    3    4    5    4    4    5    2    2    2    2    4    2    1
 397  399  400  405  406  407  409  412  416  419  421  424  425  429  430  433  434  435  437  439  446  449  450  452  453  454  455
   2    4    4    2    4    2    4    4    5    5    5    1    1    5    5    5    5    2    2    1    5    5    4    5    5    4    4
 456  459  461  463  464  466  473  477  479  486  487  488  489  492  493  494  495  497  498  499  500  501  509  512  515  516  518
   4    2    1    1    1    1    2    2    3    4    4    4    2    4    5    5    5    3    4    5    5    4    4    2    1    1    1
 519  523  524  530  535  536  538  540  545  548  550  553  554  555  556  557  564  565  566  571  574  576  580  581  582  585  587
   1    2    1    2    2    2    1    2    2    1    4    4    2    2    2    4    4    4    1    2    4    4    5    4    4    1    4

 [ reached getOption("max.print") -- omitted 671 entries ]

Within cluster sum of squares by cluster:
[1] 21.13809 17.64132 32.67836 18.63675 15.55902
 (between_SS / total_SS =  88.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
> |
```

- Comparing results of KMeans and KNN for 5 clusters using crosstable functionality provided by gmodels package

```
   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1671
```

abalone.test_60.predict.k5$cluster	abalone.test_60_kmeans_k5$cluster 1	2	3	4	5	Row Total
1	5	4	0	31	244	284
	26.225	76.319	26.513	35.173	712.356	
	0.018	0.014	0.000	0.109	0.859	0.170
	0.024	0.008	0.000	0.061	0.800	
	0.003	0.002	0.000	0.019	0.146	
2	20	125	10	347	61	563
	36.098	10.465	34.463	182.763	16.972	
	0.036	0.222	0.018	0.616	0.108	0.337
	0.096	0.253	0.064	0.686	0.200	
	0.012	0.075	0.006	0.208	0.037	
3	28	3	7	0	0	38
	113.707	6.056	3.360	11.507	6.936	
	0.737	0.079	0.184	0.000	0.000	0.023
	0.134	0.006	0.045	0.000	0.000	
	0.017	0.002	0.004	0.000	0.000	
4	77	285	67	121	0	550
	0.980	91.464	4.772	12.456	100.389	
	0.140	0.518	0.122	0.220	0.000	0.329
	0.368	0.576	0.429	0.239	0.000	
	0.046	0.171	0.040	0.072	0.000	
5	79	78	72	7	0	236
	82.950	0.936	113.323	58.149	43.076	
	0.335	0.331	0.305	0.030	0.000	0.141
	0.378	0.158	0.462	0.014	0.000	
	0.047	0.047	0.043	0.004	0.000	
Column Total	209	495	156	506	305	1671
	0.125	0.296	0.093	0.303	0.183	

- Confusion Matrix
- Calculating the metrics for k=5

```
> # confusion matrix
> abalone.test_60.cm_k5 <- abalone.test_60.ct.k5$t
> abalone.test_60.cm_k5
   y
x     1   2   3   4   5
  1   5   4   0  31 244
  2  20 125  10 347  61
  3  28   3   7   0   0
  4  77 285  67 121   0
  5  79  78  72   7   0
> # calculate the metrics
> abalone.test_60.accuracy <- sum(diag(abalone.test_60.cm_k5)) / sum(abalone.test_60.cm_k5)
> abalone.test_60.precision <- diag(abalone.test_60.cm_k5) / colSums(abalone.test_60.cm_k5)
> abalone.test_60.recall <- diag(abalone.test_60.cm_k5) / rowSums(abalone.test_60.cm_k5)
> abalone.test_60.specificity <- sapply(1:nrow(abalone.test_60.cm_k5), function(i){
+   TN <- sum(abalone.test_60.cm_k5[-i, -i])
+   FP <- sum(abalone.test_60.cm_k5[-i, i])
+   TN / (TN + FP)
+ })
> abalone.test_60.error <- 1 - abalone.test_60.accuracy
> # print results
> abalone.test_60.accuracy
[1] 0.1543986
> abalone.test_60.precision
         1          2          3          4          5
0.02392344 0.25252525 0.04487179 0.23913043 0.00000000
> abalone.test_60.recall
         1          2          3          4          5
0.01760563 0.22202487 0.18421053 0.22000000 0.00000000
> abalone.test_60.specificity
[1] 0.8529200 0.6660650 0.9087569 0.6565566 0.7874564
> abalone.test_60.error
[1] 0.8456014
```

Implementation for k=5 with 50-50 split

- Creating training labels with k-means

```
> # knn with k=5 and 50-50 split
> abalone.train_50_k5 <- kmeans(abalone.train_50, centers = 5)
> abalone.train_50_k5
K-means clustering with 5 clusters of sizes 197, 516, 477, 583, 315

Cluster means:
      length   diameter     height whole_weight shucked_weight viscera_weight shell_weight      rings
1  1.3988136  1.3931990  1.3635151    1.9265788     1.85936974     1.88997154   1.83192781  0.8565764
2  0.8192151  0.8223024  0.7114260    0.8028551     0.78508787     0.79156549   0.76557658  0.4311061
3 -0.5578181 -0.5694092 -0.5422810   -0.7519231    -0.71591350    -0.74328985  -0.73170270 -0.3871230
4  0.1951406  0.1928074  0.1401236   -0.0176486    -0.04691298    -0.02864518   0.03006721  0.2695454
5 -1.7147441 -1.7031876 -1.4282292   -1.3402950    -1.28340108    -1.30852777  -1.32791927 -1.0950139

Clustering vector:
3249 1914 2330 3683 2112 2631 1887 2158 1298  370 4040 3101 2605 1524 2007 1909 2645  969  976 1179 2845  277 1891  279 3758 4088 2555
   2    4    4    2    3    5    4    1    4    1    4    4    2    1    5    4    3    4    4    2    2    2    4    1    4    4    5
2836  562 3454 2812 1799 1227  486 2381 2383 3975  674 2370 2458 1045  949 1449 1405 2933  646 1260 2440  617  644  183 3146 2232 4095
   4    4    4    1    2    5    4    5    4    3    4    4    5    1    3    3    2    4    3    3    5    3    5    2    4    4    2
2442  417 2753 4131 2101 1804 1879  567 2881  435 1529 1572 1169 3861  849 1437   83 1330 3362  708 2629  275 3242 1623 3159 2750 3963
   4    2    4    4    3    2    4    3    3    3    1    3    4    4    4    5    4    4    3    5    5    2    1    4    4    3    1
 928 1589  701  327  222 1063 3506 1047 1590  161 1632 3777 3972 3639 1444 3125  972 3650 1881 3870 3043 2876 2655  820 2142  950 2194
   3    3    3    5    3    5    2    1    4    2    4    4    5    3    3    4    4    3    4    4    4    3    4    5    5    3    5
 561 3264 3465 1351  407 2193 3063 2719 2509 3340 2615 1706 1465 2538 1079 2262  825 1573 1557  492 2296 3697 2610  572  536 1284 2300
   3    4    2    4    3    2    2    5    3    4    2    1    4    1    5    4    5    3    3    4    4    1    2    3    3    3    3
3114 2424 2700 1320 2362 2941  187 2729  675 3670   21 2257 1844 1356  203 3607  911 2847  412 1681 2846 3913 1872   91  604 3930 3200
   3    3    2    4    4    4    2    3    4    4    5    3    3    2    4    3    5    2    4    2    2    4    4    4    3    1    3
 121 2940 4087 3545 3998 3914 2934 1594 3041  295 3628 2204 1683 1603 2533 4005 4013 1794 2350 2823 2991  721 2669 1512 4003  982 2553
   3    4    4    3    5    4    2    3    4    2    1    2    2    4    2    3    4    2    3    3    4    5    4    2    3    4    5
```

```
Within cluster sum of squares by cluster:
[1] 587.0488 830.1908 414.4468 878.1265 395.0124
 (between_SS / total_SS =  80.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
>
```

- Then the cluster vector is passed to KNN

  The output is a vector which contains the predicted values for the test dataset.

```
> abalone.test_50_k5 <- knn(abalone.train_50, abalone.test_50, abalone.train_50_k5$cluster, k = 5)
> abalone.test_50_k5
  [1] 3 3 3 3 5 3 4 4 2 2 2 2 1 3 2 3 4 5 5 3 5 4 3 3 5 3 3 2 5 4 2 2 4 2 4 4 2 2 3 2 4 2 2 1 1 3 3 5 4 4 2 4 4 4 3 3 3 4 4 4 3 4 3 5 4
 [67] 5 5 5 5 1 1 5 5 3 5 5 5 4 3 5 5 2 2 1 4 2 1 1 1 4 5 5 5 5 4 2 2 1 2 2 2 4 4 4 3 4 4 5 3 4 4 3 3 3 3 3 5 2 4 4 4 2 5 5 5 4 5 5 2 2 2 2
[133] 4 4 4 4 4 5 3 3 1 2 1 4 4 3 4 4 3 4 2 2 2 2 3 3 4 5 5 3 2 2 2 2 5 5 5 4 4 3 5 1 2 2 2 2 4 4 4 3 2 4 2 1 1 1 1 2 2 4 2 1 2 2 3 4 4 4
[199] 3 5 3 4 3 4 3 4 4 4 2 4 2 2 2 3 4 5 2 2 2 4 4 5 5 4 3 3 4 2 2 4 1 2 1 4 3 5 5 2 1 4 3 3 4 4 3 1 4 4 2 2 4 2 2 2 2 2 2 4 2 2 4 4 4
[265] 5 5 5 3 5 5 5 5 5 3 4 3 3 3 4 5 3 3 3 3 5 5 4 4 2 2 3 4 4 2 2 5 3 3 2 4 4 2 2 4 3 4 3 3 4 4 1 4 4 2 4 3 3 5 5 5 3 3 3 5 5 4 4 3 4 4
[331] 2 5 3 3 5 5 5 4 1 4 3 3 3 5 3 4 4 3 4 4 4 4 4 5 5 5 5 3 4 3 3 3 3 3 5 5 3 5 5 5 4 4 3 4 4 4 4 4 4 4 5 4 4 4 4 1 2 2 2 2 2 2 4 4 2 4 4 5
[397] 4 4 3 3 4 4 4 4 4 5 3 4 4 2 4 3 4 3 3 3 3 4 3 4 5 5 5 5 5 5 5 5 5 3 5 3 3 3 3 3 3 3 3 4 3 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2
[463] 2 2 2 2 1 2 1 1 5 5 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 3 3 4 4 3 4 4 3 4 4 2 2 4 2 4 2 2 4 4 2 2 2 2 2 2 2 2 2 2 2
[529] 2 1 2 1 1 1 1 1 1 1 5 5 5 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 3 3 3 4 3 4 3 3 4 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2
[595] 2 4 4 2 2 2 4 2 2 2 2 2 2 1 1 1 1 1 1 1 1 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 3 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 3 3 3 4 4 3 4
[661] 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 4 4 4 4 4 4 2 4 4 2 2 2 2 2 4 4 4 2 2 2 4 4 2 4 4 2 4 4 2 4 2 2 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2
[727] 2 1 1 1 2 1 1 1 1 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 3 4 4 4 4 4 4 2 4 2 4 2 4 2 4 2 2 2 1 2 1 2 2 1 1 1 1 1 1 5 5 5 5 5 5 5 5 5
[793] 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 3 4 4 4 4 4 4 4 3 4 4 3 4 4 4 4 4 4 4 4 4 4 4 2 4 2 2 4 4 4 4 2 4 4 4 4 2 4 4 2 2
[859] 2 4 4 4 2 2 2 2 2 2 4 4 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 1 2 2 2 1 1 2 2 1 1 2 1 1 1 1 1 1 1 1 1 5 3 3 4 4 4 4 3 4 4 2 4 2 4 2 2
[925] 2 2 2 2 2 2 2 1 2 2 2 2 1 1 1 1 5 5 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 4 4 3 3 3 4 3 4 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 4 4 2 4 4 4 2 2 2
[991] 4 2 4 2 2 2 2 2 2 2
 [ reached getOption("max.print") -- omitted 1089 entries ]
Levels: 1 2 3 4 5
```

- Applying KMeans to generate labels for the test records

```
> abalone.test_50_kmeans_k5 <- kmeans(abalone.test_50, centers = 5)
> abalone.test_50_kmeans_k5
K-means clustering with 5 clusters of sizes 219, 508, 568, 470, 324

Cluster means:
      length   diameter     height whole_weight shucked_weight viscera_weight shell_weight      rings
1  1.3522912  1.3586842  1.4105602    1.87456214     1.80386188     1.82587165   1.80532548  0.8504872
2  0.7958407  0.8083358  0.7010339    0.76995952     0.74010202     0.75702029   0.75751194  0.4638248
3  0.2031509  0.2106407  0.1172116   -0.02838737    -0.03862967    -0.02147687  -0.00281948  0.2411732
4 -0.5459653 -0.5597697 -0.5413735   -0.74061833    -0.69570476    -0.72427094  -0.73436480 -0.4466476
5 -1.7439799 -1.7524732 -1.5000269   -1.35837583    -1.29748001    -1.32457571  -1.35669489 -1.1348639

Clustering vector:
   1    6    9   18   19   20   23   24   25   29   32   33   34   36   37   38   40   42   43   44   48   49   50   54   57   59   60
   4    4    4    4    5    4    3    3    2    2    1    2    1    4    2    4    5    3    5    5    4    5    3    4    4    5    4
  65   68   70   71   75   76   79   80   81   85   86   87   89   90   92   93   94   95   96   98   99  101  102  104  105  107  108
   4    2    5    3    2    2    3    2    3    2    2    2    4    3    3    2    2    1    1    4    4    5    3    3    2    3    3
 109  110  111  112  115  116  117  118  119  120  122  123  124  125  126  127  129  130  133  135  136  137  138  140  141  146  148
   3    4    4    4    3    3    4    4    3    5    5    3    5    5    5    5    1    1    5    5    4    5    5    5    3    4    5
 149  152  154  158  160  163  166  167  169  174  176  177  178  179  180  181  182  185  189  190  191  197  199  202  205  209  211
   5    2    2    1    2    2    1    1    1    3    5    5    5    5    2    2    1    1    2    2    2    3    3    4    3    3    3
 212  214  215  217  219  224  226  227  228  229  230  231  232  233  236  238  240  241  247  249  252  253  254  257  261  262  263
   5    3    3    4    4    4    4    4    5    2    3    3    3    1    5    5    5    3    5    5    2    2    2    2    3    3    3
 266  267  268  269  270  271  273  278  280  281  283  285  287  289  290  291  292  293  294  299  301  302  303  304  305  313  314
   4    3    5    4    4    1    2    1    3    3    4    3    3    4    3    2    2    2    2    4    4    3    5    5    4    2    2
 315  317  320  322  326  328  331  332  333  335  337  339  341  342  344  346  347  349  350  354  355  356  357  360  363  364  366
   2    2    5    5    5    3    3    4    5    1    2    2    2    2    3    3    3    4    2    3    2    1    1    1    1    2    2
 368  374  376  377  378  384  385  387  389  392  394  396  397  398  400  401  402  406  410  411  413  415  416  419  422  423  425
```

```
Within cluster sum of squares by cluster:
[1] 1308.4194  836.4269  801.8546  408.3766  346.1336
 (between_SS / total_SS =  78.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

- Then the cluster vector is passed to KNN

  The output is a vector which contains the predicted values for the test dataset.

```
> abalone.test_50_k5.labels <- abalone.test_50_kmeans_k5$cluster
> length(abalone.test_50_k5.labels)
[1] 2089
> abalone.test_50_k5.labels
   1    6    9   18   19   20   23   24   25   29   32   33   34   36   37   38   40   42   43   44   48   49   50   54   57   59   60
   4    4    4    4    5    4    3    3    2    2    1    2    1    4    2    4    5    3    5    5    4    5    3    4    4    5    4
  65   68   70   71   75   76   79   80   81   85   86   87   89   90   92   93   94   95   96   98   99  101  102  104  105  107  108
   4    2    5    3    2    2    3    2    3    2    2    2    4    3    3    2    2    1    1    4    4    5    3    3    2    3    3
 109  110  111  112  115  116  117  118  119  120  122  123  124  125  126  127  129  130  133  135  136  137  138  140  141  146  148
   3    4    4    4    3    3    4    4    3    5    5    3    5    5    5    5    1    1    5    5    4    5    5    5    3    4    5
 149  152  154  158  160  163  166  167  169  174  176  177  178  179  180  181  182  185  189  190  191  197  199  202  205  209  211
   5    2    2    1    2    2    1    1    1    3    5    5    5    5    2    2    1    1    2    2    2    3    3    3    4    3    3
 212  214  215  217  219  224  226  227  228  229  230  231  232  233  236  238  240  241  247  249  252  253  254  257  261  262  263
   5    3    3    4    4    4    4    4    5    2    3    3    3    1    5    5    5    3    5    5    2    2    2    2    3    3    3
 266  267  268  269  270  271  273  278  280  281  283  285  287  289  290  291  292  293  294  299  301  302  303  304  305  313  314
   4    3    5    4    4    1    2    1    3    3    4    3    3    4    3    2    2    2    2    4    4    3    5    5    4    2    2
 315  317  320  322  326  328  331  332  333  335  337  339  341  342  344  346  347  349  350  354  355  356  357  360  363  364  366
   2    2    5    5    5    3    3    4    5    1    2    2    2    2    3    3    3    4    2    3    2    1    1    1    1    2    2
 368  374  376  377  378  384  385  387  389  392  394  396  397  398  400  401  402  406  410  411  413  415  416  419  422  423  425
   3    2    1    2    2    4    3    3    3    4    5    4    3    4    3    4    3    3    3    2    3    2    2    2    4    3    5
 426  427  430  431  440  441  442  443  444  445  446  448  449  450  451  452  453  456  459  464  466  468  469  471  472  473  475
   2    2    2    3    3    4    5    3    4    4    3    2    2    3    1    2    1    3    4    5    5    2    1    3    3    4    3
 476  477  481  482  483  484  485  487  488  493  494  496  497  499  501  502  504  507  508  510  513  514  515  516  517  519  520
   3    4    1    3    3    2    2    3    2    2    2    2    2    2    3    2    2    2    2    3    3    5    5    5    4    5    5
 524  525  526  530  531  532  533  535  537  539  540  542  543  544  546  547  549  550  551  552  554  556  557  558  559  566  570
   5    5    5    4    3    4    4    4    3    5    4    4    4    4    5    5    3    3    2    2    4    3    3    2    2    5    4
 571  576  577  579  580  581  582  585  587  589  590  591  593  594  596  598  599  601  603  606  610  612  613  614  615  616  618
   4    2    3    3    2    2    3    4    3    4    4    3    3    1    3    3    2    3    4    4    5    5    5    3    4    4    5
 620  622  626  627  629  640  642  647  648  650  651  654  655  660  661  662  663  664  665  666  667  668  669  670  671  679  684
```

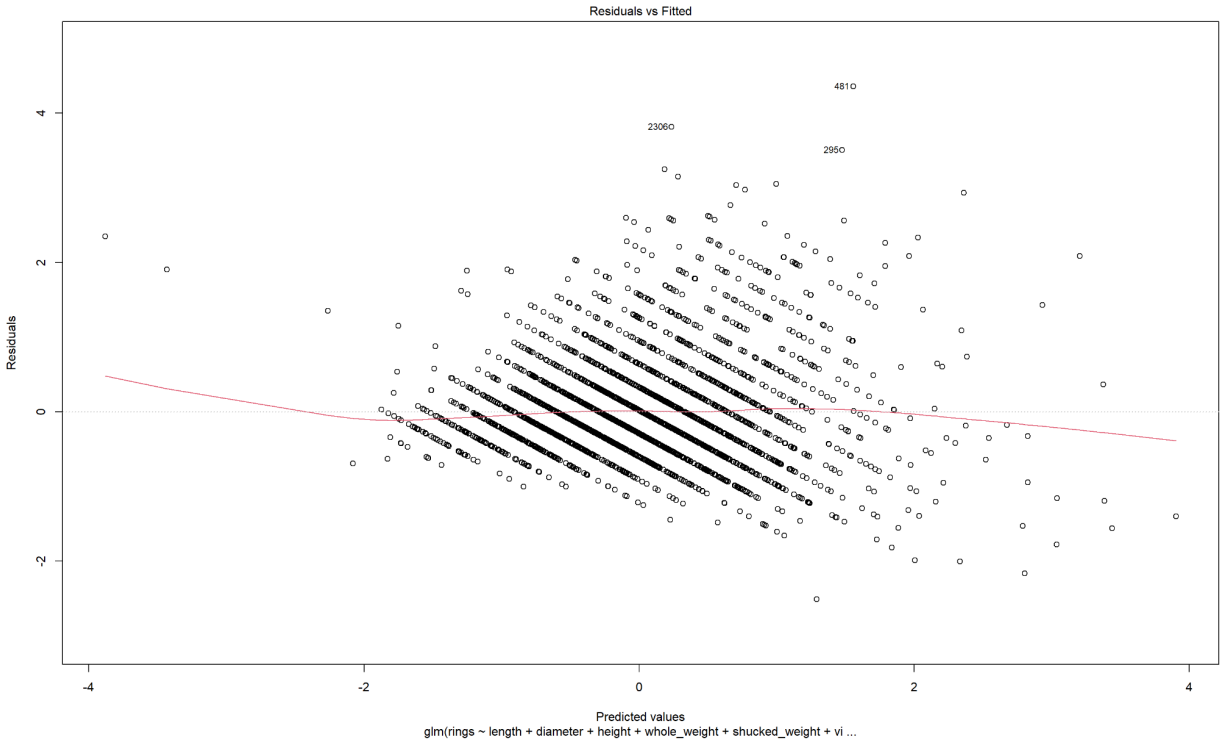
- Linear modeling using glm()

```
> # linear modelling using GLM
> abalone.train_50.glm <- glm(formula = rings ~ length + diameter + height + whole_weight + shucked_weight +  viscera_weight + shell_weight, fa
mily = gaussian, data = abalone.train_70)
> summary(abalone.train_50.glm)

Call:
glm(formula = rings ~ length + diameter + height + whole_weight +
    shucked_weight + viscera_weight + shell_weight, family = gaussian,
    data = abalone.train_70)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5111  -0.4220  -0.1166   0.2718   4.3572

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.006158   0.012720   0.484 0.628339
length         -0.044591   0.080235  -0.556 0.578426
diameter        0.315420   0.081512   3.870 0.000111 ***
height          0.345964   0.036348   9.518  < 2e-16 ***
whole_weight    1.346626   0.129331  10.412  < 2e-16 ***
shucked_weight -1.344865   0.066020 -20.371  < 2e-16 ***
viscera_weight -0.358406   0.052756  -6.794 1.32e-11 ***
shell_weight    0.322172   0.056853   5.667 1.60e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.4723417)

    Null deviance: 2951.3  on 2922  degrees of freedom
Residual deviance: 1376.9  on 2915  degrees of freedom
AIC: 6112.7

Number of Fisher Scoring iterations: 2
```

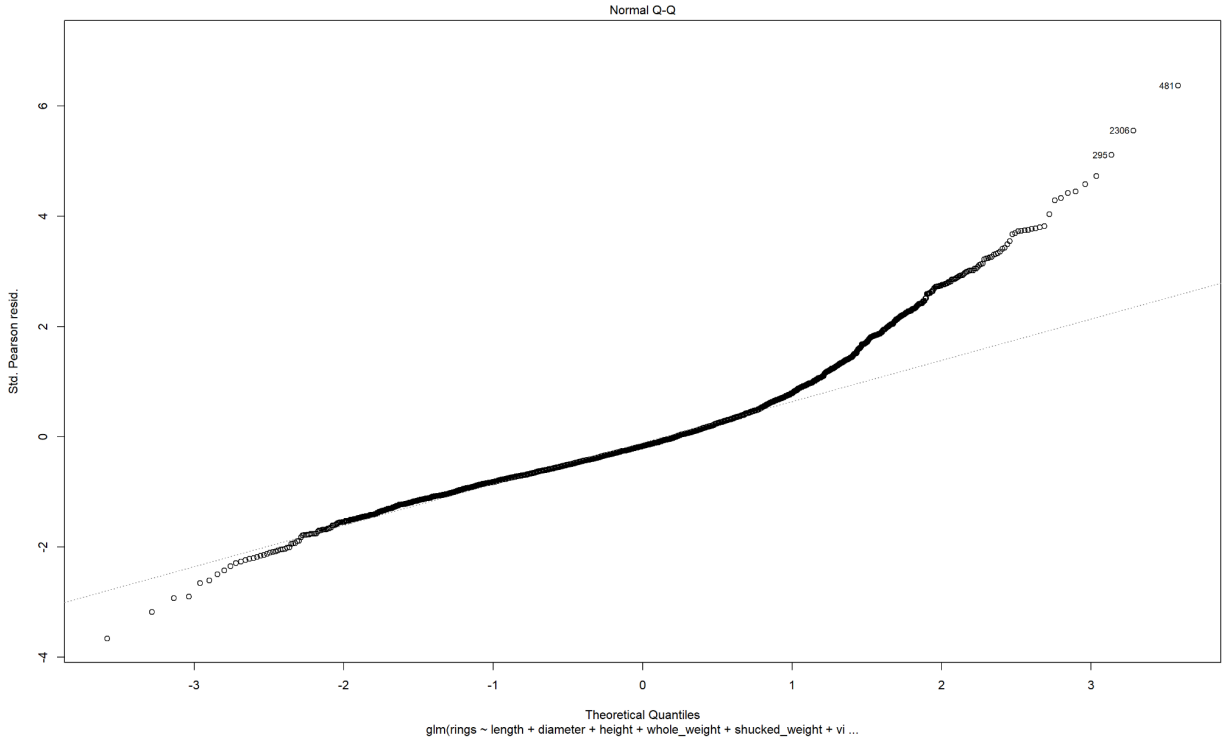
- Applying anova to the training data to get the deviance of the data

```
> abalone.train_50.glm.anova <- anova(abalone.train_50.glm, test = "Chisq")
> abalone.train_50.glm.anova
Analysis of Deviance Table

Model: gaussian, link: identity

Response: rings

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                           2922     2951.3
length          1   918.29      2921     2033.0 < 2.2e-16 ***
diameter        1    69.50      2920     1963.5 < 2.2e-16 ***
height          1   132.12      2919     1831.4 < 2.2e-16 ***
whole_weight    1     3.30      2918     1828.1  0.008244 **
shucked_weight  1   398.24      2917     1429.8 < 2.2e-16 ***
viscera_weight  1    37.78      2916     1392.0 < 2.2e-16 ***
shell_weight    1    15.17      2915     1376.9 1.455e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

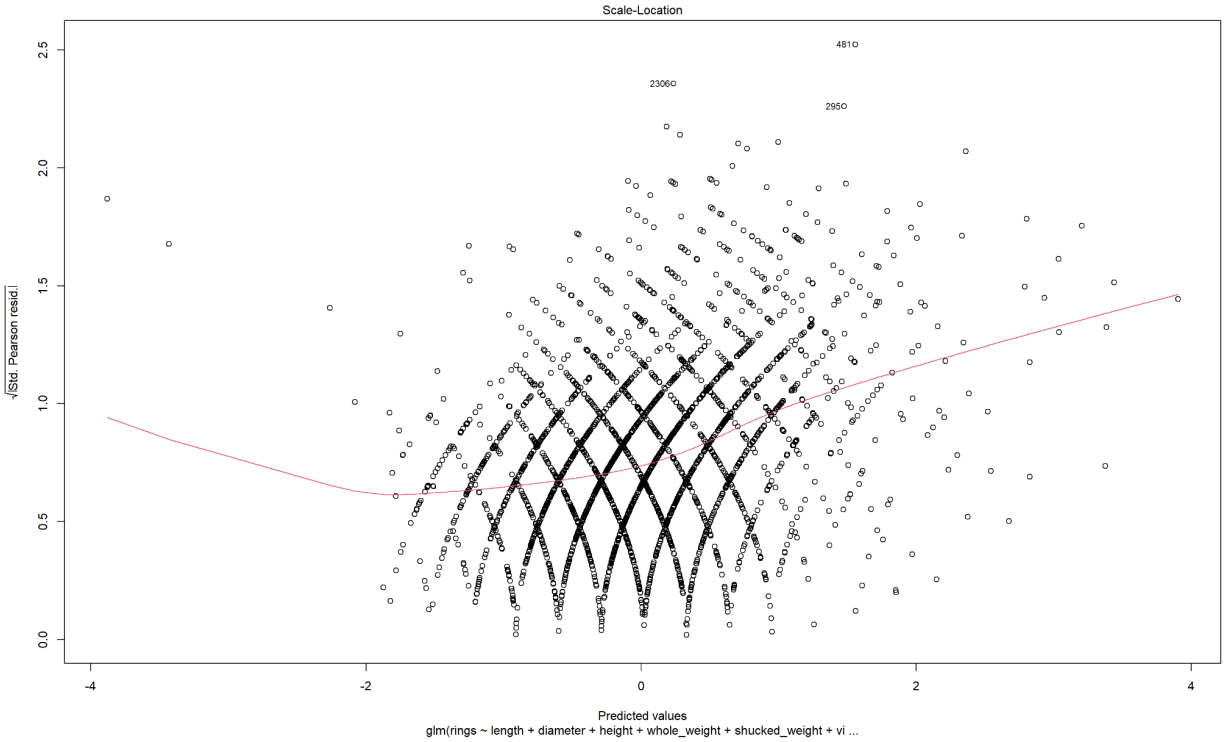
- Plotting the graphs
- Residual vs Fitted



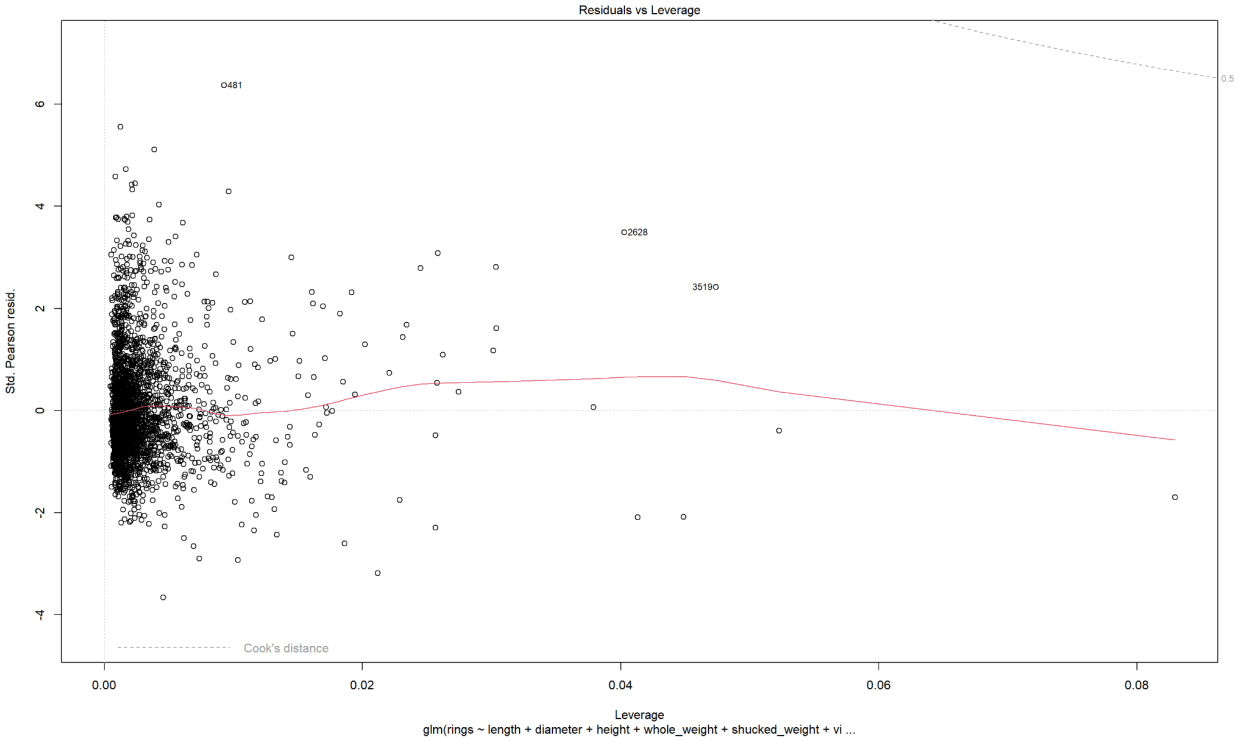
- Normal Q - Q



- Scale - Location



- Residual vs Leverage



- Predict Function
- Confidence intervals

```
> # predict
> abalone.test_50.predict <- predict(abalone.train_50.glm, newdata = abalone.test_50)
> length(abalone.test_50.predict)
[1] 2089
> summary(abalone.test_50.predict)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-3.433678 -0.481626 -0.051664  0.000136  0.425382  7.544918
> # confidence intervals
> confint(abalone.train_50.glm)
Waiting for profiling to be done...
                    2.5 %      97.5 %
(Intercept)    -0.01877274  0.03108867
length         -0.20184914  0.11266788
diameter        0.15565992  0.47517991
height          0.27472396  0.41720438
whole_weight    1.09314051  1.60011067
shucked_weight -1.47426182 -1.21546872
viscera_weight -0.46180597 -0.25500544
shell_weight    0.21074271  0.43360187
```

- Comparing the predicted values

```
> # comparing actual vs prediction
> abalone.test_50.predict.k5 <- kmeans(abalone.test_50.predict, centers = 5)
> abalone.test_50.predict.k5
K-means clustering with 5 clusters of sizes 344, 358, 679, 628, 80

Cluster means:
       [,1]
1  0.861169
2 -1.027931
3 -0.336418
4  0.219881
5  2.029791

Clustering vector:
   1    6    9   18   19   20   23   24   25   29   32   33   34   36   37   38   40   42   43   44   48   49   50   54   57   59   60
   3    3    3    3    3    3    4    3    4    1    5    4    4    3    4    3    2    4    2    2    3    2    4    3    3    2    3
  65   68   70   71   75   76   79   80   81   85   86   87   89   90   92   93   94   95   96   98   99  101  102  104  105  107  108
   3    5    2    4    4    4    1    1    4    1    5    1    3    1    4    1    1    1    5    3    3    2    4    4    4    1    4
 109  110  111  112  115  116  117  118  119  120  122  123  124  125  126  127  129  130  133  135  136  137  138  140  141  146  148
   4    3    3    3    3    3    4    3    1    2    2    4    2    2    2    2    5    5    2    2    3    2    2    3    4    3    2
 149  152  154  158  160  163  166  167  169  174  176  177  178  179  180  181  182  185  189  190  191  197  199  202  205  209  211
   2    1    1    5    5    4    1    5    5    3    2    2    2    2    1    1    5    1    4    4    1    4    4    1    3    1    4
 212  214  215  217  219  224  226  227  228  229  230  231  232  233  236  238  240  241  247  249  252  253  254  257  261  262  263
   2    4    4    3    4    4    4    3    2    1    1    4    1    5    2    2    2    1    2    2    1    4    1    5    4    4    4
 266  267  268  269  270  271  273  278  280  281  283  285  287  289  290  291  292  293  294  299  301  302  303  304  305  313  314
   2    3    3    3    3    3    3    3    3    4    3    3    3    4    3    4    2    3    3    4    4    4    3    4    4    4    4
1894 1895 1896 1898 1899 1900 1902 1903 1905 1908 1911 1912 1913 1915 1916 1917 1919 1920 1923 1924 1925 1926 1927 1928 1929 1930 1931
   4    4    4    3    3    4    3    4    1    4    1    3    4    3    3    4    4    1    4    4    4    3    4    3    4    1    4
1933
   1
 [ reached getOption("max.print") -- omitted 1089 entries ]

Within cluster sum of squares by cluster:
[1] 18.89603 32.22598 20.78724 17.39287 45.57811
 (between_SS / total_SS =  88.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
>
```

- Comparing the results KMeans and KNN for 5 clusters using the crosstable functionality provided by gmodels package

```
> abalone.test_50.ct.k5 <- CrossTable(abalone.test_50.predict.k5$cluster, abalone.test_50_kmeans_k5$cluster, prop.chisq = TRUE)

   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  2089
```

abalone.test_50.predict.k5$cluster	abalone.test_50_kmeans_k5$cluster					
	1	2	3	4	5	Row Total
1	86	187	69	2	0	344
	69.148	127.676	6.435	73.448	53.354	
	0.250	0.544	0.201	0.006	0.000	0.165
	0.393	0.368	0.121	0.004	0.000	
	0.041	0.090	0.033	0.001	0.000	
2	4	2	7	47	298	358
	29.957	83.104	83.844	13.971	1058.873	
	0.011	0.006	0.020	0.131	0.832	0.171
	0.018	0.004	0.012	0.100	0.920	
	0.002	0.001	0.003	0.022	0.143	
3	24	77	173	379	26	679
	31.275	47.026	0.731	335.030	59.731	
	0.035	0.113	0.255	0.558	0.038	0.325
	0.110	0.152	0.305	0.806	0.080	
	0.011	0.037	0.083	0.181	0.012	
4	55	213	318	42	0	628
	1.784	23.797	126.976	69.777	97.402	
	0.088	0.339	0.506	0.067	0.000	0.301
	0.251	0.419	0.560	0.089	0.000	
	0.026	0.102	0.152	0.020	0.000	
5	50	29	1	0	0	80
	206.475	4.684	19.798	17.999	12.408	
	0.625	0.362	0.012	0.000	0.000	0.038
	0.228	0.057	0.002	0.000	0.000	
	0.024	0.014	0.000	0.000	0.000	
Column Total	219	508	568	470	324	2089
	0.105	0.243	0.272	0.225	0.155	

- Confusion Matrix

- Calculating the metrices for k=5

```
> # confusion matrix
> abalone.test_50.cm_k5 <- abalone.test_50.ct.k5$t
> abalone.test_50.cm_k5
   y
x     1   2   3   4   5
  1  86 187  69   2   0
  2   4   2   7  47 298
  3  24  77 173 379  26
  4  55 213 318  42   0
  5  50  29   1   0   0
> # calculate the metrics
> abalone.test_50.accuracy <- sum(diag(abalone.test_50.cm_k5)) / sum(abalone.test_50.cm_k5)
> abalone.test_50.precision <- diag(abalone.test_50.cm_k5) / colSums(abalone.test_50.cm_k5)
> abalone.test_50.recall <- diag(abalone.test_50.cm_k5) / rowSums(abalone.test_50.cm_k5)
> abalone.test_50.specificity <- sapply(1:nrow(abalone.test_50.cm_k5), function(i){
+   TN <- sum(abalone.test_50.cm_k5[-i, -i])
+   FP <- sum(abalone.test_50.cm_k5[-i, i])
+   TN / (TN + FP)
+ })
> abalone.test_50.error <- 1 - abalone.test_50.accuracy
> # print results
> abalone.test_50.accuracy
[1] 0.1450455
> abalone.test_50.precision
          1           2           3           4           5
0.392694064 0.003937008 0.304577465 0.089361702 0.000000000
> abalone.test_50.recall
          1           2           3           4           5
0.250000000 0.005586592 0.254786451 0.066878981 0.000000000
> abalone.test_50.specificity
[1] 0.9237822 0.7076834 0.7198582 0.7070500 0.8387257
> abalone.test_50.error
[1] 0.8549545
```

Implementation for k=7 with 70-30 split

- Creating training labels with k-means

```
> abalone.train_70_k7 <- kmeans(abalone.train_70, centers = 7)
> abalone.train_70_k7
K-means clustering with 7 clusters of sizes 205, 591, 439, 208, 650, 575, 255

Cluster means:
      length   diameter     height whole_weight shucked_weight viscera_weight shell_weight       rings
1 -2.1990445 -2.1900943 -1.8182725  -1.49805052    -1.42354752    -1.45775908  -1.50017540 -1.41825965
2  0.8573546  0.8548428  0.6944156   0.83957442     0.86735573     0.84198017   0.75831882  0.15911574
3 -1.1278764 -1.1298841 -0.9954047  -1.10765025    -1.05459768    -1.09433450  -1.09377807 -0.75729811
4  0.5574467  0.6128653  0.7075367   0.55304043     0.23313754     0.45447968   0.82828116  2.25429825
5  0.2994494  0.2985336  0.1699062   0.07180672     0.08834206     0.07490821   0.07143366 -0.02285378
6 -0.3548044 -0.3634886 -0.3462261  -0.58413161    -0.56625436    -0.56890928  -0.55480251 -0.14287123
7  1.3965336  1.3907788  1.3288442   1.94060218     1.89186831     1.92164545   1.87140402  0.74305264

Clustering vector:
 755 1006 2585 2339 1448 3952 3358 3980 1134 3230 1934 1501 3783 1914 1109 1075 3146 1386 2284 2378 4044 2260  686 3857  847  983  151
   4    2    5    7    6    6    3    6    5    2    7    2    2    5    6    3    5    2    6    5    5    2    4    3    5    5    4
1638 2208 2474 1029  326 3856 2837 1956 4093  985  986 2503 1762 1584 4084 2087 2244 1793 4141 1808  344 2507 2992 3092  195 3236 3124
   2    3    2    2    3    3    5    7    7    2    2    1    7    6    5    2    6    5    2    2    6    3    2    6    5    2    5
2225 2875 2132 4023 3464 1326 3949 2667 3502 2758 3833  712 1452 1828 3501 3069 2446 3061  528 4055 1569  473 1149 2037 2313 2823 1927
   5    3    1    7    2    5    5    5    2    6    6    3    6    3    5    2    6    2    5    2    6    3    5    1    1    3    2
2234 3645 3324  458 3224  831 1078  919 1562 2283 1313  185 3837  413  627 2570 1333 3126 2253 1899  779 1561  564  794 1415 2457 3799
   4    6    3    3    5    3    3    3    3    6    6    7    3    5    3    3    5    2    4    5    6    3    5    5    7    1    2
1370  160 2516 3581 2968 3129 2505  617  357  279  270 2926 2694 2395 3201 2266  618 1905 2746  337 2845 2074  539  981 3591  956 1781
   5    4    6    2    2    2    3    6    7    4    6    5    2    2    5    7    3    2    6    4    2    5    1    5    7    6    5
1690 3625 2772 3462 1522 3294 2719 2909 2705 3008 1445 1081 2708 2211 2286 1697 2626 3588 4158 3297 1413  117 3609  648 1079 1241 2645
   2    2    5    2    7    2    3    5    7    7    3    3    7    4    3    2    7    2    6    4    2    6    5    6    3    3    6
2605 3730 1706 2463 3206 4165 3005  873 2736 3985  757 1965  988 2495 3893 3176 2869 2879  711 1492 1234  349 1425 3809 3330 4123  386
   2    6    7    3    3    3    2    2    3    2    4    2    5    3    5    6    3    6    3    2    3    3    7    6    5    6    6
```

```
 [ reached getOption("max.print") -- omitted 1923 entries ]

Within cluster sum of squares by cluster:
[1] 160.3888 563.0829 271.4512 488.5038 544.6913 545.3408 691.8639
 (between_SS / total_SS =  85.6 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
>
```

- Then the cluster vector is passed to KNN

  The output is a vector which contains the predicted values for the test dataset.

```
> abalone.test_70_k7 <- knn(abalone.train_70, abalone.test_70, abalone.train_70_k7$cluster, k = 7)
> abalone.test_70_k7
   [1] 6 3 5 3 7 1 6 5 6 3 6 6 6 6 3 6 6 6 4 2 4 4 2 4 2 6 3 2 5 6 6 6 6 3 6 6 5 3 3 1 3 3 4 7 1 6 6 1 6 2 7 7 7 5 3 1 2 2 5 5 3 3 3 3 6 6
  [67] 6 6 6 6 3 3 4 6 1 1 1 3 4 4 5 1 4 7 5 4 1 3 3 6 1 7 5 4 3 1 3 1 2 2 2 5 7 7 4 4 5 7 2 4 5 6 6 3 5 5 6 5 3 6 3 5 4 5 5 4 4 6 1 2 4 3
 [133] 3 5 3 3 5 4 5 1 2 5 4 4 5 4 4 4 5 4 4 4 5 2 6 1 1 1 3 1 4 3 1 3 3 3 1 5 6 5 6 6 6 6 3 6 1 6 4 5 2 5 5 6 5 3 4 4 5 5 4 6 6 3 6 6 6 3
 [199] 3 3 3 6 3 6 5 4 4 4 5 6 1 1 6 6 6 3 3 3 1 5 5 6 5 6 6 6 4 4 6 6 2 4 4 3 5 4 6 5 3 6 5 6 6 6 3 5 6 3 3 3 3 3 6 6 5 5 5 5 5 2 5 2 2 2 2
 [265] 2 2 2 7 1 1 1 1 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6 6 6 6 6 6 5 6 5 5 5 2 5 2 2 2 2 2 2 2 2 2 7 7 7 7 2 7 7 7 1 1 1 3 3 3 3 3 3 6 6 6 6 6
 [331] 6 6 5 5 5 5 5 5 5 2 2 2 2 2 7 7 7 7 7 7 7 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6 6 6 6 6 5 6 5 6 6 5 5 5 5 5 5 5 2 5 5 5 2 2 5 5
 [397] 2 5 5 5 2 2 2 2 2 2 2 2 2 2 2 2 2 7 2 7 7 7 7 3 3 3 3 3 6 6 3 6 6 6 6 6 6 5 5 5 5 5 2 5 5 2 5 2 2 7 7 7 7 2 7 7 2 7 7 7 7 1 1 1 1 1 3
 [463] 3 3 3 3 3 3 6 6 6 6 6 6 5 6 6 6 5 6 6 5 5 5 5 5 5 5 5 5 5 2 5 2 2 2 2 2 2 5 2 2 2 2 2 7 2 2 2 2 2 7 2 2 2 2 7 7 7 7 7 2 7 7 7 7
 [529] 7 3 6 3 6 5 5 5 5 5 5 2 5 2 2 2 2 7 2 7 7 7 1 1 3 3 3 3 3 3 3 6 6 6 6 6 5 6 5 5 5 6 5 5 5 5 5 5 5 2 5 2 2 2 2 2 2 2 2 2 2 2 2 7 7 7 7
 [595] 7 1 1 3 3 3 3 3 3 6 6 6 5 5 5 1 3 3 6 4 6 6 6 5 2 2 2 7 6 6 3 2 4 2 1 5 3 1 3 1 3 1 3 6 5 6 6 6 3 4 4 5 2 1 5 4 1 1 3 2 4 1 1 7 3 5 5 2 3
 [661] 6 6 3 6 3 6 2 3 5 2 2 4 5 4 2 6 6 3 3 6 6 6 6 3 6 4 6 7 4 4 5 5 3 7 4 4 7 5 5 4 1 6 3 1 6 6 3 4 6 5 3 5 6 6 3 3 6 5 6 7 4 4 5 3 1 1
 [727] 1 6 6 4 3 2 4 4 5 6 5 5 5 5 6 6 6 6 4 6 1 3 3 6 6 5 5 2 2 2 2 7 7 7 7 7 1 1 1 3 3 3 6 6 6 6 6 6 5 5 5 5 5 5 2 5 2 2 2 2 2 2 2 2 3 6
 [793] 6 6 6 6 5 6 5 5 5 5 5 5 2 5 2 2 2 2 2 5 2 2 2 7 2 7 1 1 3 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6 5 5 5 5 5 5 5 2 2 2 2 2 2 2 7 7 7 1 1 3 6 6 5
 [859] 5 2 2 2 2 2 2 7 7 7 7 1 3 3 3 6 6 6 6 6 5 5 5 5 5 5 5 2 5 2 5 2 2 2 2 2 2 2 2 2 2 2 7 2 2 2 2 7 7 5 5 2 2 5 2 2 2 2 2 7 7 3 3 6 6 6 6 6 5 5 5 5
 [925] 2 2 2 2 2 2 2 2 1 1 3 3 6 6 5 6 5 5 5 5 7 1 1 3 6 6 4 1 1 2 2 7 6 5 5 1 4 5 5 5 3 5 5 6 7 1 5 2 1 4 6 3 2 5 2 2 6 3 4 6 6 5 3 7 7 2 3 1
 [991] 6 5 6 6 4 6 4 6 6 6
 [ reached getOption("max.print") -- omitted 254 entries ]
Levels: 1 2 3 4 5 6 7
```

```
Within cluster sum of squares by cluster:
[1] 587.0488 830.1908 414.4468 878.1265 395.0124
 (between_SS / total_SS =  80.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
>
```

- Applying KMeans to generate labels for the test records

```
> abalone.test_70_kmeans_k7 <- kmeans(abalone.test_70, centers = 7)
> abalone.test_70_kmeans_k7
K-means clustering with 7 clusters of sizes 207, 99, 294, 260, 105, 288, 1

Cluster means:
      length   diameter      height whole_weight shucked_weight viscera_weight shell_weight       rings
1 -1.6760921 -1.6887508 -1.44832571  -1.33671483   -1.285000773    -1.29714335  -1.33347392 -1.11068229
2  1.4748498  1.4804204  1.48105681   2.11632695    2.104574497     1.99185493   1.93306876  0.90091411
3  0.2280355  0.2297365  0.09572772  -0.02141639    0.005447044    -0.01981347  -0.02412361 -0.06910311
4  0.8811144  0.8699154  0.72512285   0.89451384    0.919556675     0.91824326   0.78140015  0.10168636
5  0.3402786  0.3831475  0.62633827   0.30552174    0.012653922     0.21740058   0.57078542  1.99376033
6 -0.5340114 -0.5479079 -0.52429206  -0.74378126   -0.711131835    -0.72332125  -0.72210080 -0.36928241
7 -0.5744894 -0.5328630 23.68045187  -0.47868560   -0.123297554    -0.58928112  -0.75667270 -0.59974661

Clustering vector:
   3    6   14   22   34   43   47   50   53   55   57   58   61   62   64   65   66   69   73   74   84   85   86   91   93   98  101
   3    6    3    1    2    1    6    3    6    6    6    6    6    6    6    6    6    6    5    4    5    5    4    5    4    6    1
 105  106  109  110  111  112  113  115  116  119  124  125  126  127  128  129  130  135  145  146  149  156  157  166  168  169  174
   4    3    3    6    6    6    6    3    3    3    1    1    1    1    1    2    2    1    6    6    1    3    5    2    2    2    3
 176  177  188  189  190  196  205  208  210  212  213  219  221  222  225  226  227  228  233  235  236  238  240  248  254  257  263
   1    1    4    4    3    3    6    6    1    1    6    6    6    6    6    6    6    1    5    6    1    1    1    1    5    5    3
 265  273  274  285  293  298  300  301  305  306  308  310  313  320  324  332  334  336  339  342  345  356  360  363  364  367  371
   1    5    2    5    5    1    1    6    6    1    2    3    5    1    1    6    1    4    4    4    3    2    2    5    5    3    2
 375  381  385  389  390  391  397  399  401  402  403  404  405  410  411  412  414  416  417  423  425  426  427  437  438  440  441
   4    5    3    3    6    6    3    3    6    3    6    3    6    3    5    5    3    5    5    3    1    5    5    1    6    3    1
 444  446  447  454  465  468  475  476  478  486  490  495  496  501  505  507  508  509  511  513  514  516  519  520  521  531  534
```

```
Within cluster sum of squares by cluster:
[1] 217.9915 373.8131 276.4080 275.8722 235.3034 245.1484   0.0000
 (between_SS / total_SS =  84.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
>
```

- Obtaining labels for the test data by using K-Means on the test data

```
> abalone.test_70_k7.labels <- abalone.test_70_kmeans_k7$cluster
> length(abalone.test_70_k7.labels)
[1] 1254
> abalone.test_70_k7.labels
   3    6   14   22   34   43   47   50   53   55   57   58   61   62   64   65   66   69   73   74   84   85   86   91   93   98  101
   3    6    3    1    2    1    6    3    6    6    6    6    6    6    6    6    6    6    5    4    5    5    4    5    4    6    1
 105  106  109  110  111  112  113  115  116  119  124  125  126  127  128  129  130  135  145  146  149  156  157  166  168  169  174
   4    3    3    6    6    6    6    3    3    3    1    1    1    1    1    2    2    1    6    6    1    3    5    2    2    2    3
 176  177  188  189  190  196  205  208  210  212  213  219  221  222  225  226  227  228  233  235  236  238  240  248  254  257  263
   1    1    4    4    3    3    6    6    1    1    6    6    6    6    6    6    6    1    5    6    1    1    1    1    5    5    3
 265  273  274  285  293  298  300  301  305  306  308  310  313  320  324  332  334  336  339  342  345  356  360  363  364  367  371
   1    5    2    5    5    1    1    6    6    1    2    3    5    1    1    6    1    4    4    4    3    2    2    5    5    3    2
 375  381  385  389  390  391  397  399  401  402  403  404  405  410  411  412  414  416  417  423  425  426  427  437  438  440  441
   4    5    3    3    6    6    3    3    6    3    6    3    6    3    5    5    3    5    5    3    1    5    5    1    6    3    1
 444  446  447  454  465  468  475  476  478  486  490  495  496  501  505  507  508  509  511  513  514  516  519  520  521  531  534
   6    3    5    3    1    4    3    5    5    3    5    5    5    3    5    5    5    3    4    3    1    1    1    1    1    5    6
 538  540  545  546  547  550  555  557  560  562  563  565  566  568  569  571  573  575  576  578  588  589  591  592  593  594  595
   1    6    6    1    1    3    6    5    3    3    6    6    1    6    1    6    5    3    4    3    3    6    3    6    5    5    3
 596  600  604  608  609  611  616  631  632  636  638  641  645  652  653  662  675  676  679  687  691  692  695  699  700  704  709
   5    5    6    6    1    6    6    6    6    1    1    1    6    1    6    3    5    5    5    3    3    1    1    6    6    6    6
 714  715  717  722  724  726  731  734  736  738  744  746  751  752  759  760  762  768  769  770  775  776  784  792  795  798  800
   1    1    1    5    5    5    3    3    3    3    5    5    6    6    5    5    5    1    3    5    3    3    1    6    4    6    6
 801  806  808  809  818  819  823  824  826  840  844  846  849  852  853  857  858  862  863  864  868  870  880  885  892  895  897
   6    6    5    6    1    1    1    1    1    6    3    3    3    3    3    4    3    4    4    4    4    4    4    4    2    1    1
 898  902  908  909  910  917  918  921  926  928  935  936  937  938  939  940  941  942  944  947  955  962  966  972  973  994  995
   1    1    1    1    1    1    6    6    6    6    6    6    6    6    6    6    6    6    6    6    6    3    3    3    3    3    4
 996  999 1009 1012 1013 1014 1016 1020 1024 1027 1031 1033 1034 1038 1039 1040 1045 1046 1052 1054 1059 1067 1069 1072 1077 1083 1084
   4    3    4    4    4    4    4    4    4    4    4    2    4    4    4    4    2    2    2    1    1    1    1    1    6    6    6
```

- Linear modeling by using glm function of R.

```
>
> # linear modelling using GLM
> abalone.train_70.glm <- glm(formula = rings ~ length + diameter + height + whole_weight + shucked_weight +  viscera_weight + shell_wei
ght, family = gaussian, data = abalone.train_70)
> summary(abalone.train_70.glm)

Call:
glm(formula = rings ~ length + diameter + height + whole_weight +
    shucked_weight + viscera_weight + shell_weight, family = gaussian,
    data = abalone.train_70)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5111  -0.4220  -0.1166   0.2718   4.3572

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.006158   0.012720   0.484 0.628339
length         -0.044591   0.080235  -0.556 0.578426
diameter        0.315420   0.081512   3.870 0.000111 ***
height          0.345964   0.036348   9.518  < 2e-16 ***
whole_weight    1.346626   0.129331  10.412  < 2e-16 ***
shucked_weight -1.344865   0.066020 -20.371  < 2e-16 ***
viscera_weight -0.358406   0.052756  -6.794 1.32e-11 ***
shell_weight    0.322172   0.056853   5.667 1.60e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.4723417)

    Null deviance: 2951.3  on 2922  degrees of freedom
Residual deviance: 1376.9  on 2915  degrees of freedom
AIC: 6112.7

Number of Fisher Scoring iterations: 2

>
```

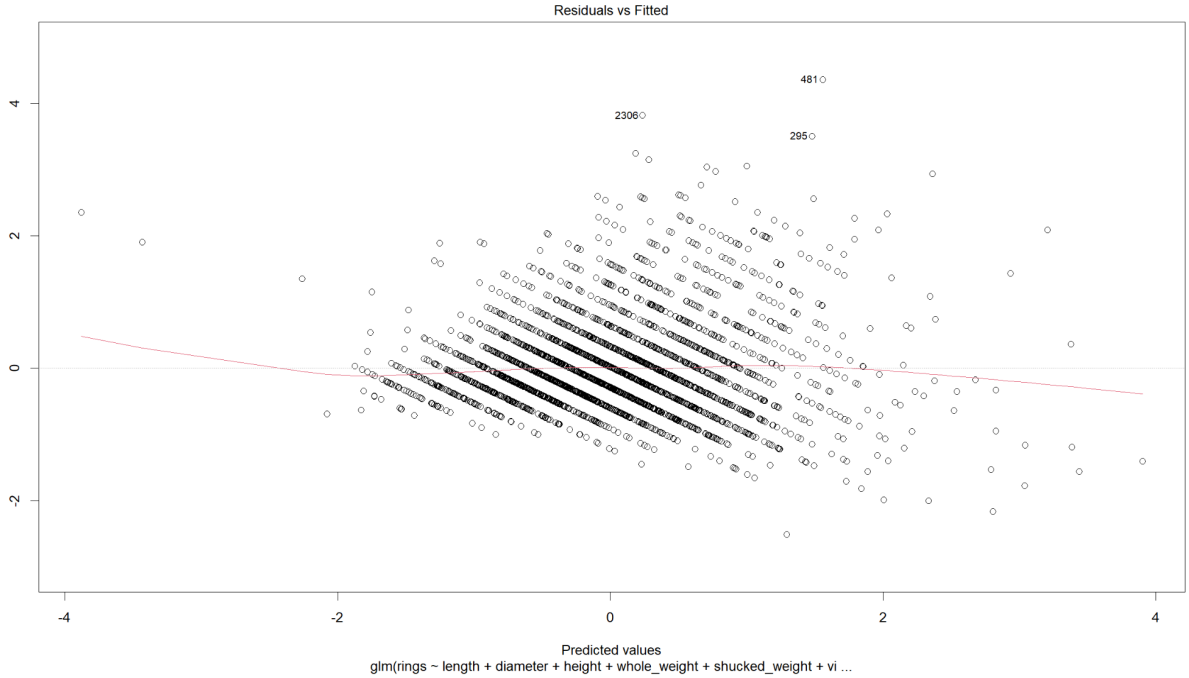
- Applying anova to the training data to get the deviance of the data

```
>
>
> # anova - analysis of variance on the model result
> abalone.train_70.glm.anova <- anova(abalone.train_70.glm, test = "Chisq")
> abalone.train_70.glm.anova
Analysis of Deviance Table

Model: gaussian, link: identity

Response: rings

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                           2922     2951.3
length          1   918.29      2921     2033.0 < 2.2e-16 ***
diameter        1    69.50      2920     1963.5 < 2.2e-16 ***
height          1   132.12      2919     1831.4 < 2.2e-16 ***
whole_weight    1     3.30      2918     1828.1  0.008244 **
shucked_weight  1   398.24      2917     1429.8 < 2.2e-16 ***
viscera_weight  1    37.78      2916     1392.0 < 2.2e-16 ***
shell_weight    1    15.17      2915     1376.9 1.455e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
>
>
```

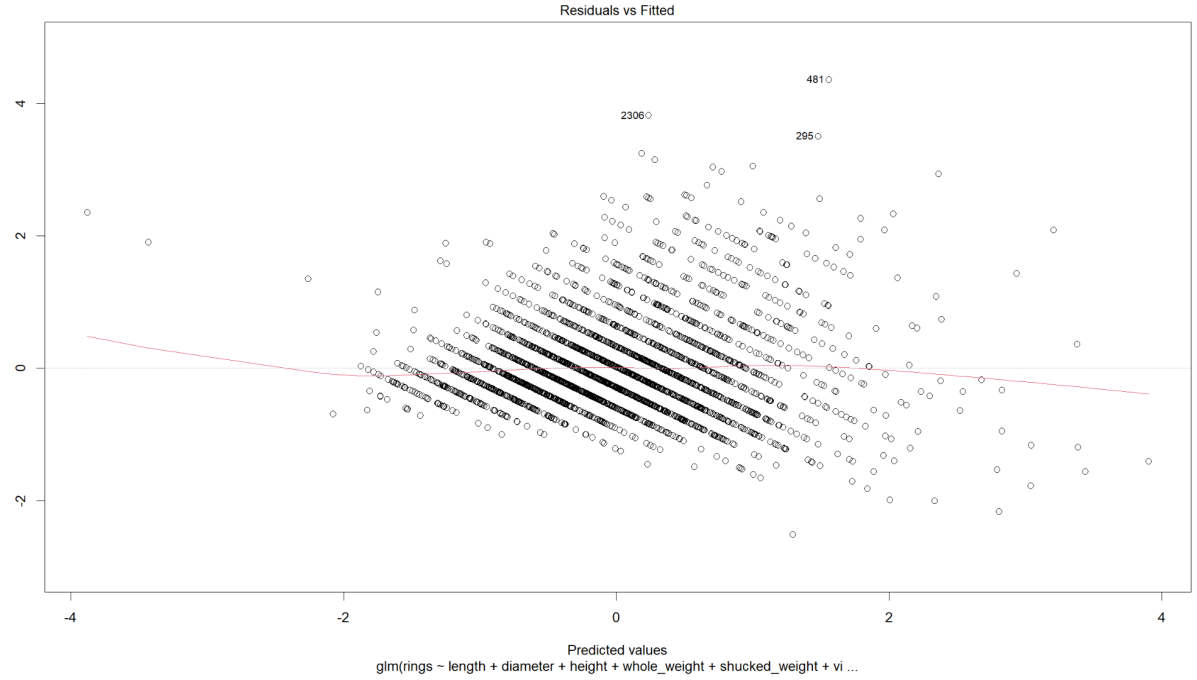
- Plotting graph

```
>
>
> plot(abalone.train_70.glm)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
>
>
>
```

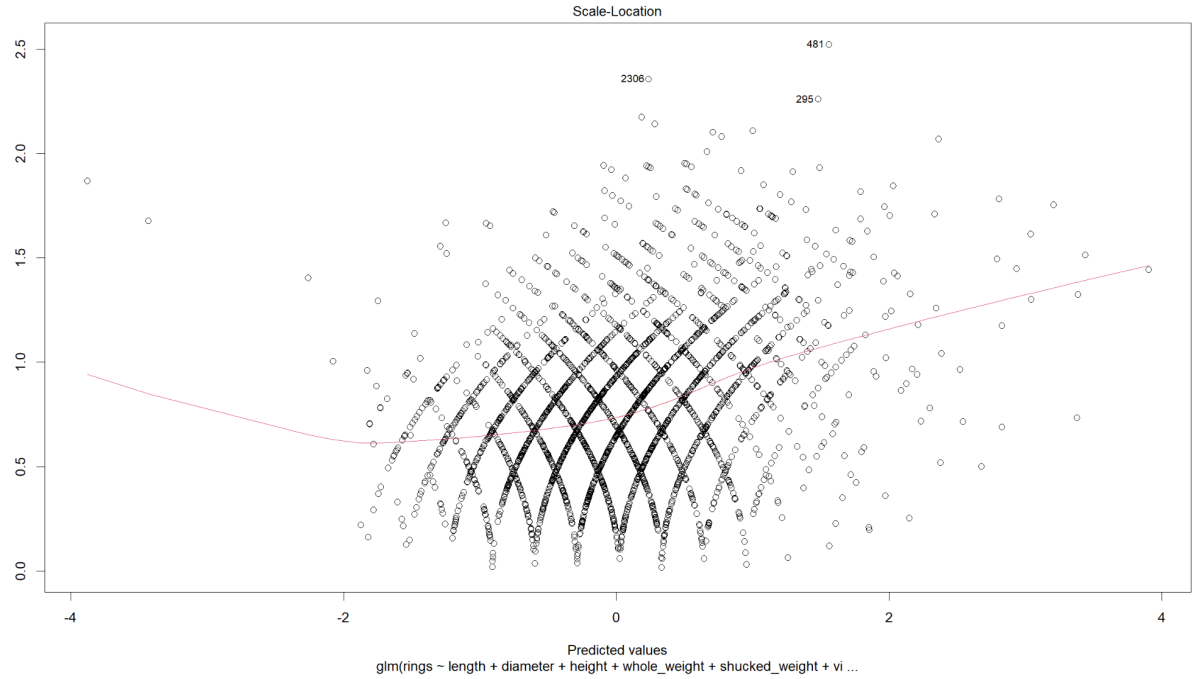
- Residual values vs Fitted values Plot



- Normal Q-Q Plot

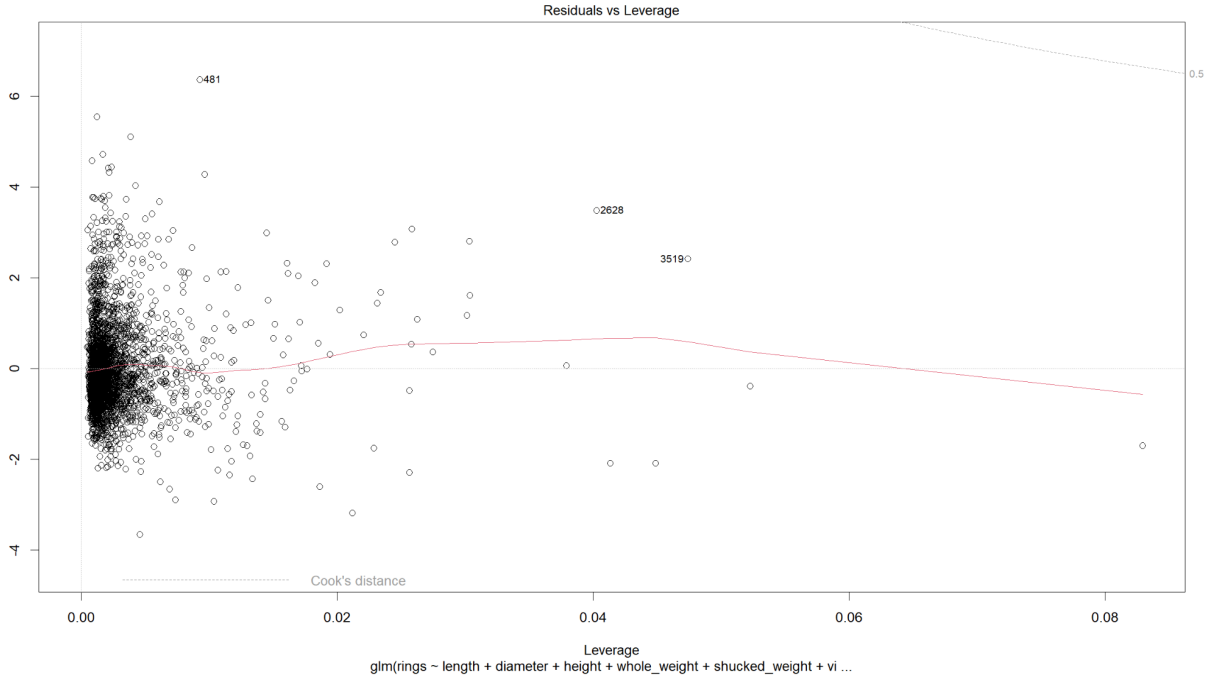


- Scale-Location Plot



-

- Residuals vs Leverage Plot



- Using predict function in R

```
>
> # predict
> abalone.test_70.predict <- predict(abalone.train_70.glm, newdata = abalone.test_70)
> length(abalone.test_70.predict)
[1] 1254
> summary(abalone.test_70.predict)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.840315 -0.494013 -0.037437 -0.005171  0.385751  7.544918
>
```

- Confidence intervals: Indicating the probability that the true parameter values lies within the range.

```
>
>
>
> # confidence intervals
> confint(abalone.train_70.glm)
Waiting for profiling to be done...
                    2.5 %      97.5 %
(Intercept)    -0.01877274  0.03108867
length         -0.20184914  0.11266788
diameter        0.15565992  0.47517991
height          0.27472396  0.41720438
whole_weight    1.09314051  1.60011067
shucked_weight -1.47426182 -1.21546872
viscera_weight -0.46180597 -0.25500544
shell_weight    0.21074271  0.43360187
>
>
>
> |
```

- Comparing predicted values

```
> abalone.test_70.predict.k7 <- kmeans(abalone.test_70.predict, centers = 7)
> abalone.test_70.predict.k7
K-means clustering with 7 clusters of sizes 109, 210, 300, 326, 198, 88, 23

Cluster means:
        [,1]
1 -1.1930596
2 -0.6985466
3 -0.2659461
4  0.1618756
5  0.6335121
6  1.3046820
7  2.4790537

Clustering vector:
   3    6   14   22   34   43   47   50   53   55   57   58   61   62   64   65   66   69   73   74   84   85   86   91   93   98  101
   4    2    4    2    5    1    2    4    3    2    3    3    2    2    2    3    2    3    6    5    7    5    7    4    6    3    2
 105  106  109  110  111  112  113  115  116  119  124  125  126  127  128  129  130  135  145  146  149  156  157  166  168  169  174
   5    4    4    3    3    3    2    3    3    5    1    2    1    2    2    6    7    1    4    3    1    4    5    6    7    7    3
 176  177  188  189  190  196  205  208  210  212  213  219  221  222  225  226  227  228  233  235  236  238  240  248  254  257  263
   2    1    5    4    5    4    3    3    2    2    3    4    3    4    3    4    3    2    6    3    1    1    1    2    5    6    5
 265  273  274  285  293  298  300  301  305  306  308  310  313  320  324  332  334  336  339  342  345  356  360  363  364  367  371
   1    6    6    6    5    1    2    2    3    1    7    4    6    2    1    3    1    5    4    6    5    7    5    5    4    4    5
 [ reached getOption("max.print") -- omitted 254 entries ]

Within cluster sum of squares by cluster:
[1]  5.448567  3.785932  4.042115  4.733147  5.161645  5.152337 27.728184
 (between_SS / total_SS =  92.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
> |
```

- Comparing results of kmeans and knn for 7 clusters using crosstable functionality

abalone.test_70.predict.k7$cluster	abalone.test_70_kmeans_k7$cluster							
	1	2	3	4	5	6	7	Row Total
1	105	2	1	0	0	1	0	109
	420.737	5.070	23.594	22.600	9.127	23.073	0.087	
	0.963	0.018	0.009	0.000	0.000	0.009	0.000	0.087
	0.507	0.020	0.003	0.000	0.000	0.003	0.000	
	0.084	0.002	0.001	0.000	0.000	0.001	0.000	
2	101	1	12	4	0	92	0	210
	126.938	14.639	28.159	35.908	17.584	39.723	0.167	
	0.481	0.005	0.057	0.019	0.000	0.438	0.000	0.167
	0.488	0.010	0.041	0.015	0.000	0.319	0.000	
	0.081	0.001	0.010	0.003	0.000	0.073	0.000	
3	1	8	99	38	0	154	0	300
	47.542	10.386	11.682	9.416	25.120	105.111	0.239	
	0.003	0.027	0.330	0.127	0.000	0.513	0.000	0.239
	0.005	0.081	0.337	0.146	0.000	0.535	0.000	
	0.001	0.006	0.079	0.030	0.000	0.123	0.000	
4	0	26	132	106	24	38	0	326
	53.813	0.003	40.402	21.825	0.398	18.157	0.260	
	0.000	0.080	0.405	0.325	0.074	0.117	0.000	0.260
	0.000	0.263	0.449	0.408	0.229	0.132	0.000	
	0.000	0.021	0.105	0.085	0.019	0.030	0.000	
5	0	21	46	85	43	3	0	198
	32.684	1.844	0.004	47.046	42.106	39.672	0.158	
	0.000	0.106	0.232	0.429	0.217	0.015	0.000	0.158
	0.000	0.212	0.156	0.327	0.410	0.010	0.000	
	0.000	0.017	0.037	0.068	0.034	0.002	0.000	
6	0	25	4	24	35	0	0	88
	14.526	46.909	13.407	1.815	103.618	20.211	0.070	
	0.000	0.284	0.045	0.273	0.398	0.000	0.000	0.070
	0.000	0.253	0.014	0.092	0.333	0.000	0.000	
	0.000	0.020	0.003	0.019	0.028	0.000	0.000	
7	0	16	0	3	3	0	1	23
	3.797	110.801	5.392	0.656	0.599	5.282	52.540	
	0.000	0.696	0.000	0.130	0.130	0.000	0.043	0.018
	0.000	0.162	0.000	0.012	0.029	0.000	1.000	
	0.000	0.013	0.000	0.002	0.002	0.000	0.001	
Column Total	207	99	294	260	105	288	1	1254
	0.165	0.079	0.234	0.207	0.084	0.230	0.001	

- Confusion matrix
- Calculating the metrics for k=7

```
> # confusion matrix
> abalone.test_70.cm_k7 <- abalone.test_70.ct.k7$t
> abalone.test_70.cm_k7
   y
x     1   2   3   4   5   6   7
  1 105   2   1   0   0   1   0
  2 101   1  12   4   0  92   0
  3   1   8  99  38   0 154   0
  4   0  26 132 106  24  38   0
  5   0  21  46  85  43   3   0
  6   0  25   4  24  35   0   0
  7   0  16   0   3   3   0   1
> # calculate the metrics
> abalone.test_70.accuracy <- sum(diag(abalone.test_70.cm_k7)) / sum(abalone.test_70.cm_k7)
> abalone.test_70.precision <- diag(abalone.test_70.cm_k7) / colSums(abalone.test_70.cm_k7)
> abalone.test_70.recall <- diag(abalone.test_70.cm_k7) / rowSums(abalone.test_70.cm_k7)
> abalone.test_70.specificity <- sapply(1:nrow(abalone.test_70.cm_k7), function(i){
+   TN <- sum(abalone.test_70.cm_k7[-i, -i])
+   FP <- sum(abalone.test_70.cm_k7[-i, i])
+   TN / (TN + FP)
+ })
> abalone.test_70.error <- 1 - abalone.test_70.accuracy
> # print results
> abalone.test_70.accuracy
[1] 0.2830941
> abalone.test_70.precision
         1          2          3          4          5          6          7
0.50724638 0.01010101 0.33673469 0.40769231 0.40952381 0.00000000 1.00000000
> abalone.test_70.recall
         1          2          3          4          5          6          7
0.963302752 0.004761905 0.330000000 0.325153374 0.217171717 0.000000000 0.043478261
> abalone.test_70.specificity
[1] 0.9109170 0.9061303 0.7955975 0.8340517 0.9412879 0.7530017 1.0000000
> abalone.test_70.error
[1] 0.7169059
```

Implementation for k=7 with 60-40 split

- Creating training labels with k-means

```
> abalone.train_60_k7 <- kmeans(abalone.train_60, centers = 7)
> abalone.train_60_k7
K-means clustering with 7 clusters of sizes 491, 383, 516, 565, 208, 137, 206

Cluster means:
       length    diameter      height whole_weight shucked_weight viscera_weight shell_weight      rings
1  0.84235321  0.83691779  0.67158743   0.81978683     0.85666453     0.83894533   0.71931411  0.1298498
2 -1.72481714 -1.73337231 -1.47503939  -1.35015230    -1.28539492    -1.31716003  -1.34233754 -1.1350314
3  0.29579934  0.28185708  0.12415177   0.04184917     0.09923679     0.03531055   0.01875683 -0.1801926
4 -0.58200566 -0.58841805 -0.56709559  -0.76600851    -0.72043939    -0.75783681  -0.75405984 -0.4800753
5 -0.07427569 -0.03920672  0.07018241  -0.22274672    -0.39830230    -0.20938683  -0.07480166  1.3282994
6  0.81859261  0.89220297  1.05164585   0.93737325     0.55254024     0.76060864   1.28981041  2.3003386
7  1.43528140  1.41862136  1.35377390   1.97073197     1.97739854     1.95014050   1.84074848  0.6363663

Clustering vector:
1089  870 2407 3354 3280  754 2813 2581 2468 3927 3767 1356   55 1280 3079 1557 3641  603 3951  819 1467 3346  910 3814 3445  986 3760
   4    1    3    4    6    1    2    3    5    5    3    1    4    4    7    4    4    4    5    2    3    5    2    2    4    1    3
1019 2809  444  666 3742  938 2866 1526  162 1962  843 1501 2786 1468 3957 1358  132 2994 3180 1700  125 2385 3082 3325 3386   65 1733
   1    7    4    4    1    4    2    7    1    1    4    1    1    4    5    3    4    7    2    1    2    4    7    4    5    4    1
 333  956 1684 3360 3756 2232 2464 2035 3436  299 4062 2594 2403 3171 3887 3090 3238 3363 2422 1482 3690 2882 3298 1618 3684 1193 3661
   2    4    1    6    4    3    5    6    2    4    3    1    5    5    4    4    6    4    5    3    1    4    6    3    1    7    3
1783  398  217 3832 1357  999 2788 1289 3798 1641 4002 1612  809 1161 3067 3154 2463  905 3176 2298 3801  124  490  482 3042 2490  165
   3    4    4    3    3    3    3    4    7    3    4    3    4    1    1    4    4    2    5    4    7    2    5    5    3    5    7
 362  706 1300  614 1855 1380  212  746  980 3880 3189 1266 1296 3263 2306 4012  528  788 2286 3137 3473  370  777 4006 4000 1768 1256
   1    4    4    5    4    1    2    5    3    6    7    4    3    1    5    1    5    5    4    3    2    6    5    3    4    4    4
 601 1226 3058 1114 1456 1534 3872 3091 3560 2944 1739   79  903 1372 3459 1980 3410 2867 3709 1682  537   18  993  371 1663 3550 2692
   5    2    1    3    4    2    4    4    1    1    7    3    2    1    1    7    4    2    7    1    5    4    1    6    3    4    1
3978  275 4046 3584 3221  164 3272 1652 3056 1607 2081 1776 3234 2024  269 2147 3184 3693   32 3340 3045  772  729  240 2011 1331 4166
```

```
 [ reached getOption("max.print") -- omitted 1506 entries ]

Within cluster sum of squares by cluster:
[1] 467.9879 438.8661 393.1555 438.8740 286.2247 322.7681 539.6066
 (between_SS / total_SS =  85.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

- Then the cluster vector is passed to KNN

The output is a vector which contains the predicted values for the test dataset.

```
> abalone.test_60_k7 <- knn(abalone.train_60, abalone.test_60, abalone.train_60_k7$cluster, k = 7)
> abalone.test_60_k7
   [1] 3 4 5 5 5 5 4 3 4 2 4 4 3 3 6 1 6 6 3 4 4 2 2 2 4 4 2 3 4 4 4 3 2 4 3 3 1 2 5 4 6 1 1 6 3 1 6 5 1 5 1 7 7 3 4 5 3 5 5 3 4 4 3 4 6 2 2
  [67] 2 4 2 3 6 4 2 1 1 6 1 7 7 3 2 2 2 1 1 1 1 3 1 3 3 6 5 5 5 4 4 4 5 2 3 3 4 4 4 4 4 4 4 4 2 5 6 4 2 2 2 2 2 6 6 2 5 2 4 6 2 4 5 3 3 6 2
 [133] 3 2 7 3 6 5 6 1 2 4 2 2 2 4 3 4 1 7 7 7 1 1 1 3 5 4 4 5 4 2 3 3 3 4 3 4 3 5 6 6 6 2 2 5 6 5 5 4 2 4 3 1 5 6 6 3 1 3 4 4 2 2 2 4 4 6
 [199] 5 3 6 3 5 1 6 6 1 6 1 3 3 3 4 2 2 2 2 2 2 4 4 4 2 4 4 2 3 3 5 4 3 5 3 5 2 5 5 1 6 1 5 4 3 5 4 5 5 3 3 5 1 4 2 5 2 2 2 5 5 5 2 6 2 4
 [265] 4 4 4 2 2 6 6 3 5 4 5 5 5 5 2 4 4 5 3 5 2 2 5 4 4 4 4 2 2 3 4 5 4 3 5 3 3 5 5 2 7 3 5 6 1 6 5 3 3 5 4 3 3 5 5 4 3 4 4 4 4 5 2 2 2 2 2
 [331] 2 2 2 2 4 4 4 4 4 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 7 6 7 7 7 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 4 4 3 3 3 3 3 3 3 3 4 3 3 3 1
 [397] 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 7 7 7 1 7 7 7 7 7 7 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3
 [463] 3 3 3 1 1 1 1 1 1 1 1 1 1 1 7 1 7 7 7 1 7 7 7 7 7 7 7 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 3 4 4 5 3 3 4 3 3 3 3 3 3 3
 [529] 3 3 3 3 1 1 3 3 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 7 1 1 1 7 7 7 7 7 7 7 2 2 4 2 4 4 2 4 4 4 4 4 4 4 4 4 4 4 3 3 3 3 3 1 3 1 1 1 1
 [595] 1 1 1 7 1 1 1 1 1 3 1 1 7 1 7 7 7 7 7 7 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 4 3 4 3 3 3 3 3 4 3 3 4 3 3 3 3 3
 [661] 3 1 1 3 3 1 3 3 3 3 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 7 1 1 7 1 1 1 1 1 7 1 1 1 7 7 7 7 7 7 7 7 7 7 7 2 4 4 4 4 3 3 3 3 3 3 3 1 1 1 1 1 1
 [727] 1 1 7 1 7 7 7 7 2 2 2 2 4 4 4 4 4 4 4 3 3 3 3 4 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 6 1 1 1 1 1 7 1
 [793] 7 7 1 7 7 2 2 2 2 2 2 2 2 2 4 4 4 4 4 3 3 3 2 2 2 4 4 6 4 4 3 4 3 3 3 3 3 1 7 1 4 5 4 5 4 5 4 1 4 2 2 4 4 2 2 1 3 4 3 6 4 2 2 4 7 4 2 3
 [859] 1 6 7 7 3 2 2 2 2 3 6 5 3 5 3 1 2 7 5 6 4 2 2 6 6 6 6 6 5 3 6 6 6 3 3 2 2 5 5 3 5 6 6 4 4 4 4 4 3 1 6 1 5 4 1 3 5 1 3 6 1 6 6 4 4 4
 [925] 4 4 2 5 3 5 4 3 1 5 4 2 5 4 5 4 4 5 5 4 5 7 7 6 6 3 2 5 3 5 6 1 6 6 3 4 2 4 5 2 2 3 3 2 1 6 1 6 5 5 2 1 1 2 5 4 2 2 3 5 5 5 5 5 3 4
 [991] 4 5 5 4 4 2 2 2 5 5
 [ reached getOption("max.print") -- omitted 671 entries ]
Levels: 1 2 3 4 5 6 7
```

- Applying KMeans to generate labels for the test records

```
> # apply kmeans test data to generate labels for the test records
> abalone.test_60_kmeans_k7 <- kmeans(abalone.test_60, centers = 7)
> abalone.test_60_kmeans_k7
K-means clustering with 7 clusters of sizes 75, 151, 367, 215, 371, 122, 370

Cluster means:
       length    diameter     height whole_weight shucked_weight viscera_weight shell_weight      rings
1  0.83941590  0.91547966  1.0156966  1.030320463     0.61295145     0.834195602   1.48198169  2.2702435
2  0.05499015  0.07940695  0.1865176 -0.093101732    -0.29749583    -0.125191267   0.12776360  1.4789244
3  0.85900564  0.85287481  0.7086063  0.849626462     0.87376970     0.866517685   0.76200962  0.1608575
4 -1.84230617 -1.83555592 -1.5424561 -1.391283330    -1.34026161    -1.358235627  -1.38902825 -1.1551448
5  0.24069259  0.23958544  0.1214355  0.001726771     0.04369445     0.004961567  -0.02491566 -0.2135128
6  1.43624887  1.43000825  1.5851766  2.061360387     2.07805567     2.057332409   1.80381146  0.5544458
7 -0.60565891 -0.61810557 -0.5871066 -0.784581737    -0.74586311    -0.763134031  -0.77113732 -0.4865811

Clustering vector:
   3    4    7    8   10   11   12   14   15   19   21   22   24   28   29   31   33   37   39   40   41   43   45   46   47   48   49
   5    7    2    2    2    2    7    5    7    4    7    7    5    5    2    3    1    1    5    7    7    4    4    4    7    7    4
  50   52   53   54   56   59   60   62   63   68   70   71   72   73   74   75   76   78   80   84   85   86   91   93   95   96   97
   5    7    7    7    5    4    7    5    5    1    4    2    7    1    3    3    2    5    3    1    2    3    2    3    1    6    5
  99  102  104  106  107  110  111  115  120  131  134  137  138  139  140  141  143  146  150  154  157  158  160  166  168  174  175
   7    2    5    2    5    7    7    5    7    1    4    4    4    7    7    5    1    7    4    3    3    1    2    6    1    5    4
 176  177  184  187  188  189  190  192  195  197  198  202  203  204  205  207  208  209  210  214  216  218  219  220  221  223  225
   4    4    3    3    3    3    5    3    5    5    1    2    2    2    7    7    7    2    4    5    5    7    7    7    7    7    7
 226  228  232  233  235  242  245  247  248  250  254  259  265  267  268  270  276  282  283  285  286  290  294  296  302  304  308
   7    4    2    1    7    4    4    4    4    4    1    1    4    2    7    7    1    4    7    2    5    5    1    4    2    4    6
 310  311  312  313  317  320  323  324  325  329  332  347  349  351  356  357  360  369  377  380  383  384  386  388  389  391  395
```

- Obtaining labels for the test data by using K-Means on the test data

```
> abalone.test_60_k7.labels <- abalone.test_60_kmeans_k7$cluster
> length(abalone.test_60_k7.labels)
[1] 1671
> abalone.test_60_k7.labels
   3    4    7    8   10   11   12   14   15   19   21   22   24   28   29   31   33   37   39   40   41   43   45   46   47   48   49
   5    7    2    2    2    2    7    5    7    4    7    7    5    5    2    3    1    1    5    7    7    4    4    4    7    7    4
  50   52   53   54   56   59   60   62   63   68   70   71   72   73   74   75   76   78   80   84   85   86   91   93   95   96   97
   5    7    7    7    5    4    7    5    5    1    4    2    7    1    3    3    2    5    3    1    2    3    2    3    1    6    5
  99  102  104  106  107  110  111  115  120  131  134  137  138  139  140  141  143  146  150  154  157  158  160  166  168  174  175
   7    2    5    2    5    7    7    5    7    1    4    4    4    7    7    5    1    7    4    3    3    1    2    6    1    5    4
 176  177  184  187  188  189  190  192  195  197  198  202  203  204  205  207  208  209  210  214  216  218  219  220  221  223  225
   4    4    3    3    3    3    5    3    5    5    1    2    2    2    7    7    7    2    4    5    5    7    7    7    7    7    7
 226  228  232  233  235  242  245  247  248  250  254  259  265  267  268  270  276  282  283  285  286  290  294  296  302  304  308
   7    4    2    1    7    4    4    4    4    4    1    1    4    2    7    7    1    4    7    2    5    5    1    4    2    4    6
 310  311  312  313  317  320  323  324  325  329  332  347  349  351  356  357  360  369  377  380  383  384  386  388  389  391  395
   5    1    2    1    1    4    7    4    4    7    7    5    7    3    6    6    6    3    3    3    5    7    7    7    2    7    4
 397  399  400  405  406  407  409  412  416  419  421  424  425  429  430  433  434  435  437  439  446  449  450  452  453  454  455
   5    2    5    7    5    7    5    2    2    1    1    4    4    2    1    2    2    7    4    7    5    3    2    1    1    5    3
 456  459  461  463  464  466  473  477  479  486  487  488  489  492  493  494  495  497  498  499  500  501  509  512  515  516  518
   5    7    7    4    4    4    7    7    1    2    5    1    5    2    3    1    1    3    1    3    5    5    5    7    4    4    4
 519  523  524  530  535  536  538  540  545  548  550  553  554  555  556  557  564  565  566  571  574  576  580  581  582  585  587
   4    4    4    7    7    7    4    7    7    4    5    5    2    7    5    2    5    2    4    2    2    3    1    3    2    7    5
 591  592  593  595  596  597  598  599  602  613  615  618  620  621  622  623  626  628  629  635  645  646  648  649  654  656  658
   2    7    2    2    2    5    2    3    7    4    2    4    4    4    2    2    2    7    2    7    7    7    7    7    4    4    1
 661  662  665  667  670  673  677  679  680  682  683  684  685  687  695  698  702  703  704  709  712  717  718  722  725  726  728
   1    5    2    7    2    2    2    2    4    7    7    2    5    2    4    4    2    7    7    7    7    4    4    2    7    2    7
 731  734  736  737  741  742  747  748  751  755  756  757  762  766  769  770  774  775  776  778  780  784  791  793  801  803  806
```

- Linear modeling by using glm function of R.

```
> abalone.train_60.glm <- glm(formula = rings ~ length + diameter + height + whole_weight + shucked_weight +  viscera_weight + shell_weight, fa
mily = gaussian, data = abalone.train_70)
> summary(abalone.train_60.glm)

Call:
glm(formula = rings ~ length + diameter + height + whole_weight +
    shucked_weight + viscera_weight + shell_weight, family = gaussian,
    data = abalone.train_70)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5111  -0.4220  -0.1166   0.2718   4.3572

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.006158   0.012720   0.484 0.628339
length         -0.044591   0.080235  -0.556 0.578426
diameter        0.315420   0.081512   3.870 0.000111 ***
height          0.345964   0.036348   9.518  < 2e-16 ***
whole_weight    1.346626   0.129331  10.412  < 2e-16 ***
shucked_weight -1.344865   0.066020 -20.371  < 2e-16 ***
viscera_weight -0.358406   0.052756  -6.794 1.32e-11 ***
shell_weight    0.322172   0.056853   5.667 1.60e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.4723417)

    Null deviance: 2951.3  on 2922  degrees of freedom
Residual deviance: 1376.9  on 2915  degrees of freedom
AIC: 6112.7

Number of Fisher Scoring iterations: 2
```

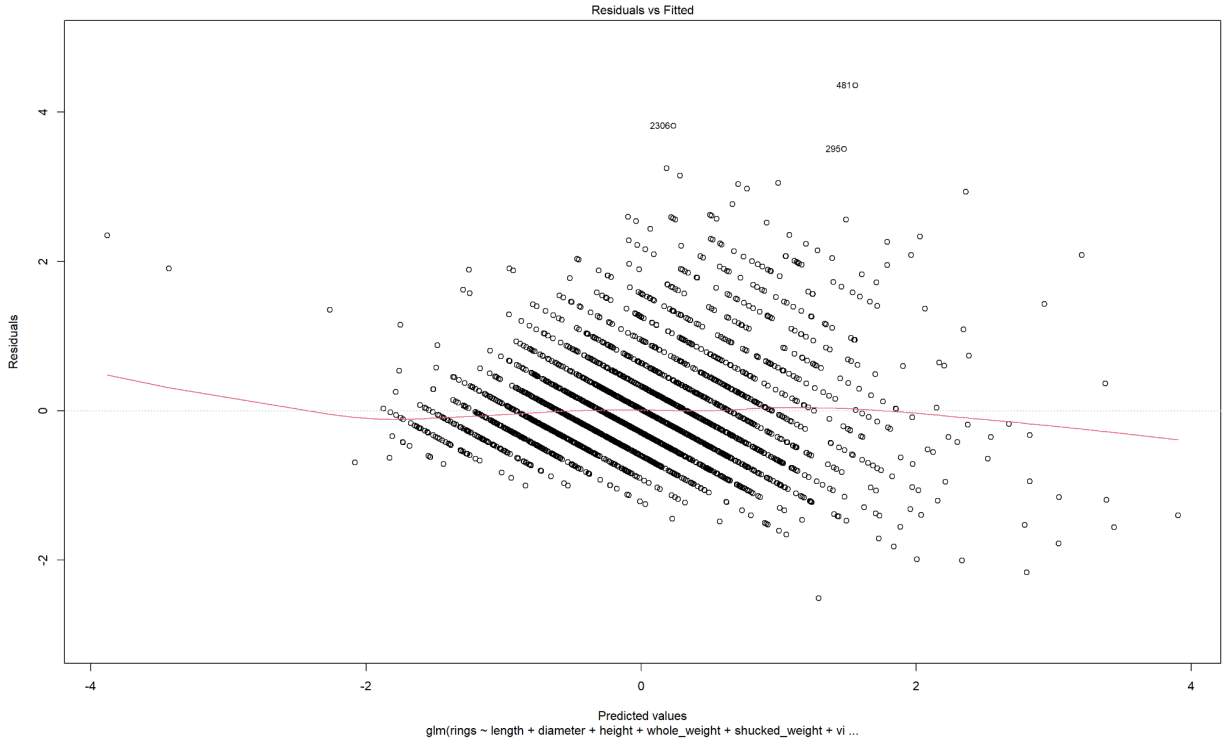
- Applying anova to the training data

```
> # anova - analysis of variance on the model result
> abalone.train_60.glm.anova <- anova(abalone.train_60.glm, test = "Chisq")
> abalone.train_60.glm.anova
Analysis of Deviance Table

Model: gaussian, link: identity

Response: rings

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                          2922     2951.3
length          1   918.29      2921     2033.0 < 2.2e-16 ***
diameter        1    69.50      2920     1963.5 < 2.2e-16 ***
height          1   132.12      2919     1831.4 < 2.2e-16 ***
whole_weight    1     3.30      2918     1828.1  0.008244 **
shucked_weight  1   398.24      2917     1429.8 < 2.2e-16 ***
viscera_weight  1    37.78      2916     1392.0 < 2.2e-16 ***
shell_weight    1    15.17      2915     1376.9 1.455e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

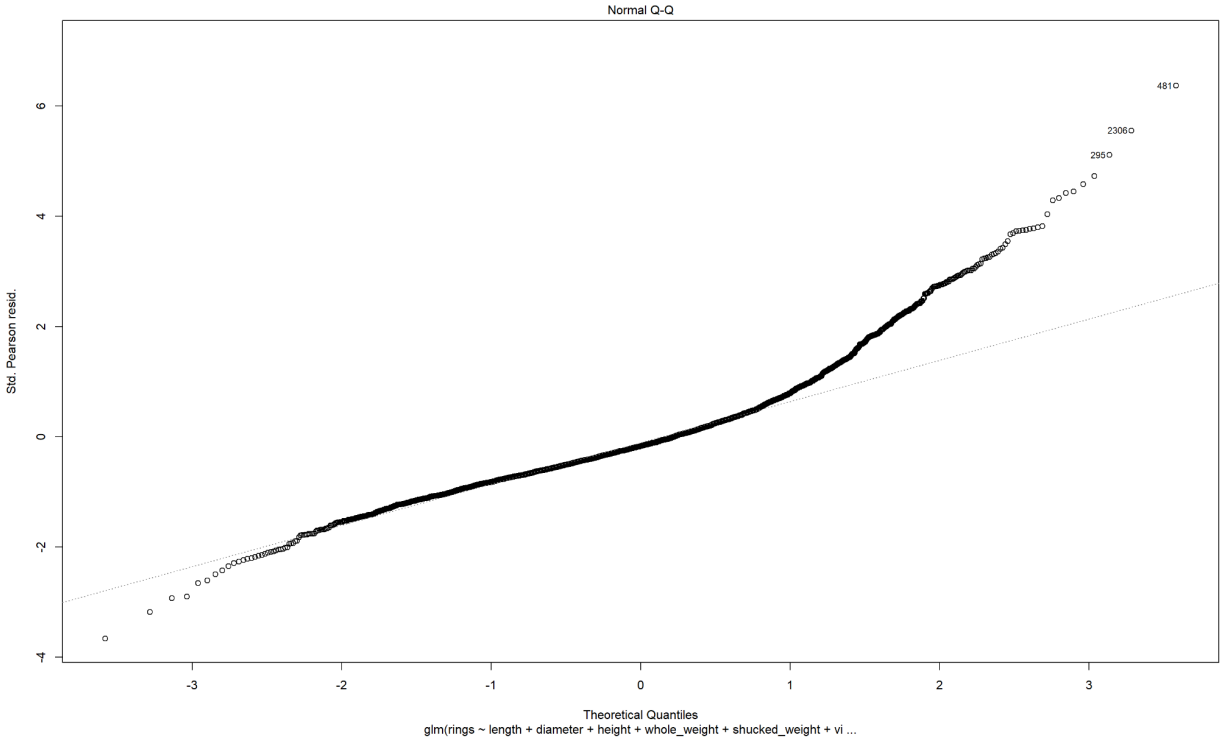
- Plotting the graphs

```
> # plotting graphs
> plot(abalone.train_60.glm)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
>
```

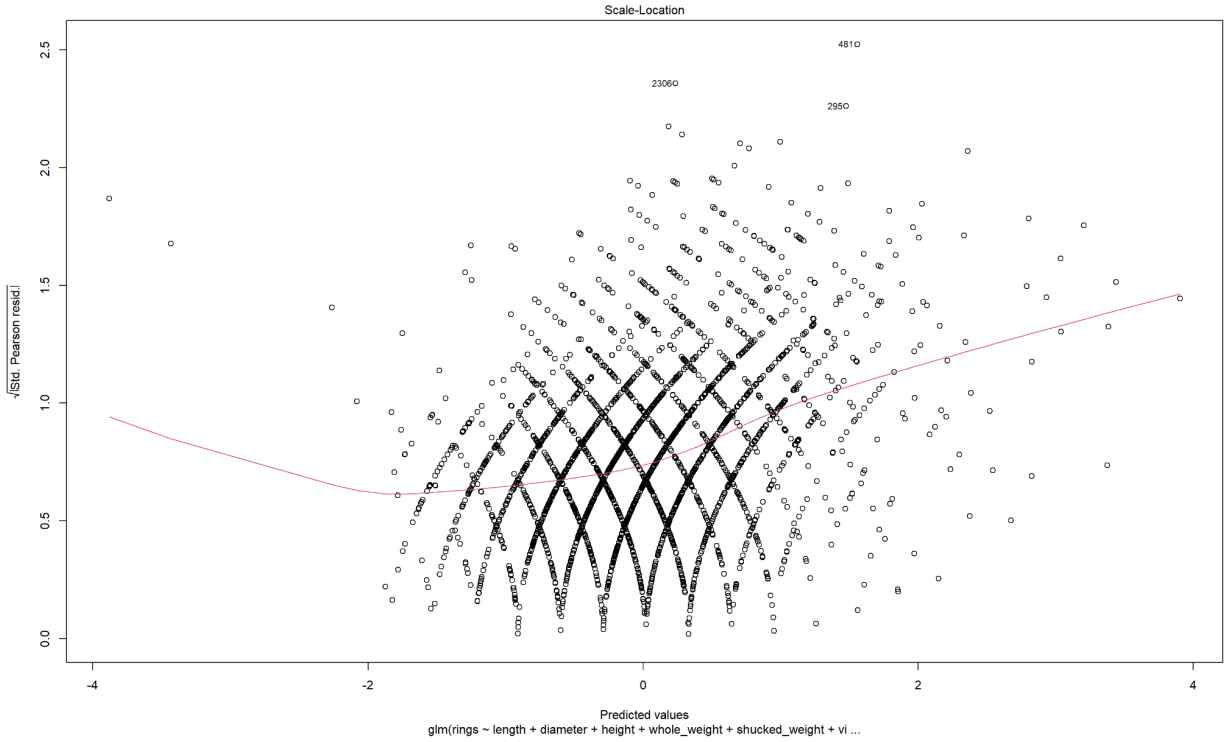
- Residual vs Fitted



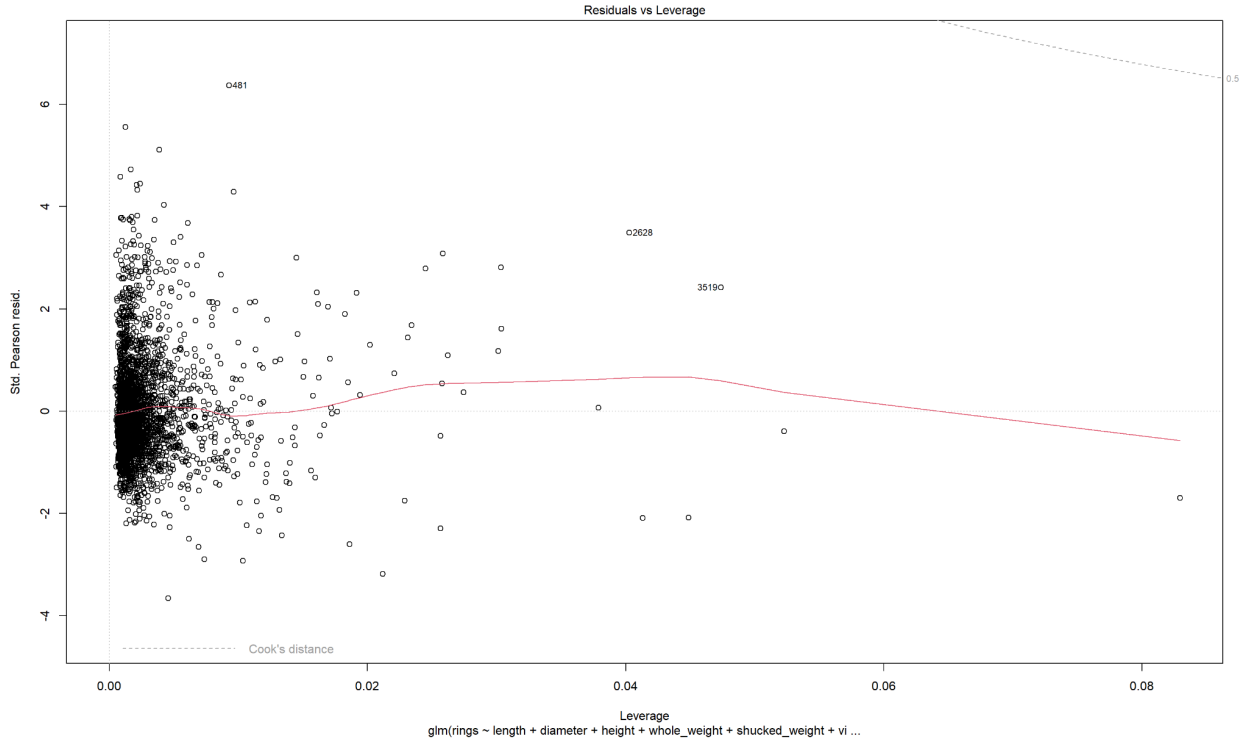
Normal Q-Q Plot



- Scale-Location Plot



Residuals vs Leverage Plot



- Using predict function in R
- Confidence intervals

```
> # predict
> abalone.test_60.predict <- predict(abalone.train_60.glm, newdata = abalone.test_60)
> length(abalone.test_60.predict)
[1] 1671
> summary(abalone.test_60.predict)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.873713 -0.434182 -0.000945  0.029797  0.461208  7.544918
> # confidence intervals
> confint(abalone.train_60.glm)
Waiting for profiling to be done...
                     2.5 %      97.5 %
(Intercept)    -0.01877274  0.03108867
length         -0.20184914  0.11266788
diameter        0.15565992  0.47517991
height          0.27472396  0.41720438
whole_weight    1.09314051  1.60011067
shucked_weight -1.47426182 -1.21546872
viscera_weight -0.46180597 -0.25500544
shell_weight    0.21074271  0.43360187
```

- Comparing the predicted values

```
> # comparing actual vs prediction
> abalone.test_60.predict.k7 <- kmeans(abalone.test_60.predict, centers = 7)
> abalone.test_60.predict.k7
K-means clustering with 7 clusters of sizes 265, 410, 119, 134, 30, 291, 422

Cluster means:
        [,1]
1 -0.7093000
2 -0.2556295
3 -1.2592845
4  1.2354772
5  2.4183205
6  0.6283278
7  0.1693641

Clustering vector:
   3    4    7    8   10   11   12   14   15   19   21   22   24   28   29   31   33   37   39   40   41   43   45   46   47   48   49
   7    2    4    7    6    7    2    7    2    1    1    1    2    7    6    6    6    7    7    1    1    3    3    1    1    2    3
  50   52   53   54   56   59   60   62   63   68   70   71   72   73   74   75   76   78   80   84   85   86   91   93   95   96   97
   7    1    2    2    7    3    2    1    2    5    3    6    1    4    6    7    7    6    6    5    6    5    7    4    4    4    6
  99  102  104  106  107  110  111  115  120  131  134  137  138  139  140  141  143  146  150  154  157  158  160  166  168  174  175
   2    7    7    7    6    2    2    2    1    6    1    3    1    2    1    7    4    2    3    4    6    5    5    4    5    2    3
 176  177  184  187  188  189  190  192  195  197  198  202  203  204  205  207  208  209  210  214  216  218  219  220  221  223  225
   1    3    6    7    6    7    6    6    1    7    4    6    7    6    2    2    2    4    1    7    6    2    7    2    2    7    2
 226  228  232  233  235  242  245  247  248  250  254  259  265  267  268  270  276  282  283  285  286  290  294  296  302  304  308
   7    1    4    5    2    3    1    1    1    1    6    6    3    7    1    2    4    1    2    4    6    6    4    3    4    1    5
 310  311  312  313  317  320  323  324  325  329  332  347  349  351  356  357  360  369  377  380  383  384  386  388  389  391  395
   7    4    7    4    6    1    1    3    1    2    2    6    2    7    5    6    6    6    7    4    2    2    1    2    7    1    1
 397  399  400  405  406  407  409  412  416  419  421  424  425  429  430  433  434  435  437  439  446  449  450  452  453  454  455
   2    6    7    2    7    2    7    7    6    4    4    3    3    6    4    4    6    2    1    1    4    6    7    6    6    7    6
 456  459  461  463  464  466  473  477  479  486  487  488  489  492  493  494  495  497  498  499  500  501  509  512  515  516  518
   7    7    6    6    7    1    7    7    6    6    6    6    7    6    2    5    2    7    1    5    7    7    1    7    6    6    7
2400 2402 2404 2408 2410 2418 2419 2420 2421 2425 2426 2427 2436 2437 2439 2441 2442 2443 2444 2446 2447 2451 2453 2455 2456 2457 2462
   7    7    3    6    6    3    2    2    1    3    6    6    7    4    6    2    6    2    2    7    4    2    2    3    3    3    6
2470
   7
 [ reached getOption("max.print") -- omitted 671 entries ]

Within cluster sum of squares by cluster:
[1]  5.488927  5.947029  5.261364  6.962943 29.596944  6.296052  6.335788
 (between_SS / total_SS =  92.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
>
```

- Comparing results of KMeans and KNN for 7 clusters using the crosstable functionality

Total Observations in Table: 1671

abalone.test_60.predict.k7$cluster	abalone.test_60_kmeans_k7$cluster 1	2	3	4	5	6	7	Row Total
1	0	0	7	99	13	4	142	265
	11.894	23.947	45.044	123.546	35.708	12.175	118.319	
	0.000	0.000	0.026	0.374	0.049	0.015	0.536	0.159
	0.000	0.000	0.019	0.460	0.035	0.033	0.384	
	0.000	0.000	0.004	0.059	0.008	0.002	0.085	
2	0	10	52	0	144	8	196	410
	18.402	19.749	16.076	52.753	30.824	16.072	121.942	
	0.000	0.024	0.127	0.000	0.351	0.020	0.478	0.245
	0.000	0.066	0.142	0.000	0.388	0.066	0.530	
	0.000	0.006	0.031	0.000	0.086	0.005	0.117	
3	0	0	0	116	0	3	0	119
	5.341	10.753	26.136	662.146	26.421	3.724	26.349	
	0.000	0.000	0.000	0.975	0.000	0.025	0.000	0.071
	0.000	0.000	0.000	0.540	0.000	0.025	0.000	
	0.000	0.000	0.000	0.069	0.000	0.002	0.000	
4	39	19	36	0	6	34	0	134
	180.909	3.922	1.467	17.241	18.961	59.943	29.671	
	0.291	0.142	0.269	0.000	0.045	0.254	0.000	0.080
	0.520	0.126	0.098	0.000	0.016	0.279	0.000	
	0.023	0.011	0.022	0.000	0.004	0.020	0.000	
5	16	1	2	0	0	11	0	30
	159.469	1.080	3.196	3.860	6.661	35.434	6.643	
	0.533	0.033	0.067	0.000	0.000	0.367	0.000	0.018
	0.213	0.007	0.005	0.000	0.000	0.090	0.000	
	0.010	0.001	0.001	0.000	0.000	0.007	0.000	
6	18	56	135	0	45	36	1	291
	1.868	33.553	79.070	37.442	5.951	10.246	62.450	
	0.062	0.192	0.464	0.000	0.155	0.124	0.003	0.174
	0.240	0.371	0.368	0.000	0.121	0.295	0.003	
	0.011	0.034	0.081	0.000	0.027	0.022	0.001	
7	2	65	135	0	163	26	31	422
	15.152	18.927	19.321	54.297	51.267	0.751	41.726	
	0.005	0.154	0.320	0.000	0.386	0.062	0.073	0.253
	0.027	0.430	0.368	0.000	0.439	0.213	0.084	
	0.001	0.039	0.081	0.000	0.098	0.016	0.019	
Column Total	75	151	367	215	371	122	370	1671
	0.045	0.090	0.220	0.129	0.222	0.073	0.221	

- Confusion Matrix
- Calculating the metrics for k=7

```
> # confusion matrix
> abalone.test_60.cm_k7 <- abalone.test_60.ct.k7$t
> abalone.test_60.cm_k7
   y
x     1   2   3   4   5   6   7
  1   0   0   7  99  13   4 142
  2   0  10  52   0 144   8 196
  3   0   0   0 116   0   3   0
  4  39  19  36   0   6  34   0
  5  16   1   2   0   0  11   0
  6  18  56 135   0  45  36   1
  7   2  65 135   0 163  26  31
> # calculate the metrics
> abalone.test_60.accuracy <- sum(diag(abalone.test_60.cm_k7)) / sum(abalone.test_60.cm_k7)
> abalone.test_60.precision <- diag(abalone.test_60.cm_k7) / colSums(abalone.test_60.cm_k7)
> abalone.test_60.recall <- diag(abalone.test_60.cm_k7) / rowSums(abalone.test_60.cm_k7)
> abalone.test_60.specificity <- sapply(1:nrow(abalone.test_60.cm_k7), function(i){
+   TN <- sum(abalone.test_60.cm_k7[-i, -i])
+   FP <- sum(abalone.test_60.cm_k7[-i, i])
+   TN / (TN + FP)
+ })
> abalone.test_60.error <- 1 - abalone.test_60.accuracy
> # print results
> abalone.test_60.accuracy
[1] 0.04608019
> abalone.test_60.precision
         1          2          3          4          5          6          7
0.00000000 0.06622517 0.00000000 0.00000000 0.00000000 0.29508197 0.08378378
> abalone.test_60.recall
         1          2          3          4          5          6          7
0.00000000 0.02439024 0.00000000 0.00000000 0.00000000 0.12371134 0.07345972
> abalone.test_60.specificity
[1] 0.9466572 0.8881840 0.7635309 0.8601171 0.7739183 0.9376812 0.7285829
> abalone.test_60.error
[1] 0.9539198
```

Implementation for k=7 with 50-50 split

- Creating training labels with k-means

```
> abalone.train_50_k7 <- kmeans(abalone.train_50, centers = 7)
> abalone.train_50_k7
K-means clustering with 7 clusters of sizes 373, 152, 177, 383, 416, 424, 163

Cluster means:
      length   diameter     height whole_weight shucked_weight viscera_weight shell_weight       rings
1 -1.0228687 -1.0303443 -0.9229641   -1.0524108     -1.0030908     -1.0459398   -1.0379183 -0.68705635
2 -2.1497499 -2.1229100 -1.7870409   -1.4808907     -1.4121417     -1.4371108   -1.4766330 -1.39554530
3  1.4230224  1.4161328  1.3947122    1.9832823      1.9219870      1.9419375    1.8871577  0.84239785
4  0.9066310  0.8946985  0.7350433    0.9097496      0.9469596      0.9116625    0.7988574  0.14851832
5  0.3658883  0.3564692  0.2397221    0.1551886      0.1788904      0.1574371    0.1458230 -0.07262809
6 -0.2647842 -0.2717984 -0.2497905   -0.5040577     -0.4954647     -0.4875844   -0.4690423 -0.06355463
7  0.4604966  0.5134338  0.6290622    0.4293953      0.1218934      0.3329155    0.7113726  2.07559912

Clustering vector:
3249 1914 2330 3683 2112 2631 1887 2158 1298  370 4040 3101 2605 1524 2007 1909 2645  969  976 1179 2845  277 1891  279 3758 4088 2555
   4    5    6    4    1    1    5    3    6    3    5    5    4    3    1    5    6    6    5    4    4    4    5    7    6    5    2
2836  562 3454 2812 1799 1227  486 2381 2383 3975  674 2370 2458 1045  949 1449 1405 2933  646 1260 2440  617  644  183 3146 2232 4095
   5    6    5    3    4    1    5    2    5    1    7    5    2    3    6    6    4    5    1    1    1    6    1    5    5    5    4
2442  417 2753 4131 2101 1804 1879  567 2881  435 1529 1572 1169 3861  849 1437   83 1330 3362  708 2629  275 3242 1623 3159 2750 3963
   5    7    6    5    1    5    5    6    6    6    3    6    5    7    5    1    7    5    1    1    2    4    3    5    5    6    3
 928 1589  701  327  222 1063 3506 1047 1590  161 1632 3777 3972 3639 1444 3125  972 3650 1881 3870 3043 2876 2655  820 2142  950 2194
   1    6    1    1    6    2    4    3    6    7    5    5    1    1    1    5    6    6    5    6    5    1    5    1    2    6    2
 561 3264 3465 1351  407 2193 3063 2719 2509 3340 2615 1706 1465 2538 1079 2262  825 1573 1557  492 2296 3697 2610  572  536 1284 2300
   1    7    4    5    6    7    4    1    1    5    4    3    6    3    1    5    1    6    1    5    5    3    4    6    6    6    6
3114 2424 2700 1320 2362 2941  187 2729  675 3670   21 2257 1844 1356  203 3607  911 2847  412 1681 2846 3913 1872   91  604 3930 3200
   1    1    4    5    5    5    4    6    7    5    1    6    1    5    6    6    2    4    7    4    4    6    5    7    6    3    6
 121 2940 4087 3545 3998 3914 2934 1594 3041  295 3628 2204 1683 1603 2533 4005 4013 1794 2350 2823 2991  721 2669 1512 4003  982 2553
   6    5    5    6    1    5    4    6    5    7    3    7    4    5    4    6    5    5    6    1    5    2    5    4    6    5    2
```

```
 [ reached getOption("max.print") -- omitted 1088 entries ]

Within cluster sum of squares by cluster:
[1] 243.0035 139.6985 509.0077 369.4773 316.4782 392.2801 337.2894
 (between_SS / total_SS =  85.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
> |
```

- Then the cluster vector is passed to KNN

  The output is a vector which contains the predicted values for the test dataset.

```
> abalone.test_50_k7 <- knn(abalone.train_50, abalone.test_50, abalone.train_50_k7$cluster, k = 7)
> abalone.test_50_k7
   [1] 6 1 6 1 1 1 5 5 4 7 4 7 3 6 7 6 1 7 2 2 6 2 5 6 6 2 6 6 4 2 5 4 7 5 4 5 7 7 7 6 5 5 4 4 3 3 6 6 1 7 6 4 5 6 6 6 6 6 6 6 6 1 5 1 1 7
  [67] 1 1 2 1 3 3 1 2 1 2 1 1 5 6 2 2 4 5 3 7 4 3 3 3 6 1 2 1 2 5 7 7 3 4 5 4 6 7 5 1 5 6 1 6 6 6 6 6 6 1 1 5 7 5 7 7 2 2 2 7 1 2 4 5 7 7
 [133] 5 5 5 6 7 1 6 6 7 7 3 6 7 6 7 6 6 5 7 4 7 7 6 1 5 1 1 6 7 7 7 4 2 2 1 6 6 1 2 3 7 4 4 4 6 6 5 1 4 5 7 3 3 3 7 7 7 5 4 3 4 5 6 5 5 6
 [199] 6 1 1 6 6 5 6 5 5 5 7 5 4 7 7 6 6 2 7 7 7 7 6 1 1 7 1 1 5 7 4 7 3 7 3 5 1 2 2 4 3 5 6 1 5 7 6 3 7 7 7 5 5 4 4 7 7 4 4 5 7 4 7 7 7 6
 [265] 2 2 2 1 2 1 2 2 2 1 7 6 6 6 6 2 1 6 6 1 1 2 5 5 7 4 6 6 7 4 4 1 1 6 4 5 5 7 4 7 1 5 6 1 5 7 7 7 5 5 5 7 1 6 1 2 1 6 6 6 2 2 7 6 1 7 6
 [331] 4 2 6 6 2 2 1 5 3 5 6 1 6 1 6 6 5 6 6 7 7 5 2 2 2 2 2 6 6 1 1 6 1 2 2 1 2 1 2 5 7 6 6 6 5 6 7 6 6 1 7 7 7 7 3 7 7 4 7 7 7 7 7 4 5 5 1
 [397] 7 5 6 1 5 6 6 6 6 2 6 5 5 5 5 1 5 1 1 6 6 1 6 2 2 2 2 2 1 1 1 1 1 1 6 1 6 6 6 6 6 6 6 5 6 5 5 5 5 5 5 5 5 5 5 4 4 4 4 4 4 4 4 4 4 4
 [463] 4 4 4 4 3 4 3 3 2 2 2 2 2 2 2 2 2 1 1 1 1 6 1 1 1 1 6 6 1 6 6 6 6 6 6 6 6 6 6 6 6 5 6 6 5 5 4 5 5 5 5 4 5 5 5 5 4 4 4 4 4 4 4 4 4 4
 [529] 4 4 4 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 6 6 6 6 6 6 5 6 6 6 6 6 5 6 5 5 5 5 5 5 5 5 5 5 5 5 5 4 4 5
 [595] 4 5 5 4 4 4 5 4 4 4 4 4 4 3 4 3 3 3 3 3 3 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 6 1 6 6 1 6 6 6 6 6 6 6 6 6 6 6 6 5
 [661] 6 6 5 5 5 5 5 6 5 5 5 5 5 5 4 5 5 5 5 5 5 5 5 5 4 5 4 5 4 5 5 4 5 4 5 5 4 5 5 4 5 5 4 5 5 4 4 4 4 4 4 4 4 4 4 4 4 5 3 4 4 4 4 4 4
 [727] 4 3 4 3 4 3 3 3 3 2 2 1 1 1 1 1 1 1 1 1 6 1 6 6 6 6 6 6 5 6 6 6 5 5 5 4 5 5 5 4 5 4 5 4 5 4 4 4 4 4 4 4 4 3 3 3 3 3 3 2 2 2 2 2 2 1 1 1
 [793] 1 1 1 1 1 1 1 1 1 1 6 1 6 6 6 6 6 6 6 6 6 6 6 5 6 6 6 6 5 6 5 6 6 6 6 6 5 6 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 5 5 5 5 4 5 5 5 5 5 5 5 4
 [859] 4 5 5 5 5 4 4 4 4 4 5 5 4 4 4 4 4 4 4 4 3 4 4 4 4 4 4 4 4 4 3 4 4 4 4 3 4 4 4 3 4 3 3 3 3 3 3 3 3 3 2 6 1 6 5 5 5 6 5 5 4 5 4 5 5
 [925] 4 4 4 4 4 5 3 4 4 4 4 4 3 3 3 2 2 2 1 1 1 1 1 1 1 1 1 6 6 6 6 6 6 6 6 6 6 6 5 6 5 5 6 6 5 5 5 5 5 5 5 5 5 5 4 5 5 5 5 5 5 4 4 4
 [991] 5 4 5 4 4 4 4 4 5 4
 [ reached getOption("max.print") -- omitted 1089 entries ]
Levels: 1 2 3 4 5 6 7
> |
```

- Applying KMeans to generate labels for the test records

```
> abalone.test_50_kmeans_k7 <- kmeans(abalone.test_50, centers = 7)
> abalone.test_50_kmeans_k7
K-means clustering with 7 clusters of sizes 321, 441, 92, 148, 181, 454, 452

Cluster means:
       length    diameter      height whole_weight shucked_weight viscera_weight shell_weight      rings
1 -1.750759286 -1.75963360 -1.5033543  -1.35983485    -1.29968878    -1.32719879 -1.358934485 -1.1369663
2  0.284539404  0.28046220  0.1088744   0.01824507     0.07614943     0.01985356 -0.007190433 -0.1981595
3  0.854657723  0.92112401  1.4096584   1.01296512     0.56135451     0.82174416  1.439999318  2.3636054
4  1.468692184  1.46237100  1.3652699   2.10089220     2.12596693     2.04443272  1.930712663  0.6387875
5  0.006092427  0.04111323  0.1588403  -0.12838303    -0.31548034    -0.11980451  0.032291819  1.4479779
6  0.849701623  0.85344571  0.6990477   0.84843439     0.86980882     0.87239256  0.762781744  0.1817955
7 -0.557909270 -0.57076172 -0.5578392  -0.75282230    -0.70519827    -0.73588332 -0.745849362 -0.4933873

Clustering vector:
   1    6    9   18   19   20   23   24   25   29   32   33   34   36   37   38   40   42   43   44   48   49   50   54   57   59   60
   5    7    7    7    1    7    2    2    6    5    3    3    4    7    3    7    1    5    1    1    7    1    2    7    7    1    7
  65   68   70   71   75   76   79   80   81   85   86   87   89   90   92   93   94   95   96   98   99  101  102  104  105  107  108
   7    6    1    5    6    5    2    6    2    5    6    6    7    2    5    6    6    3    4    7    7    1    5    2    6    2    2
 109  110  111  112  115  116  117  118  119  120  122  123  124  125  126  127  129  130  133  135  136  137  138  140  141  146  148
   2    7    7    7    2    2    7    7    2    7    1    5    1    1    1    1    3    4    1    1    7    1    1    1    2    7    1
 149  152  154  158  160  163  166  167  169  174  176  177  178  179  180  181  182  185  189  190  191  197  199  202  205  209  211
   1    6    6    3    5    6    4    4    4    2    1    1    1    1    2    5    3    4    6    2    6    2    5    5    7    5    7
 212  214  215  217  219  224  226  227  228  229  230  231  232  233  236  238  240  241  247  249  252  253  254  257  261  262  263
   1    2    5    7    7    7    7    7    1    6    5    5    5    3    1    1    1    5    1    1    6    6    3    3    2    2    2
 266  267  268  269  270  271  273  278  280  281  283  285  287  289  290  291  292  293  294  299  301  302  303  304  305  313  314
  ⌄
 [ reached getOption("max.print") -- omitted 1089 entries ]

Within cluster sum of squares by cluster:
[1] 341.1866 331.1311 739.6720 450.2233 255.0627 475.4506 348.0893
 (between_SS / total_SS =  82.9 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

- Obtaining labels for the test data by using K-Means on the test data

```
> abalone.test_50_k7.labels <- abalone.test_50_kmeans_k7$cluster
> length(abalone.test_50_k7.labels)
[1] 2089
> abalone.test_50_k7.labels
   1    6    9   18   19   20   23   24   25   29   32   33   34   36   37   38   40   42   43   44   48   49   50   54   57   59   60
   5    7    7    7    1    7    2    2    6    5    3    3    4    7    3    7    1    5    1    1    7    1    2    7    7    1    7
  65   68   70   71   75   76   79   80   81   85   86   87   89   90   92   93   94   95   96   98   99  101  102  104  105  107  108
   7    6    1    5    6    5    2    6    2    5    6    6    7    2    5    6    6    3    4    7    7    1    5    2    6    2    2
 109  110  111  112  115  116  117  118  119  120  122  123  124  125  126  127  129  130  133  135  136  137  138  140  141  146  148
   2    7    7    7    2    2    7    7    2    7    1    5    1    1    1    1    3    4    1    1    7    1    1    1    2    7    1
 149  152  154  158  160  163  166  167  169  174  176  177  178  179  180  181  182  185  189  190  191  197  199  202  205  209  211
   1    6    6    3    5    6    4    4    4    2    1    1    1    1    2    5    3    4    6    2    6    2    5    5    7    5    7
 212  214  215  217  219  224  226  227  228  229  230  231  232  233  236  238  240  241  247  249  252  253  254  257  261  262  263
   1    2    5    7    7    7    7    7    1    6    5    5    5    3    1    1    1    5    1    1    6    6    3    3    2    2    2
 266  267  268  269  270  271  273  278  280  281  283  285  287  289  290  291  292  293  294  299  301  302  303  304  305  313  314
   7    5    1    7    7    3    3    3    2    5    7    5    2    7    2    5    6    5    3    7    7    5    1    1    7    3    3
 315  317  320  322  326  328  331  332  333  335  337  339  341  342  344  346  347  349  350  354  355  356  357  360  363  364  366
   3    3    1    1    1    5    5    7    1    4    6    6    6    3    5    5    2    7    6    2    3    4    4    4    3    5    3
 368  374  376  377  378  384  385  387  389  392  394  396  397  398  400  401  402  406  410  411  413  415  416  419  422  423  425
   2    6    3    6    6    7    2    5    5    7    1    7    2    7    2    7    2    2    2    3    2    6    5    3    7    5    1
 426  427  430  431  440  441  442  443  444  445  446  448  449  450  451  452  453  456  459  464  466  468  469  471  472  473  475
   6    3    3    5    5    7    1    5    7    7    2    5    6    5    4    3    3    2    7    1    1    6    3    2    2    7    2
 476  477  481  482  483  484  485  487  488  493  494  496  497  499  501  502  504  507  508  510  513  514  515  516  517  519  520
   5    7    3    5    5    5    6    2    6    6    3    5    6    6    2    3    6    5    3    5    2    1    1    1    7    1    1
 524  525  526  530  531  532  533  535  537  539  540  542  543  544  546  547  549  550  551  552  554  556  557  558  559  566  570
   1    1    1    7    5    7    7    7    2    1    7    7    7    7    1    1    2    2    3    6    5    2    5    6    6    1    7
 571  576  577  579  580  581  582  585  587  589  590  591  593  594  596  598  599  601  603  606  610  612  613  614  615  616  618
   5    6    2    2    3    6    5    7    2    5    7    5    5    3    5    5    6    5    7    7    1    1    1    5    5    7    1
 620  622  626  627  629  640  642  647  648  650  651  654  655  660  661  662  663  664  665  666  667  668  669  670  671  679  684
```

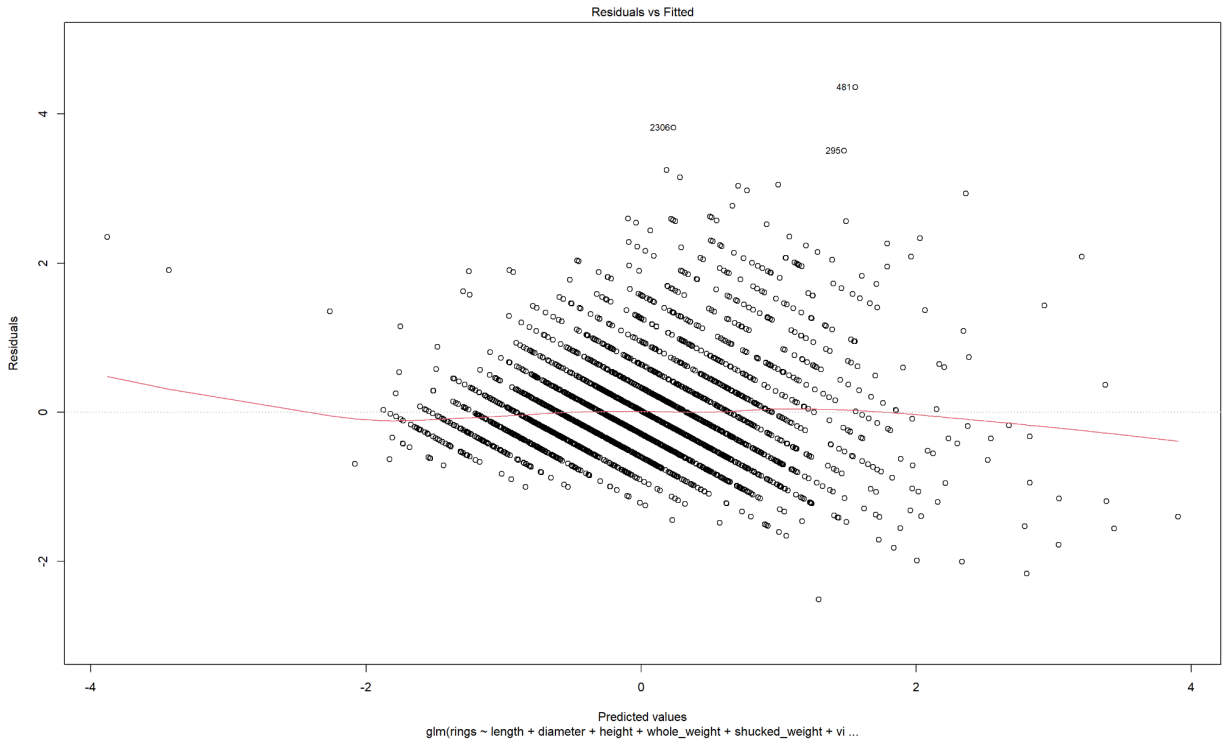
- Linear modeling using glm()

```
> # linear modelling using GLM
> abalone.train_50.glm <- glm(formula = rings ~ length + diameter + height + whole_weight + shucked_weight +  viscera_weight + shell_weight, fa
mily = gaussian, data = abalone.train_70)
> summary(abalone.train_50.glm)

Call:
glm(formula = rings ~ length + diameter + height + whole_weight +
    shucked_weight + viscera_weight + shell_weight, family = gaussian,
    data = abalone.train_70)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5111  -0.4220  -0.1166   0.2718   4.3572

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.006158   0.012720   0.484 0.628339
length         -0.044591   0.080235  -0.556 0.578426
diameter        0.315420   0.081512   3.870 0.000111 ***
height          0.345964   0.036348   9.518  < 2e-16 ***
whole_weight    1.346626   0.129331  10.412  < 2e-16 ***
shucked_weight -1.344865   0.066020 -20.371  < 2e-16 ***
viscera_weight -0.358406   0.052756  -6.794 1.32e-11 ***
shell_weight    0.322172   0.056853   5.667 1.60e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.4723417)

    Null deviance: 2951.3  on 2922  degrees of freedom
Residual deviance: 1376.9  on 2915  degrees of freedom
AIC: 6112.7

Number of Fisher Scoring iterations: 2
```

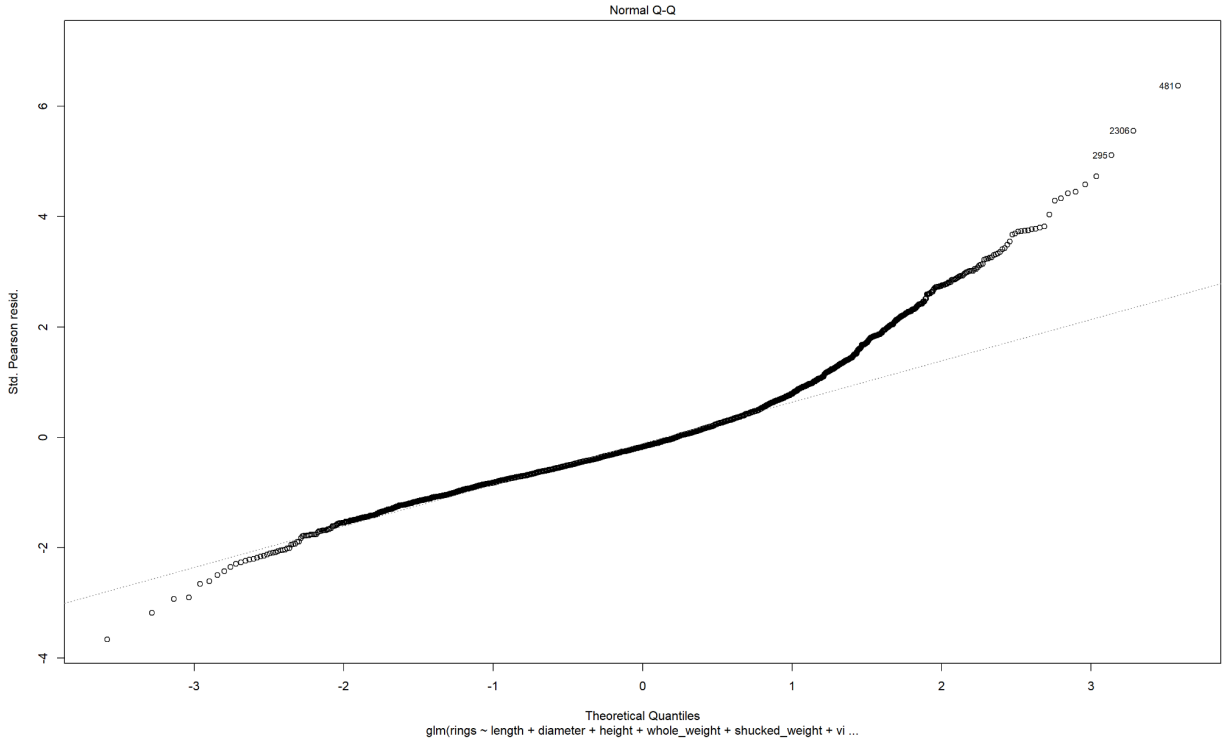
- Applying anova to the training data to get the deviance of the data

```
> abalone.train_50.glm.anova <- anova(abalone.train_50.glm, test = "Chisq")
> abalone.train_50.glm.anova
Analysis of Deviance Table

Model: gaussian, link: identity

Response: rings

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                          2922     2951.3
length          1   918.29      2921     2033.0 < 2.2e-16 ***
diameter        1    69.50      2920     1963.5 < 2.2e-16 ***
height          1   132.12      2919     1831.4 < 2.2e-16 ***
whole_weight    1     3.30      2918     1828.1  0.008244 **
shucked_weight  1   398.24      2917     1429.8 < 2.2e-16 ***
viscera_weight  1    37.78      2916     1392.0 < 2.2e-16 ***
shell_weight    1    15.17      2915     1376.9 1.455e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

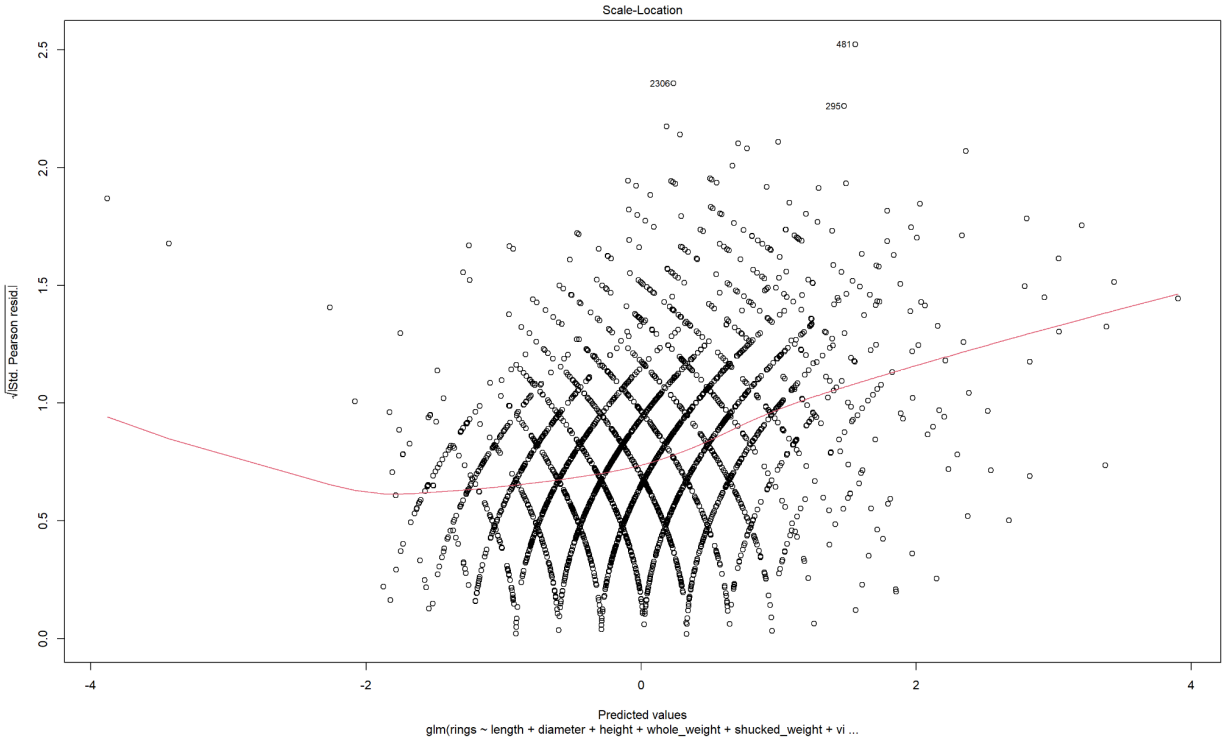
- Plotting the graphs
- Residual vs Fitted



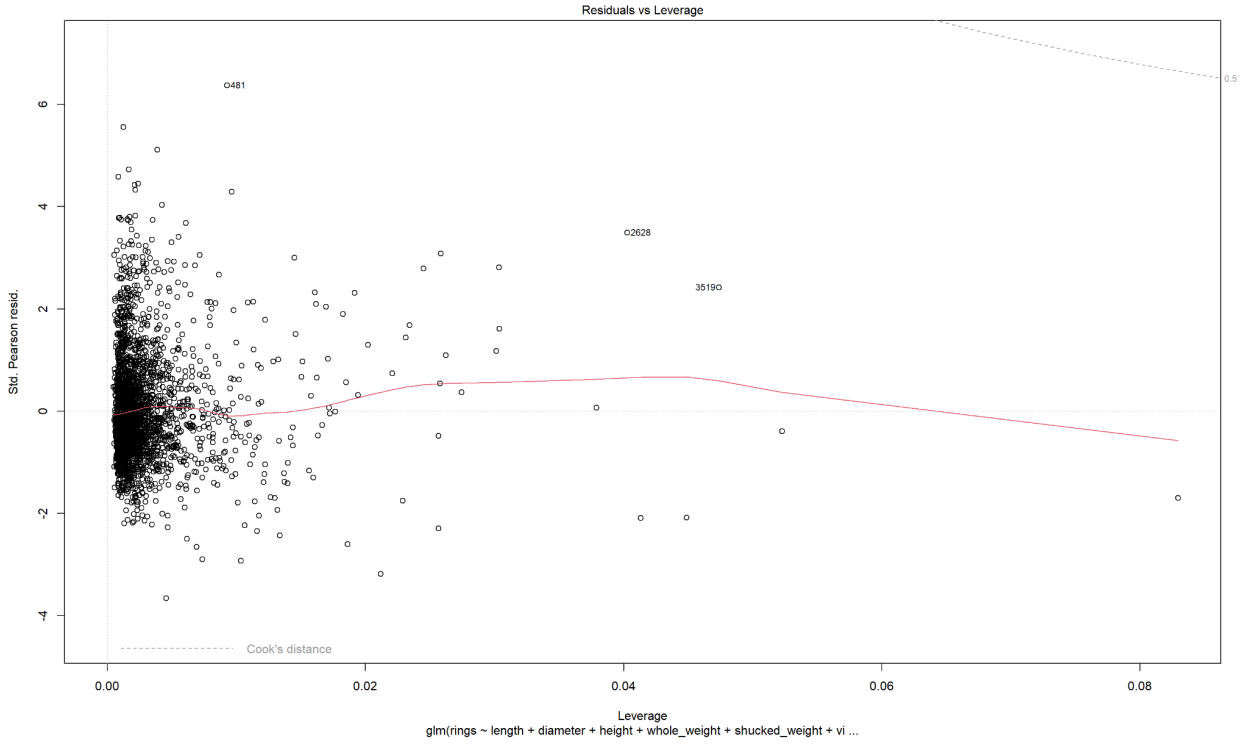
- Normal Q - Q



- Scale - Location



- Residual vs Leverage



- Predict Function

- Confidence intervals

```
> # predict
> abalone.test_50.predict <- predict(abalone.train_50.glm, newdata = abalone.test_50)
> length(abalone.test_50.predict)
[1] 2089
> summary(abalone.test_50.predict)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-3.433678 -0.481626 -0.051664  0.000136  0.425382  7.544918
> # confidence intervals
> confint(abalone.train_50.glm)
Waiting for profiling to be done...
                    2.5 %      97.5 %
(Intercept)    -0.01877274  0.03108867
length         -0.20184914  0.11266788
diameter        0.15565992  0.47517991
height          0.27472396  0.41720438
whole_weight    1.09314051  1.60011067
shucked_weight -1.47426182 -1.21546872
viscera_weight -0.46180597 -0.25500544
shell_weight    0.21074271  0.43360187
```

- Comparing the predicted values

```
> # comparing actual vs prediction
> abalone.test_50.predict.k7 <- kmeans(abalone.test_50.predict, centers = 7)
> abalone.test_50.predict.k7
K-means clustering with 7 clusters of sizes 614, 410, 547, 56, 1, 298, 163

Cluster means:
        [,1]
1  0.04406467
2  0.55619181
3 -0.45119232
4  2.13287623
5  7.54491816
6 -1.08868204
7  1.16216332

Clustering vector:
   1    6    9   18   19   20   23   24   25   29   32   33   34   36   37   38   40   42   43   44   48   49   50   54   57   59   60
   3    3    1    3    3    3    1    1    1    2    7    2    2    3    1    3    3    1    6    6    1    6    1    3    3    6    1
  65   68   70   71   75   76   79   80   81   85   86   87   89   90   92   93   94   95   96   98   99  101  102  104  105  107  108
   1    4    6    2    2    1    2    2    1    2    4    2    1    7    2    7    2    7    7    1    1    6    2    1    2    2    1
 109  110  111  112  115  116  117  118  119  120  122  123  124  125  126  127  129  130  133  135  136  137  138  140  141  146  148
   1    1    3    3    1    1    1    1    2    3    3    1    6    6    6    6    4    4    6    6    3    6    6    3    1    1    6
 149  152  154  158  160  163  166  167  169  174  176  177  178  179  180  181  182  185  189  190  191  197  199  202  205  209  211
   6    2    7    4    4    2    7    4    4    1    3    6    6    6    2    2    7    7    1    2    2    1    1    2    3    7    2
 212  214  215  217  219  224  226  227  228  229  230  231  232  233  236  238  240  241  247  249  252  253  254  257  261  262  263
   6    1    2    1    1    1    1    3    6    7    2    2    7    4    6    6    6    2    6    6    7    1    2    7    2    1    2
 266  267  268  269  270  271  273  278  280  281  283  285  287  289  290  291  292  293  294  299  301  302  303  304  305  313  314
   3    2    3    2    1    4    4    4    2    7    1    7    1    2    2    2    7    2    7    1    3    7    3    3    1    7    4
 315  317  320  322  326  328  331  332  333  335  337  339  341  342  344  346  347  349  350  354  355  356  357  360  363  364  366
   7    2    3    6    3    1    1    3    6    4    7    1    3    4    1    1    2    3    1    1    2    4    2    2    2    2    2
```

```
1832 1833 1834 1835 1842 1847 1848 1850 1851 1853 1854 1856 1858 1861 1862 1863 1866 1875 1878 1880 1883 1884 1885 1886 1888 1889 1893
   6    3    3    3    3    1    1    3    3    1    1    3    1    2    3    1    3    3    1    2    1    1    3    2    2    1    1
1894 1895 1896 1898 1899 1900 1902 1903 1905 1908 1911 1912 1913 1915 1916 1917 1919 1920 1923 1924 1925 1926 1927 1928 1929 1930 1931
   1    1    2    3    1    1    1    1    2    1    2    3    1    1    1    2    2    2    1    1    1    1    1    1    1    2    2
1933
   7
 [ reached getOption("max.print") -- omitted 1089 entries ]

Within cluster sum of squares by cluster:
[1] 13.417371 10.708502 14.338154  8.980542  0.000000 25.626869  7.605721
 (between_SS / total_SS =  93.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
> |
```

- Comparing results of KMeans and KNN for 7 clusters

Total Observations in Table: 2089

	abalone.test_50_kmeans_k7$cluster							
abalone.test_50.predict.k7$cluster	1	2	3	4	5	6	7	Row Total
1	0	255	1	28	77	153	100	614
	94.348	121.282	25.078	5.523	10.648	2.867	8.124	
	0.000	0.415	0.002	0.046	0.125	0.249	0.163	0.294
	0.000	0.578	0.011	0.189	0.425	0.337	0.221	
	0.000	0.122	0.000	0.013	0.037	0.073	0.048	
2	0	80	15	47	75	189	4	410
	63.001	0.496	0.517	11.096	43.867	111.992	80.893	
	0.000	0.195	0.037	0.115	0.183	0.461	0.010	0.196
	0.000	0.181	0.163	0.318	0.414	0.416	0.009	
	0.000	0.038	0.007	0.022	0.036	0.090	0.002	
3	57	95	0	14	10	48	323	547
	8.707	3.630	24.090	15.811	29.504	42.260	353.846	
	0.104	0.174	0.000	0.026	0.018	0.088	0.590	0.262
	0.178	0.215	0.000	0.095	0.055	0.106	0.715	
	0.027	0.045	0.000	0.007	0.005	0.023	0.155	
4	0	0	29	22	1	4	0	56
	8.605	11.822	285.470	81.960	3.058	5.485	12.117	
	0.000	0.000	0.518	0.393	0.018	0.071	0.000	0.027
	0.000	0.000	0.315	0.149	0.006	0.009	0.000	
	0.000	0.000	0.014	0.011	0.000	0.002	0.000	
5	0	0	1	0	0	0	0	1
	0.154	0.211	20.751	0.071	0.087	0.217	0.216	
	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
	0.000	0.000	0.011	0.000	0.000	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
6	264	5	0	3	0	1	25	298
	1039.828	53.307	13.124	15.539	25.820	62.779	24.172	
	0.886	0.017	0.000	0.010	0.000	0.003	0.084	0.143
	0.822	0.011	0.000	0.020	0.000	0.002	0.055	
	0.126	0.002	0.000	0.001	0.000	0.000	0.012	
7	0	6	46	34	18	59	0	163
	25.047	23.456	209.945	43.651	1.064	15.690	35.269	
	0.000	0.037	0.282	0.209	0.110	0.362	0.000	0.078
	0.000	0.014	0.500	0.230	0.099	0.130	0.000	
	0.000	0.003	0.022	0.016	0.009	0.028	0.000	
Column Total	321	441	92	148	181	454	452	2089
	0.154	0.211	0.044	0.071	0.087	0.217	0.216	

- Confusion Matrix
- Calculating the metrics for k=7

```
> # confusion matrix
> abalone.test_50.cm_k7 <- abalone.test_50.ct.k7$t
> abalone.test_50.cm_k7
   y
x     1   2   3   4   5   6   7
  1   0 255   1  28  77 153 100
  2   0  80  15  47  75 189   4
  3  57  95   0  14  10  48 323
  4   0   0  29  22   1   4   0
  5   0   0   1   0   0   0   0
  6 264   5   0   3   0   1  25
  7   0   6  46  34  18  59   0
> # calculate the metrics
> abalone.test_50.accuracy <- sum(diag(abalone.test_50.cm_k7)) / sum(abalone.test_50.cm_k7)
> abalone.test_50.precision <- diag(abalone.test_50.cm_k7) / colSums(abalone.test_50.cm_k7)
> abalone.test_50.recall <- diag(abalone.test_50.cm_k7) / rowSums(abalone.test_50.cm_k7)
> abalone.test_50.specificity <- sapply(1:nrow(abalone.test_50.cm_k7), function(i){
+   TN <- sum(abalone.test_50.cm_k7[-i, -i])
+   FP <- sum(abalone.test_50.cm_k7[-i, i])
+   TN / (TN + FP)
+ })
> abalone.test_50.error <- 1 - abalone.test_50.accuracy
> # print results
> abalone.test_50.accuracy
[1] 0.04930589
> abalone.test_50.precision
          1           2           3           4           5           6           7
0.000000000 0.181405896 0.000000000 0.148648649 0.000000000 0.002202643 0.000000000
> abalone.test_50.recall
          1           2           3           4           5           6           7
0.000000000 0.195121951 0.000000000 0.392857143 0.000000000 0.003355705 0.000000000
> abalone.test_50.specificity
[1] 0.7823729 0.7849911 0.9403372 0.9380226 0.9133142 0.7470687 0.7653167
> abalone.test_50.error
[1] 0.9506941
```

Implementation for k=9 with 70-30 split

- Creating training labels with k-means

```
> abalone.train_70_k9 <- kmeans(abalone.train_70, centers = 9)
> abalone.train_70_k9
K-means clustering with 9 clusters of sizes 373, 188, 183, 478, 446, 458, 122, 459, 216

Cluster means:
      length    diameter       height whole_weight shucked_weight viscera_weight shell_weight       rings
1 -1.2453283 -1.24592399 -1.086409994  -1.17238720     -1.1225344    -1.15597717   -1.1595031 -0.79515411
2  1.4733488  1.44440088  1.346211599   2.05109603      2.0913475     2.05488025    1.8594616  0.48085509
3 -2.2585224 -2.24671022 -1.863188716  -1.51280939     -1.4351027    -1.47253252   -1.5136576 -1.46920435
4  0.5528630  0.54520623  0.358427537   0.37342801      0.4246849     0.36848039    0.3190443 -0.07676148
5  0.9435804  0.94281600  0.799828040   0.97768311      0.9953041     1.00612207    0.8813880  0.26535619
6 -0.5168557 -0.52329245 -0.518798290  -0.72028596     -0.6792885    -0.70820750   -0.7040257 -0.39929552
7  0.9966985  1.09467236  1.189323366   1.27881004      0.8107861     1.11314950    1.7144265  2.40522133
8  0.0566487  0.04593751 -0.009012486  -0.20112818     -0.1741019    -0.19707586   -0.1876837 -0.13890267
9  0.1309829  0.16503453  0.276652592   0.01713675     -0.2133401    -0.03259407    0.2366498  1.75371652

Clustering vector:
 755 1006 2585 2339 1448 3952 3358 3980 1134 3230 1934 1501 3783 1914 1109 1075 3146 1386 2284 2378 4044 2260  686 3857  847  983  151
   7    4    8    7    6    8    1    6    4    5    7    5    5    4    6    1    9    5    6    8    4    4    9    1    8    4    9
1638 2208 2474 1029  326 3856 2837 1956 4093  985  986 2503 1762 1584 4084 2087 2244 1793 4141 1808  344 2507 2992 3092  195 3236 3124
   4    6    5    5    1    6    4    2    2    4    4    3    2    8    4    5    6    4    5    5    8    1    4    6    8    5    4
2225 2875 2132 4023 3464 1326 3949 2667 3502 2758 3833  712 1452 1828 3501 3069 2446 3061  528 4055 1569  473 1149 2037 2313 2823 1927
   8    6    1    2    5    8    8    8    5    8    6    1    6    1    4    5    8    5    9    5    6    6    4    3    1    6    4
2234 3645 3324  458 3224  831 1078  919 1562 2283 1313  185 3837  413  627 2570 1333 3126 2253 1899  779 1561  564  794 1415 2457 3799
   9    6    6    1    8    1    1    1    6    8    8    2    1    4    6    6    8    4    9    8    9    1    8    4    2    3    5
1370  160 2516 3581 2968 3129 2505  617  357  279  270 2926 2694 2395 3201 2266  618 1905 2746  337 2845 2074  539  981 3591  956 1781
   4    9    6    4    5    4    1    6    5    7    6    4    5    5    8    2    1    4    6    5    5    8    3    4    2    8    8
```

```
 [ reached getOption("max.print") -- omitted 1923 entries ]

Within cluster sum of squares by cluster:
[1] 208.8928 376.9397 136.2298 327.8483 415.5183 297.8493 319.8783 297.7122 340.9615
 (between_SS / total_SS =  88.0 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

- Then the cluster vector is passed to KNN

  The output is a vector which contains the predicted values for the test dataset.

```
> abalone.test_70_k9 <- knn(abalone.train_70, abalone.test_70, abalone.train_70_k9$cluster, k = 9)
> abalone.test_70_k9
   [1] 8 1 8 1 7 3 8 8 6 1 6 6 6 8 1 6 6 6 7 5 7 9 5 9 5 6 1 4 8 8 6 6 6 1 8 8 8 1 1 3 1 1 7 7 3 6 6 3 8 5 2 7 2 8 1 3 5 5 4 8 6 6 1 1 6 6
  [67] 6 6 6 6 1 1 7 6 1 3 3 1 9 9 4 3 5 7 9 9 3 1 1 6 3 7 4 7 1 3 1 3 5 5 5 8 7 7 7 9 4 7 5 5 8 9 6 6 8 8 6 8 6 8 6 8 9 9 8 9 7 8 3 9 7 1
 [133] 1 9 1 1 8 9 4 3 5 8 9 7 9 9 7 9 8 7 4 7 4 5 8 3 3 3 1 3 9 6 3 1 1 1 3 4 6 9 8 8 6 9 1 6 3 9 9 4 4 4 9 6 9 1 9 7 8 9 9 6 6 1 6 6 6 6
 [199] 1 1 1 6 1 6 8 9 9 9 9 8 3 3 6 6 6 1 1 1 3 4 9 9 8 8 8 8 9 9 8 6 4 9 9 1 8 9 8 8 1 6 4 6 6 6 1 9 6 1 1 1 1 1 6 8 8 8 4 4 4 4 4 4 5 4 5
 [265] 5 5 5 2 3 3 3 1 1 1 1 1 1 1 1 6 6 6 6 6 6 6 6 6 6 6 6 6 8 8 8 8 4 4 4 4 4 5 5 4 5 5 5 5 4 2 2 5 2 5 2 2 2 3 3 3 1 1 1 6 6 6 6 6 6 6
 [331] 8 6 8 4 4 4 8 4 4 4 5 5 5 5 2 2 2 2 2 2 2 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 6 6 6 6 6 6 6 8 8 6 8 8 8 8 8 6 8 8 8 8 4 8 8 4 4 8 8 4 8 8 5 4 8 4
 [397] 4 4 4 4 5 5 4 5 5 4 4 4 4 5 5 5 5 5 5 5 2 5 2 2 2 2 2 1 1 1 6 6 6 6 6 6 6 6 6 6 8 8 8 4 4 4 4 4 4 4 4 4 4 4 5 2 2 2 5 5 2 5 2 5 2 2 2 3 3 3 3 1 1 1
 [463] 1 1 1 1 1 6 6 6 6 6 8 8 8 8 8 8 8 8 8 8 8 8 8 4 4 8 8 8 8 4 8 4 4 4 4 4 4 4 4 4 5 4 5 5 4 5 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 2 2 2 2 5 2 2 2
 [529] 2 1 6 6 6 8 8 8 8 4 4 4 5 4 5 4 4 2 5 5 2 2 3 3 1 1 1 6 6 6 6 6 8 8 8 8 8 8 4 8 4 8 8 4 4 9 4 4 8 8 4 4 4 5 5 5 4 5 5 5 4 5 5 5 5 5 2 2 2 2
 [595] 2 3 3 1 1 1 6 6 6 6 8 8 8 8 1 1 6 6 7 6 8 6 8 4 4 5 2 8 8 1 4 9 5 3 8 1 3 1 3 1 8 4 8 6 6 1 9 9 4 4 3 4 9 3 3 1 5 7 3 3 7 1 9 8 5 1
 [661] 6 6 1 6 6 6 5 6 4 5 5 9 4 7 5 8 8 1 1 6 9 6 9 1 6 9 8 7 9 9 8 4 1 7 9 7 7 9 8 9 3 6 6 3 9 6 1 4 6 8 1 4 6 6 1 1 6 9 8 7 7 9 9 1 3 3
 [727] 3 8 8 9 1 4 9 5 4 6 9 9 8 9 9 9 8 9 9 8 3 1 1 6 6 8 4 5 4 5 5 2 7 2 2 2 3 3 3 1 1 1 6 6 6 6 8 6 8 8 8 4 8 4 4 4 4 4 4 5 5 5 5 5 5 1 6
 [793] 6 6 8 8 8 8 8 4 4 8 4 4 4 4 4 4 4 5 4 4 5 5 5 2 5 2 3 3 1 1 1 1 1 1 1 1 1 1 6 6 6 6 6 6 8 8 8 8 4 8 8 4 5 5 5 4 5 4 2 2 2 1 1 1 6 6 8
 [859] 8 4 4 5 5 5 2 2 2 2 1 1 1 1 6 6 6 8 4 8 8 8 8 8 4 4 8 4 4 4 4 5 5 4 5 5 5 5 5 5 5 5 5 2 2 8 8 5 5 4 5 5 5 2 2 6 6 6 6 6 6 6 8 8 8 4 8
 [925] 5 4 5 5 5 5 5 5 3 3 1 1 6 6 8 8 4 8 8 5 3 3 1 8 6 9 3 3 5 4 7 6 4 8 3 5 8 4 8 1 4 4 6 2 3 9 4 1 7 9 1 5 9 4 5 8 1 9 6 6 4 1 7 7 4 6 3
 [991] 6 9 8 6 9 6 9 6 6 9
 [ reached getOption("max.print") -- omitted 254 entries ]
Levels: 1 2 3 4 5 6 7 8 9
```

- Applying KMeans to generate labels for the test records.

```
> abalone.test_70_kmeans_k9 <- kmeans(abalone.test_70, centers = 9)
> abalone.test_70_kmeans_k9
K-means clustering with 9 clusters of sizes 186, 83, 105, 253, 31, 108, 218, 230, 40

Cluster means:
      length   diameter      height whole_weight shucked_weight viscera_weight shell_weight       rings
1 -1.1925144 -1.2189925 -1.08422003  -1.15633500    -1.10514068    -1.13062232  -1.14409420 -0.78984311
2 -2.1871965 -2.1645431 -1.80025892  -1.49531148    -1.44100363    -1.44793300  -1.50716031 -1.48164005
3  0.1670000  0.2113655  0.39978180   0.07437386    -0.17721046     0.03242289   0.31617208  1.72791109
4 -0.3885332 -0.4056111 -0.41651359  -0.63681205    -0.61295482    -0.60365059  -0.61843481 -0.28591142
5  1.6912295  1.6904890  2.66457532   2.78613519     2.84457644     2.58210428   2.38981868  0.77094915
6  1.2655212  1.2603978  1.11398496   1.53838271     1.61037100     1.50683219   1.32635462  0.29052010
7  0.2509419  0.2561622  0.09820088  -0.01323879     0.02726688    -0.02918519  -0.02102826 -0.16581075
8  0.7567263  0.7429216  0.61497827   0.70980297     0.74633446     0.74845355   0.60858723  0.07316017
9  1.0700706  1.1045838  1.26673032   1.40237203     0.96460924     1.16881602   1.77668389  2.31573328

Clustering vector:
   3    6   14   22   34   43   47   50   53   55   57   58   61   62   64   65   66   69   73   74   84   85   86   91   93   98  101
   7    1    7    1    9    2    4    7    4    1    4    4    4    4    1    4    4    4    9    8    9    3    8    3    8    4    1
 105  106  109  110  111  112  113  115  116  119  124  125  126  127  128  129  130  135  145  146  149  156  157  166  168  169  174
   8    7    4    4    4    4    1    7    4    7    1    1    2    1    1    9    9    2    4    4    2    7    8    5    9    5    7
 176  177  188  189  190  196  205  208  210  212  213  219  221  222  225  226  227  228  233  235  236  238  240  248  254  257  263
   1    2    8    8    8    7    4    4    1    1    4    4    4    4    4    4    1    1    9    4    1    2    2    1    3    3    7
 265  273  274  285  293  298  300  301  305  306  308  310  313  320  324  332  334  336  339  342  345  356  360  363  364  367  371
   2    8    9    3    3    2    1    1    4    2    9    7    9    1    2    1    2    8    8    9    7    9    9    9    3    7    9
 375  381  385  389  390  391  397  399  401  402  403  404  405  410  411  412  414  416  417  423  425  426  427  437  438  440  441
   8    3    7    3    4    1    7    7    4    7    4    7    4    7    3    3    7    3    9    4    2    3    3    1    1    3    1
 444  446  447  454  465  468  475  476  478  486  490  495  496  501  505  507  508  509  511  513  514  516  519  520  521  531  534
   1    7    3    7    2    6    7    3    9    3    3    9    3    7    9    3    3    7    8    7    2    2    2    1    2    3    4
 538  540  545  546  547  550  555  557  560  562  563  565  566  568  569  571  573  575  576  578  588  589  591  592  593  594  595

 [ reached getOption("max.print") -- omitted 254 entries ]

Within cluster sum of squares by cluster:
[1] 107.17326  52.57918 181.08647 175.26266 640.28568 111.40794 171.28295 190.87276  93.78271
 (between_SS / total_SS =  83.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

- Obtaining labels for the test data by using K-Means on the test data

```
> abalone.test_70_k9.labels <- abalone.test_70_kmeans_k9$cluster
> length(abalone.test_70_k9.labels)
[1] 1254
> abalone.test_70_k9.labels
  3   6  14  22  34  43  47  50  53  55  57  58  61  62  64  65  66  69  73  74  84  85  86  91  93  98 101
  7   1   7   1   9   2   4   7   4   1   4   4   4   4   1   4   4   4   9   8   9   3   8   3   8   4   1
105 106 109 110 111 112 113 115 116 119 124 125 126 127 128 129 130 135 145 146 149 156 157 166 168 169 174
  8   7   4   4   4   4   1   7   4   7   1   1   2   1   1   9   9   2   4   4   2   7   8   5   9   5   7
176 177 188 189 190 196 205 208 210 212 213 219 221 222 225 226 227 228 233 235 236 238 240 248 254 257 263
  1   2   8   8   8   7   4   4   1   1   4   4   4   4   4   4   1   1   9   4   1   2   2   1   3   3   7
265 273 274 285 293 298 300 301 305 306 308 310 313 320 324 332 334 336 339 342 345 356 360 363 364 367 371
  2   8   9   3   3   2   1   1   4   2   9   7   9   1   2   1   2   8   8   9   7   9   9   9   3   7   9
375 381 385 389 390 391 397 399 401 402 403 404 405 410 411 412 414 416 417 423 425 426 427 437 438 440 441
  8   3   7   3   4   1   7   7   4   7   4   7   4   7   3   3   7   3   9   4   2   3   3   1   1   3   1
444 446 447 454 465 468 475 476 478 486 490 495 496 501 505 507 508 509 511 513 514 516 519 520 521 531 534
  1   7   3   7   2   6   7   3   9   3   3   9   3   7   9   3   3   7   8   7   2   2   2   1   2   3   4
538 540 545 546 547 550 555 557 560 562 563 565 566 568 569 571 573 575 576 578 588 589 591 592 593 594 595
  2   1   1   1   2   7   4   3   3   4   4   4   1   4   2   3   3   8   8   8   3   4   3   1   3   9   7
596 600 604 608 609 611 616 631 632 636 638 641 645 652 653 662 675 676 679 687 691 692 695 699 700 704 709
  3   3   4   4   1   4   4   4   4   1   1   1   4   1   4   7   3   3   3   3   4   2   2   4   4   1   1
714 715 717 722 724 726 731 734 736 738 744 746 751 752 759 760 762 768 769 770 775 776 784 792 795 798 800
  1   1   2   3   3   3   7   3   4   7   3   3   4   4   8   3   3   1   7   3   4   7   1   4   8   4   4
801 806 808 809 818 819 823 824 826 840 844 846 849 852 853 857 858 862 863 864 868 870 880 885 892 895 897
  4   1   3   4   1   1   1   1   1   4   7   7   7   7   8   8   8   8   8   8   8   8   6   6   5   2   2
898 902 908 909 910 917 918 921 926 928 935 936 937 938 939 940 941 942 944 947 955 962 966 972 973 994 995
  2   2   1   1   1   1   1   1   1   4   4   4   4   4   4   4   4   4   4   4   4   4   7   7   7   8   8
```

- Linear modelling by using glm function of R

```
[ reached getOption("max.print") -- omitted 254 entries ]
>
> # linear modelling using GLM
> abalone.train_70.glm <- glm(formula = rings ~ length + diameter + height + whole_weight + shucked_weight +  viscera_weight + shell_wei
ght, family = gaussian, data = abalone.train_70)
> summary(abalone.train_70.glm)

Call:
glm(formula = rings ~ length + diameter + height + whole_weight +
    shucked_weight + viscera_weight + shell_weight, family = gaussian,
    data = abalone.train_70)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5111  -0.4220  -0.1166   0.2718   4.3572

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.006158   0.012720   0.484 0.628339
length        -0.044591   0.080235  -0.556 0.578426
diameter       0.315420   0.081512   3.870 0.000111 ***
height         0.345964   0.036348   9.518  < 2e-16 ***
whole_weight   1.346626   0.129331  10.412  < 2e-16 ***
shucked_weight -1.344865   0.066020 -20.371  < 2e-16 ***
viscera_weight -0.358406   0.052756  -6.794 1.32e-11 ***
shell_weight   0.322172   0.056853   5.667 1.60e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.4723417)

    Null deviance: 2951.3  on 2922  degrees of freedom
Residual deviance: 1376.9  on 2915  degrees of freedom
AIC: 6112.7

Number of Fisher Scoring iterations: 2
```

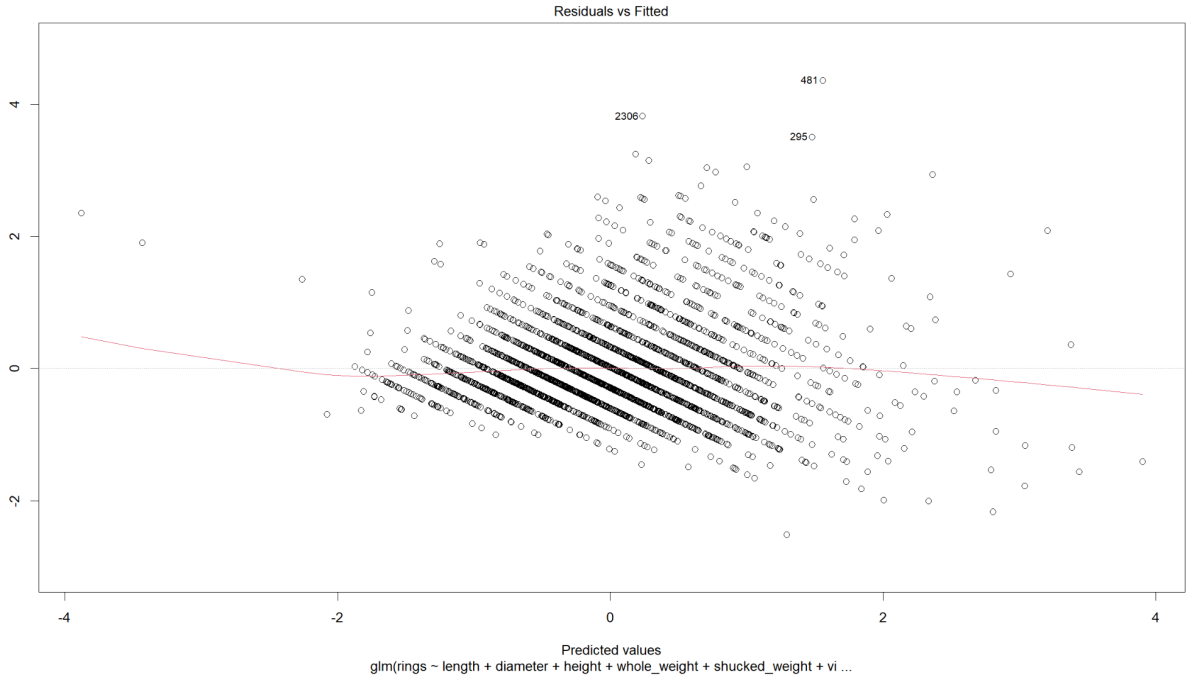
- Applying anova to the training data to get the deviance of the data

```
>
> # anova - analysis of variance on the model result
> abalone.train_70.glm.anova <- anova(abalone.train_70.glm, test = "Chisq")
> abalone.train_70.glm.anova
Analysis of Deviance Table

Model: gaussian, link: identity

Response: rings

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                          2922     2951.3
length          1   918.29      2921     2033.0 < 2.2e-16 ***
diameter        1    69.50      2920     1963.5 < 2.2e-16 ***
height          1   132.12      2919     1831.4 < 2.2e-16 ***
whole_weight    1     3.30      2918     1828.1  0.008244 **
shucked_weight  1   398.24      2917     1429.8 < 2.2e-16 ***
viscera_weight  1    37.78      2916     1392.0 < 2.2e-16 ***
shell_weight    1    15.17      2915     1376.9 1.455e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

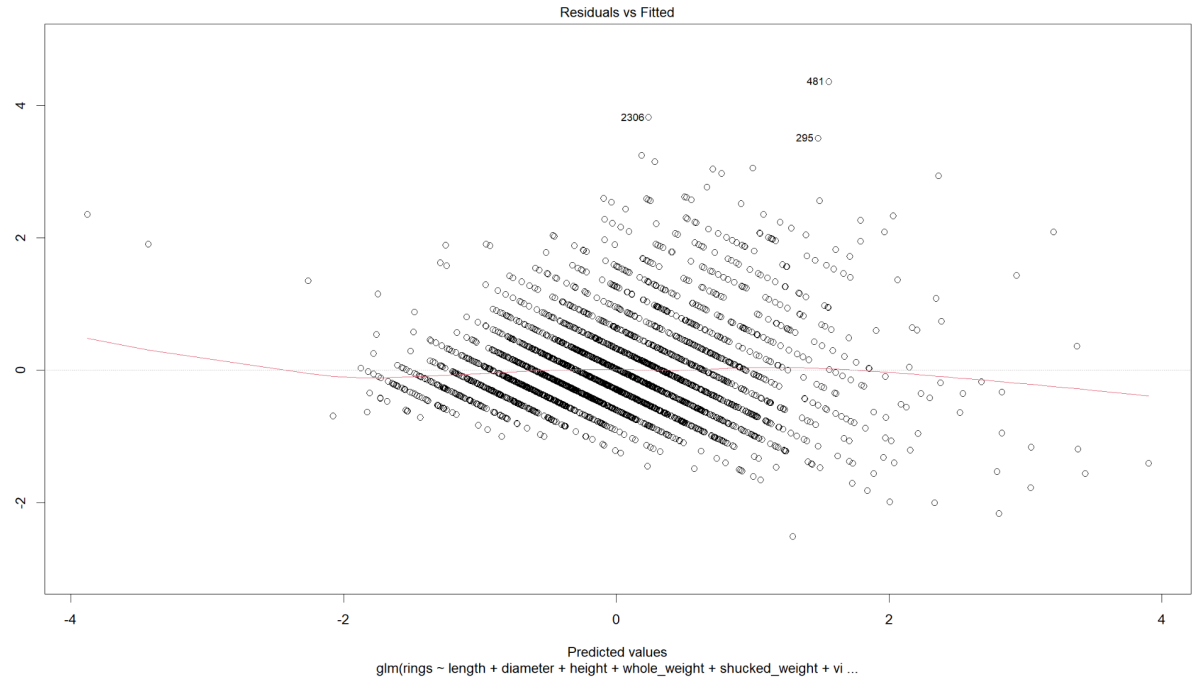
- Plotting graph

```
>
>
> plot(abalone.train_70.glm)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
>
>
>
```

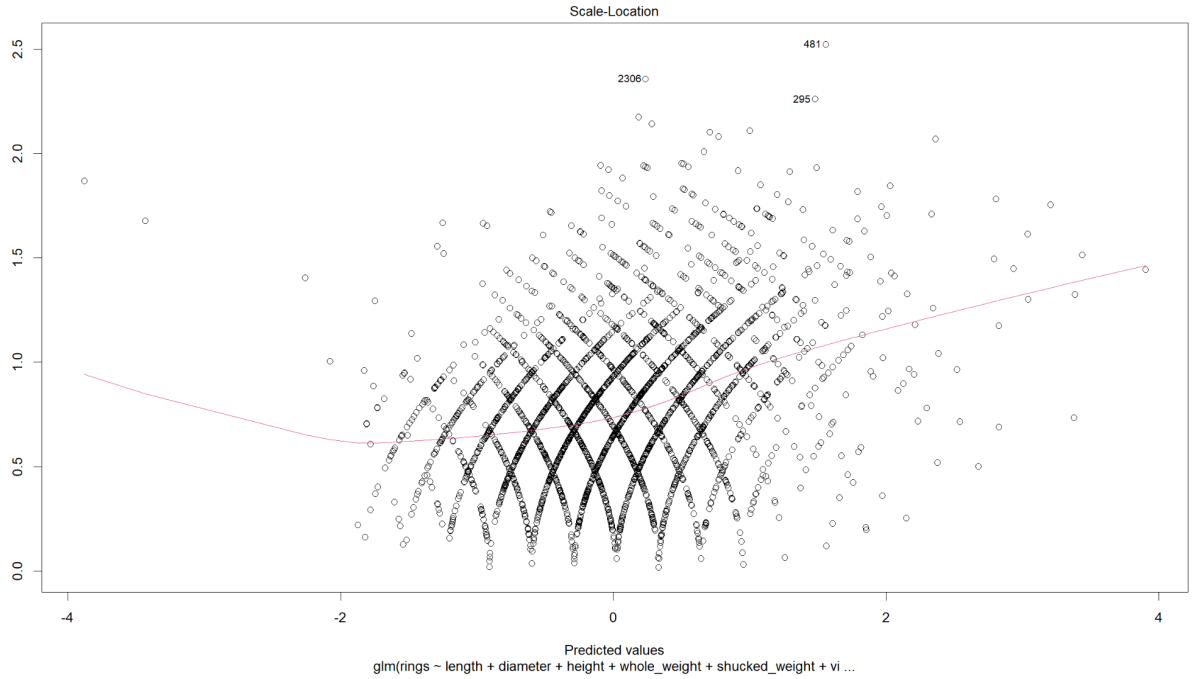
- Residual values vs Fitted values Plot



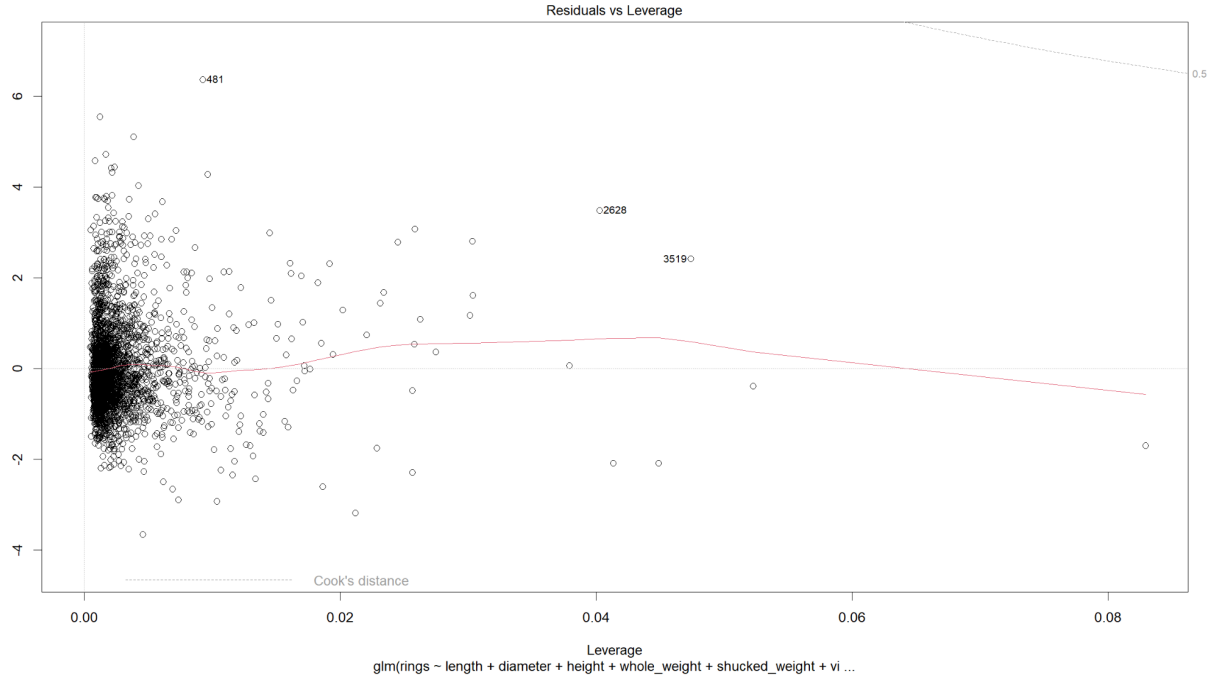
- Normal Q-Q Plot



- Scale-Location Plot



- Residuals vs Leverage Plot



- Using predict function in R

```
>
>
> # predict
> abalone.test_70.predict <- predict(abalone.train_70.glm, newdata = abalone.test_70)
> length(abalone.test_70.predict)
[1] 1254
>
```

- Confidence intervals: Indicating the probability that the true parameter values lies within the range

```
>
> # confidence intervals
> confint(abalone.train_70.glm)
Waiting for profiling to be done...
                    2.5 %      97.5 %
(Intercept)     -0.01877274  0.03108867
length          -0.20184914  0.11266788
diameter         0.15565992  0.47517991
height           0.27472396  0.41720438
whole_weight     1.09314051  1.60011067
shucked_weight -1.47426182 -1.21546872
viscera_weight -0.46180597 -0.25500544
shell_weight     0.21074271  0.43360187
>
```

- Comparing the predicted values

```
Within cluster sum of squares by cluster:
[1]  1.988371  2.316529  1.473202  3.005395  2.043458  2.256823 27.064216  2.189746  1.405194
 (between_SS / total_SS =  93.9 %)
```

- Comparing results of kmeans and knn for 9 clusters using the crosstable functionality provided gmodels pacakge

Total Observations in Table: 1254

| abalone.test_70_kmeans_k9$cluster
abalone.test_70.predict.k9$cluster	1	2	3	4	5	6	7	8	9	Row Total
1	0	0	0	0	54	0	0	0	0	54
	5.211	1.766	9.431	9.603	314.455	1.593	4.005	5.469	9.904	
	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.043
	0.000	0.000	0.000	0.000	0.331	0.000	0.000	0.000	0.000	
	0.000	0.000	0.000	0.000	0.043	0.000	0.000	0.000	0.000	
2	49	0	70	40	0	5	1	7	81	253
	24.764	8.272	15.084	0.554	32.886	0.814	16.816	13.535	25.794	
	0.194	0.000	0.277	0.158	0.000	0.020	0.004	0.028	0.320	0.202
	0.405	0.000	0.320	0.179	0.000	0.135	0.011	0.055	0.352	
	0.039	0.000	0.056	0.032	0.000	0.004	0.001	0.006	0.065	
3	0	0	35	2	97	2	0	0	3	139
	13.412	4.545	4.738	20.880	344.829	1.077	10.309	14.077	19.847	
	0.000	0.000	0.252	0.014	0.698	0.014	0.000	0.000	0.022	0.111
	0.000	0.000	0.160	0.009	0.595	0.054	0.000	0.000	0.013	
	0.000	0.000	0.028	0.002	0.077	0.002	0.000	0.000	0.002	
4	0	19	0	5	0	7	17	13	1	62
	5.982	142.113	10.828	3.293	8.059	14.615	33.450	7.194	9.460	
	0.000	0.306	0.000	0.081	0.000	0.113	0.274	0.210	0.016	0.049
	0.000	0.463	0.000	0.022	0.000	0.189	0.183	0.102	0.004	
	0.000	0.015	0.000	0.004	0.000	0.006	0.014	0.010	0.001	
5	15	2	0	71	0	4	19	36	41	188
	0.544	2.797	32.833	42.215	24.437	0.431	1.834	15.108	1.232	
	0.080	0.011	0.000	0.378	0.000	0.021	0.101	0.191	0.218	0.150
	0.124	0.049	0.000	0.318	0.000	0.108	0.204	0.283	0.178	
	0.012	0.002	0.000	0.057	0.000	0.003	0.015	0.029	0.033	
6	43	0	6	77	0	8	15	32	77	258
	13.167	8.435	33.856	21.108	33.536	0.020	0.893	1.319	18.615	
	0.167	0.000	0.023	0.298	0.000	0.031	0.058	0.124	0.298	0.206
	0.355	0.000	0.027	0.345	0.000	0.216	0.161	0.252	0.335	
	0.034	0.000	0.005	0.061	0.000	0.006	0.012	0.026	0.061	
7	0	10	0	1	0	9	0	1	0	21

```
                                  |     2.026 |   126.331 |     3.667 |     2.002 |     2.730 |   113.345 |     1.557 |     0.597 |
3.852 |           |
                                  |     0.000 |     0.476 |     0.000 |     0.048 |     0.000 |     0.429 |     0.000 |     0.048 |
0.000 |     0.017 |
                                  |     0.000 |     0.244 |     0.000 |     0.004 |     0.000 |     0.243 |     0.000 |     0.008 |
0.000 |           |
                                  |     0.000 |     0.008 |     0.000 |     0.001 |     0.000 |     0.007 |     0.000 |     0.001 |
0.000 |           |
----------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|----
-------|-----------|
                                8 |         4 |        10 |         0 |        18 |         0 |         1 |        41 |        35 |
8 |       117 |
                                  |     4.707 |     9.967 |    20.433 |     0.378 |    15.208 |     1.742 |   120.407 |    45.231 |
8.442 |           |
                                  |     0.034 |     0.085 |     0.000 |     0.154 |     0.000 |     0.009 |     0.350 |     0.299 |
0.068 |     0.093 |
                                  |     0.033 |     0.244 |     0.000 |     0.081 |     0.000 |     0.027 |     0.441 |     0.276 |
0.035 |           |
                                  |     0.003 |     0.008 |     0.000 |     0.014 |     0.000 |     0.001 |     0.033 |     0.028 |
0.006 |           |
----------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|----
-------|-----------|
                                9 |        10 |         0 |       108 |         9 |        12 |         1 |         0 |         3 |
19 |       162 |
                                  |     2.029 |     5.297 |   224.566 |    13.620 |     3.896 |     2.989 |    12.014 |    10.955 |
3.863 |           |
                                  |     0.062 |     0.000 |     0.667 |     0.056 |     0.074 |     0.006 |     0.000 |     0.019 |
0.117 |     0.129 |
                                  |     0.083 |     0.000 |     0.493 |     0.040 |     0.074 |     0.027 |     0.000 |     0.024 |
0.083 |           |
                                  |     0.008 |     0.000 |     0.086 |     0.007 |     0.010 |     0.001 |     0.000 |     0.002 |
0.015 |           |
----------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|----
-------|-----------|
                     Column Total |       121 |        41 |       219 |       223 |       163 |        37 |        93 |       127 |
230 |      1254 |
                                  |     0.096 |     0.033 |     0.175 |     0.178 |     0.130 |     0.030 |     0.074 |     0.101 |
0.183 |           |
----------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|----
-------|-----------|
```

- Confusion Matrix

```
>
> # confusion matrix
> abalone.test_70.cm_k9 <- abalone.test_70.ct.k9$t
> abalone.test_70.cm_k9
   y
x     1   2   3   4   5   6   7   8   9
  1   0   0   0   0  54   0   0   0   0
  2  49   0  70  40   0   5   1   7  81
  3   0   0  35   2  97   2   0   0   3
  4   0  19   0   5   0   7  17  13   1
  5  15   2   0  71   0   4  19  36  41
  6  43   0   6  77   0   8  15  32  77
  7   0  10   0   1   0   9   0   1   0
  8   4  10   0  18   0   1  41  35   8
  9  10   0 108   9  12   1   0   3  19
```

- Calculating the metrices for k=9

```
>
> # calculate the metrics
> abalone.test_70.accuracy <- sum(diag(abalone.test_70.cm_k9)) / sum(abalone.test_70.cm_k9)
> abalone.test_70.precision <- diag(abalone.test_70.cm_k9) / colSums(abalone.test_70.cm_k9)
> abalone.test_70.recall <- diag(abalone.test_70.cm_k9) / rowSums(abalone.test_70.cm_k9)
> abalone.test_70.specificity <- sapply(1:nrow(abalone.test_70.cm_k9), function(i){
+     TN <- sum(abalone.test_70.cm_k9[-i, -i])
+     FP <- sum(abalone.test_70.cm_k9[-i, i])
+     TN / (TN + FP)
+ })
> abalone.test_70.error <- 1 - abalone.test_70.accuracy
> # print results
> abalone.test_70.accuracy
[1] 0.08133971
> abalone.test_70.precision
          1           2           3           4           5           6           7           8           9
0.00000000 0.00000000 0.15981735 0.02242152 0.00000000 0.21621622 0.00000000 0.27559055 0.08260870
> abalone.test_70.recall
          1           2           3           4           5           6           7           8           9
0.00000000 0.00000000 0.25179856 0.08064516 0.00000000 0.03100775 0.00000000 0.29914530 0.11728395
> abalone.test_70.specificity
[1] 0.8991667 0.9590410 0.8349776 0.8171141 0.8470919 0.9708835 0.9245742 0.9190853 0.8067766
> abalone.test_70.error
[1] 0.9186603
>
```

Implementation for k=9 with 60-40 split

- Creating training labels with k-means

```
> # knn with k=9 and 60-40 split
> abalone.train_60_k9 <- kmeans(abalone.train_60, centers = 9)
> abalone.train_60_k9
K-means clustering with 9 clusters of sizes 453, 248, 131, 150, 441, 182, 365, 469, 67

Cluster means:
       length    diameter      height whole_weight shucked_weight viscera_weight shell_weight      rings
1  0.74025794  0.73282857  0.57733108  0.660750826     0.69091566    0.690865355   0.57489383  0.1040985
2  1.20638983  1.19397293  1.08886659  1.457059516     1.48654138    1.452795759   1.35131489  0.3432320
3  0.82741501  0.89939813  1.06095752  0.944106000     0.56604107    0.789565347   1.27222145  2.3171539
4 -2.25957353 -2.24957902 -1.87397996 -1.509710587    -1.42798978   -1.465839896  -1.50828664 -1.4909323
5  0.26792340  0.25875525  0.08827342  0.002689834     0.06624502   -0.006429082  -0.02535348 -0.1876100
6 -0.04994128 -0.01325395  0.11533826 -0.181158855    -0.37604986   -0.176745520  -0.01938893  1.4691057
7 -1.23356809 -1.24277988 -1.09770238 -1.169247218    -1.11628075   -1.151826713  -1.15822577 -0.8215304
8 -0.44070878 -0.45390462 -0.45436313 -0.667996158    -0.62884688   -0.656380154  -0.65083845 -0.3795282
9  1.68744089  1.68173206  1.64765207  2.630207767     2.57713297    2.482413409   2.53397658  1.0204788

Clustering vector:
1089  870 2407 3354 3280  754 2813 2581 2468 3927 3767 1356   55 1280 3079 1557 3641  603 3951  819 1467 3346  910 3814 3445  986 3760
   8    1    1    8    3    1    4    5    6    6    5    1    7    8    2    7    8    7    6    7    5    6    7    8    1    5
1019 2809  444  666 3742  938 2866 1526  162 1962  843 1501 2786 1468 3957 1358  132 2994 3180 1700  125 2385 3082 3325 3386   65 1733
   1    2    7    7    2    8    4    2    1    2    8    1    1    8    6    1    8    2    4    2    7    8    2    7    8    8    2
 333  956 1684 3360 3756 2232 2464 2035 3436  299 4062 2594 2403 3171 3887 3090 3238 3363 2422 1482 3690 2882 3298 1618 3684 1193 3661
   4    8    1    3    8    5    6    3    7    8    5    1    6    6    8    8    3    7    6    5    1    8    3    5    1    2    5
1783  398  217 3832 1357  999 2788 1289 3798 1641 4002 1612  809 1161 3067 3154 2463  905 3176 2298 3801  124  490  482 3042 2490  165
   5    8    8    5    5    1    5    8    9    1    8    5    8    1    1    8    7    7    8    8    9    7    1    6    5    6    9
 362  706 1300  614 1855 1380  212  746  980 3880 3189 1266 1296 3263 2306 4012  528  788 2286 3137 3473  370  777 4006 4000 1768 1256
   1    8    8    6    8    1    7    6    5    3    9    8    5    1    6    1    6    6    8    5    4    3    6    5    7    8    7

 [ reached getOption("max.print") -- omitted 1506 entries ]

Within cluster sum of squares by cluster:
[1] 382.5423 288.0392 279.7420 123.7834 295.6980 245.8989 203.5749 303.1434 232.3263
 (between_SS / total_SS =  88.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

- Then the cluster vector is passed to KNN

  The output is a vector which contains the predicted values for the test dataset.

```
> abalone.test_60_k9 <- knn(abalone.train_60, abalone.test_60, abalone.train_60_k9$cluster, k = 9)
> abalone.test_60_k9
   [1] 5 8 6 6 6 6 8 5 8 7 7 7 5 5 3 1 3 3 5 7 8 4 4 7 8 8 4 5 7 8 8 5 4 8 8 5 1 4 6 7 3 1 1 3 5 1 3 6 1 6 1 2 2 5 8 6 5 6 5 8 8 5 7 3 7 7
  [67] 7 8 7 5 3 8 4 1 1 3 1 9 9 5 4 7 4 1 1 2 1 1 1 5 5 3 6 6 6 8 7 8 6 7 8 5 7 8 7 8 8 8 8 7 6 3 8 4 7 7 7 7 3 3 4 6 7 8 3 7 8 6 5 5 3 4
 [133] 6 7 2 5 3 6 3 3 7 7 4 7 7 7 5 7 1 9 2 2 1 1 1 5 8 8 8 6 7 7 5 5 1 8 5 8 5 6 6 3 2 4 4 6 3 6 6 8 7 7 5 1 6 3 3 5 3 5 7 7 4 4 4 8 8 3
 [199] 6 5 3 5 6 1 3 3 1 3 1 5 5 5 8 4 4 4 4 7 4 7 8 8 4 7 7 4 5 5 8 8 8 6 5 6 7 6 6 1 3 1 6 7 5 6 7 6 5 6 5 1 1 7 7 6 7 4 7 6 8 6 7 6 7 8
 [265] 8 8 8 4 4 3 3 5 6 8 6 6 5 6 7 8 8 6 5 6 4 4 6 8 7 7 7 4 4 1 8 6 8 5 6 8 6 6 7 2 5 6 3 2 3 6 5 5 6 8 5 5 6 6 7 5 8 8 8 7 6 4 4 7 7 7
 [331] 7 7 7 7 7 8 7 8 8 5 5 5 1 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 9 3 2 2 9 4 4 4 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 5 5 5 5 5 5 5 5 8 5 5 5 1
 [397] 5 5 1 1 1 1 1 1 2 1 1 1 2 1 2 1 2 2 2 1 2 2 2 2 9 4 4 4 4 4 4 4 4 7 7 7 8 7 8 7 8 8 8 8 8 8 8 8 8 5 8 8 5 8 5 5 5 5 5 5 5 5 1 5 5 5 1
 [463] 5 5 5 1 1 1 1 1 1 1 2 1 2 1 2 2 2 2 9 2 1 2 9 9 9 9 9 9 4 4 7 7 7 7 7 7 8 7 7 7 7 8 8 8 8 8 8 8 8 8 5 5 8 8 8 6 8 5 8 5 5 5 5 5 8 5
 [529] 5 5 8 5 1 1 5 5 1 1 1 1 1 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 2 9 2 9 9 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 5 5 5 5 5 1 5 1 1 1 1
 [595] 1 1 1 2 2 2 1 2 1 1 1 2 2 2 2 2 9 2 2 7 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 5 8 5 8 8 5 8 5 5 5 5 8 5 5 8 5 5 5 5 5
 [661] 5 1 1 5 5 1 1 5 5 1 1 5 5 5 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 2 2 1 2 1 2 1 2 1 2 2 2 2 9 2 9 9 2 9 7 8 8 8 7 5 5 5 1 1 5 5 1 1 1 1 1 1
 [727] 1 1 2 2 2 2 2 2 9 4 4 7 7 7 8 8 8 8 8 8 8 5 8 5 8 8 5 5 5 5 5 5 5 5 5 1 5 1 1 5 5 5 1 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 2 2 1 2 1
 [793] 2 2 2 2 9 4 4 4 4 7 7 7 7 7 8 8 8 8 8 5 5 5 4 7 7 7 7 3 8 8 5 8 5 5 5 5 5 1 2 2 8 8 7 5 8 6 7 1 8 7 4 8 7 4 7 1 5 8 5 3 8 7 4 8 9 7 4 5
 [859] 1 3 2 2 5 7 4 4 4 5 3 6 5 6 5 1 7 2 6 3 8 4 4 3 3 3 3 3 6 5 3 2 3 5 5 7 4 6 6 1 6 3 3 8 8 8 7 8 1 1 1 2 6 8 1 5 6 1 5 3 1 3 3 8 8 8
 [925] 7 7 7 5 5 6 8 1 1 6 8 7 6 7 6 7 8 6 6 8 6 2 9 3 3 5 7 6 5 6 3 1 3 3 5 7 4 8 5 7 4 5 5 7 1 1 6 3 6 5 7 1 1 7 6 8 7 4 1 6 6 6 6 8 5 8
 [991] 8 6 6 8 7 4 4 4 6 6
 [ reached getOption("max.print") -- omitted 671 entries ]
Levels: 1 2 3 4 5 6 7 8 9
```

- Applying KMeans to generate labels for the test records.

```
> abalone.test_60_kmeans_k9 <- kmeans(abalone.test_60, centers = 9)
> abalone.test_60_kmeans_k9
K-means clustering with 9 clusters of sizes 96, 213, 263, 202, 105, 283, 264, 115, 130

Cluster means:
       length    diameter      height whole_weight shucked_weight viscera_weight shell_weight       rings
1  0.60862238  0.66373271  0.73378183    0.6578058     0.30014962     0.51910389   1.02254516  2.28213289
2  1.02814301  1.01907463  0.87359575    1.1510687     1.15622700     1.17313066   1.02849422  0.25646264
3  0.15498386  0.14855239  0.05201456   -0.1019071    -0.06330741    -0.09775445  -0.11681182 -0.27189959
4 -1.21157858 -1.22101468 -1.06488851   -1.1594393    -1.11713035    -1.13651971  -1.14253363 -0.70722691
5  1.46678729  1.48149643  1.69650453    2.1598804     2.14102287     2.10796985   1.96686566  0.80039267
6  0.64379587  0.64303652  0.48634208    0.5075293     0.54569322     0.50870681   0.45267568  0.01728037
7 -0.44832441 -0.46339558 -0.47746474   -0.6782527    -0.63728076    -0.65606145  -0.66823613 -0.47051435
8 -2.21416183 -2.19243450 -1.80544378   -1.5053062    -1.45031104    -1.46523831  -1.51687176 -1.44391424
9 -0.04477129 -0.02011932  0.12926268   -0.2141076    -0.38200324    -0.21457549  -0.01950735  1.18962321

Clustering vector:
   3    4    7    8   10   11   12   14   15   19   21   22   24   28   29   31   33   37   39   40   41   43   45   46   47   48   49
   3    7    1    9    1    9    7    3    7    4    4    4    3    6    1    6    1    1    3    4    7    8    8    4    7    7    8
  50   52   53   54   56   59   60   62   63   68   70   71   72   73   74   75   76   78   80   84   85   86   91   93   95   96   97
   3    4    7    7    3    8    7    7    3    1    8    9    4    1    6    6    1    6    1    1    1    6    9    6    2    5    3
  99  102  104  106  107  110  111  115  120  131  134  137  138  139  140  141  143  146  150  154  157  158  160  166  168  174  175
   7    9    3    9    3    7    7    3    4    1    4    8    4    4    4    3    1    7    8    6    6    1    1    5    5    3    8
 176  177  184  187  188  189  190  192  195  197  198  202  203  204  205  207  208  209  210  214  216  218  219  220  221  223  225
   4    8    6    6    2    2    6    6    3    3    1    9    9    9    7    4    7    9    4    3    3    4    7    4    7    7    7
 226  228  232  233  235  242  245  247  248  250  254  259  265  267  268  270  276  282  283  285  286  290  294  296  302  304  308
   7    4    9    1    7    8    4    4    4    4    1    2    8    9    4    9    5    4    7    9    3    3    1    8    9    4    5
 310  311  312  313  317  320  323  324  325  329  332  347  349  351  356  357  360  369  377  380  383  384  386  388  389  391  395
   6    1    9    1    2    4    4    8    4    4    4    3    4    6    5    2    5    6    2    6    3    9    7    7    9    4    4
 397  399  400  405  406  407  409  412  416  419  421  424  425  429  430  433  434  435  437  439  446  449  450  452  453  454  455

 [ reached getOption("max.print") -- omitted 671 entries ]

Within cluster sum of squares by cluster:
[1] 214.45603 203.20774 154.98646 118.46952 858.37887 207.47303 155.98790  63.91469 142.30520
 (between_SS / total_SS =  84.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

- Obtaining labels for the test data by using K-Means on the test data

```
> abalone.test_60_k9.labels <- abalone.test_60_kmeans_k9$cluster
> length(abalone.test_60_k9.labels)
[1] 1671
> abalone.test_60_k9.labels
   3    4    7    8   10   11   12   14   15   19   21   22   24   28   29   31   33   37   39   40   41   43   45   46   47   48   49
   3    7    1    9    1    9    7    3    7    4    4    4    3    6    1    6    1    1    3    4    7    8    8    4    7    7    8
  50   52   53   54   56   59   60   62   63   68   70   71   72   73   74   75   76   78   80   84   85   86   91   93   95   96   97
   3    4    7    7    3    8    7    7    3    1    8    9    4    1    6    6    1    6    1    1    1    6    9    6    2    5    3
  99  102  104  106  107  110  111  115  120  131  134  137  138  139  140  141  143  146  150  154  157  158  160  166  168  174  175
   7    9    3    9    3    7    7    3    4    1    4    8    4    4    4    3    1    7    8    6    6    1    1    5    5    3    8
 176  177  184  187  188  189  190  192  195  197  198  202  203  204  205  207  208  209  210  214  216  218  219  220  221  223  225
   4    8    6    6    2    2    6    6    3    3    1    9    9    9    7    4    7    9    4    3    3    4    7    4    7    7    7
 226  228  232  233  235  242  245  247  248  250  254  259  265  267  268  270  276  282  283  285  286  290  294  296  302  304  308
   7    4    9    1    7    8    4    4    4    4    1    2    8    9    4    9    5    4    7    9    3    3    1    8    9    4    5
 310  311  312  313  317  320  323  324  325  329  332  347  349  351  356  357  360  369  377  380  383  384  386  388  389  391  395
   6    1    9    1    2    4    4    8    4    4    4    3    4    6    5    2    5    6    2    6    3    9    7    7    9    4    4
 397  399  400  405  406  407  409  412  416  419  421  424  425  429  430  433  434  435  437  439  446  449  450  452  453  454  455
   3    9    6    7    9    7    6    9    1    1    2    8    8    1    1    1    1    7    4    4    3    6    1    1    1    6    2
 456  459  461  463  464  466  473  477  479  486  487  488  489  492  493  494  495  497  498  499  500  501  509  512  515  516  518
   3    4    4    8    8    8    7    7    5    9    6    1    3    9    2    1    1    2    1    6    6    9    6    7    8    8    8
 519  523  524  530  535  536  538  540  545  548  550  553  554  555  556  557  564  565  566  571  574  576  580  581  582  585  587
   8    4    8    4    7    7    8    4    4    8    6    9    9    7    3    9    3    9    4    9    1    6    1    2    9    4    3
 591  592  593  595  596  597  598  599  602  613  615  618  620  621  622  623  626  628  629  635  645  646  648  649  654  656  658
   9    4    9    9    9    3    6    6    4    4    9    4    8    4    9    9    9    4    1    4    7    7    7    7    8    8    1
 661  662  665  667  670  673  677  679  680  682  683  684  685  687  695  698  702  703  704  709  712  717  718  722  725  726  728
   1    3    9    7    9    9    9    1    4    7    7    9    3    9    8    8    9    7    4    4    4    8    8    6    7    9    7
 731  734  736  737  741  742  747  748  751  755  756  757  762  766  769  770  774  775  776  778  780  784  791  793  801  803  806
   3    9    3    9    9    4    5    3    9    1    2    1    1    3    3    1    7    3    9    9    9    4    6    7    7    7    4
```

- Linear modeling by using glm function of R.

```
> abalone.train_60.glm <- glm(formula = rings ~ length + diameter + height + whole_weight + shucked_weight +  viscera_weight + shell_weight, fa
mily = gaussian, data = abalone.train_70)
> summary(abalone.train_60.glm)

Call:
glm(formula = rings ~ length + diameter + height + whole_weight +
    shucked_weight + viscera_weight + shell_weight, family = gaussian,
    data = abalone.train_70)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5111  -0.4220  -0.1166   0.2718   4.3572

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.006158   0.012720   0.484 0.628339
length         -0.044591   0.080235  -0.556 0.578426
diameter        0.315420   0.081512   3.870 0.000111 ***
height          0.345964   0.036348   9.518  < 2e-16 ***
whole_weight    1.346626   0.129331  10.412  < 2e-16 ***
shucked_weight -1.344865   0.066020 -20.371  < 2e-16 ***
viscera_weight -0.358406   0.052756  -6.794 1.32e-11 ***
shell_weight    0.322172   0.056853   5.667 1.60e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.4723417)

    Null deviance: 2951.3  on 2922  degrees of freedom
Residual deviance: 1376.9  on 2915  degrees of freedom
AIC: 6112.7

Number of Fisher Scoring iterations: 2
```

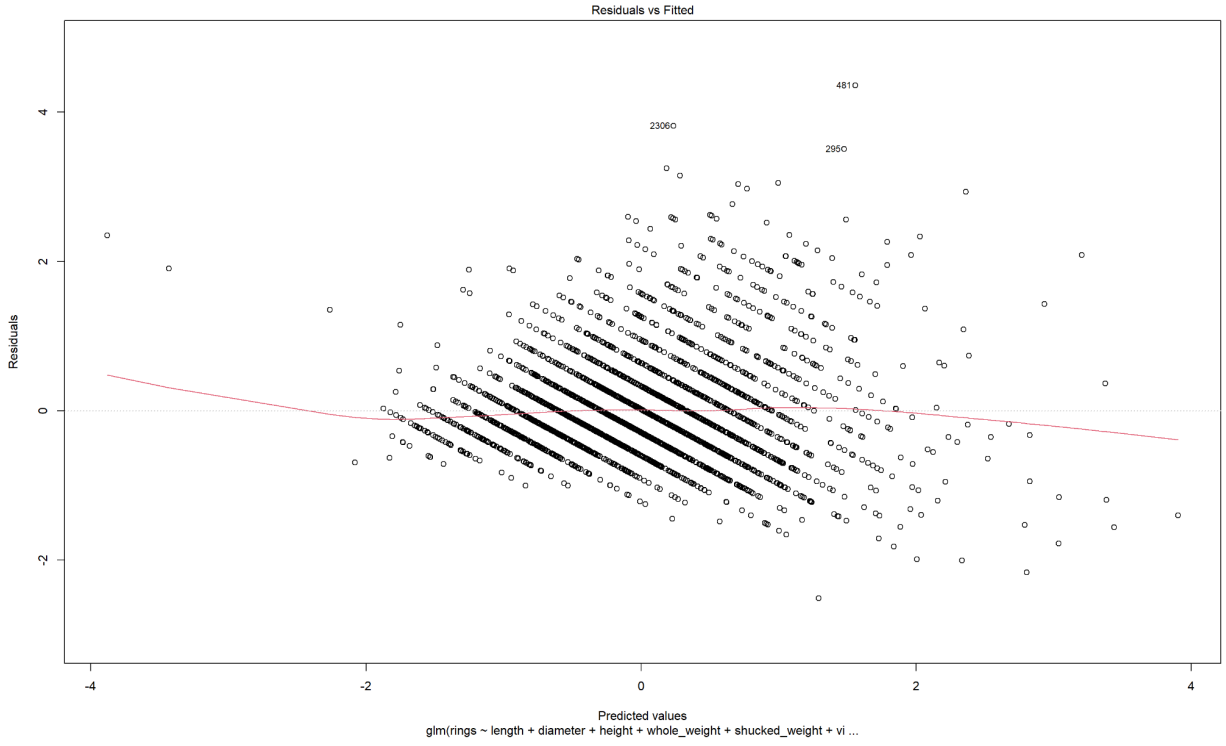
- Applying anova to the training data

```
> # anova - analysis of variance on the model result
> abalone.train_60.glm.anova <- anova(abalone.train_60.glm, test = "Chisq")
> abalone.train_60.glm.anova
Analysis of Deviance Table

Model: gaussian, link: identity

Response: rings

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                          2922     2951.3
length          1   918.29      2921     2033.0 < 2.2e-16 ***
diameter        1    69.50      2920     1963.5 < 2.2e-16 ***
height          1   132.12      2919     1831.4 < 2.2e-16 ***
whole_weight    1     3.30      2918     1828.1  0.008244 **
shucked_weight  1   398.24      2917     1429.8 < 2.2e-16 ***
viscera_weight  1    37.78      2916     1392.0 < 2.2e-16 ***
shell_weight    1    15.17      2915     1376.9 1.455e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

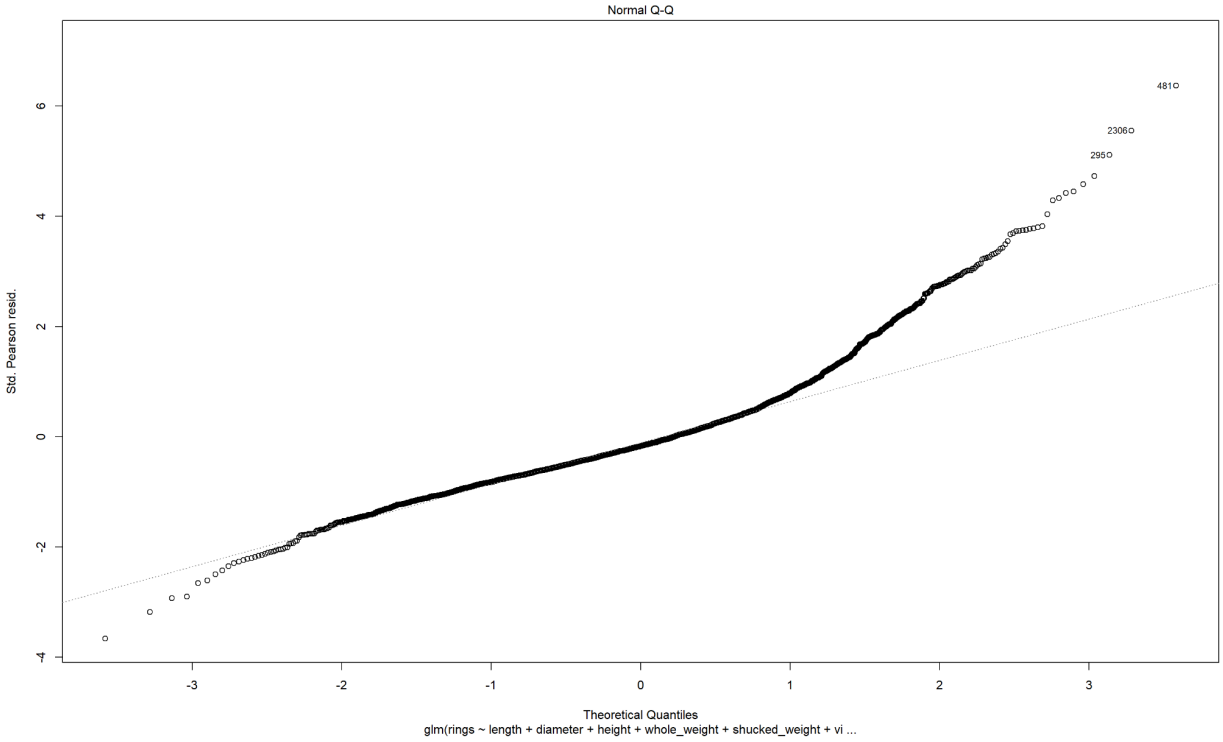
- Plotting the graphs

```
> # plotting graphs
> plot(abalone.train_60.glm)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
>
```

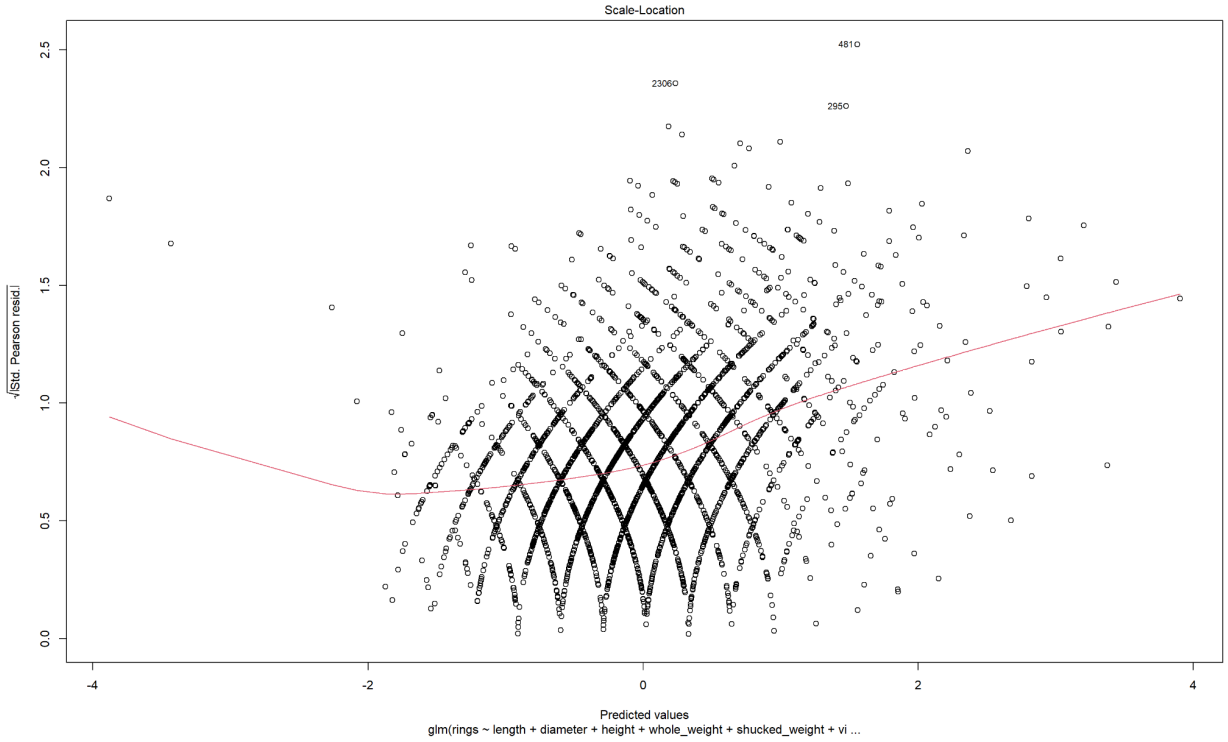
- Residual vs Fitted



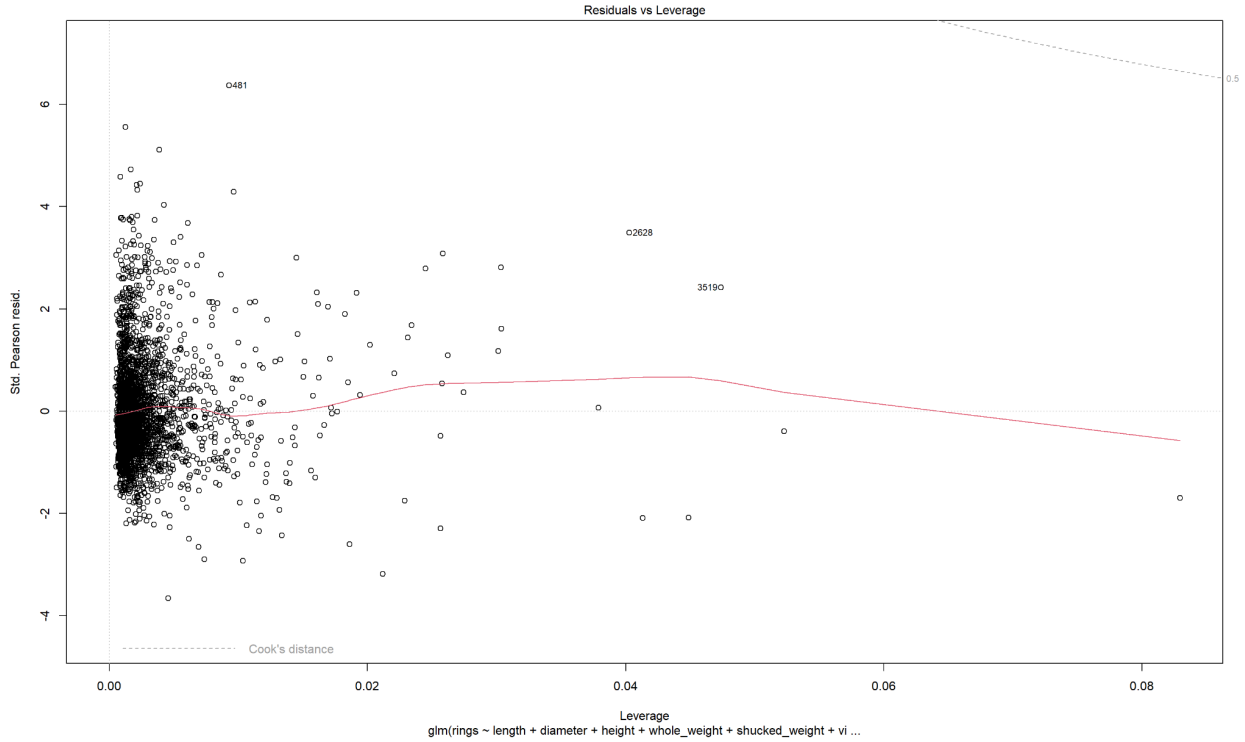
Normal Q-Q Plot



- Scale-Location Plot



Residuals vs Leverage Plot



- Using predict function in R
- Confidence intervals

```
> # predict
> abalone.test_60.predict <- predict(abalone.train_60.glm, newdata = abalone.test_60)
> length(abalone.test_60.predict)
[1] 1671
> summary(abalone.test_60.predict)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.873713 -0.434182 -0.000945  0.029797  0.461208  7.544918
> # confidence intervals
> confint(abalone.train_60.glm)
Waiting for profiling to be done...
                    2.5 %      97.5 %
(Intercept)    -0.01877274  0.03108867
length         -0.20184914  0.11266788
diameter        0.15565992  0.47517991
height          0.27472396  0.41720438
whole_weight    1.09314051  1.60011067
shucked_weight -1.47426182 -1.21546872
viscera_weight -0.46180597 -0.25500544
shell_weight    0.21074271  0.43360187
```

- Comparing the predicted values

```
> abalone.test_60.predict.k9
K-means clustering with 9 clusters of sizes 269, 319, 58, 19, 235, 88, 341, 147, 195

Cluster means:
        [,1]
1 -0.4457843
2 -0.1294083
3  1.5961947
4  2.6789917
5  0.5675036
6 -1.3375211
7  0.2057042
8  0.9909412
9 -0.8408551

Clustering vector:
   3    4    7    8   10   11   12   14   15   19   21   22   24   28   29   31   33   37   39   40   41   43   45   46   47   48   49
   7    2    8    7    8    7    1    7    2    1    9    1    2    7    5    5    5    7    7    9    1    6    6    9    9    2    9
  50   52   53   54   56   59   60   62   63   68   70   71   72   73   74   75   76   78   80   84   85   86   91   93   95   96   97
   7    1    1    1    7    6    2    1    2    3    6    5    1    8    8    5    7    5    5    3    5    4    7    8    3    3    5
  99  102  104  106  107  110  111  115  120  131  134  137  138  139  140  141  143  146  150  154  157  158  160  166  168  174  175
   2    7    2    7    5    2    2    2    9    8    9    9    9    1    1    7    8    2    6    8    8    4    3    3    4    2    6
 176  177  184  187  188  189  190  192  195  197  198  202  203  204  205  207  208  209  210  214  216  218  219  220  221  223  225
   9    6    5    7    8    7    5    5    9    7    3    8    7    8    1    1    1    8    9    7    5    1    7    1    2    7    1
 226  228  232  233  235  242  245  247  248  250  254  259  265  267  268  270  276  282  283  285  286  290  294  296  302  304  308
   2    9    8    3    2    6    9    9    9    1    8    5    9    7    1    2    3    1    2    8    8    5    8    9    8    9    4
 310  311  312  313  317  320  323  324  325  329  332  347  349  351  356  357  360  369  377  380  383  384  386  388  389  391  395
   2    8    2    8    5    9    1    6    9    1    1    5    1    7    3    5    8    5    5    8    2    2    1    2    7    1    9
 397  399  400  405  406  407  409  412  416  419  421  424  425  429  430  433  434  435  437  439  446  449  450  452  453  454  455
   2    5    7    1    7    1    7    7    8    8    3    9    6    8    8    8    5    1    1    9    8    5    7    5    8    7    5
 456  459  461  463  464  466  473  477  479  486  487  488  489  492  493  494  495  497  498  499  500  501  509  512  515  516  518
   7    1    9    6    6    6    1    2    4    5    7    5    2    5    5    8    8    3    5    5    8    5    5    2    6    9    9
[illegible]
2274 2276 2277 2281 2282 2284 2285 2293 2294 2295 2296 2297 2303 2305 2307 2308 2311 2312 2319 2321 2322 2326 2327 2328 2330 2331 2332
   5    3    3    2    2    1    1    9    9    2    7    7    2    7    5    7    1    9    7    1    7    1    2    2    7    1    8
2334 2335 2336 2337 2338 2342 2347 2348 2351 2352 2357 2364 2365 2370 2371 2372 2375 2378 2380 2381 2383 2389 2394 2395 2396 2397 2398
   3    3    8    5    7    9    7    2    5    5    8    5    8    5    1    6    1    7    9    6    7    7    9    7    5    5    8
2400 2402 2404 2408 2410 2418 2419 2420 2421 2425 2426 2427 2436 2437 2439 2441 2442 2443 2444 2446 2447 2451 2453 2455 2456 2457 2462
   7    7    9    5    8    6    2    2    9    6    5    5    7    8    8    2    5    2    2    7    8    2    1    6    6    6    5
2470
   7
 [ reached getOption("max.print") -- omitted 671 entries ]

Within cluster sum of squares by cluster:
[1]  2.679333  2.731211  2.694240 25.996447  2.566551  3.167734  3.355259  2.912442  2.888273
 (between_SS / total_SS =  94.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
>
```

- Comparing results of kmeans and knn for 9 clusters using the crosstable functionality provided gmodels pacakge

Total Observations in Table: 1671

abalone.test_60.predict.k7$cluster	abalone.test_60_kmeans_k9$cluster								
	1	2	3	4	5	6	7	8	9
1	0	2	11	165	4	6	68	9	0
265	15.224	29.898	22.610	551.894	9.613	33.682	16.312	4.679	20.616
	0.000	0.008	0.042	0.623	0.015	0.023	0.257	0.034	0.000
0.159	0.000	0.009	0.042	0.817	0.038	0.021	0.258	0.078	0.000
	0.000	0.001	0.007	0.099	0.002	0.004	0.041	0.005	0.000
2	0	23	114	27	4	61	166	0	15
410	23.555	16.384	37.924	10.272	18.384	1.025	158.183	28.217	8.951
	0.000	0.056	0.278	0.066	0.010	0.149	0.405	0.000	0.037
0.245	0.000	0.108	0.433	0.134	0.038	0.216	0.629	0.000	0.115
	0.000	0.014	0.068	0.016	0.002	0.037	0.099	0.000	0.009
3	0	0	0	10	3	0	0	106	0
119	6.837	15.169	18.730	1.337	2.681	20.154	18.801	1168.156	9.258
	0.000	0.000	0.000	0.084	0.025	0.000	0.000	0.891	0.000
0.071	0.000	0.000	0.000	0.050	0.029	0.000	0.000	0.922	0.000
	0.000	0.000	0.000	0.006	0.002	0.000	0.000	0.063	0.000
4	42	29	3	0	32	18	0	0	10
134	152.837	8.317	15.517	16.199	66.034	0.971	21.171	9.222	0.017
	0.313	0.216	0.022	0.000	0.239	0.134	0.000	0.000	0.075
0.080	0.438	0.136	0.011	0.000	0.305	0.064	0.000	0.000	0.077
	0.025	0.017	0.002	0.000	0.019	0.011	0.000	0.000	0.006

5	11	4	0	0	14	1	0	0	0
30									
	49.929	0.008	4.722	3.627	77.858	3.278	4.740	2.065	2.334
	0.367	0.133	0.000	0.000	0.467	0.033	0.000	0.000	0.000
0.018									
	0.115	0.019	0.000	0.000	0.133	0.004	0.000	0.000	0.000
	0.007	0.002	0.000	0.000	0.008	0.001	0.000	0.000	0.000
6	33	81	25	0	29	79	1	0	43
291									
	15.857	51.971	9.447	35.178	6.278	17.918	43.997	20.027	18.312
	0.113	0.278	0.086	0.000	0.100	0.271	0.003	0.000	0.148
0.174									
	0.344	0.380	0.095	0.000	0.276	0.279	0.004	0.000	0.331
	0.020	0.048	0.015	0.000	0.017	0.047	0.001	0.000	0.026
7	10	74	110	0	19	118	29	0	62
422									
	8.369	7.592	28.596	51.014	2.131	30.293	21.286	29.042	25.916
	0.024	0.175	0.261	0.000	0.045	0.280	0.069	0.000	0.147
0.253									
	0.104	0.347	0.418	0.000	0.181	0.417	0.110	0.000	0.477
	0.006	0.044	0.066	0.000	0.011	0.071	0.017	0.000	0.037
Column Total	96	213	263	202	105	283	264	115	130
1671									
	0.057	0.127	0.157	0.121	0.063	0.169	0.158	0.069	0.078

- Confusion Matrix
- Calculating the metrices for k=9

```
> # confusion matrix
> abalone.test_60.cm_k9 <- abalone.test_60.ct.k9$t
> abalone.test_60.cm_k9
   y
x     1   2   3   4   5   6   7   8   9
  1   0   2  11 165   4   6  68   9   0
  2   0  23 114  27   4  61 166   0  15
  3   0   0   0  10   3   0   0 106   0
  4  42  29   3   0  32  18   0   0  10
  5  11   4   0   0  14   1   0   0   0
  6  33  81  25   0  29  79   1   0  43
  7  10  74 110   0  19 118  29   0  62
> # calculate the metrics
> abalone.test_60.accuracy <- sum(diag(abalone.test_60.cm_k9)) / sum(abalone.test_60.cm_k9)
> abalone.test_60.precision <- diag(abalone.test_60.cm_k9) / colSums(abalone.test_60.cm_k9)
Warning message:
In diag(abalone.test_60.cm_k9)/colSums(abalone.test_60.cm_k9) :
  longer object length is not a multiple of shorter object length
> abalone.test_60.recall <- diag(abalone.test_60.cm_k9) / rowSums(abalone.test_60.cm_k9)
> abalone.test_60.specificity <- sapply(1:nrow(abalone.test_60.cm_k9), function(i){
+   TN <- sum(abalone.test_60.cm_k9[-i, -i])
+   FP <- sum(abalone.test_60.cm_k9[-i, i])
+   TN / (TN + FP)
+ })
> abalone.test_60.error <- 1 - abalone.test_60.accuracy
> # print results
> abalone.test_60.accuracy
[1] 0.08677439
> abalone.test_60.precision
        1         2         3         4         5         6         7         8         9
0.0000000 0.1079812 0.0000000 0.0000000 0.1333333 0.2791519 0.1098485 0.0000000 0.1769231
> abalone.test_60.recall
         1          2          3          4          5          6          7
0.00000000 0.05609756 0.00000000 0.00000000 0.46666667 0.27147766 0.06872038
> abalone.test_60.specificity
[1] 0.9317212 0.8493259 0.8305412 0.8685751 0.9445460 0.8521739 0.8118495
> abalone.test_60.error
[1] 0.9132256
>
```

Implementation for k=9 with 50-50 split

- Creating training labels with k-means

```
> # knn with k=9 and 50-50 split
> abalone.train_50_k9 <- kmeans(abalone.train_50, centers = 9)
> abalone.train_50_k9
K-means clustering with 9 clusters of sizes 148, 118, 298, 109, 356, 367, 247, 85, 360

Cluster means:
      length   diameter      height whole_weight shucked_weight viscera_weight shell_weight       rings
1  0.1558011  0.1850944  0.32575786   0.04777179     -0.2055489    -0.01872993    0.2996328  1.82493010
2 -2.2945559 -2.2552777 -1.88081467  -1.52002420     -1.4474611    -1.47559418   -1.5111208 -1.52233356
3 -1.2166096 -1.2216558 -1.07432523  -1.15979121     -1.1141894    -1.14404843   -1.1449217 -0.76002932
4  1.5301502  1.5152744  1.43836322   2.18792043      2.2289216     2.16039485    1.9720067  0.56405514
5  0.1171568  0.1061233  0.02096392  -0.12631401     -0.1068711    -0.13026592   -0.1148452 -0.12841186
6  0.6494721  0.6415958  0.49069847   0.49273358      0.5432287     0.49933955    0.4259677  0.02394872
7  1.0394769  1.0149347  0.90060928   1.15950816      1.1826340     1.18818880    1.0134432  0.22022426
8  0.9831282  1.0740576  1.17320797   1.26899052      0.8486188     1.08536160    1.6487917  2.31573328
9 -0.4752606 -0.4848592 -0.45497555  -0.68900332     -0.6582655    -0.67485900   -0.6654687 -0.33611279

Clustering vector:
3249 1914 2330 3683 2112 2631 1887 2158 1298  370 4040 3101 2605 1524 2007 1909 2645  969  976 1179 2845  277 1891  279 3758 4088 2555
   6    5    1    7    9    3    5    8    5    8    5    5    6    4    3    5    9    5    5    7    6    7    6    8    5    6    3
2836  562 3454 2812 1799 1227  486 2381 2383 3975  674 2370 2458 1045  949 1449 1405 2933  646 1260 2440  617  644  183 3146 2232 4095
   5    5    5    4    7    3    1    2    5    9    1    6    2    7    9    9    7    6    9    3    3    9    3    6    1    5    7
2442  417 2753 4131 2101 1804 1879  567 2881  435 1529 1572 1169 3861  849 1437   83 1330 3362  708 2629  275 3242 1623 3159 2750 3963
   5    8    5    5    3    6    5    9    9    9    4    9    6    1    5    3    1    5    3    3    2    6    7    5    5    9    4
 928 1589  701  327  222 1063 3506 1047 1590  161 1632 3777 3972 3639 1444 3125  972 3650 1881 3870 3043 2876 2655  820 2142  950 2194
   9    9    3    3    9    2    6    4    5    6    5    6    3    9    9    5    5    9    5    5    5    3    5    3    3    9    2
 561 3264 3465 1351  407 2193 3063 2719 2509 3340 2615 1706 1465 2538 1079 2262  825 1573 1557  492 2296 3697 2610  572  536 1284 2300
   9    1    7    6    9    6    7    3    3    6    7    4    5    7    3    5    3    9    3    1    5    7    7    9    9    9    9
   .
 [ reached getOption("max.print") -- omitted 1088 entries ]

Within cluster sum of squares by cluster:
[1] 247.05197  94.70387 169.49059 243.94381 238.59283 276.44841 231.02705 202.15595 250.20736
 (between_SS / total_SS =  87.9 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

- Then the cluster vector is passed to KNN

  The output is a vector which contains the predicted values for the test dataset.

```
> abalone.test_50_k9 <- knn(abalone.train_50, abalone.test_50, abalone.train_50_k9$cluster, k = 9)
> abalone.test_50_k9
   [1] 9 3 9 9 3 9 6 5 6 1 8 8 8 9 8 9 3 1 2 2 9 3 5 9 9 2 9 9 7 2 1 6 1 5 6 6 1 6 6 9 6 5 6 7 8 7 9 9 3 1 5 6 5 5 5 9 9 9 5 5 5 9 5 3 3 1
  [67] 3 3 2 3 8 8 3 2 9 3 3 3 5 9 3 2 6 6 8 1 7 4 8 4 5 3 2 3 2 6 1 8 7 7 6 6 5 1 1 9 1 5 3 5 1 9 9 9 9 3 3 6 1 1 1 8 3 2 2 1 3 3 6 6 1 1
 [133] 5 5 5 9 1 3 9 9 8 8 8 5 1 9 1 5 9 5 1 7 1 1 9 3 1 3 3 9 8 8 8 7 3 2 3 5 5 3 2 8 6 7 6 7 5 5 5 3 6 6 8 8 7 8 8 8 1 6 7 8 7 6 9 5 5 1
 [199] 9 3 3 5 9 6 9 5 5 5 8 6 7 1 1 9 5 2 6 8 1 1 5 3 3 1 3 3 5 1 6 1 4 8 8 5 3 2 2 7 8 5 5 9 5 1 9 8 1 1 1 6 6 8 7 8 1 7 6 6 8 6 6 8 1 5
 [265] 2 2 2 3 2 3 2 2 2 3 1 9 9 9 5 2 3 9 9 9 3 2 5 6 8 6 9 5 1 7 7 3 9 1 6 5 5 8 6 1 3 5 9 9 5 1 8 1 1 6 1 3 9 3 2 3 1 1 9 3 2 1 5 3 1 5
 [331] 7 2 9 9 2 3 3 6 8 5 9 3 1 3 9 1 1 9 1 1 1 5 2 3 2 2 9 1 9 9 9 3 2 3 3 2 3 2 6 1 1 5 1 5 5 1 5 5 3 1 1 1 1 7 6 8 7 1 8 6 1 1 6 5 6 3
 [397] 1 6 9 3 5 1 5 1 5 2 9 6 6 6 5 9 5 9 9 9 5 3 1 2 2 3 3 3 3 3 3 3 3 3 3 9 3 9 9 9 9 9 5 9 5 5 5 5 6 5 6 5 6 6 6 6 6 7 6 7 7 7 6 7 7 7 6
 [463] 7 7 7 7 4 7 4 4 2 2 2 2 3 3 3 3 3 3 3 3 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 5 5 9 9 5 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 7 6 7 7 6 7 7 7 6 7 7
 [529] 7 7 7 7 4 4 4 7 4 4 2 2 2 2 2 2 3 3 3 3 3 3 9 9 9 9 9 3 3 9 9 9 9 9 9 5 9 9 9 6 5 5 9 9 5 5 5 5 5 5 5 5 5 6 6 6 5 6 5 6 6 6 6 6 6 6
 [595] 6 6 6 7 6 7 5 6 7 7 7 7 7 4 7 7 4 4 4 4 4 2 2 2 3 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 9 9 9 9 9 9 9 9 9 9 9 5 9 9 9 5 5 9 5
 [661] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 5 5 5 6 5 6 6 6 6 6 6 6 6 6 6 5 6 6 6 6 6 6 6 5 6 6 6 6 6 6 6 6 6 6 7 6 6 7 6 7 7 6 8 7 7 7 6 7 7
 [727] 7 7 7 4 7 4 4 4 4 2 3 3 3 3 3 3 9 9 9 9 9 9 9 9 9 9 9 9 5 9 5 5 5 5 5 6 6 6 6 6 6 6 7 6 7 7 7 7 7 4 7 4 4 4 4 2 2 2 3 3 3 3 3 3 3
 [793] 3 3 3 3 3 3 3 3 3 9 9 9 9 9 9 9 9 9 9 9 9 9 9 5 5 5 9 5 5 5 5 5 5 5 9 5 5 9 5 5 5 5 5 5 5 5 5 5 5 5 6 5 6 6 5 5 5 5 6 6 5 5 6 6 6 6 6
 [859] 6 5 5 6 6 6 7 6 7 6 5 6 6 6 6 6 6 6 7 7 7 7 7 7 7 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 4 7 4 4 4 4 4 4 4 2 9 9 5 5 5 5 9 5 5 6 5 7 6 6
 [925] 7 6 7 6 7 6 7 7 7 7 7 7 7 4 4 4 2 2 3 3 3 3 3 3 3 3 3 9 9 9 9 9 9 9 5 5 9 9 9 6 5 5 5 5 5 5 5 5 5 5 6 5 1 5 5 6 6 6 5 6 6 5 6 6 7 6 6
 [991] 6 6 6 6 6 6 7 6 6 6
 [ reached getOption("max.print") -- omitted 1089 entries ]
Levels: 1 2 3 4 5 6 7 8 9
```

- Applying KMeans to generate labels for the test records.

```
> abalone.test_50_kmeans_k9 <- kmeans(abalone.test_50, centers = 9)
> abalone.test_50_kmeans_k9
K-means clustering with 9 clusters of sizes 164, 70, 285, 98, 158, 364, 354, 465, 131

Cluster means:
      length   diameter     height whole_weight shucked_weight viscera_weight shell_weight      rings
1 -0.2129806 -0.1678936 -0.0890265   -0.4043481     -0.5278778     -0.3657700   -0.2654824  0.8980869
2  1.0444952  1.0959467  1.6099805    1.3492364      0.8426616      1.1782810    1.7623164  2.3777648
3 -1.1735869 -1.1867810 -1.0538108   -1.1385520     -1.0819450     -1.1192417   -1.1284139 -0.7912824
4  1.5905010  1.5698551  1.4679964    2.3408585      2.3960602      2.2747048    2.1078924  0.5934101
5 -2.1629219 -2.1658455 -1.7875885   -1.4902307     -1.4232636     -1.4488836   -1.5048078 -1.3967334
6 -0.3350914 -0.3565226 -0.4117537   -0.5892672     -0.5300393     -0.5722133   -0.5983732 -0.5196510
7  0.9655141  0.9713678  0.8405741    1.0432354      1.0671910      1.0739089    0.9486813  0.2168261
8  0.4391523  0.4345985  0.2380449    0.2181081      0.2807610      0.2106306    0.1697401 -0.1501851
9  0.3993114  0.4440283  0.5618102    0.3321678      0.0939521      0.2697341    0.5333350  1.8152197

Clustering vector:
   1    6    9   18   19   20   23   24   25   29   32   33   34   36   37   38   40   42   43   44   48   49   50   54   57   59   60
   1    3    6    6    3    3    8    8    7    9    2    2    2    6    9    6    3    9    5    5    6    5    8    6    6    5    6
  65   68   70   71   75   76   79   80   81   85   86   87   89   90   92   93   94   95   96   98   99  101  102  104  105  107  108
   6    7    5    1    7    9    8    9    8    9    7    9    6    8    1    9    7    2    2    6    6    3    9    8    7    8    6
 109  110  111  112  115  116  117  118  119  120  122  123  124  125  126  127  129  130  133  135  136  137  138  140  141  146  148
   6    6    6    6    1    1    1    6    8    3    3    1    3    3    5    3    2    2    3    5    6    5    3    3    8    6    5
 149  152  154  158  160  163  166  167  169  174  176  177  178  179  180  181  182  185  189  190  191  197  199  202  205  209  211
   5    7    8    2    9    7    4    4    4    6    3    5    3    5    8    9    2    7    7    8    9    1    9    1    3    9    1
 212  214  215  217  219  224  226  227  228  229  230  231  232  233  236  238  240  241  247  249  252  253  254  257  261  262  263
   3    6    1    6    6    1    6    3    3    8    9    1    9    2    5    5    5    9    3    5    9    8    9    9    8    8    8

 [ reached getOption("max.print") -- omitted 1089 entries ]

Within cluster sum of squares by cluster:
[1] 152.0794 682.6480 170.3895 304.1626 103.4136 205.5020 371.7328 368.0881 188.2668
 (between_SS / total_SS =  85.2 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

- Obtaining labels for the test data by using K-Means on the test data

```
> abalone.test_50_k9.labels <- abalone.test_50_kmeans_k9$cluster
> length(abalone.test_50_k9.labels)
[1] 2089
> abalone.test_50_k9.labels
   1    6    9   18   19   20   23   24   25   29   32   33   34   36   37   38   40   42   43   44   48   49   50   54   57   59   60
   1    3    6    6    3    3    8    8    7    9    2    2    2    6    9    6    3    9    5    5    6    5    8    6    6    5    6
  65   68   70   71   75   76   79   80   81   85   86   87   89   90   92   93   94   95   96   98   99  101  102  104  105  107  108
   6    7    5    1    7    9    8    9    8    9    7    9    6    8    1    9    7    2    2    6    6    3    9    8    7    8    6
 109  110  111  112  115  116  117  118  119  120  122  123  124  125  126  127  129  130  133  135  136  137  138  140  141  146  148
   6    6    6    6    1    1    1    6    8    3    3    1    3    3    5    3    2    2    3    5    6    5    3    3    8    6    5
 149  152  154  158  160  163  166  167  169  174  176  177  178  179  180  181  182  185  189  190  191  197  199  202  205  209  211
   5    7    8    2    9    7    4    4    4    6    3    5    3    5    8    9    2    7    7    8    9    1    9    1    3    9    1
 212  214  215  217  219  224  226  227  228  229  230  231  232  233  236  238  240  241  247  249  252  253  254  257  261  262  263
   3    6    1    6    6    1    6    3    3    8    9    1    9    2    5    5    5    9    3    5    9    8    9    9    8    8    8
 266  267  268  269  270  271  273  278  280  281  283  285  287  289  290  291  292  293  294  299  301  302  303  304  305  313  314
   1    9    3    6    1    2    9    2    1    9    6    9    8    6    8    9    7    9    9    1    3    9    3    3    6    2    2
 315  317  320  322  326  328  331  332  333  335  337  339  341  342  344  346  347  349  350  354  355  356  357  360  363  364  366
   2    7    3    5    3    1    1    3    5    4    9    7    9    2    1    1    8    3    9    8    9    2    7    2    2    9    9
 368  374  376  377  378  384  385  387  389  392  394  396  397  398  400  401  402  406  410  411  413  415  416  419  422  423  425
   8    7    2    7    8    1    8    1    1    6    3    3    1    6    8    6    8    8    8    9    8    7    9    9    1    1    5
 426  427  430  431  440  441  442  443  444  445  446  448  449  450  451  452  453  456  459  464  466  468  469  471  472  473  475
   9    9    9    9    1    3    3    9    3    3    8    9    7    9    4    2    2    8    3    5    5    7    2    8    6    6    8
 476  477  481  482  483  484  485  487  488  493  494  496  497  499  501  502  504  507  508  510  513  514  515  516  517  519  520
   9    6    2    9    9    9    8    8    9    7    2    9    7    8    8    2    7    9    9    9    1    5    5    5    3    5    3
 524  525  526  530  531  532  533  535  537  539  540  542  543  544  546  547  549  550  551  552  554  556  557  558  559  566  570
   5    5    5    3    9    1    1    6    1    5    3    6    1    6    3    5    8    8    9    9    1    6    1    7    7    3    6
 571  576  577  579  580  581  582  585  587  589  590  591  593  594  596  598  599  601  603  606  610  612  613  614  615  616  618
   1    8    8    8    2    7    9    3    8    1    1    1    9    2    9    9    8    9    3    1    3    5    3    1    1    1    3
 620  622  626  627  629  640  642  647  648  650  651  654  655  660  661  662  663  664  665  666  667  668  669  670  671  679  684
   5    1    1    3    9    1    7    5    6    6    5    5    3    8    2    8    6    3    1    3    6    1    9    1    1    9    9
```

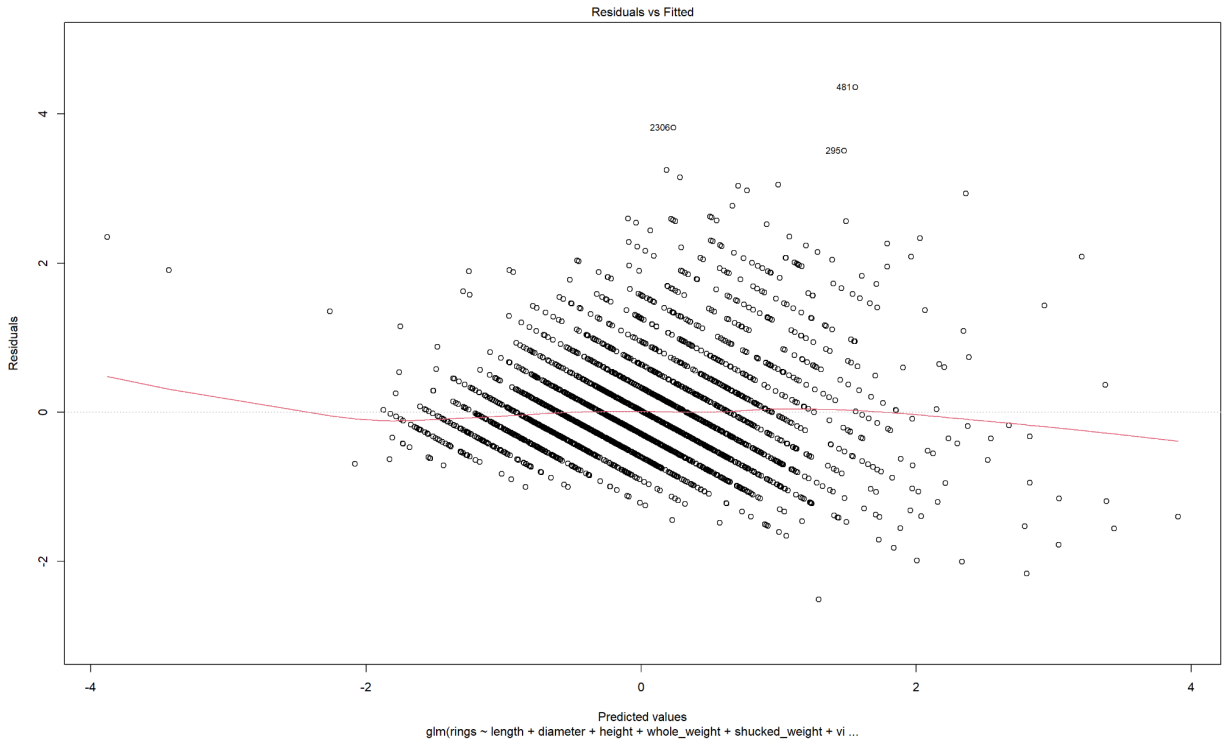
- Linear modeling using glm()

```
> # linear modelling using GLM
> abalone.train_50.glm <- glm(formula = rings ~ length + diameter + height + whole_weight + shucked_weight +  viscera_weight + shell_weight, fa
mily = gaussian, data = abalone.train_70)
> summary(abalone.train_50.glm)

Call:
glm(formula = rings ~ length + diameter + height + whole_weight +
    shucked_weight + viscera_weight + shell_weight, family = gaussian,
    data = abalone.train_70)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5111  -0.4220  -0.1166   0.2718   4.3572

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.006158   0.012720   0.484 0.628339
length         -0.044591   0.080235  -0.556 0.578426
diameter        0.315420   0.081512   3.870 0.000111 ***
height          0.345964   0.036348   9.518  < 2e-16 ***
whole_weight    1.346626   0.129331  10.412  < 2e-16 ***
shucked_weight -1.344865   0.066020 -20.371  < 2e-16 ***
viscera_weight -0.358406   0.052756  -6.794 1.32e-11 ***
shell_weight    0.322172   0.056853   5.667 1.60e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.4723417)

    Null deviance: 2951.3  on 2922  degrees of freedom
Residual deviance: 1376.9  on 2915  degrees of freedom
AIC: 6112.7

Number of Fisher Scoring iterations: 2
```

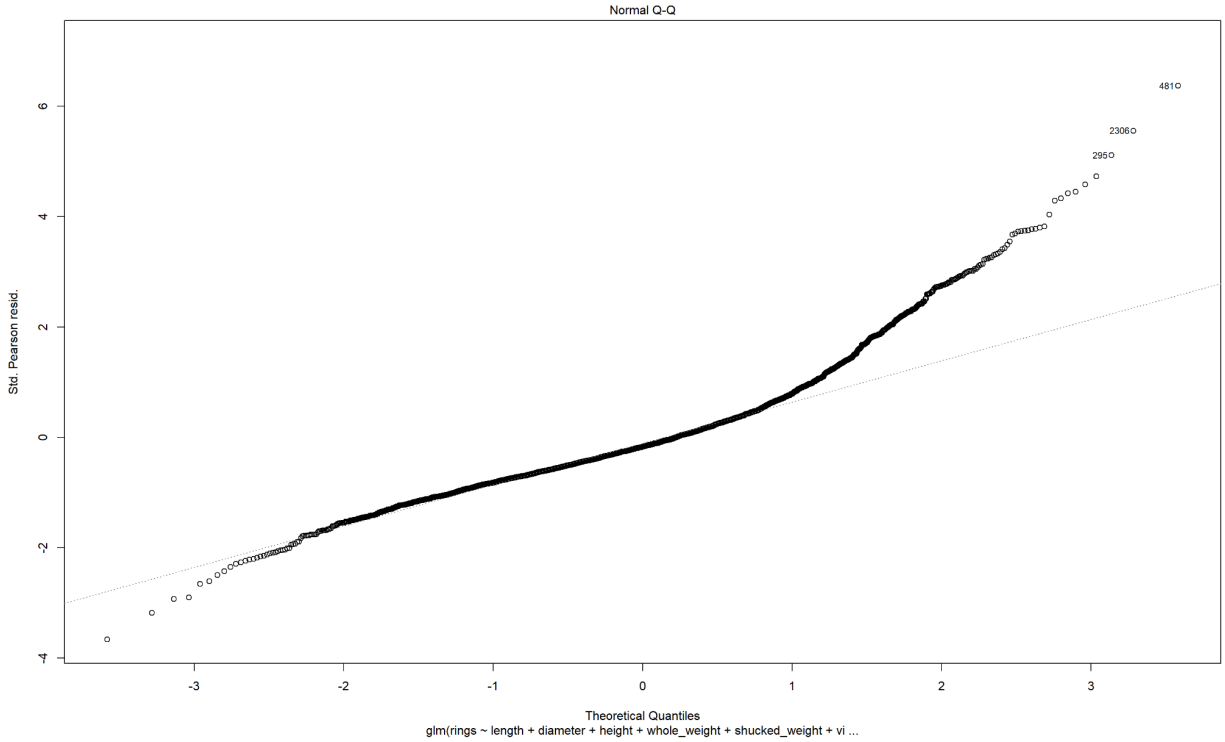
- Applying anova to the training data to get the deviance of the data

```
> abalone.train_50.glm.anova <- anova(abalone.train_50.glm, test = "Chisq")
> abalone.train_50.glm.anova
Analysis of Deviance Table

Model: gaussian, link: identity

Response: rings

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                          2922     2951.3
length          1   918.29      2921     2033.0 < 2.2e-16 ***
diameter        1    69.50      2920     1963.5 < 2.2e-16 ***
height          1   132.12      2919     1831.4 < 2.2e-16 ***
whole_weight    1     3.30      2918     1828.1  0.008244 **
shucked_weight  1   398.24      2917     1429.8 < 2.2e-16 ***
viscera_weight  1    37.78      2916     1392.0 < 2.2e-16 ***
shell_weight    1    15.17      2915     1376.9 1.455e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

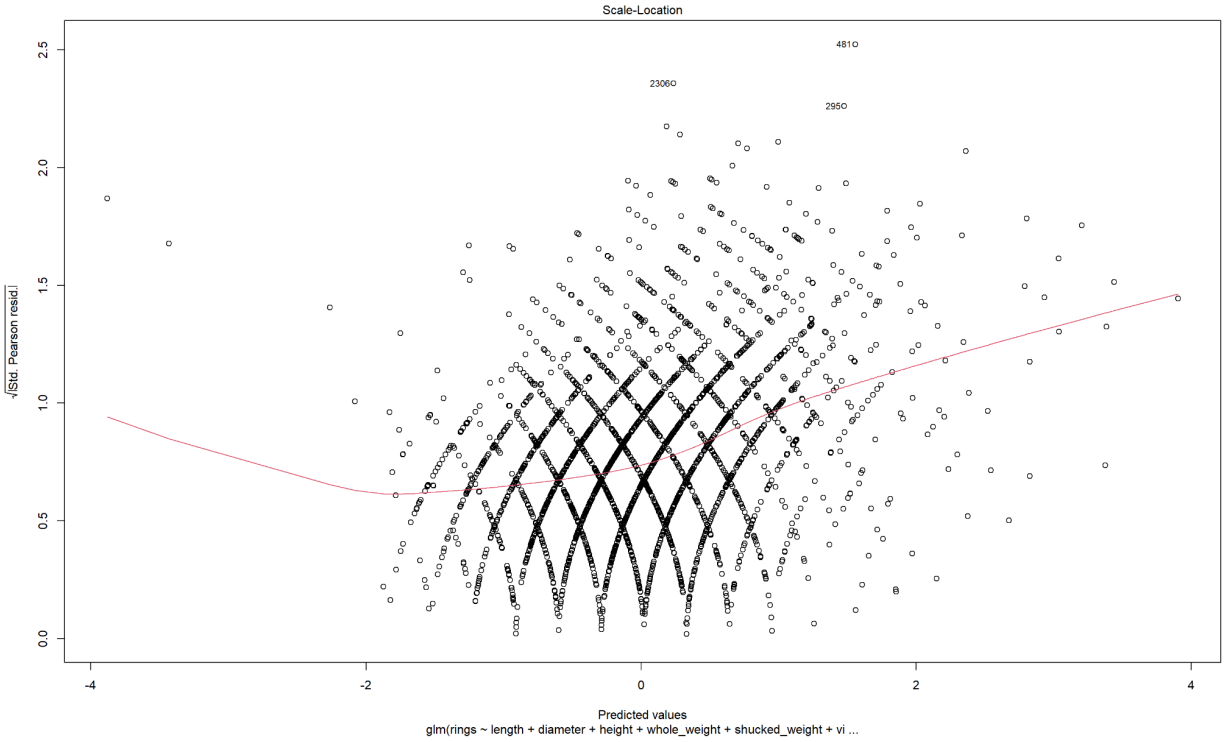
- Plotting the graphs
- Residual vs Fitted



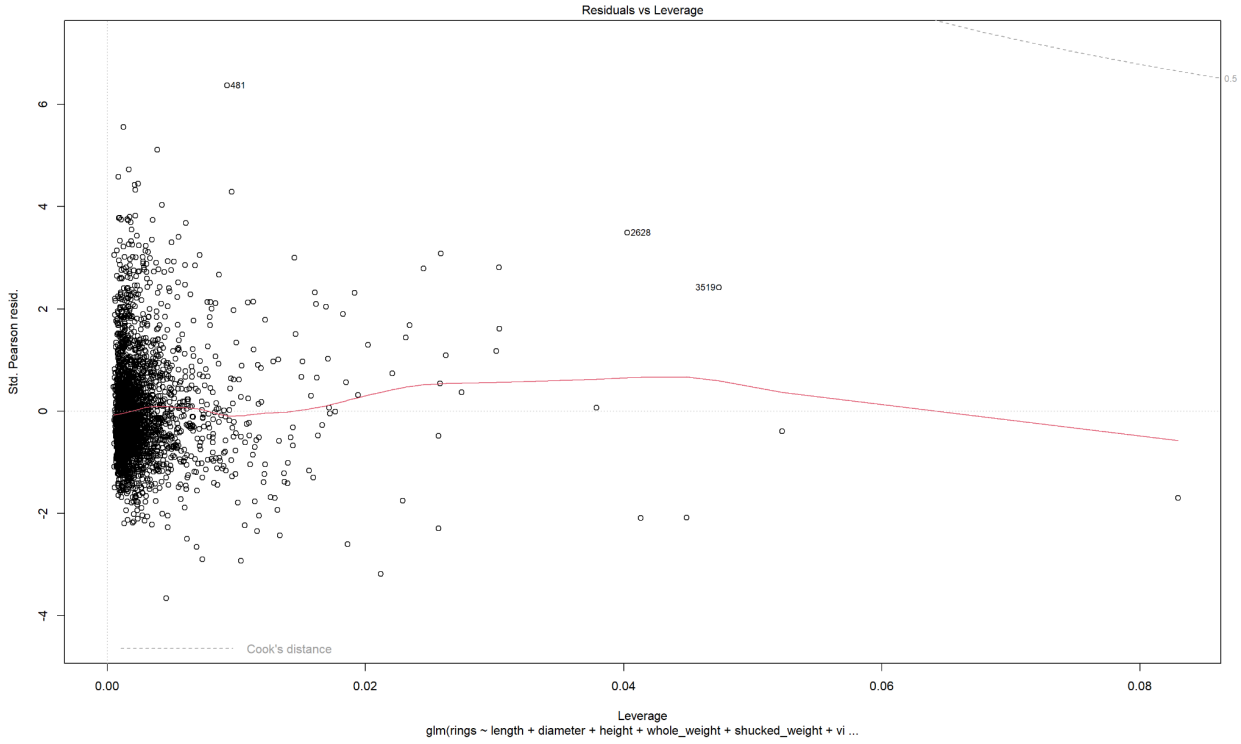
- Normal Q - Q



- Scale - Location



- Residual vs Leverage



- Predict Function

- Confidence intervals

```
> # predict
> abalone.test_50.predict <- predict(abalone.train_50.glm, newdata = abalone.test_50)
> length(abalone.test_50.predict)
[1] 2089
> summary(abalone.test_50.predict)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-3.433678 -0.481626 -0.051664  0.000136  0.425382  7.544918
> # confidence intervals
> confint(abalone.train_50.glm)
Waiting for profiling to be done...
                    2.5 %      97.5 %
(Intercept)    -0.01877274  0.03108867
length         -0.20184914  0.11266788
diameter        0.15565992  0.47517991
height          0.27472396  0.41720438
whole_weight    1.09314051  1.60011067
shucked_weight -1.47426182 -1.21546872
viscera_weight -0.46180597 -0.25500544
shell_weight    0.21074271  0.43360187
```

- Comparing the predicted values

```
> # comparing actual vs prediction
> abalone.test_50.predict.k9 <- kmeans(abalone.test_50.predict, centers = 9)
> abalone.test_50.predict.k9
K-means clustering with 9 clusters of sizes 451, 134, 269, 415, 369, 89, 1, 53, 308

Cluster means:
        [,1]
1 -0.1179088
2  1.2344863
3 -0.8911925
4  0.2452105
5 -0.4729756
6 -1.4412201
7  7.5449182
8  2.1593056
9  0.6714808

Clustering vector:
   1    6    9   18   19   20   23   24   25   29   32   33   34   36   37   38   40   42   43   44   48   49   50   54   57   59   60
   5    5    1    5    5    5    4    1    4    9    2    4    4    5    4    5    3    4    6    6    1    3    4    5    5    6    1
  65   68   70   71   75   76   79   80   81   85   86   87   89   90   92   93   94   95   96   98   99  101  102  104  105  107  108
   1    8    3    9    4    4    9    9    4    9    8    9    1    2    4    2    9    2    2    1    1    3    4    1    9    9    4
 109  110  111  112  115  116  117  118  119  120  122  123  124  125  126  127  129  130  133  135  136  137  138  140  141  146  148
   4    1    1    1    1    1    4    1    9    3    3    4    3    3    6    3    8    8    3    6    1    3    3    5    4    1    3
 149  152  154  158  160  163  166  167  169  174  176  177  178  179  180  181  182  185  189  190  191  197  199  202  205  209  211
   6    9    2    8    8    4    2    8    8    1    3    6    3    6    9    9    2    2    1    9    9    4    4    9    5    2    4
 212  214  215  217  219  224  226  227  228  229  230  231  232  233  236  238  240  241  247  249  252  253  254  257  261  262  263
   3    4    9    1    4    4    1    5    3    2    9    4    2    8    3    6    6    9    3    3    9    1    9    2    4    4    9
 266  267  268  269  270  271  273  278  280  281  283  285  287  289  290  291  292  293  294  299  301  302  303  304  305  313  314
   5    4    5    9    1    8    8    8    4    9    1    2    4    4    9    9    2    9    2    1    3    2    3    3    1    2    8
 315  317  320  322  326  328  331  332  333  335  337  339  341  342  344  346  347  349  350  354  355  356  357  360  363  364  366
   2    9    3    6    3    1    4    5    3    8    2    4    1    8    4    1    9    5    1    4    9    8    9    9    9    4    9
```

```
1894 1895 1896 1898 1899 1900 1902 1903 1905 1908 1911 1912 1913 1915 1916 1917 1919 1920 1923 1924 1925 1926 1927 1928 1929 1930 1931
   4    4    4    5    1    4    1    4    9    4    9    1    4    1    1    9    4    9    4    1    4    1    1    1    1    9    4
1933
   2
 [ reached getOption("max.print") -- omitted 1089 entries ]

Within cluster sum of squares by cluster:
[1] 4.575516 5.554876 4.814150 5.001656 4.402981 7.180316 0.000000 8.289457 5.860407
 (between_SS / total_SS =  96.2 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
> |
```

- Comparing results of kmeans and knn for 9 clusters using the crosstable functionality provided gmodels pacakge

```
> abalone.test_50.ct.k9 <- CrossTable(abalone.test_50.predict.k9$cluster, abalone.test_50_kmeans_k9$cluster, prop.chisq = TRUE)

   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  2089


                                 | abalone.test_50_kmeans_k9$cluster
abalone.test_50.predict.k9$cluster |         1 |         2 |         3 |         4 |         5 |         6 |         7 |         8 |         9
| Row Total |
-----------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------
|-----------|
                                 1 |        55 |         0 |         2 |        15 |         0 |       160 |        51 |       162 |         6
|       451 |
                                   |    10.843 |    15.112 |    57.594 |     1.792 |    34.111 |    84.347 |     8.459 |    37.810 |    17.555
|           |
                                   |     0.122 |     0.000 |     0.004 |     0.033 |     0.000 |     0.355 |     0.113 |     0.359 |     0.013
|     0.216 |
                                   |     0.335 |     0.000 |     0.007 |     0.153 |     0.000 |     0.440 |     0.144 |     0.348 |     0.046
|           |
                                   |     0.026 |     0.000 |     0.001 |     0.007 |     0.000 |     0.077 |     0.024 |     0.078 |     0.003
|           |
-----------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------
|-----------|
                                 2 |         1 |        31 |         0 |        16 |         0 |         0 |        41 |        14 |        31
|       134 |
                                   |     8.615 |   156.512 |    18.281 |    15.010 |    10.135 |    23.349 |    14.736 |     8.399 |    60.766
|           |
                                   |     0.007 |     0.231 |     0.000 |     0.119 |     0.000 |     0.000 |     0.306 |     0.104 |     0.231
|     0.064 |
                                   |     0.006 |     0.443 |     0.000 |     0.163 |     0.000 |     0.000 |     0.116 |     0.030 |     0.237
|           |
                                   |     0.000 |     0.015 |     0.000 |     0.008 |     0.000 |     0.000 |     0.020 |     0.007 |     0.015
|           |
-----------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------
|-----------|
                                 3 |         0 |         0 |       169 |         1 |        73 |        17 |         3 |         6 |         0
|       269 |
                                   |    21.118 |     9.014 |   476.941 |    10.699 |   136.269 |    19.038 |    39.782 |    48.479 |    16.869
|           |
                                   |     0.000 |     0.000 |     0.628 |     0.004 |     0.271 |     0.063 |     0.011 |     0.022 |     0.000
|     0.129 |
                                   |     0.000 |     0.000 |     0.593 |     0.010 |     0.462 |     0.047 |     0.008 |     0.013 |     0.000
|           |
                                   |     0.000 |     0.000 |     0.081 |     0.000 |     0.035 |     0.008 |     0.001 |     0.003 |     0.000
```

	4	70	2	0	22	0	17	110	165	29
415		42.978	10.194	56.618	0.329	31.388	42.309	22.383	57.094	0.340
		0.169	0.005	0.000	0.053	0.000	0.041	0.265	0.398	0.070
0.199		0.427	0.029	0.000	0.224	0.000	0.047	0.311	0.355	0.221
		0.034	0.001	0.000	0.011	0.000	0.008	0.053	0.079	0.014
	5	13	0	112	5	0	167	18	54	0
369		8.803	12.365	75.517	8.755	27.909	164.051	31.712	9.639	23.140
		0.035	0.000	0.304	0.014	0.000	0.453	0.049	0.146	0.000
0.177		0.079	0.000	0.393	0.051	0.000	0.459	0.051	0.116	0.000
		0.006	0.000	0.054	0.002	0.000	0.080	0.009	0.026	0.000
	6	0	0	2	1	85	1	0	0	0
89		6.987	2.982	8.472	2.415	910.051	13.572	15.082	19.811	5.581
		0.000	0.000	0.022	0.011	0.955	0.011	0.000	0.000	0.000
0.043		0.000	0.000	0.007	0.010	0.538	0.003	0.000	0.000	0.000
		0.000	0.000	0.001	0.000	0.041	0.000	0.000	0.000	0.000
	7	0	1	0	0	0	0	0	0	0
1		0.079	27.876	0.136	0.047	0.076	0.174	0.169	0.223	0.063
		0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000		0.000	0.014	0.000	0.000	0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	8	0	27	0	18	0	0	5	0	3
53		4.161	358.256	7.231	96.797	4.009	9.235	1.765	11.798	0.032
		0.000	0.509	0.000	0.340	0.000	0.000	0.094	0.000	0.057
0.025										
		0.000	0.386	0.000	0.184	0.000	0.000	0.014	0.000	0.023
		0.000	0.013	0.000	0.009	0.000	0.000	0.002	0.000	0.001
	9	25	9	0	20	0	2	126	64	62
308		0.028	0.169	42.020	2.133	23.295	49.742	104.370	0.303	94.336
		0.081	0.029	0.000	0.065	0.000	0.006	0.409	0.208	0.201
0.147		0.152	0.129	0.000	0.204	0.000	0.005	0.356	0.138	0.473
		0.012	0.004	0.000	0.010	0.000	0.001	0.060	0.031	0.030
	Column Total	164	70	285	98	158	364	354	465	131
2089		0.079	0.034	0.136	0.047	0.076	0.174	0.169	0.223	0.063

- Confusion Matrix
- Calculating the metrices for k=9

```
> # confusion matrix
> abalone.test_50.cm_k9 <- abalone.test_50.ct.k9$t
> abalone.test_50.cm_k9
   y
x     1   2   3   4   5   6   7   8   9
  1  55   0   2  15   0 160  51 162   6
  2   1  31   0  16   0   0  41  14  31
  3   0   0 169   1  73  17   3   6   0
  4  70   2   0  22   0  17 110 165  29
  5  13   0 112   5   0 167  18  54   0
  6   0   0   2   1  85   1   0   0   0
  7   0   1   0   0   0   0   0   0   0
  8   0  27   0  18   0   0   5   0   3
  9  25   9   0  20   0   2 126  64  62
> # calculate the metrics
> abalone.test_50.accuracy <- sum(diag(abalone.test_50.cm_k9)) / sum(abalone.test_50.cm_k9)
> abalone.test_50.precision <- diag(abalone.test_50.cm_k9) / colSums(abalone.test_50.cm_k9)
> abalone.test_50.recall <- diag(abalone.test_50.cm_k9) / rowSums(abalone.test_50.cm_k7)
Warning message:
In diag(abalone.test_50.cm_k9)/rowSums(abalone.test_50.cm_k7) :
  longer object length is not a multiple of shorter object length
> abalone.test_50.specificity <- sapply(1:nrow(abalone.test_50.cm_k9), function(i){
+   TN <- sum(abalone.test_50.cm_k9[-i, -i])
+   FP <- sum(abalone.test_50.cm_k9[-i, i])
+   TN / (TN + FP)
+ })
> abalone.test_50.error <- 1 - abalone.test_50.accuracy
> # print results
> abalone.test_50.accuracy
[1] 0.1627573
> abalone.test_50.precision
         1          2          3          4          5          6          7          8          9
0.335365854 0.442857143 0.592982456 0.224489796 0.000000000 0.002747253 0.000000000 0.000000000 0.473282443
> abalone.test_50.recall
         1          2          3          4          5          6          7          8          9
0.089576547 0.075609756 0.308957952 0.392857143 0.000000000 0.003355705 0.000000000 0.000000000 0.151219512
> abalone.test_50.specificity
[1] 0.9334554 0.9800512 0.9362637 0.9545998 0.9081395 0.8185000 0.8304598 0.7716110 0.9612577
> abalone.test_50.error
[1] 0.8372427
> 
```

Linear Modelling of the data:

1. 70-30 split data

```
> # linear modelling on the data
> # 70-30 split
> abalone.train_70.glm <- glm(formula = rings ~ length + diameter + height + whole_weight + shucked_weight +  viscera_weight + shell_wei
ght, family = gaussian, data = abalone.train_70)
> summary(abalone.train_70.glm)

Call:
glm(formula = rings ~ length + diameter + height + whole_weight +
    shucked_weight + viscera_weight + shell_weight, family = gaussian,
    data = abalone.train_70)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5111  -0.4220  -0.1166   0.2718   4.3572

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.006158   0.012720   0.484 0.628339
length         -0.044591   0.080235  -0.556 0.578426
diameter        0.315420   0.081512   3.870 0.000111 ***
height          0.345964   0.036348   9.518  < 2e-16 ***
whole_weight    1.346626   0.129331  10.412  < 2e-16 ***
shucked_weight -1.344865   0.066020 -20.371  < 2e-16 ***
viscera_weight -0.358406   0.052756  -6.794 1.32e-11 ***
shell_weight    0.322172   0.056853   5.667 1.60e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.4723417)

    Null deviance: 2951.3  on 2922  degrees of freedom
Residual deviance: 1376.9  on 2915  degrees of freedom
AIC: 6112.7

Number of Fisher Scoring iterations: 2

>
```

- The linear fit of the model can be found in the previous section of knn

2. 60-40 split data

```
> # 60-40 split
> abalone.train_60.glm <- glm(formula = rings ~ length + diameter + height + whole_weight + shucked_weight +  viscera_weight + shell_weigh
t, family = gaussian, data = abalone.train_60)
> summary(abalone.train_60.glm)

Call:
glm(formula = rings ~ length + diameter + height + whole_weight +
    shucked_weight + viscera_weight + shell_weight, family = gaussian,
    data = abalone.train_60)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1259  -0.4239  -0.1193   0.2825   4.2863

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.01130    0.01358   0.832    0.406
length         -0.10258    0.08510  -1.205    0.228
diameter        0.36785    0.08691   4.232 2.40e-05 ***
height          0.37630    0.03821   9.847  < 2e-16 ***
whole_weight    1.48207    0.13893  10.668  < 2e-16 ***
shucked_weight -1.39747    0.07008 -19.940  < 2e-16 ***
viscera_weight -0.37863    0.05650  -6.702 2.54e-11 ***
shell_weight    0.24076    0.06029   3.993 6.71e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.4616922)

    Null deviance: 2544.4  on 2505  degrees of freedom
Residual deviance: 1153.3  on 2498  degrees of freedom
AIC: 5184.9

Number of Fisher Scoring iterations: 2
```

- The linear fit of the model can be found in the previous section of knn

3. 50-50 split data

```
>
> # 50-50 split
> abalone.train_50.glm <- glm(formula = rings ~ length + diameter + height + whole_weight + shucked_weight +  viscera_weight + shell_wei
ght, family = gaussian, data = abalone.train_50)
> summary(abalone.train_50.glm)

Call:
glm(formula = rings ~ length + diameter + height + whole_weight +
    shucked_weight + viscera_weight + shell_weight, family = gaussian,
    data = abalone.train_50)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5805  -0.4203  -0.1056   0.2923   3.4587

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.003018   0.014912   0.202 0.839620
length         -0.121917   0.093646  -1.302 0.193094
diameter        0.363054   0.096180   3.775 0.000165 ***
height          0.361072   0.042672   8.462  < 2e-16 ***
whole_weight    1.145549   0.149993   7.637 3.36e-14 ***
shucked_weight -1.265420   0.077567 -16.314  < 2e-16 ***
viscera_weight -0.340962   0.060029  -5.680 1.54e-08 ***
shell_weight    0.445937   0.067555   6.601 5.17e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.4641387)

    Null deviance: 2064.31  on 2087  degrees of freedom
Residual deviance:  965.41  on 2080  degrees of freedom
AIC: 4332.8

Number of Fisher Scoring iterations: 2
```

- The linear fit of the model can be found in the previous section of knn

Predictions on the dataset using dependent variables:

1. 70-30 split

```
>
>
> # prediction using dependent variables
> # 70-30 split
> summary(abalone.test_70.predict)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.840315 -0.494013 -0.037437 -0.005171  0.385751  7.544918
> summary(abalone.test_70$rings)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-2.15053 -0.59975 -0.28959 -0.02568  0.33073  4.67293
>
```

2. 60-40 split

```
>
> # 60-40 split
> summary(abalone.test_60.predict)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.873713 -0.434182 -0.000945  0.029797  0.461208  7.544918
> summary(abalone.test_60$rings)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-2.150534 -0.599747  0.020568  0.007575  0.330726  5.293245
>
```

3. 50-50 split

```
>
> # 50-50 split
> summary(abalone.test_50.predict)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-3.433678 -0.481626 -0.051664  0.000136  0.425382  7.544918
> summary(abalone.test_50$rings)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-2.150534 -0.599747 -0.289589 -0.008978  0.330726  5.913560
> |
```

Analysis of Variance:

1. 70-30 split

```
>
>
> # analysis of variance
> # 70-30 split
> abalone.train_70.glm.anova <- anova(abalone.train_70.glm, test = "Chisq")
> abalone.train_70.glm.anova
Analysis of Deviance Table

Model: gaussian, link: identity

Response: rings

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                            2922     2951.3
length          1   918.29      2921     2033.0 < 2.2e-16 ***
diameter        1    69.50      2920     1963.5 < 2.2e-16 ***
height          1   132.12      2919     1831.4 < 2.2e-16 ***
whole_weight    1     3.30      2918     1828.1  0.008244 **
shucked_weight  1   398.24      2917     1429.8 < 2.2e-16 ***
viscera_weight  1    37.78      2916     1392.0 < 2.2e-16 ***
shell_weight    1    15.17      2915     1376.9 1.455e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
>
> |
```

2. 60-40 split

```
> # 60-40 split
> abalone.train_60.glm.anova <- anova(abalone.train_60.glm, test = "Chisq")
> abalone.train_60.glm.anova
Analysis of Deviance Table

Model: gaussian, link: identity

Response: rings

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                          2505     2544.4
length          1   816.53      2504     1727.9 < 2.2e-16 ***
diameter        1    62.36      2503     1665.5 < 2.2e-16 ***
height          1   128.97      2502     1536.6 < 2.2e-16 ***
whole_weight    1     3.83      2501     1532.7  0.003968 **
shucked_weight  1   338.73      2500     1194.0 < 2.2e-16 ***
viscera_weight  1    33.33      2499     1160.7 < 2.2e-16 ***
shell_weight    1     7.36      2498     1153.3 6.518e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

3. 50-50 split

```
>
> # 50-50 split
> abalone.train_50.glm.anova <- anova(abalone.train_50.glm, test = "Chisq")
> abalone.train_50.glm.anova
Analysis of Deviance Table

Model: gaussian, link: identity

Response: rings

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                          2087    2064.31
length          1   620.12      2086    1444.19 < 2.2e-16 ***
diameter        1    51.02      2085    1393.17 < 2.2e-16 ***
height          1    99.36      2084    1293.81 < 2.2e-16 ***
whole_weight    1     2.59      2083    1291.22    0.0182 *
shucked_weight  1   275.04      2082    1016.18 < 2.2e-16 ***
viscera_weight  1    30.55      2081     985.63 4.957e-16 ***
shell_weight    1    20.22      2080     965.41 4.082e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

Confidence Intervals:

1. 70-30 split

```
>
> # confidence intervals
> # 70-30 split
> confint(abalone.train_70.glm)
Waiting for profiling to be done...
                    2.5 %      97.5 %
(Intercept)    -0.01877274  0.03108867
length         -0.20184914  0.11266788
diameter        0.15565992  0.47517991
height          0.27472396  0.41720438
whole_weight    1.09314051  1.60011067
shucked_weight -1.47426182 -1.21546872
viscera_weight -0.46180597 -0.25500544
shell_weight    0.21074271  0.43360187
>
>
```

2. 60-40 split

```
> # 60-40 split
> confint(abalone.train_60.glm)
Waiting for profiling to be done...
                     2.5 %      97.5 %
(Intercept)    -0.01532237  0.03791532
length         -0.26937661  0.06421438
diameter        0.19749809  0.53819688
height          0.30139743  0.45119374
whole_weight    1.20978302  1.75435910
shucked_weight -1.53482854 -1.26010966
viscera_weight -0.48936757 -0.26789579
shell_weight    0.12259029  0.35893186
>
```

3. 50-50 split

```
>
> # 50-50 split
> confint(abalone.train_50.glm)
Waiting for profiling to be done...
                    2.5 %      97.5 %
(Intercept)    -0.02620845  0.03224492
length         -0.30545904  0.06162459
diameter        0.17454547  0.55156264
height          0.27743686  0.44470773
whole_weight    0.85156771  1.43953040
shucked_weight -1.41744921 -1.11339173
viscera_weight -0.45861595 -0.22330773
shell_weight    0.31353080  0.57834314
>
```

Observations:

- Accuracy Table

Ration Clusters	70-30%	60-40%	50-50
5	26%	15%	14%
7	28%	4%	4%
9	8%	8%	16%

- Cluster metrics for 70-30 split

Clusters	Accuracy	Precision	Recall	Specificity	Error
5	0.26874	0.25552050	0.47928994	0.7824885	0.73126
7	0.2830941	1	0.04347	1	0.7169059
9	0.08133	0.2755	0.29914	0.80677	0.91866

- Cluster metrics for 60-40 split

Clusters	Accuracy	Precision	Recall	Specificity	Error
5	0.154398	0.23913043	0.22	0.6565566	0.8456014
7	0.04608019	0.08378378	0.07345972	0.7285829	0.953919
9	0.08677	0.1769	0.068	0.811	0.9132

- Cluster metrics for 50-50 split

Clusters	Accuracy	Precision	Recall	Specificity	Error
5	0.1450455	0.0893	0.06687	0.8387257	0.8549

7	0.04930	0.0022	0.0033	0.765316	0.95069
9	0.1627	0.4732	0.151219	0.961	0.837