Assignement2_DS

Yassine Lahbabi

27/04/2020

```
# To clear everything in the WorkSpace.
rm(list = ls())
```

1 Assignement 2 : Data Statistics

My answers to Assignment 2 for Data Statistics are found below.

1.1 Question 1: Clustering

1.1.1 .a : Exploring the Dataset

In 1.a, we need to explore the dataset and summarise the variables that we have, and include any plots that we need for our study :

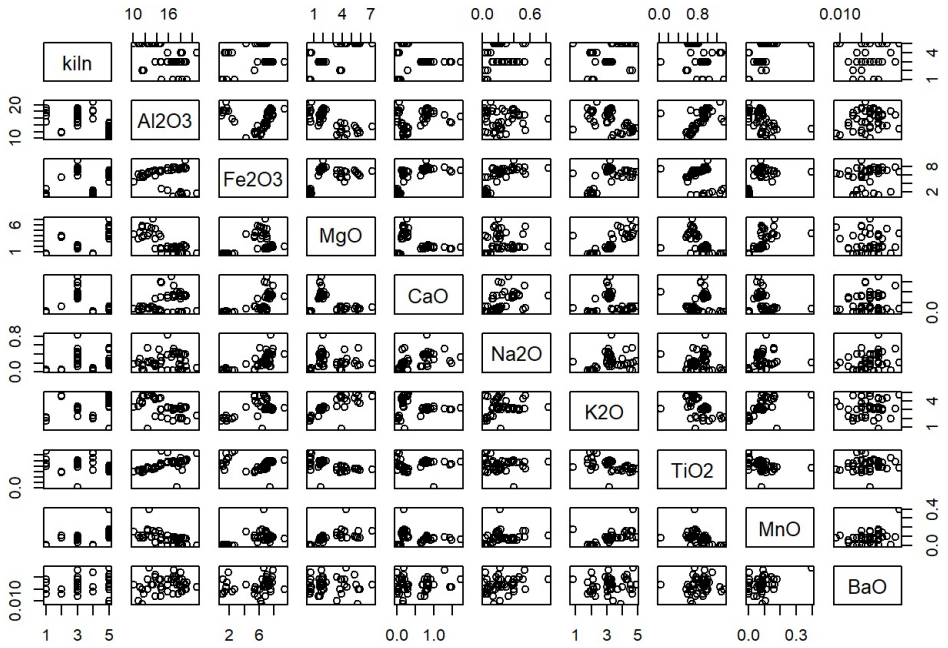
```
pottery.complete <- read.csv("pottery.csv")

# Looking at the Top 6 values to make sure that the dataset has been read correctly.
head(pottery.complete)
```

```
##         kiln Al2O3 Fe2O3  MgO  CaO Na2O  K2O TiO2   MnO   BaO
## 1 Gloucester  18.8  9.52 2.00 0.79 0.40 3.20 1.01 0.077 0.015
## 2 Gloucester  16.9  7.33 1.65 0.84 0.40 3.05 0.99 0.067 0.018
## 3 Gloucester  18.2  7.64 1.82 0.77 0.40 3.07 0.98 0.087 0.014
## 4 Gloucester  17.4  7.48 1.71 1.01 0.40 3.16 0.03 0.084 0.017
## 5 Gloucester  16.9  7.29 1.56 0.76 0.40 3.05 1.00 0.063 0.019
## 6 Gloucester  17.8  7.24 1.83 0.92 0.43 3.12 0.93 0.061 0.019
```

```
# Looking at the different structures of the variables in our dataset.
str(pottery.complete)
```

```
## 'data.frame':    48 obs. of  10 variables:
##  $ kiln : Factor w/ 5 levels "Ashley Rails",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ Al2O3: num  18.8 16.9 18.2 17.4 16.9 17.8 18.8 16.5 18 15.8 ...
##  $ Fe2O3: num  9.52 7.33 7.64 7.48 7.29 7.24 7.45 7.05 7.42 7.15 ...
##  $ MgO  : num  2 1.65 1.82 1.71 1.56 1.83 2.06 1.81 2.06 1.62 ...
##  $ CaO  : num  0.79 0.84 0.77 1.01 0.76 0.92 0.87 1.73 1 0.71 ...
##  $ Na2O : num  0.4 0.4 0.4 0.4 0.4 0.43 0.25 0.33 0.28 0.38 ...
##  $ K2O  : num  3.2 3.05 3.07 3.16 3.05 3.12 3.26 3.2 3.37 3.25 ...
##  $ TiO2 : num  1.01 0.99 0.98 0.03 1 0.93 0.98 0.95 0.96 0.93 ...
##  $ MnO  : num  0.077 0.067 0.087 0.084 0.063 0.061 0.072 0.066 0.072 0.062 ...
##  $ BaO  : num  0.015 0.018 0.014 0.017 0.019 0.019 0.017 0.019 0.017 0.017 ...
```

```
# Plotting the pottery dataset (Since our variables may not be related we will not compare them between themselv
es).
plot(pottery.complete)
```



I then look at the top 6 rows to make sure the dataset has been read in correctly:

1.1.2 .b : Adjusting the Dataset

In 1.b, we need to remove the column 'kiln'.

```
# Removing the kiln column :
pottery = pottery.complete[c(-1)]
# Checking out if the kiln column has been removed.
str(pottery)
```

```
## 'data.frame':    48 obs. of  9 variables:
##  $ Al2O3: num  18.8 16.9 18.2 17.4 16.9 17.8 18.8 16.5 18 15.8 ...
##  $ Fe2O3: num  9.52 7.33 7.64 7.48 7.29 7.24 7.45 7.05 7.42 7.15 ...
##  $ MgO  : num  2 1.65 1.82 1.71 1.56 1.83 2.06 1.81 2.06 1.62 ...
##  $ CaO  : num  0.79 0.84 0.77 1.01 0.76 0.92 0.87 1.73 1 0.71 ...
##  $ Na2O : num  0.4 0.4 0.4 0.4 0.4 0.43 0.25 0.33 0.28 0.38 ...
##  $ K2O  : num  3.2 3.05 3.07 3.16 3.05 3.12 3.26 3.2 3.37 3.25 ...
##  $ TiO2 : num  1.01 0.99 0.98 0.03 1 0.93 0.98 0.95 0.96 0.93 ...
##  $ MnO  : num  0.077 0.067 0.087 0.084 0.063 0.061 0.072 0.066 0.072 0.062 ...
##  $ BaO  : num  0.015 0.018 0.014 0.017 0.019 0.019 0.017 0.019 0.017 0.017 ...
```

We will now standardize the data because it will make our dataset have the same scale, so our data becomes internally consistent because it will allow us to have the same content and format.

```
scale.pottery = scale(pottery)

# Checking out the structure and the values again.
head(scale.pottery)
```

```
##          Al2O3     Fe2O3        MgO       CaO      Na2O         K2O       TiO2
## [1,] 1.1787012 1.5744454 -0.3148639 0.6196585 0.8884038  0.02101780  0.7323502
## [2,] 0.4756433 0.6410724 -0.5176903 0.7308080 0.8884038 -0.14170065  0.6388587
## [3,] 0.9566829 0.7731937 -0.4191746 0.5751987 0.8884038 -0.12000486  0.5921129
## [4,] 0.6606585 0.7050021 -0.4829200 1.1087164 0.8884038 -0.02237379 -3.8487339
## [5,] 0.4756433 0.6240245 -0.5698457 0.5529688 0.8884038 -0.14170065  0.6856044
## [6,] 0.8086707 0.6027146 -0.4133796 0.9086472 1.0608164 -0.06576537  0.3583841
##              MnO         BaO
## [1,] -0.04129390 -0.55791357
## [2,] -0.19145353  0.41003287
## [3,]  0.10886573 -0.88056239
## [4,]  0.06381784  0.08738405
## [5,] -0.25151738  0.73268168
## [6,] -0.28154931  0.73268168
```

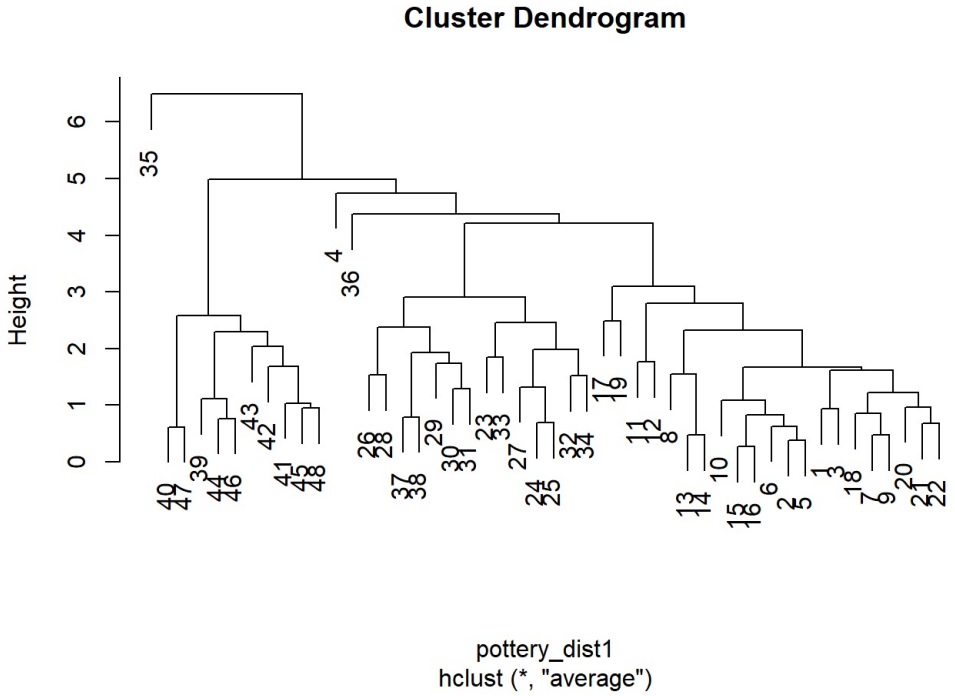
```
str(scale.pottery)
```

```
##  num [1:48, 1:9] 1.179 0.476 0.957 0.661 0.476 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:9] "Al2O3" "Fe2O3" "MgO" "CaO" ...
##  - attr(*, "scaled:center")= Named num [1:9] 15.615 5.826 2.543 0.511 0.245 ...
##   ..- attr(*, "names")= chr [1:9] "Al2O3" "Fe2O3" "MgO" "CaO" ...
##  - attr(*, "scaled:scale")= Named num [1:9] 2.702 2.346 1.726 0.45 0.174 ...
##   ..- attr(*, "names")= chr [1:9] "Al2O3" "Fe2O3" "MgO" "CaO" ...
```

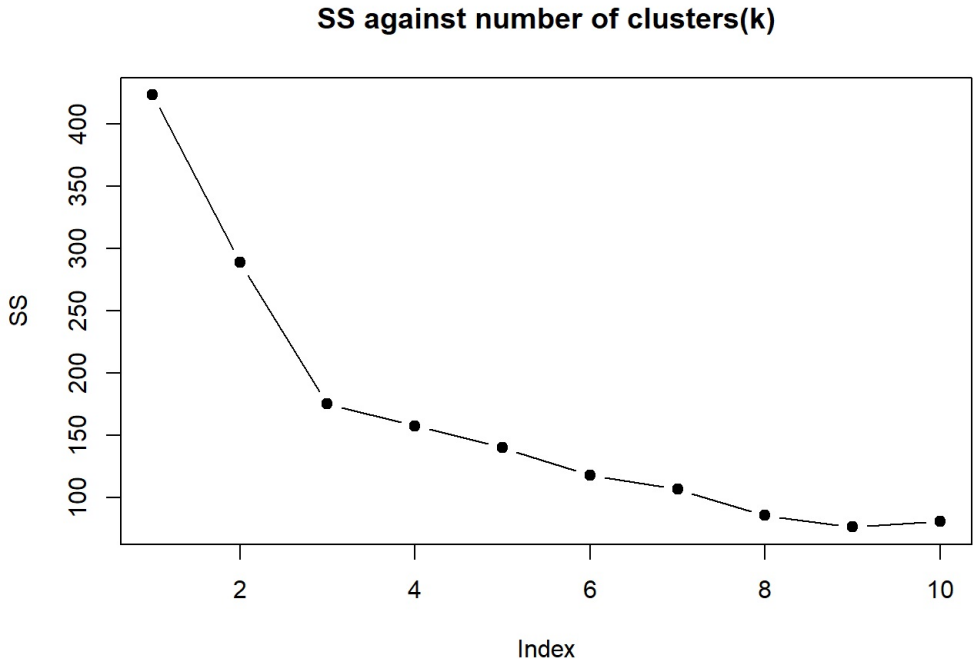
1.1.3 .c : Hierarchical Clustering and average linkage.

Now we will try to plot the Sum of square against the number of cluster, so we can find an elbow.

```
pottery_dist1 <- dist(scale.pottery,method = "euclidean")
cl.average <- hclust(pottery_dist1,method = "average")
plot(cl.average)
```




```
SS <- rep(0,10)

n <- nrow(scale.pottery)
SS[1] <- (n-1)*sum(apply(scale.pottery,2,var))

set.seed(13)
for(k in 2:10){
  SS[k] <- sum(kmeans(scale.pottery,centers =k)$withinss)
}

plot(SS, main = "SS against number of clusters(k)",pch =19, type ="b")
```




```
hierachical.cluster <- cutree(tree = cl.average,k=3)
hierachical.cluster
```

```
##  [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 2 1 1 1
## [39] 3 3 3 3 3 3 3 3 3 3
```

```
table(hierachical.cluster)
```

```
## hierachical.cluster
##  1  2  3
## 37  1 10
```

```
table(hierachical.cluster,pottery.complete$kiln)
```

```
##
## hierachical.cluster Ashley Rails Caldicot Gloucester Islands Thorns Llanedeyrn
##                   1      0     2        22        0      13
##                   2      0     0         0        0       1
##                   3      5     0         0        5       0
```

In the last table, when we do a comparison between the real results and the results that we have just found, we can clearly notice that there is 5 groups instead of 3, and the hierarchical algorithm isn't very accurate because it did not manage to cluster the whole group of values.

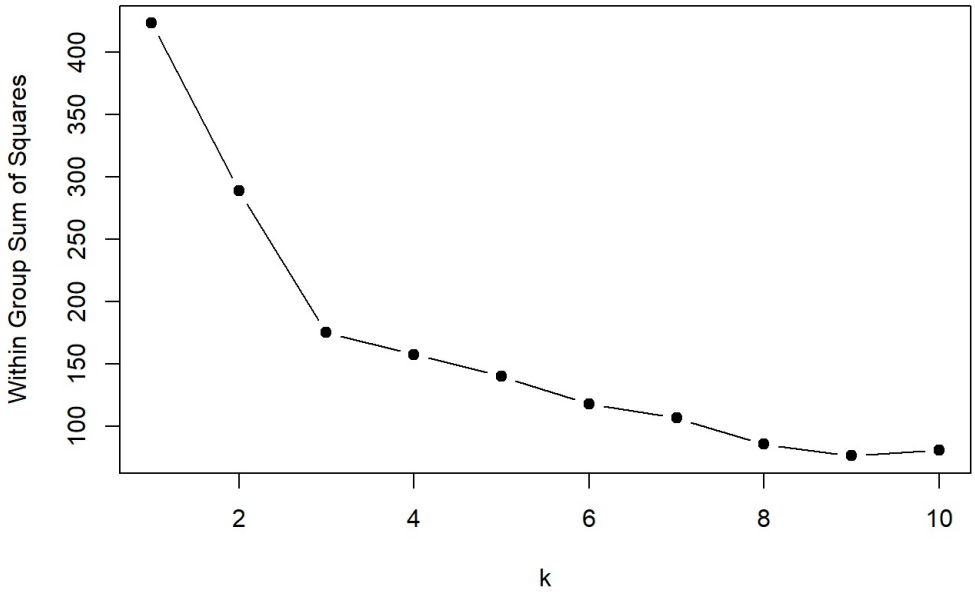
1.1.4 .d : K-Means Clustering

Let's move on now to the K-means Clustering :

```
library(ggplot2)
#install.packages("HSAUR2")
library(HSAUR2)

##########################################
# We're interested in how 'tightly packed' clusters are - this is the Within Group SS
# Let's find this for clustering solutions for k=1 up to k=10:
SS <- rep(0, 10)
SS
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

```r
# Finding the k = 1 solution separately, as kmeans() doesn't do this:
n <- nrow(scale.pottery)
SS[1] <- (n - 1) * sum(apply(scale.pottery, 2, var))

set.seed(13)
for(k in 2:10) {
  SS[k] <- sum(kmeans(scale.pottery, centers = k)$withinss)
}

# Let's plot k against the SS:
plot(1:10, SS,
     type = "b", xlab = "k",
     ylab = "Within Group Sum of Squares", pch = 19)
```



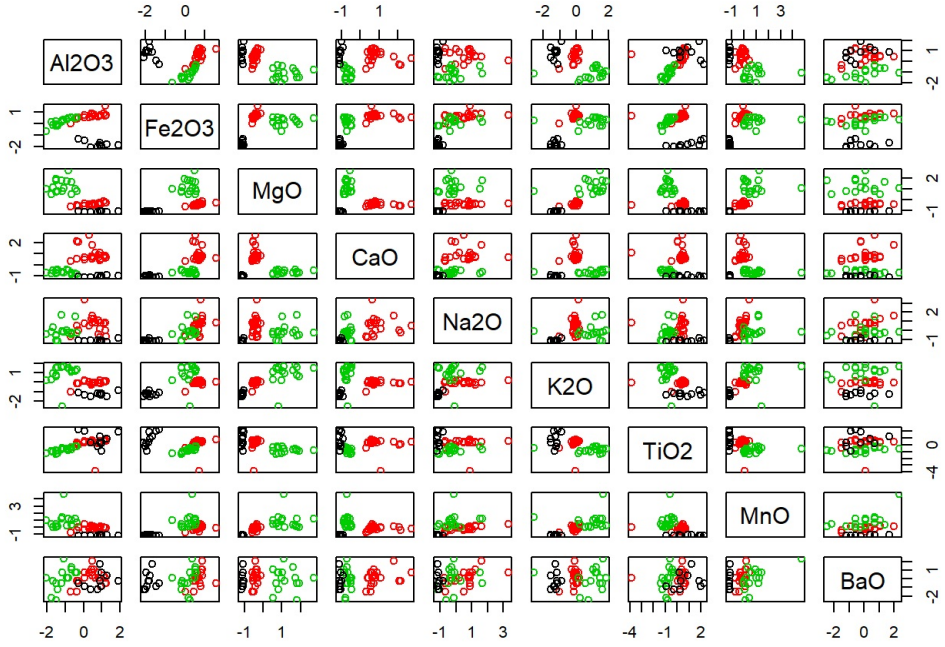

```r
# Seems to suggest 3 groups but we know they are 5 different locations
# Let's try with 3 and 5.
k <- 3
kcl1 <- kmeans(scale.pottery, center = k)
table(kcl1$cluster)
```

```
##
##  1  2  3
## 10 22 16
```

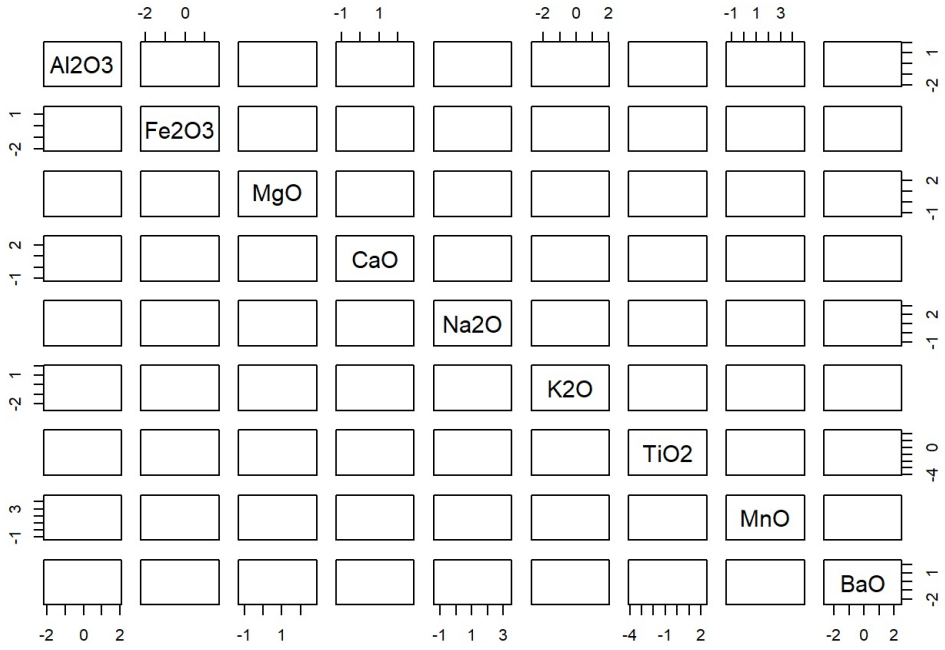
```r
table(kcl1$cluster, pottery.complete$kiln)
```

```
##
##     Ashley Rails Caldicot Gloucester Islands Thorns Llanedeyrn
##   1            5        0          0              5          0
##   2            0        0         22              0          0
##   3            0        2          0              0         14
```

```r
# Visualising these:
pairs(scale.pottery, col = kcl1$cluster)
```




```
pairs(scale.pottery, col = pottery$kiln)
```



At this time, 3 clusters would be an ideal solution for clustering the pottery data using K-means algorithm.

1.1.5 .e : Comparisons and agreement between the two solutions

Let's now compare between the cluster solutions that we have obtained in 1.c and 1.d using Rand index :

```
#install.packages("fossil")
library(fossil)
table(kcl1$cluster, pottery.complete$kiln)
```

```
##
##     Ashley Rails Caldicot Gloucester Islands Thorns Llanedeyrn
##   1             5        0           0               5          0
##   2             0        0          22               0          0
##   3             0        2           0               0         14
```

```
rand.index(kcl1$cluster, as.numeric(pottery.complete$kiln))
```

```
## [1] 0.9530142
```

```
adj.rand.index(kcl1$cluster, as.numeric(pottery.complete$kiln)) # We have 1 so it's perfect !
```

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

```
# Let's check the agreement between both solutions:
hcl <- cutree(hclust(dist(scale.pottery)), 3)
pcl <- kmeans(scale.pottery, centers = 3)
tab <- table(hcl, pcl$cluster)
tab
```

```
##
## hcl  1  2  3
##   1 22 15  0
##   2  0  1  0
##   3  0  0 10
```

We can see that the values are close so k = 3 seems to be a good solution.

1.1.6 .f : Conclusion

The 2 clustering solutions are close. But, in fact, the kmeans clustering seems a little more accurate even if we have approximately the same results. As for the hierarchical clustering we can clearly see that it did not cluster the whole data that we have. For the kmeans we can see that 5 would be a good solution too from the graph as we increase in the value, but i think that we do not have enough data for each category so some values got clustered into another category. This is why we chose 3 as the good solution.

1.2 Question 2: Logistic Regression

1.2.1 Introduction :

In this study, aim was to predict if a person has a heart disease or not based on attributes blood pressure,heart beat, exang, fbs and others. Our Dataset contains many medical indicators that we will explain here :

- age : age in year.
- sex : (1 = male; 0 = female)
- cp : the chest pain experienced(value 1: typical angina, value 2: atypical angina, value 3: non-anginal pain, value 4: asymptomatic)
- trestbps : resting blood pressure (in mm hg on admission to the hospital)
- chol : serum cholestoral in mg/dl
- fbs : (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- restecg: resting electrocardiographic measurement (0 = normal, 1 = having st-t wave abnormality, 2 = showing probable or definite left ventricular hypertrophy by estes' criteria)
- thalach: maximum heart rate achieved
- exang: exercise induced angina (1 = yes; 0 = no)
- oldpeak: the slope of the peak exercise st segment (value 1: upsloping, value 2: flat, value 3: downsloping)
- slope: the slope of the peak exercise st segment (value 1: upsloping, value 2: flat, value 3: downsloping)
- ca: number of major vessels (0–3) colored by flourosopy
- thal: a blood disorder called thalassemia (3 = normal; 6 = fixed defect; 7 = reversable defect)
- target: heart disease (0 = no, 1 = yes)

1.2.2 Data Exploration : Importing and exploring the Dataset

```
rm(list=ls())


# Importing the dataset :
heart = read.csv("heart-disease.csv",header = F)

# The head of the dataset :
head(heart)
```

```
##   V1 V2 V3  V4  V5 V6 V7  V8 V9 V10 V11 V12 V13 V14
## 1 63  1  1 145 233  1  2 150  0 2.3   3 0.0 6.0   0
## 2 67  1  4 160 286  0  2 108  1 1.5   2 3.0 3.0   2
## 3 67  1  4 120 229  0  2 129  1 2.6   2 2.0 7.0   1
## 4 37  1  3 130 250  0  0 187  0 3.5   3 0.0 3.0   0
## 5 41  0  2 130 204  0  2 172  0 1.4   1 0.0 3.0   0
## 6 56  1  2 120 236  0  0 178  0 0.8   1 0.0 3.0   0
```

```
tail(heart)
```

```
##     V1 V2 V3  V4  V5 V6 V7  V8 V9 V10 V11 V12 V13 V14
## 298 57  0  4 140 241  0  0 123  1 0.2   2 0.0 7.0   1
## 299 45  1  1 110 264  0  0 132  0 1.2   2 0.0 7.0   1
## 300 68  1  4 144 193  1  0 141  0 3.4   2 2.0 7.0   2
## 301 57  1  4 130 131  0  0 115  1 1.2   2 1.0 7.0   3
## 302 57  0  2 130 236  0  2 174  0 0.0   2 1.0 3.0   1
## 303 38  1  3 138 175  0  0 173  0 0.0   1   ? 3.0   0
```

```
# The number of rows of the dataset :
nrow(heart)
```

```
## [1] 303
```

```
# Preparing column names :
names <- c("age",
           "sex",
           "cp",
           "trestbps",
           "chol",
           "fbs",
           "restecg",
           "thalach",
           "exang",
           "oldpeak",
           "slope",
           "ca",
           "thal",
           "target")

#Apply column names to the dataframe :
colnames(heart) <- names

#Glimpse data to verify that new column names are in place :
colnames(heart)
```

```
##  [1] "age"      "sex"      "cp"       "trestbps" "chol"     "fbs"
##  [7] "restecg"  "thalach"  "exang"    "oldpeak"  "slope"    "ca"
## [13] "thal"     "target"
```

```
#Replacing values 1,2,3,4 by 1. So that, 0 is the absence of heart diseace and 1 is the presence of it.
for(i in 1:length(heart$target)){
  if(heart$target[i] >= 1){
    heart$target[i] = 1
  }
}

# Exploring now the structure and the values of our modified data :
str(heart)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : num  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp      : num  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : num  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg : num  2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : num  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : num  3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : Factor w/ 5 levels "?","0.0","1.0",..: 2 5 4 2 2 2 4 2 3 2 ...
##  $ thal    : Factor w/ 4 levels "?","3.0","6.0",..: 3 2 4 2 2 2 2 2 4 4 ...
##  $ target  : num  0 1 1 0 0 0 1 0 1 1 ...
```

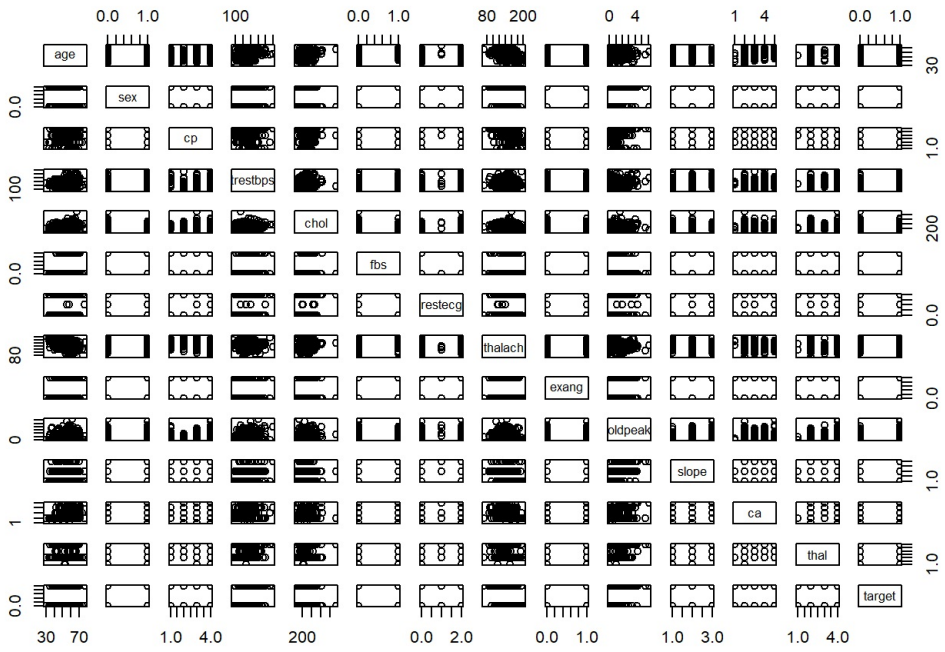
```
head(heart)
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope  ca thal
## 1  63   1  1      145  233   1       2     150     0     2.3     3 0.0  6.0
## 2  67   1  4      160  286   0       2     108     1     1.5     2 3.0  3.0
## 3  67   1  4      120  229   0       2     129     1     2.6     2 2.0  7.0
## 4  37   1  3      130  250   0       0     187     0     3.5     3 0.0  3.0
## 5  41   0  2      130  204   0       2     172     0     1.4     1 0.0  3.0
## 6  56   1  2      120  236   0       0     178     0     0.8     1 0.0  3.0
##   target
## 1      0
## 2      1
## 3      1
## 4      0
## 5      0
## 6      0
```

```
tail(heart)
```

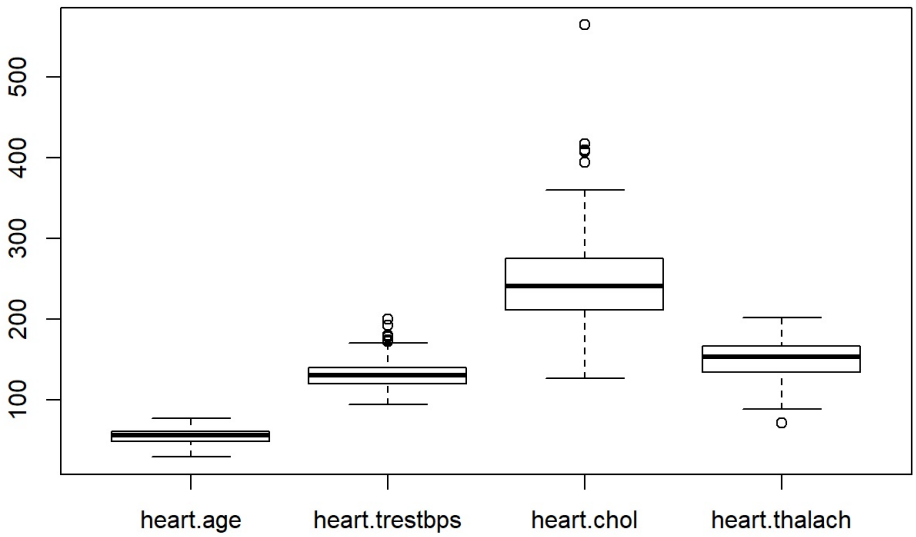
```
##     age sex cp trestbps chol fbs restecg thalach exang oldpeak slope  ca thal
## 298  57   0  4      140  241   0       0     123     1     0.2     2 0.0  7.0
## 299  45   1  1      110  264   0       0     132     0     1.2     2 0.0  7.0
## 300  68   1  4      144  193   1       0     141     0     3.4     2 2.0  7.0
## 301  57   1  4      130  131   0       0     115     1     1.2     2 1.0  7.0
## 302  57   0  2      130  236   0       2     174     0     0.0     2 1.0  3.0
## 303  38   1  3      138  175   0       0     173     0     0.0     1   ?  3.0
##     target
## 298      1
## 299      1
## 300      1
## 301      1
## 302      1
## 303      0
```

```
plot(heart)
```

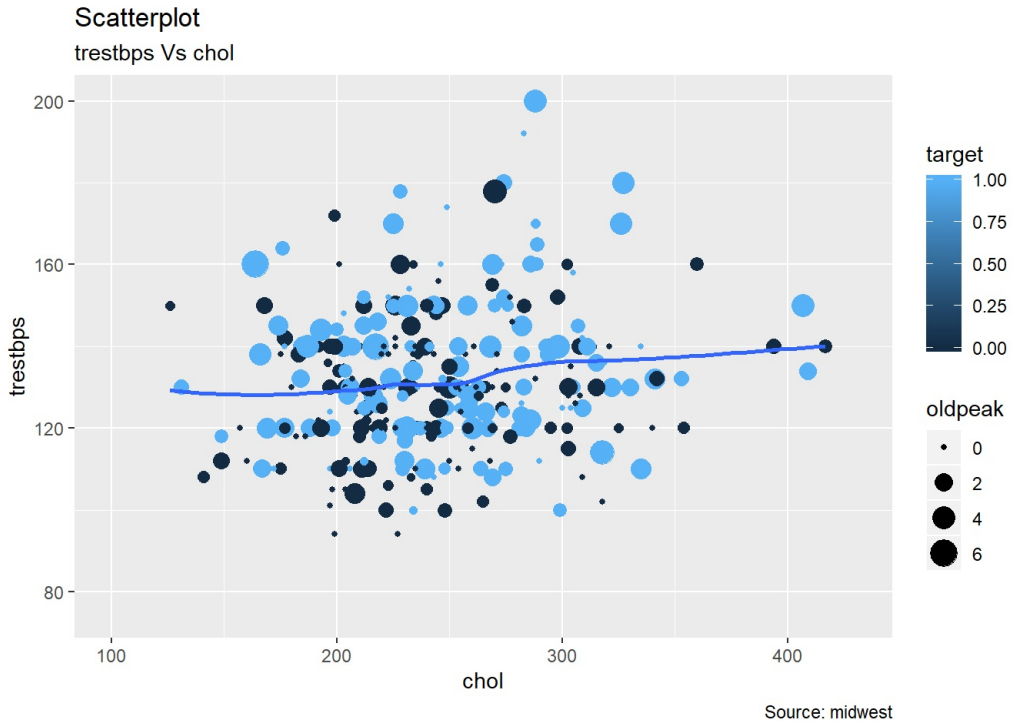


Let's explore the dataset by using different plots and functions :

```
# Boxplot of the variables that we suspect to be important :
databox=data.frame(heart$age, heart$trestbps,heart$chol,heart$thalach)
boxplot(databox)
```




```
# Scatterplot to compare the resting blood pressure vs serum cholestoral in mg/dl :
library(ggplot2)
gg <- ggplot(heart, aes(x=chol, y=trestbps)) +
  geom_point(aes(col=target, size=oldpeak)) +
  geom_smooth(method="loess", se=F) +
  xlim(c(100, 430)) +
  ylim(c(75, 200)) +
  labs(subtitle="trestbps Vs chol",
       y="trestbps",
       x="chol",
       title="Scatterplot",
       caption = "Source: midwest",
       bins = 30)
plot(gg)
```



1.2.3 Methodology :

1.2.3.1 Step 1 : Splitting the Dataset into training and testing set.

First, we will split our data into a training set and a testing set.

```
set.seed(13)
n <- length(heart$target)
index <- sample(1:n,floor(n*0.7))

# Splitting the data :
train <- heart[index,]
test <- heart[-index,]
```

1.2.3.2 Step 2 : Building our linear model

We will now build our linear model using interactions to have the best model possible.

```
# Linear model :
model.linear <- lm(target ~ .,data = train)

# The ^2 factor allows us to check for all the possible interactions between variables, which results in 163 par
ameters.
model.interaction <- lm(target ~(age+sex+cp+trestbps+chol+fbs+restecg+thalach+exang+oldpeak+slope+ca+thal)^2,dat
a = train)

summary(model.linear)
```

```
##
## Call:
## lm(formula = target ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.96964 -0.22942 -0.02863  0.15233  0.95379
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0586388  0.6158153   0.095  0.92424
## age          0.0001205  0.0034769   0.035  0.97239
## sex          0.1558496  0.0603651   2.582  0.01057 *
## cp           0.0886237  0.0307806   2.879  0.00444 **
## trestbps     0.0019268  0.0014789   1.303  0.19417
## chol         0.0000515  0.0005087   0.101  0.91947
## fbs         -0.0157118  0.0712061  -0.221  0.82560
## restecg      0.0085162  0.0262884   0.324  0.74633
## thalach     -0.0013128  0.0014599  -0.899  0.36962
## exang        0.1415769  0.0602288   2.351  0.01975 *
## oldpeak      0.0409300  0.0270933   1.511  0.13250
## slope        0.0835976  0.0519676   1.609  0.10933
## ca0.0        0.0384242  0.2086141   0.184  0.85406
## ca1.0        0.2924220  0.2123495   1.377  0.17008
## ca2.0        0.4125649  0.2216814   1.861  0.06425 .
## ca3.0        0.4372338  0.2300578   1.901  0.05885 .
## thal3.0     -0.5613145  0.3665651  -1.531  0.12734
## thal6.0     -0.4923669  0.3724973  -1.322  0.18780
## thal7.0     -0.3387630  0.3647219  -0.929  0.35414
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3503 on 193 degrees of freedom
## Multiple R-squared:  0.5462, Adjusted R-squared:  0.5039
## F-statistic: 12.91 on 18 and 193 DF,  p-value: < 2.2e-16
```

```
library(MASS)
# This function will allow us to have the lowest AIC possible with all the interactions that we've included in o
ur factored model.
model.linear.auto <- stepAIC(model.interaction,direction = "both",trace = F)

summary(model.linear.auto)
```

```
##
## Call:
## lm(formula = target ~ age + sex + cp + trestbps + chol + fbs +
##     restecg + thalach + exang + oldpeak + slope + ca + thal +
##     age:fbs + age:thalach + age:exang + age:slope + age:ca +
##     age:thal + sex:thalach + sex:exang + sex:oldpeak + sex:ca +
##     sex:thal + cp:chol + cp:ca + cp:thal + trestbps:fbs + trestbps:thalach +
##     trestbps:exang + trestbps:ca + trestbps:thal + chol:thalach +
##     chol:oldpeak + chol:slope + chol:ca + fbs:restecg + fbs:thalach +
##     fbs:exang + fbs:oldpeak + fbs:thal + restecg:exang + restecg:slope +
##     restecg:ca + restecg:thal + thalach:exang + thalach:slope +
##     thalach:ca + exang:oldpeak + exang:thal + oldpeak:ca + oldpeak:thal,
##     data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.57221 -0.12865 -0.00583  0.09848  0.78077
##
## Coefficients: (14 not defined because of singularities)
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.077e+02  3.257e+01   3.308 0.001219 **
## age              -1.749e+00  4.787e-01  -3.653 0.000377 ***
## sex               2.589e+00  7.569e-01   3.421 0.000839 ***
## cp               -1.052e+01  3.384e+00  -3.109 0.002311 **
## trestbps          1.785e-02  1.409e-02   1.267 0.207374
## chol              2.232e-02  7.360e-03   3.032 0.002937 **
## fbs              -1.056e-02  1.324e+00  -0.008 0.993650
## restecg          -3.149e-01  2.780e-01  -1.133 0.259513
## thalach           3.350e-02  1.937e-02   1.729 0.086142 .
## exang            -9.850e-01  8.249e-01  -1.194 0.234646
## oldpeak          -4.562e-01  2.249e-01  -2.029 0.044570 *
## slope            -7.068e-01  7.252e-01  -0.975 0.331574
## ca0.0            -1.080e+02  3.244e+01  -3.330 0.001136 **
## ca1.0            -1.082e+02  3.233e+01  -3.347 0.001071 **
```

```
## ca2.0            -1.036e+02  3.226e+01  -3.211 0.001670 **
## ca3.0            -1.223e+02  3.574e+01  -3.423 0.000832 ***
## thal3.0          -2.451e-01  7.486e-01  -0.327 0.743919
## thal6.0          -2.460e+00  1.594e+00  -1.544 0.125145
## thal7.0           2.195e-01  3.601e-01   0.610 0.543182
## age:fbs           3.466e-02  1.268e-02   2.733 0.007168 **
## age:thalach       2.202e-04  1.647e-04   1.337 0.183722
## age:exang        -1.142e-02  8.618e-03  -1.325 0.187549
## age:slope         2.037e-02  7.107e-03   2.866 0.004864 **
## age:ca0.0         1.671e+00  4.826e-01   3.463 0.000728 ***
## age:ca1.0         1.673e+00  4.823e-01   3.468 0.000715 ***
## age:ca2.0         1.665e+00  4.818e-01   3.457 0.000742 ***
## age:ca3.0         1.702e+00  4.876e-01   3.490 0.000663 ***
## age:thal3.0       2.865e-02  7.307e-03   3.921 0.000143 ***
## age:thal6.0      -6.197e-03  1.854e-02  -0.334 0.738687
## age:thal7.0              NA         NA      NA       NA
## sex:thalach      -8.116e-03  3.205e-03  -2.533 0.012532 *
## sex:exang        -6.363e-01  1.437e-01  -4.427 2.03e-05 ***
## sex:oldpeak       1.651e-01  7.757e-02   2.128 0.035242 *
## sex:ca0.0        -1.240e+00  5.913e-01  -2.097 0.037985 *
## sex:ca1.0        -1.046e+00  6.132e-01  -1.706 0.090444 .
## sex:ca2.0        -1.685e+00  6.616e-01  -2.547 0.012062 *
## sex:ca3.0                NA         NA      NA       NA
## sex:thal3.0       1.613e-01  1.901e-01   0.848 0.397814
## sex:thal6.0       1.751e+00  5.827e-01   3.004 0.003206 **
## sex:thal7.0              NA         NA      NA       NA
## cp:chol          -1.345e-03  7.094e-04  -1.895 0.060287 .
## cp:ca0.0          1.095e+01  3.391e+00   3.231 0.001569 **
## cp:ca1.0          1.119e+01  3.399e+00   3.291 0.001290 **
## cp:ca2.0          1.098e+01  3.407e+00   3.224 0.001606 **
## cp:ca3.0          1.199e+01  3.651e+00   3.285 0.001316 **
## cp:thal3.0       -1.434e-01  6.703e-02  -2.139 0.034306 *
## cp:thal6.0       -6.202e-02  1.379e-01  -0.450 0.653702
## cp:thal7.0               NA         NA      NA       NA
## trestbps:fbs     -7.589e-03  4.846e-03  -1.566 0.119769
## trestbps:thalach  1.813e-04  8.308e-05   2.182 0.030928 *
## trestbps:exang    1.119e-02  3.523e-03   3.176 0.001872 **
## trestbps:ca0.0   -4.104e-02  1.169e-02  -3.510 0.000619 ***
## trestbps:ca1.0   -4.469e-02  1.213e-02  -3.686 0.000335 ***
## trestbps:ca2.0   -5.038e-02  1.218e-02  -4.137 6.32e-05 ***
## trestbps:ca3.0           NA         NA      NA       NA
## trestbps:thal3.0 -7.741e-03  3.460e-03  -2.237 0.027001 *
## trestbps:thal6.0  5.233e-03  7.799e-03   0.671 0.503447
## trestbps:thal7.0         NA         NA      NA       NA
## chol:thalach     -1.373e-04  3.468e-05  -3.959 0.000124 ***
## chol:oldpeak      1.682e-03  7.270e-04   2.314 0.022256 *
## chol:slope       -2.685e-03  1.186e-03  -2.263 0.025293 *
## chol:ca0.0        6.974e-03  3.296e-03   2.116 0.036311 *
## chol:ca1.0        1.826e-03  3.223e-03   0.566 0.572082
## chol:ca2.0       -8.250e-04  3.664e-03  -0.225 0.822225
## chol:ca3.0               NA         NA      NA       NA
## fbs:restecg      -3.803e-01  1.164e-01  -3.267 0.001397 **
## fbs:thalach      -7.412e-03  5.773e-03  -1.284 0.201498
## fbs:exang         3.949e-01  1.927e-01   2.049 0.042489 *
## fbs:oldpeak       2.277e-01  9.308e-02   2.446 0.015809 *
## fbs:thal3.0       4.441e-01  2.048e-01   2.168 0.032020 *
## fbs:thal6.0       3.621e-01  2.919e-01   1.240 0.217081
## fbs:thal7.0              NA         NA      NA       NA
## restecg:exang    -7.159e-02  6.064e-02  -1.181 0.239990
## restecg:slope    -2.287e-01  5.033e-02  -4.545 1.26e-05 ***
## restecg:ca0.0     5.682e-01  2.712e-01   2.095 0.038123 *
## restecg:ca1.0     7.848e-01  2.725e-01   2.880 0.004662 **
## restecg:ca2.0     1.072e+00  2.914e-01   3.677 0.000346 ***
## restecg:ca3.0            NA         NA      NA       NA
## restecg:thal3.0   4.936e-02  6.055e-02   0.815 0.416438
## restecg:thal6.0   2.715e-01  1.300e-01   2.089 0.038733 *
## restecg:thal7.0          NA         NA      NA       NA
## thalach:exang     4.860e-03  3.289e-03   1.478 0.141985
## thalach:slope     3.981e-03  2.985e-03   1.334 0.184650
## thalach:ca0.0    -3.734e-02  1.130e-02  -3.306 0.001230 **
## thalach:ca1.0    -3.034e-02  1.188e-02  -2.554 0.011830 *
## thalach:ca2.0    -3.949e-02  1.164e-02  -3.394 0.000918 ***
## thalach:ca3.0            NA         NA      NA       NA
## exang:oldpeak     1.230e-01  5.794e-02   2.123 0.035679 *
## exang:thal3.0    -2.854e-01  1.432e-01  -1.994 0.048322 *
## exang:thal6.0     6.345e-01  3.082e-01   2.059 0.041549 *
## exang:thal7.0            NA         NA      NA       NA
## oldpeak:ca0.0    -1.409e-01  1.147e-01  -1.229 0.221359
## oldpeak:ca1.0    -1.134e-01  1.227e-01  -0.924 0.357111
```

```
## oldpeak:ca2.0    -3.524e-01  1.463e-01  -2.409 0.017404 *
## oldpeak:ca3.0            NA         NA      NA       NA
## oldpeak:thal3.0   1.393e-01  6.045e-02   2.304 0.022849 *
## oldpeak:thal6.0   1.441e-02  1.073e-01   0.134 0.893407
## oldpeak:thal7.0          NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2833 on 128 degrees of freedom
## Multiple R-squared:  0.8032, Adjusted R-squared:  0.6756
## F-statistic: 6.295 on 83 and 128 DF,  p-value: < 2.2e-16
```

1.2.3.3 Step 3 : Testing our model.

Now, we will test our model to see if it is efficient enough to predict our target.

```
# Removing the target column.
targetpredict.test <- test[-c(14)]
# Checking out if it has been removed :
targetpredict.test
```

```
##     age sex cp trestbps chol fbs restecg thalach exang oldpeak slope  ca thal
## 4    37   1  3      130  250   0       0     187     0     3.5     3 0.0  3.0
## 10   53   1  4      140  203   1       2     155     1     3.1     3 0.0  7.0
## 24   58   1  3      132  224   0       2     173     0     3.2     1 2.0  7.0
## 26   50   0  3      120  219   0       0     158     0     1.6     2 0.0  3.0
## 27   58   0  3      120  340   0       0     172     0     0.0     1 0.0  3.0
## 28   66   0  1      150  226   0       0     114     0     2.6     3 0.0  3.0
## 30   40   1  4      110  167   0       2     114     1     2.0     2 0.0  7.0
## 34   59   1  4      135  234   0       0     161     0     0.5     2 0.0  7.0
## 37   43   1  4      120  177   0       2     120     1     2.5     2 0.0  7.0
## 38   57   1  4      150  276   0       2     112     1     0.6     2 1.0  6.0
## 40   61   1  3      150  243   1       0     137     1     1.0     2 0.0  3.0
## 41   65   0  4      150  225   0       2     114     0     1.0     2 3.0  7.0
## 42   40   1  1      140  199   0       0     178     1     1.4     1 0.0  7.0
## 44   59   1  3      150  212   1       0     157     0     1.6     1 0.0  3.0
## 45   61   0  4      130  330   0       2     169     0     0.0     1 0.0  3.0
## 48   50   1  4      150  243   0       2     128     0     2.6     2 0.0  7.0
## 51   41   0  2      105  198   0       0     168     0     0.0     1 1.0  3.0
## 52   65   1  4      120  177   0       0     140     0     0.4     1 0.0  7.0
## 56   54   1  4      124  266   0       2     109     1     2.2     2 1.0  7.0
## 69   59   1  4      170  326   0       2     140     1     3.4     3 0.0  7.0
## 71   65   0  3      155  269   0       0     148     0     0.8     1 0.0  3.0
## 75   44   1  4      110  197   0       2     177     0     0.0     1 1.0  3.0
## 76   65   0  3      160  360   0       2     151     0     0.8     1 0.0  3.0
## 80   58   1  4      150  270   0       2     111     1     0.8     1 0.0  7.0
## 85   52   1  2      120  325   0       0     172     0     0.2     1 0.0  3.0
## 88   53   0  3      128  216   0       2     115     0     0.0     1 0.0    ?
## 89   53   0  4      138  234   0       2     160     0     0.0     1 0.0  3.0
## 93   62   1  3      130  231   0       0     146     0     1.8     2 3.0  7.0
## 95   63   0  3      135  252   0       2     172     0     0.0     1 0.0  3.0
## 96   52   1  4      128  255   0       0     161     1     0.0     1 1.0  7.0
## 98   60   0  4      150  258   0       2     157     0     2.6     2 2.0  7.0
## 102  34   1  1      118  182   0       2     174     0     0.0     1 0.0  3.0
## 104  71   0  3      110  265   1       2     130     0     0.0     1 1.0  3.0
## 106  54   1  2      108  309   0       0     156     0     0.0     1 0.0  7.0
## 110  39   1  4      118  219   0       0     140     0     1.2     2 0.0  7.0
## 115  62   0  3      130  263   0       0      97     0     1.2     2 1.0  7.0
## 117  58   1  3      140  211   1       2     165     0     0.0     1 0.0  3.0
## 118  35   0  4      138  183   0       0     182     0     1.4     1 0.0  3.0
## 120  65   1  4      135  254   0       2     127     0     2.8     2 1.0  7.0
## 122  63   0  4      150  407   0       2     154     0     4.0     2 3.0  7.0
## 133  29   1  2      130  204   0       2     202     0     0.0     1 0.0  3.0
## 138  62   1  2      120  281   0       2     103     0     1.4     2 1.0  7.0
## 142  59   1  1      170  288   0       2     159     0     0.2     2 0.0  7.0
## 144  64   1  3      125  309   0       0     131     1     1.8     2 0.0  7.0
## 145  58   1  3      105  240   0       2     154     1     0.6     2 0.0  7.0
## 152  42   0  4      102  265   0       2     122     0     0.6     2 0.0  3.0
## 155  64   1  4      120  246   0       2      96     1     2.2     3 1.0  3.0
## 159  60   1  4      140  293   0       2     170     0     1.2     2 2.0  7.0
## 166  57   1  4      132  207   0       0     168     1     0.0     1 0.0  7.0
## 169  35   1  4      126  282   0       2     156     1     0.0     1 0.0  7.0
## 176  57   1  4      152  274   0       0      88     1     1.2     2 1.0  7.0
## 177  52   1  4      108  233   1       0     147     0     0.1     1 3.0  7.0
## 181  48   1  4      124  274   0       2     166     0     0.5     2 0.0  7.0
## 183  42   1  1      148  244   0       2     178     0     0.8     1 2.0  3.0
## 185  60   0  4      158  305   0       2     161     0     0.0     1 0.0  3.0
## 186  63   0  2      140  195   0       0     179     0     0.0     1 2.0  3.0
```

```
## 192  51   1  4      140  298   0       0     122     1     4.2     2 3.0  7.0
## 195  68   0  3      120  211   0       2     115     0     1.5     2 0.0  3.0
## 201  50   0  4      110  254   0       2     159     0     0.0     1 0.0  3.0
## 203  57   1  3      150  126   1       0     173     0     0.2     1 1.0  7.0
## 207  58   1  4      128  259   0       2     130     1     3.0     2 2.0  7.0
## 208  50   1  4      144  200   0       2     126     1     0.9     2 0.0  7.0
## 214  66   0  4      178  228   1       0     165     1     1.0     2 2.0  7.0
## 216  56   1  1      120  193   0       2     162     0     1.9     2 0.0  7.0
## 217  46   0  2      105  204   0       0     172     0     0.0     1 0.0  3.0
## 220  59   1  4      138  271   0       2     182     0     0.0     1 0.0  3.0
## 226  34   0  2      118  210   0       0     192     0     0.7     1 0.0  3.0
## 229  54   1  4      110  206   0       2     108     1     0.0     2 1.0  3.0
## 230  66   1  4      112  212   0       2     132     1     0.1     1 1.0  3.0
## 233  49   1  3      118  149   0       2     126     0     0.8     1 3.0  3.0
## 234  74   0  2      120  269   0       2     121     1     0.2     1 1.0  3.0
## 237  56   1  4      130  283   1       2     103     1     1.6     3 0.0  7.0
## 242  41   0  2      126  306   0       0     163     0     0.0     1 0.0  3.0
## 244  61   1  1      134  234   0       0     145     0     2.6     2 2.0  3.0
## 245  60   0  3      120  178   1       0      96     0     0.0     1 0.0  3.0
## 247  58   1  4      100  234   0       0     156     0     0.1     1 1.0  7.0
## 254  51   0  3      120  295   0       2     157     0     0.6     1 0.0  3.0
## 257  67   0  4      106  223   0       0     142     0     0.3     1 2.0  3.0
## 263  60   0  1      150  240   0       0     171     0     0.9     1 0.0  3.0
## 266  42   1  4      136  315   0       0     125     1     1.8     2 0.0  6.0
## 271  61   1  4      140  207   0       2     138     1     1.9     1 1.0  7.0
## 273  46   1  4      140  311   0       0     120     1     1.8     2 2.0  7.0
## 275  59   1  1      134  204   0       0     162     0     0.8     1 2.0  3.0
## 281  57   1  4      110  335   0       0     143     1     3.0     2 1.0  7.0
## 284  35   1  2      122  192   0       0     174     0     0.0     1 0.0  3.0
## 285  61   1  4      148  203   0       0     161     0     0.0     1 1.0  7.0
## 290  56   1  2      120  240   0       0     169     0     0.0     3 0.0  3.0
## 298  57   0  4      140  241   0       0     123     1     0.2     2 0.0  7.0
## 299  45   1  1      110  264   0       0     132     0     1.2     2 0.0  7.0
## 300  68   1  4      144  193   1       0     141     0     3.4     2 2.0  7.0
## 303  38   1  3      138  175   0       0     173     0     0.0     1   ?  3.0
```

```
predicted.value <- predict(model.interaction,newdata = test)
table(round(predicted.value))
```

```
##
## -1  0  1  2  3 25
##  1 48 32  8  1  1
```

```
predicted.value
```

```
##            4           10           24           26           27           28
##  0.222433967  0.644780312  1.043507465  0.052867674 -0.194588364  0.127938168
##           30           34           37           38           40           41
##  0.047136771  0.837726893  0.186266562  0.786288489  1.933450431  2.070012022
##           42           44           45           48           51           52
##  0.294820916  0.814696966 -0.298065761  0.472737602  0.171762307 -0.135196008
##           56           69           71           75           76           80
##  0.865261095  1.744869105 -0.133464132  1.665103332 -0.004254227  0.500874676
##           85           88           89           93           95           96
##  0.373686195  0.235345883 -0.044197352  1.429092312  0.033200770  0.784985327
##           98          102          104          106          110          115
##  1.376159590 -0.127314197  1.299912834  0.510278776  0.764749735 -0.448523789
##          117          118          120          122          133          138
## -0.281734813 -0.347219289  0.953619289 -0.307370371 -0.155405724  0.031055074
##          142          144          145          152          155          159
##  0.112431136  1.395786213 -0.153996191 -0.036436944  0.219530894  0.934247872
##          166          169          176          177          181          183
##  0.177198145  0.922573427  0.660119789  2.442129736  0.225878066  0.436495711
##          185          186          192          195          201          203
## -0.304693297  1.066648554  1.508375905 -0.164683368  0.054184457  1.507749673
##          207          208          214          216          217          220
##  1.118738315  0.269909227  3.160538415 -0.014806686 -0.062876168 -0.008640224
##          226          229          230          233          234          237
## -0.528490113  0.009253044  0.148843501  1.132429777 -0.197257758 -0.012117250
##          242          244          245          247          254          257
## -0.304876999  0.102897085  0.772604622  0.650046318  0.106512281  1.135827268
##          263          266          271          273          275          281
## -0.163932549  2.008400899  1.188706619  0.915418919  0.125961845  1.019266084
##          284          285          290          298          299          300
## -0.186494647  1.352563189  1.142120740  0.958067983  0.679148604  1.155079533
##          303
## 25.341662376
```

```
# Readjusting the values so that if they are below 0 then we replace them by 0 and if they are above 1 we replac
e them by 1.

for(i in 1:length(predicted.value)){
  if(predicted.value[i]<0){
    predicted.value[i] = 0
  }
  if(predicted.value[i]>1){
    predicted.value[i] = 1
  }
}

test$pred_target = round(predicted.value)
table(round(predicted.value),test$target)
```

```
##
##      0  1
##   0 32 17
##   1 13 29
```

1.2.4 Conclusion :

We can see that our model is working good enough to predict our target. As our model grew up from 54% of R-squared to 80% so our model can expain 80% of our dataset which is a really good value but not sufficient for our purposes given that we are in the domain of health care where false classifications have dire consequences. Evaluating other algorithms would be a logical next step for improving the accuracy and reducing patient risk with the data that we have been given to study.

1.3 Question 3: Principal Components Analysis (PCA).

1.3.1 Introduction :

We have been given a dataset representing the quality of life in different Cities. The data includes ratings for 9 different indicators of the quality of life in 329 cities. These are climate, housing, health, crime, transportation, education, arts, recreation, and economics. For each category, a higher rating is better. We will then try to perform a weighted principal components analysis to be able to interpret the results.

1.3.2 Data Exploration :

```
rm(list=ls())
ratings.complete = read.csv(file = "ratings.txt", header = F, sep = ' ')

# Preparing column names :
names2 <- c("Climate & Terrain","Housing","HealthCare","Crime","Transportation","Education","Art","Recreation","
Economics","Index")

# Apply column names to the dataframe :
colnames(ratings.complete) <- names2

# Checking out the new names for our columns :
colnames(ratings.complete)
```

```
##  [1] "Climate & Terrain" "Housing"           "HealthCare"
##  [4] "Crime"             "Transportation"    "Education"
##  [7] "Art"               "Recreation"        "Economics"
## [10] "Index"
```

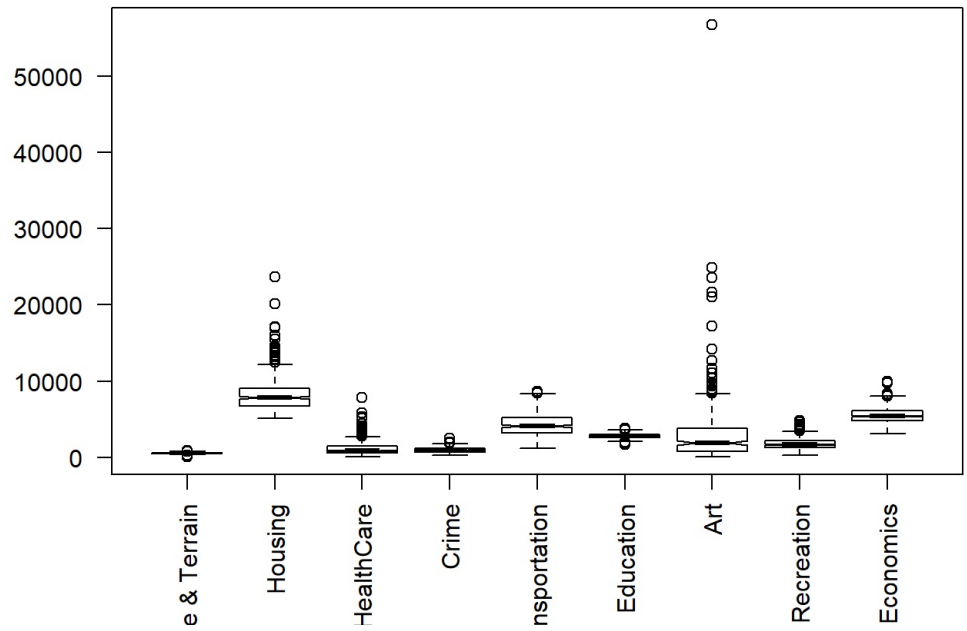
```
# Removing the last column because it's only the index :
ratings = ratings.complete[-c(10)]

# Checking out the structure and the values of our modified dataset :
str(ratings)
```

```
## 'data.frame':    329 obs. of  9 variables:
##  $ Climate & Terrain: int  521 575 468 476 659 520 559 537 561 609 ...
##  $ Housing          : int  6200 8138 7339 7908 8393 5819 8288 6487 6191 6546 ...
##  $ HealthCare       : int  237 1656 618 1431 1853 640 621 965 432 669 ...
##  $ Crime            : int  923 886 970 610 1483 727 514 706 399 1073 ...
##  $ Transportation   : int  4031 4883 2531 6883 6558 2444 2881 4975 4246 4902 ...
##  $ Education        : int  2757 2438 2560 3399 3026 2972 3144 2945 2778 2852 ...
##  $ Art              : int  996 5564 237 4655 4496 334 2333 1487 256 1235 ...
##  $ Recreation       : int  1405 2632 859 1617 2612 1018 1117 1280 1210 1109 ...
##  $ Economics        : int  7633 4350 5250 5864 5727 5254 5097 5795 4230 6241 ...
```

```
head(ratings)
```

```
##   Climate & Terrain Housing HealthCare Crime Transportation Education  Art
## 1               521    6200        237   923           4031      2757  996
## 2               575    8138       1656   886           4883      2438 5564
## 3               468    7339        618   970           2531      2560  237
## 4               476    7908       1431   610           6883      3399 4655
## 5               659    8393       1853  1483           6558      3026 4496
## 6               520    5819        640   727           2444      2972  334
##   Recreation Economics
## 1       1405      7633
## 2       2632      4350
## 3        859      5250
## 4       1617      5864
## 5       2612      5727
## 6       1018      5254
```

```
boxplot(ratings,varwidth = T,notch = T ,outline = T, las = 2)
```



We can notice that there is more variability in the ratings of the arts and housing than in the ratings of crime and climate.

1.3.3 Methodology :

We need now to change the order of some parameters in order to have them all in this following shape : The higher the better the Housing and Crime need to be inverted so that the higher value will be the best.

```
ratings$Housing <- -ratings$Housing
ratings$Crime <- -ratings$Crime
```

We will now scale our data as we did before to remove eventual variations.

```
scale.ratings = scale(ratings)
head(scale.ratings)
```

```
##      Climate & Terrain     Housing HealthCare       Crime Transportation
## [1,]        -0.1467824  0.89992576 -0.9458990  0.10654981     -0.1234045
## [2,]         0.3002069  0.08743661  0.4688539  0.21014653      0.4637042
## [3,]        -0.5854941  0.42241020 -0.5660393 -0.02504601     -1.1570466
## [4,]        -0.5192735  0.18386205  0.2445273  0.98292201      1.8418937
## [5,]         0.9955236 -0.01946986  0.6652643 -1.46140045      1.6179379
## [6,]        -0.1550600  1.05965660 -0.5441052  0.65533240     -1.2169979
##       Education        Art Recreation  Economics
## [1,] -0.1804514 -0.4641863 -0.5458150  1.9434730
## [2,] -1.1748623  0.5198122  0.9729596 -1.0838164
## [3,] -0.7945547 -0.6276834 -1.2216511 -0.2539168
## [4,]  1.8208394  0.3240034 -0.2834024  0.3122592
## [5,]  0.6580957  0.2897530  0.9482037  0.1859300
## [6,]  0.4897628 -0.6067885 -1.0248417 -0.2502283
```

Now that our variables are set up correctly, we can move on to the PCA.

```
fit <- prcomp(scale.ratings)
fit
```

```
## Standard deviations (1, .., p=9):
## [1] 1.8461560 1.1018059 1.0684003 0.9596446 0.8679199 0.7940793 0.7021736
## [8] 0.5639490 0.3469900
##
## Rotation (n x k) = (9 x 9):
##                           PC1        PC2          PC3         PC4        PC5
## Climate & Terrain  0.2064140  0.2178353 -0.689955982  0.13732125 -0.3691499
## Housing           -0.3565216 -0.2506240  0.208172230 -0.51182871 -0.2334878
## HealthCare         0.4602146 -0.2994653 -0.007324926  0.01470183 -0.1032405
## Crime             -0.2812984 -0.3553423 -0.185104981  0.53905047  0.5239397
## Transportation     0.3511508 -0.1796045  0.146376283 -0.30290371  0.4043485
## Education          0.2752926 -0.4833821  0.229702548  0.33541103 -0.2088191
## Art                0.4630545 -0.1947899 -0.026484298 -0.10108039 -0.1050976
## Recreation         0.3278879  0.3844746 -0.050852640 -0.18980082  0.5295406
## Economics          0.1354123  0.4712833  0.607314475  0.42176994 -0.1596201
##                           PC6         PC7         PC8           PC9
## Climate & Terrain  0.37460469 -0.08470577 -0.36230833  0.0013913515
## Housing            0.14163983  0.23063862 -0.61385513 -0.0136003402
## HealthCare        -0.37384804  0.01386761 -0.18567612 -0.7163548935
## Crime             -0.08092329 -0.01860646 -0.43002477  0.0586084614
## Transportation     0.46759180 -0.58339097 -0.09359866  0.0036294527
## Education          0.50216981  0.42618186  0.18866756  0.1108401911
## Art               -0.46188072 -0.02152515 -0.20398969  0.6857582127
## Recreation         0.08991578  0.62787789 -0.15059597 -0.0255062915
## Economics          0.03260813 -0.14974066 -0.40480926  0.0004377942
```

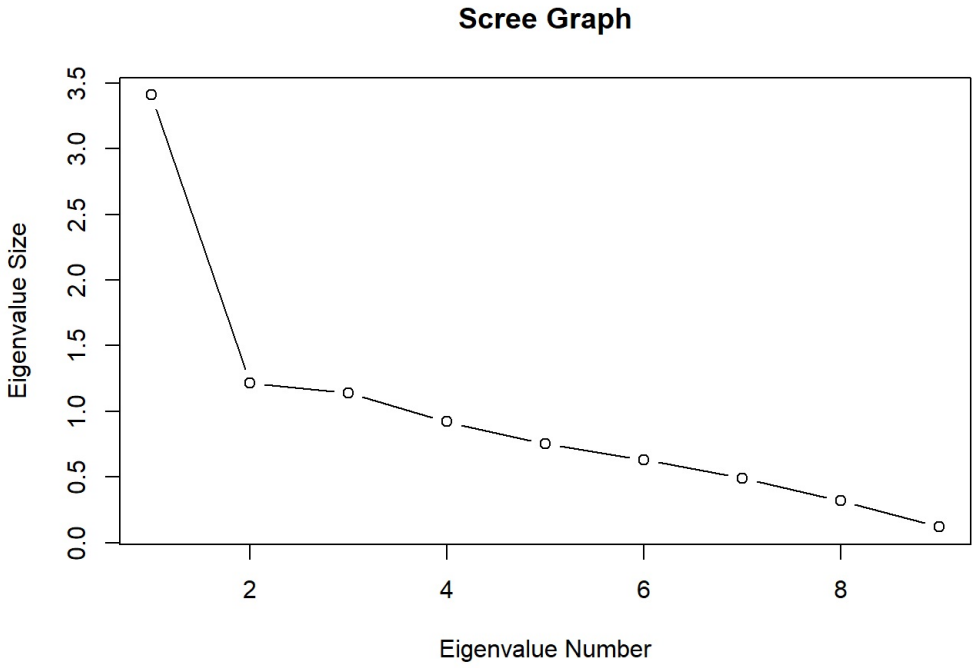
```
s1 <- summary(fit)
R <- cor(ratings[,])
# Computing eigenvalues for our numeric matrice R :
r.eigen <- eigen(R)

fit
```


```
## Standard deviations (1, .., p=9):
## [1] 1.8461560 1.1018059 1.0684003 0.9596446 0.8679199 0.7940793 0.7021736
## [8] 0.5639490 0.3469900
##
## Rotation (n x k) = (9 x 9):
##                           PC1        PC2          PC3         PC4        PC5
## Climate & Terrain  0.2064140  0.2178353 -0.689955982  0.13732125 -0.3691499
## Housing           -0.3565216 -0.2506240  0.208172230 -0.51182871 -0.2334878
## HealthCare         0.4602146 -0.2994653 -0.007324926  0.01470183 -0.1032405
## Crime             -0.2812984 -0.3553423 -0.185104981  0.53905047  0.5239397
## Transportation     0.3511508 -0.1796045  0.146376283 -0.30290371  0.4043485
## Education          0.2752926 -0.4833821  0.229702548  0.33541103 -0.2088191
## Art                0.4630545 -0.1947899 -0.026484298 -0.10108039 -0.1050976
## Recreation         0.3278879  0.3844746 -0.050852640 -0.18980082  0.5295406
## Economics          0.1354123  0.4712833  0.607314475  0.42176994 -0.1596201
##                           PC6         PC7         PC8           PC9
## Climate & Terrain  0.37460469 -0.08470577 -0.36230833  0.0013913515
## Housing            0.14163983  0.23063862 -0.61385513 -0.0136003402
## HealthCare        -0.37384804  0.01386761 -0.18567612 -0.7163548935
## Crime             -0.08092329 -0.01860646 -0.43002477  0.0586084614
## Transportation     0.46759180 -0.58339097 -0.09359866  0.0036294527
## Education          0.50216981  0.42618186  0.18866756  0.1108401911
## Art               -0.46188072 -0.02152515 -0.20398969  0.6857582127
## Recreation         0.08991578  0.62787789 -0.15059597 -0.0255062915
## Economics          0.03260813 -0.14974066 -0.40480926  0.0004377942
```


As we can see through the PC1 value, the ratings will be more impacted by the Art parameter, then the Health Care one and finally the Climate & Terrain one. We can say that we can have a better community with a higher score if we have cheaper housing but higher Crime rate.

In PC2, the economics and Recreation parameters influence a lot more the ratings but the Climate & Terrain one has a bigger penalty.

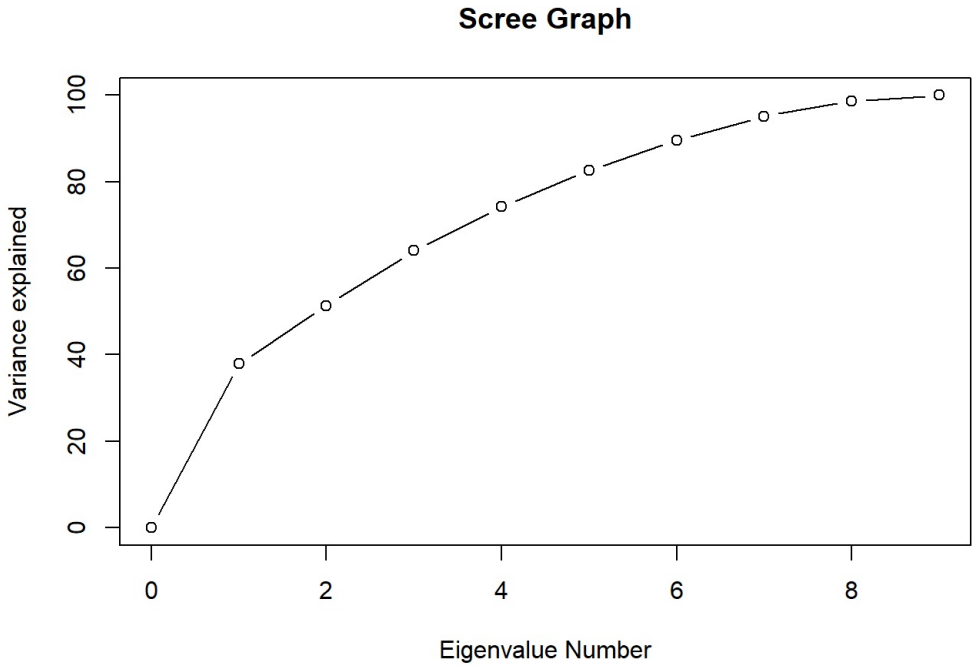
```
plot(r.eigen$values, xlab = "Eigenvalue Number",
                     ylab = "Eigenvalue Size",
                     main = "Scree Graph",type = "b")
```




```
pred.ratings = predict(fit)

cumsum(s1$importance[1,]^2) / sum(s1$importance[1,]^2)
```

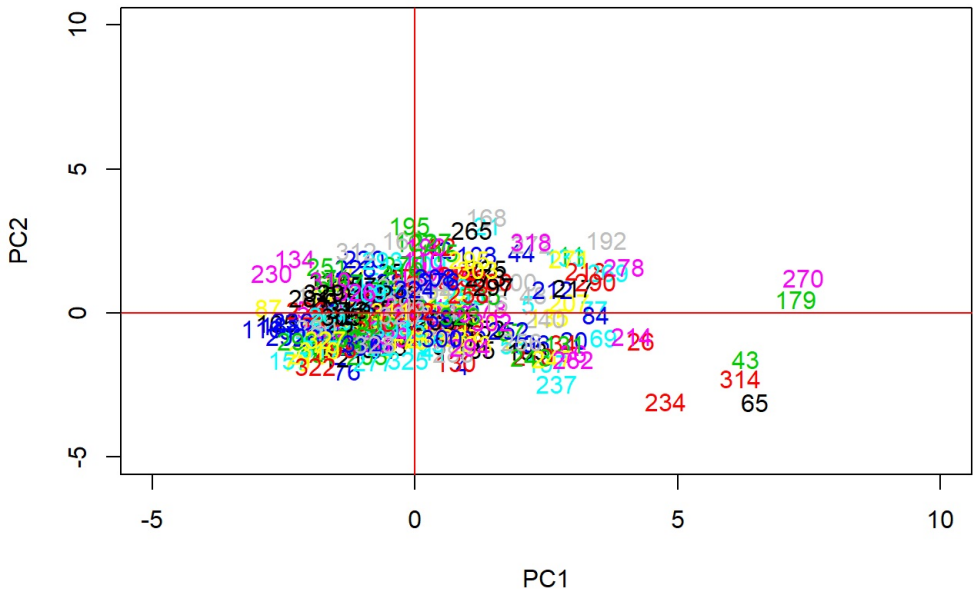
```
##       PC1       PC2       PC3       PC4       PC5       PC6       PC7       PC8
## 0.3786991 0.5135853 0.6404163 0.7427405 0.8264389 0.8965013 0.9512844 0.9866220
##       PC9
## 1.0000000
```

```
vals <- cumsum(s1$importance[1,]^2) / sum(s1$importance[1,]^2)

plot(c(0, 1:length(vals)), c(0, 100* vals),
     ylim = c(0, 100),
     xlab = 'Eigenvalue Number',
     ylab = 'Variance explained',
     main = 'Scree Graph', type = "b")
```



We can clearly see from the graph, that the elbow is formed in PC2, and the cumulative proportion of the variance is 51%, which is not good. But, the second scree plot show that the only clear break in the amount of variance accounted for by each component is between the first and second components. However, the first component by itself explains less than 40% of the variance, so more components might be needed. We can see that the first three principal components explain roughly two-thirds of the toal variability in the standardized ratings, so that might be a reasonable way to reduce the dimensions.

```
{plot(pred.ratings[,1],pred.ratings[,2],type ="n",xlab ="PC1",ylab ="PC2",xlim = range(-5,10),ylim=range(-5,10))
  text(pred.ratings[,1],pred.ratings[,2],
       labels = ratings.complete$Index,col = ratings.complete$Index)
  abline(h=0,col = "red")
  abline(v=0,col ="red")}
```



We can see that there's a huge concentration of points near the origin (0,0) coordinates because we have scaled the data. We also have some outliers, let's take a look at them.

```
points <- data.frame(pred.ratings[,1],pred.ratings[,2])
colnames(points)[1] <- "x"
colnames(points)[2] <- "y"

# Assigning a value(0,1,2) based on the number of negative signs in the coordinate. So that we can classify them easily
sign <- rep(0,length(points$x))
sign
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [149] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [186] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [223] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [260] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [297] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
  for(i in 1:length(points$x)){
    if(points$x[i] > 0 & points$y[i] > 0){
      sign[i] = 2
    }
    else if(points$x[i] < 0 & points$y[i] < 0){
      sign[i] = 0
    }
    else{
      sign[i] = 1
    }
  }

points$sign <- sign


# Calculating the distance from the origin (0,0) and the point.
distance.origin <- sqrt((points$x)^2 + (points$y)^2)
points$distance.origin <- distance.origin

# Getting the index from the ratings.complete.
points$index <- ratings.complete$Index
points <- points[order(points[,3],points[,4],decreasing = T),]
```

```
points
points$rank <- seq(1,length(points$x))
```

```
ranking <- data.frame(points$rank,points$index)
ranking
```

1.3.4 Conclusion :

We can conclude from our analysis that from the plot component graph we have some outliers points that appears to be more extreme than the remainder of the data. And from the Scree plot we can see that the first three principal components explain a big amount of our standardized dataset.