



MSc. in Artificial Intelligence

Fundamentals and Background on Artificial Intelligence

Main Exercise Report

Delikonstantis Dimitrios

mtn 2007

Bochalis Christodoulos

mtn 2019

March 4, 2020

Contents

1	Abstract	3
1	Exercises Part 1	4
1.1	Exercise 1.....	4
1.2	Exercise 2.....	12
1.3	Exercise 3.....	17
1.4	Exercise 4.....	22
1.5	Exercise 5.....	26
2	Exercises Part 2	36
2.1	Exercise 1.....	36
2.2	Exercise 2.....	38
2.3	Exercise 3.....	41
2.4	Exercise 4.....	41
2.5	Exercise 5.....	43
2.6	Exercise 6.....	44
2.7	Exercise 7.1	47
2.8	Exercise 7.2	50
2.9	Exercise 8.....	53
3	Exercises Part 3	56
3.1	Exercise 1.....	56
3.2	Exercise 2.....	59
3.3	Exercise 3.....	61

1 Abstract

The present report is the main assignment for the Fundamentals and Background on Artificial Intelligence class. It contains the results for the exercises for all three parts of the assignment. The first two parts are implemented in R and the last part in Matlab.

1 Exercises Part 1

In this part, the exercises from the following file are presented:

ΜποχαληςΔεληκωνσταντής.pdf

1.1 Exercise 1

In this exercise we implemented linear regression, logistic regression and the perceptron algorithm for a dataset that describes coronary heart disease versus age (Coronary Heart Disease.txt). Then, for each algorithm used, we report back the intercept (w_0), the slope (w_1), R-squared parameter and the plot graphs. According to the exercise description, age is the independent variable and coronary heart disease is the dependent one. In the plots presented throughout the exercise, X axis displays the independent variable (Age) whereas Y axis displays the dependent variable (Coronary heart disease).

Implementation

At start, we load the required libraries. Not all the libraries that are loaded are used as there were many changes and experimentation before the result. They are inserted as a group in each exercise and there are further libraries added to the group if needed.

```
# Load Libraries
library(tidyverse)

library(ggpubr)
library(aod)
library(ggplot2)
library(readtext)
library(rgl)

library(corrplot)
library(DescTools)
```

We start by implementing a simple logistic regression example for a single independent variable and we extract the intercept (w_0), the slope (w_1) of the model and the pseudo-R-squared value. We used the **glm** function and summary function to extract these parameters. We initialize a vector x with values and a vector y with

labels. Variable x is the independent variable whereas variable y is the dependent one. The model presented below is of type $y = w_0 + w_1 * x$. This implementation is to demonstrate a simple logistic regression example.

```
# create vector with values
x <- c(1, 58, 43, 91, 65, 12, 27, 43, 77)
# create vector with labels
y <- c(0, 1, 1, 1, 0, 1, 0, 1, 1)
# estimate a logistic regression model
glmmodel <- glm(y ~ x)
# view basic descriptives of the data
summary(glmmodel)

##
## Call:
## glm(formula = y ~ x)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -0.7885  -0.3709   0.1332   0.3551   0.5574
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.364350   0.317691   1.147   0.289
## x            0.006525   0.005869   1.112   0.303
##
## (Dispersion parameter for gaussian family taken to be 0.2428369)
##
##      Null deviance: 2.0000  on 8  degrees of freedom
## Residual deviance: 1.6999  on 7  degrees of freedom
## AIC: 16.541
##
## Number of Fisher Scoring iterations: 2

# calculate R-squared value for logistic regression
PseudoR2(glmmodel)

## McFadden
## 0.1219091
```

As we can see, the intercept is $w_0=0.36435$ and the slope is $w_1=0.006525$. So now, the model generalizes to $y = 0.36435 + 0.006525 * x$. Furthermore, the R-squared factor from the probabilities is 12.19% which is a poor explanation of the variability of the response data around its mean. The R-squared factor is reported back with the PseudoR2 function.

Now we will use the simple linear regression algorithm on the Coronary Heart Disease.txt dataset and we will report back the parameters of the model (w_0 , w_1 and R^2). First, we read the file with the read.table function and put it in a table. Then, we view the column names and the first few rows to get an idea on how the dataset looks like. The summary function gives us some basic descriptives of the data (min value, max value, median, mean, 1st Qu, 3rd Qu). Then we view the standard

deviations by applying the sd function to each variable on the dataset. The correlation coefficient between the variables is reported back with the cor function. Then we do a scatterplot of the data to visualize and see the relationship between the variables.

```
# read file
mydata <- read.table("Coronary Heart Disease.txt", header=TRUE, sep=",")
# view what is on the data frame
names(mydata)

## [1] "Age" "Coronary.Heart.Disease"

# view the first few rows of the data
head(mydata)

##   Age Coronary.Heart.Disease
## 1  25                      0
## 2  25                      0
## 3  25                      0
## 4  25                      0
## 5  25                      1
## 6  25                      0

# view basic descriptives of the data
summary(mydata)

##      Age      Coronary.Heart.Disease
##  Min.   :25.00   Min.      :0.00
## 1st Qu.:35.75   1st Qu.:0.00
##  Median :42.00   Median    :0.00
##   Mean   :44.20   Mean      :0.43
## 3rd Qu.:57.00   3rd Qu.:1.00
##   Max.   :62.00   Max.      :1.00

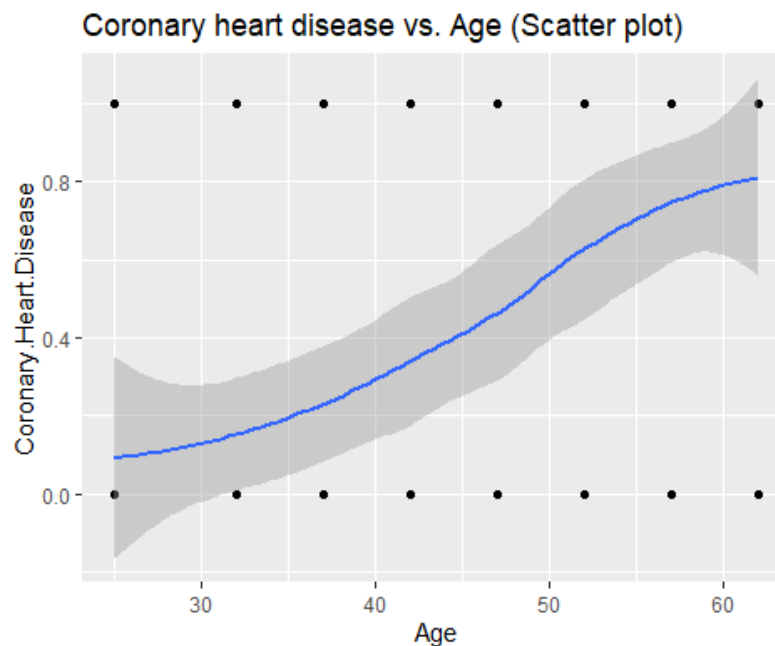
# view the standard deviations by applying the sd function to each variable on the data-set
sapply(mydata, sd)

##      Age Coronary.Heart.Disease
## 11.5277443      0.4975699

# correlation coefficient between the variables
cor(mydata$Age, mydata$Coronary.Heart.Disease)

## [1] 0.5078807

# scatter plot displaying Age versus Coronary heart disease
ggplot(mydata, aes(x = Age, y = Coronary.Heart.Disease)) + geom_point() + stat_smooth() + ggtitle("Coronary heart disease vs. Age (Scatter plot)")
```

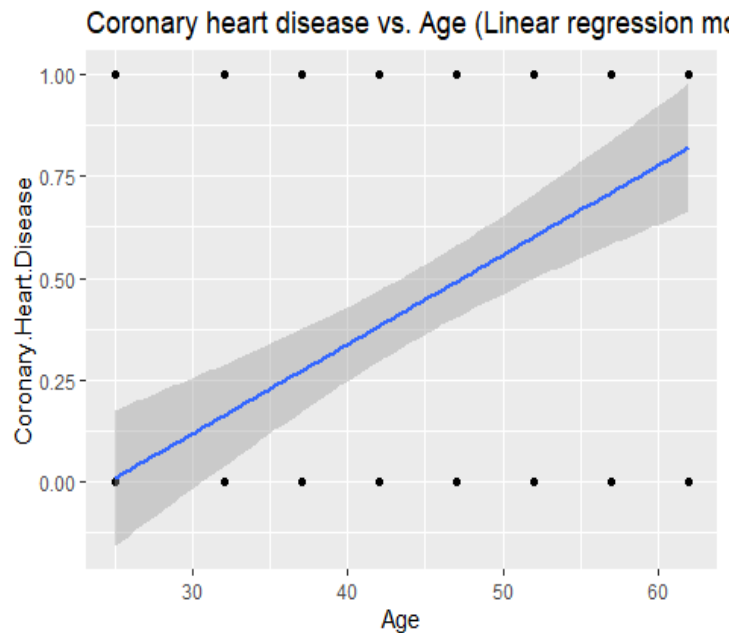


Now that we have prepared our data, we are ready to implement the linear regression algorithm with the `lm` function. Age is the independent variable whereas Coronary heart disease is the dependent variable. The independent variable (Age) is displayed on the X axis whereas the dependent variable (Coronary heart disease) is displayed on the Y axis. The linear model (`lm` function) reports back that the intercept is $w_0 = -0.538933$ and the slope is $w_1 = 0.021922$. So, the simple linear model generalizes to $\text{Chd} = -0.538933 + 0.021922 * \text{Age}$. The multiple R-squared is 25.79%. This means that the linear model poorly explains the variability of the response data around its mean. Also, we created the linear regression plot between the variables.

```
# Age is the independent variable whereas Coronary heart disease is the dependent variable
# The independent variable (Age) is displayed on the X axis whereas the dependent variable (Coronary heart disease) is displayed on the Y axis
# Chd = w0 + w1 * Age
# estimate a linear regression model
lmmodel <- lm(Coronary.Heart.Disease ~ Age, data = mydata)
print(lmmodel)

##
## Call:
## lm(formula = Coronary.Heart.Disease ~ Age, data = mydata)
##
## Coefficients:
## (Intercept)      Age
##   -0.53893    0.02192

# scatter plot of the linear regression line
ggplot(mydata, aes(Age, Coronary.Heart.Disease)) + geom_point() + stat_smooth(
  method = lm) + ggtitle("Coronary heart disease vs. Age (Linear regression model)")
```



```
# view the statistical summary of the linear regression model
summary(lmmodel)
```

```
##
## Call:
## lm(formula = Coronary.Heart.Disease ~ Age, data = mydata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.82020 -0.38177 -0.00911  0.28940  0.99089
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.538933   0.171510  -3.142  0.00222 **
## Age          0.021922   0.003756   5.837 6.91e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4308 on 98 degrees of freedom
## Multiple R-squared:  0.2579, Adjusted R-squared:  0.2504
## F-statistic: 34.07 on 1 and 98 DF, p-value: 6.905e-08
```

The linear model predicts a probability of 35.9851% for someone aged 41 years old to suffer from Coronary heart disease. Below we can observe the model prediction probability outcome.

```
# model prediction probability (linear regression) of someone 41 years old suffering of Coronary heart disease
Age_41 <- data.frame(Age=41)
Age_41$prediction <- predict(lmmodel, newdata=Age_41)
print(Age_41$prediction)

## [1] 0.359851
```


As a next step, we implement the logistic regression algorithm on the same dataset with the glm function. Again, Age is the independent variable whereas Coronary heart disease is the dependent variable. The independent variable (Age) is displayed on the X axis whereas the dependent variable (Coronary heart disease) is displayed on the Y axis. The logistic regression model (glm function) reports back that the intercept value is $w_0 = -5.2264$ and the slope value is $w_1 = 0.1095$. So, the logistic regression model generalizes to $\text{Chd} = -5.2264 + 0.1095 * \text{Age}$. The pseudo-R-squared value is 20.78806%. This means that the logistic model explains less the variability of the response data around its mean than the linear model algorithm (Linear model R-squared=25.79%). Below, we can observe the original (probability) data plot.

```
# estimate a logistic regression model
glmmodel <- glm(Coronary.Heart.Disease ~ Age, data = mydata, family = "binomial")
print(glmmodel)

##
## Call:  glm(formula = Coronary.Heart.Disease ~ Age, family = "binomial",
##      data = mydata)
##
## Coefficients:
## (Intercept)      Age
##      -5.2264      0.1095
##
## Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
## Null Deviance:      136.7
## Residual Deviance: 108.3    AIC: 112.3

# view the statistical summary of the logistic regression model
summary(glmmodel)

##
## Call:
## glm(formula = Coronary.Heart.Disease ~ Age, family = "binomial",
##      data = mydata)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.8724  -0.9251  -0.3994   0.7864   2.2665
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.22638    1.12035  -4.665 3.09e-06 ***
## Age          0.10950    0.02384   4.593 4.36e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 136.66  on 99  degrees of freedom
## Residual deviance: 108.25  on 98  degrees of freedom
```

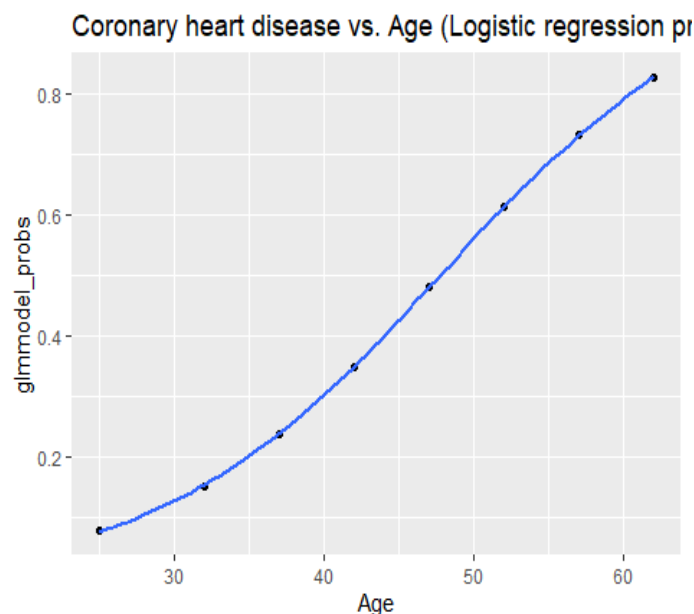
```
## AIC: 112.25
##
## Number of Fisher Scoring iterations: 4

# calculate R-squared value for logistic regression
PseudoR2(glmmodel)

## McFadden
## 0.2078806

# make predictions on the training data used to fit the model and get a vector of fitted probabilities
glmmodel_probs <- predict(glmmodel, type = "response")
# plot predicted probability
ggplot(mydata, aes(x = Age, y = glmmodel_probs)) + geom_point() + stat_smooth() + ggtitle("Coronary heart disease vs. Age (Logistic regression predicted probability)")

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The logistic regression model predicts a probability of 32.37096% for someone aged 41 years old to suffer from Coronary heart disease. Below we can observe the model prediction probability outcome.

```
# model prediction probability (logistic regression) of someone 41 years old
suffering of Coronary heart disease
Age_41 <- data.frame(Age=41)
Age_41$prediction <- predict(glmmodel, newdata=Age_41, type='response')
print(Age_41$prediction)

## [1] 0.3237096
```

Furthermore, we continue to implement the perceptron algorithm for the same dataset. At start we divide the dataset into train and test. Then, we do a scatter plot of the training dataset.

```
#perceptron implementation
# divide complete data-set into test and train data-sets
indexes <- sample(1:nrow(mydata), size = 0.8 * nrow(mydata))
train <- mydata[indexes,]
# train
test <- mydata[-indexes,]
# test

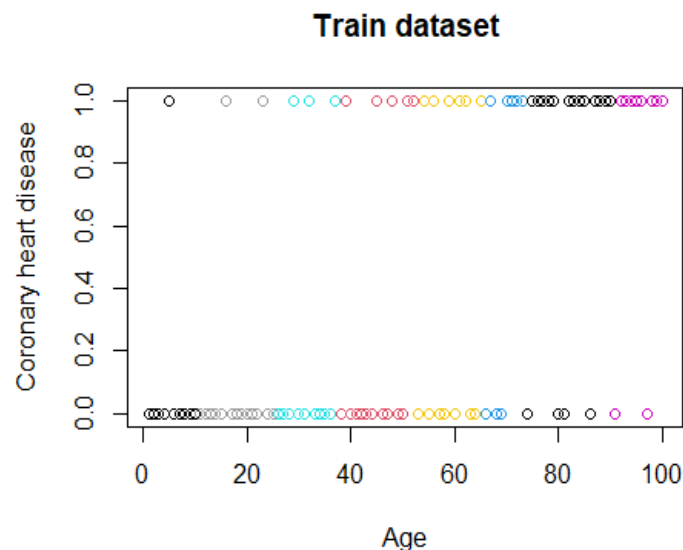
# plot train data-set
str(train)

## 'data.frame': 80 obs. of 2 variables:
## $ Age : int 62 52 52 42 42 37 47 25 62 57 ...
## $ Coronary.Heart.Disease: int 1 1 1 0 1 0 1 1 1 1 ...

summary(train)

##      Age      Coronary.Heart.Disease
## Min.   :25.00   Min.   :0.0000
## 1st Qu.:37.00   1st Qu.:0.0000
## Median :42.00   Median :0.0000
## Mean   :44.76   Mean   :0.4375
## 3rd Qu.:57.00   3rd Qu.:1.0000
## Max.   :62.00   Max.   :1.0000

with(train, plot(mydata$Coronary.Heart.Disease, col=mydata$Age, xlab = "Age",
ylab = "Coronary heart disease", main = "Train dataset"))
```



1.2 Exercise 2

In this exercise, we first present the theoretical background of linear regression and then we use the dataset **faithful** to fit a linear regression model.

Linear Regression

Linear Regression is a regression learning algorithm that learns a model which is a linear combination of features of the input variable. We have a collection of labeled examples $\{x_i, y_i\}_{i=1}^N$ where N is the size of the dataset and x_i is the D -dimensional feature vector of example i .

The purpose is to build a model as a **linear combination of features of x** :

$$h_{\theta,b} = \theta x + b$$

Where θ is a D -dimensional vector of parameters and b is a real number. The model is then used to predict the unknown y for a given x .

To estimate the parameters θ , we must minimize the following objective (loss) function:

$$\frac{1}{N} \sum_{i=1..N} (h_{\theta,b}(x_i) - y_i)^2$$

This function is also called **squared error loss**. The optimization method used to minimize this equation is called **gradient descent**. We implemented gradient descent from scratch in the Exercise Part 3 section of this report.

The next step is to load the dataset **faithful**.

```
# Apply the lm() function.
lm_faithfull <- lm(eruptions ~ waiting, data = faithful)

print(lm_faithfull)

##
## Call:
## lm(formula = eruptions ~ waiting, data = faithful)
##
## Coefficients:
## (Intercept)      waiting
##    -1.87402      0.07563
```

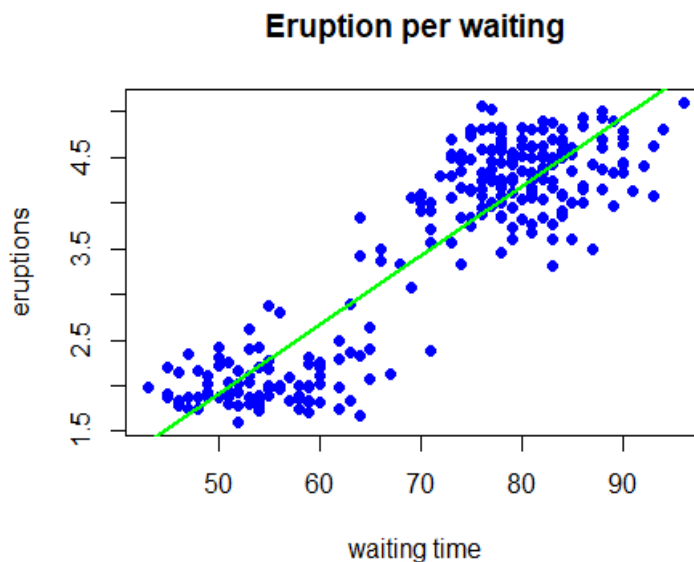
```

print(summary(lm_faithfull))

##
## Call:
## lm(formula = eruptions ~ waiting, data = faithful)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.29917 -0.37689  0.03508  0.34909  1.19329
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.874016   0.160143  -11.70  <2e-16 ***
## waiting      0.075628   0.002219   34.09  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4965 on 270 degrees of freedom
## Multiple R-squared:  0.8115, Adjusted R-squared:  0.8108
## F-statistic: 1162 on 1 and 270 DF,  p-value: < 2.2e-16

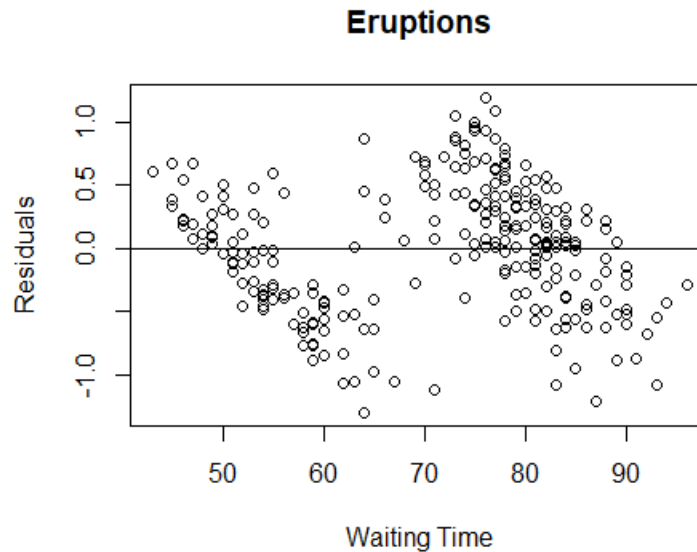
abline(lm_faithfull, col = "green", lwd = 2)

```



We observe that the R-squared value of the model is 0.8115. This means that 81% of the variance in the dataset can be explained by the linear model. In the next plot we present simple residuals analysis. We observe that no significant pattern emerges in the residuals.

```
residuals <- resid(lm_faithfull)
plot(faithful$waiting, residuals, ylab="Residuals", xlab="Waiting Time", ma
in="Eruptions")
abline(0, 0)
```



```
# Predict eruption for waiting value = 80
waiting80 <- data.frame(waiting = c(80))
pred <- predict(lm_faithfull,waiting80)
print(pred)

##      1
## 4.17622

summary(aov(lm_faithfull))

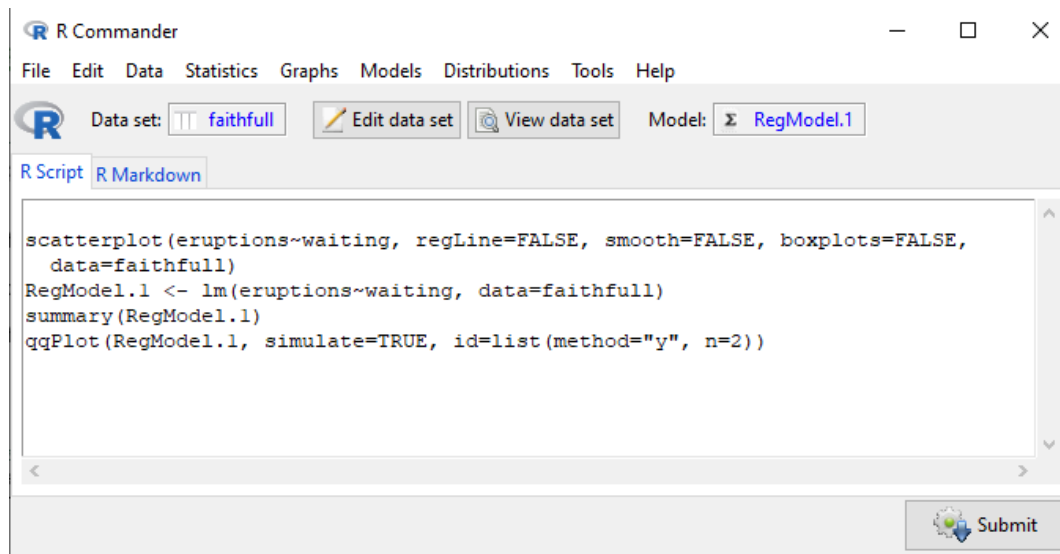
##           Df Sum Sq Mean Sq F value Pr(>F)
## waiting      1 286.48  286.48    1162 <2e-16 ***
## Residuals  270   66.56    0.25
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We also predict the value of the eruption variable if waiting time is equal to 80. The predicted value is 4.17.

Finally, we confirm that for a confidence level of 0.05 there is a statistically significant relationship between the variables eruption and waiting.

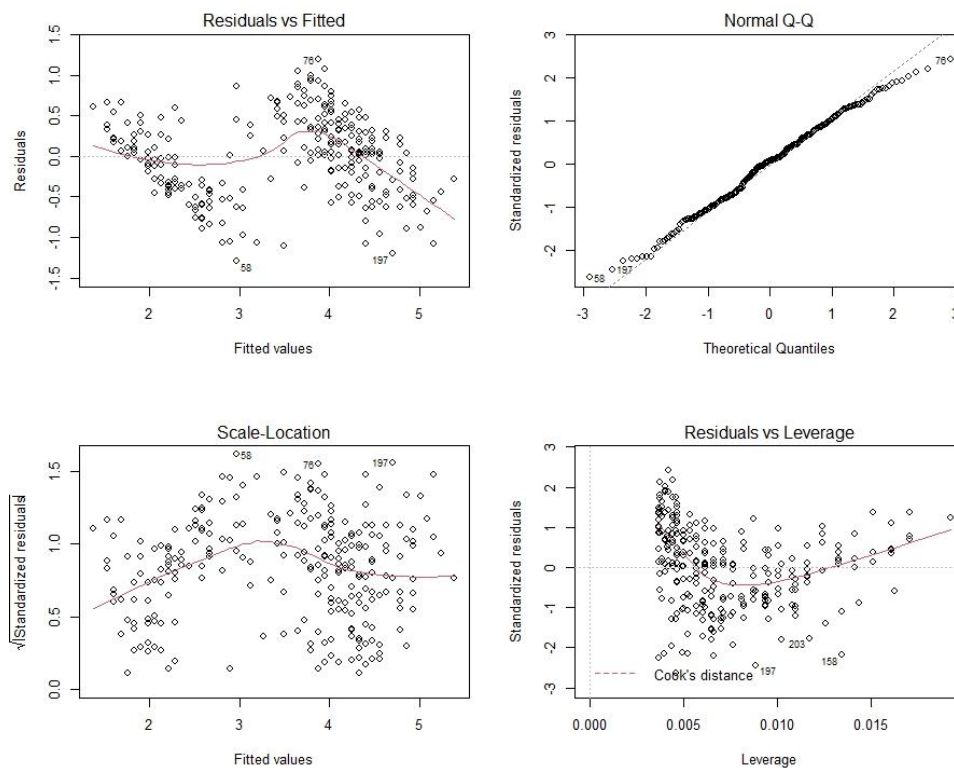
We also used the package **Rcmdr** to perform the above analysis. The result were the same. The package is easy to use and helpful in this kind of analysis. To start the R commander environment, we run the following command:

```
library('Rcmdr')
```



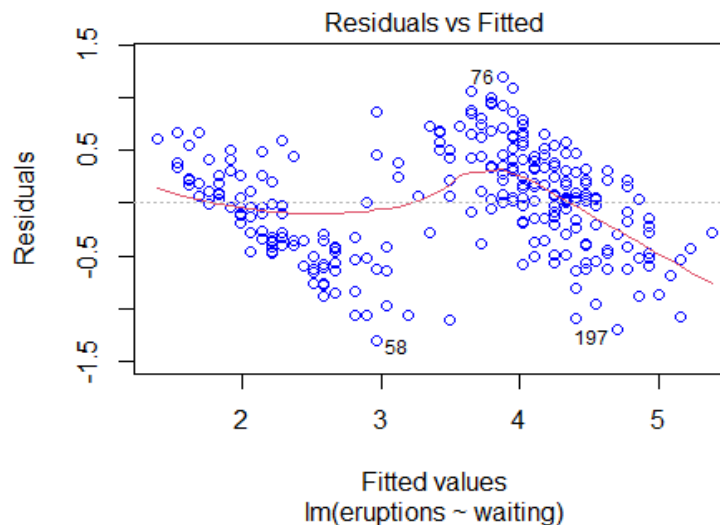
To perform a residuals analysis for the waiting variable, we use the R commander environment.

lm(eruptions ~ waiting)

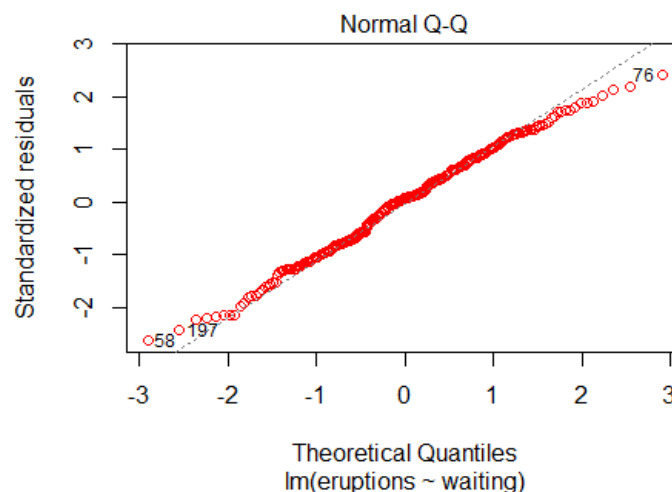


Residual plots are used to look for underlying patterns in the residuals that may mean that the model has a problem.

In the following plot, the data does not have any obvious distinct pattern. While it is slightly curved, it has equally spread residuals around the horizontal line without a distinct pattern.

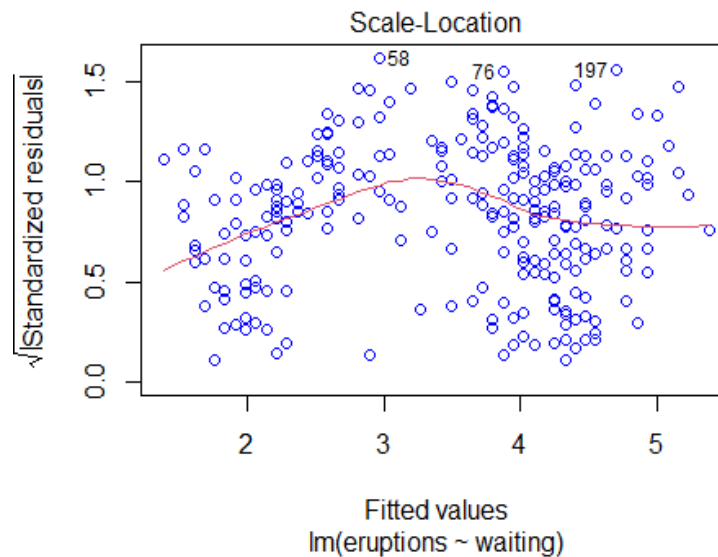


Residuals should be **normally distributed**, and the Q-Q Plot will show this. If residuals follow close to a straight line on this plot, it is a good indication they are normally distributed. For our model, the Q-Q plot shows pretty good alignment to the line with a few points at the top slightly offset. Probably not significant and a reasonable alignment.



This plot tests the linear regression assumption of equal variance (homoscedasticity) i.e. that the residuals have equal variance along the regression line. It is also called the Spread-Location plot.

For our example, the residuals are reasonably well spread above and below a horizontal line however the beginning of the line does have fewer points so slightly less variance there.



1.3 Exercise 3

In this exercise we analyzed the data-set `decathlon.csv` which refer to the performance records of different athletes per game for the Olympic games (2004, Greece). Firstly, we normalized the data because we wanted to perform a reliable analysis. If we had not normalized the data, the algorithm would be dominated by the variables that use a larger scale, adversely affecting our model's performance. Therefore, we decided it was imperative to normalize the data. Then, we created a dendrogram which shows a picture of the athletes' performances. Also, we performed k-means clustering on the dataset and estimated the best value for k.

At start, we load the required libraries. Then, we read the dataset with the command `read.csv` function which puts the data into a data-frame. Then, we view the column names and the first few rows to get an idea on how the dataset looks like. The summary function gives us some basic descriptives of the data (min value, max value, median, mean, 1st Qu, 3rd Qu).

```
# read file
mydata <- read.csv("decathlon.csv", header=TRUE, sep=";")
mydata <- mydata
```

```
# view what is on the data frame
```

```
names(mydata)
```

```
## [1] "X"           "X100m"       "Long.jump"   "Shot.put"    "High.jump"
## [6] "X400m"       "X110m.H"     "Discus"      "Pole.vault"  "Javeline"
## [11] "X1500m"      "Rank"        "Points"      "Competition"
```

```
# view the first few rows of the data
```

```
head(mydata)
```

```
##           X X100m Long.jump Shot.put High.jump X400m X110m.H Discus Pole.v
ault
## 1   Sebrle 10.85      7.84    16.36      2.12 48.36   14.05  48.72
5.0
## 2     Clay 10.44      7.96    15.23      2.06 49.19   14.13  50.11
4.9
## 3   Karpov 10.50      7.81    15.93      2.09 46.81   13.97  51.65
4.6
## 4    Macey 10.89      7.47    15.73      2.15 48.97   14.56  48.34
4.4
## 5  Warners 10.62      7.74    14.48      1.97 47.97   14.01  43.73
4.9
## 6 Zsivoczky 10.91      7.14    15.31      2.12 49.40   14.95  45.62
4.7
## Javeline X1500m Rank Points Competition
## 1   70.52 280.01    1   8893   OlympicG
## 2   69.71 282.00    2   8820   OlympicG
## 3   55.54 278.11    3   8725   OlympicG
## 4   58.46 265.42    4   8414   OlympicG
## 5   55.39 278.05    5   8343   OlympicG
## 6   63.45 269.54    6   8287   OlympicG
```

```
# view basic descriptives of the data
```

```
summary(mydata)
```

```
##           X           X100m       Long.jump       Shot.put
## Length:41      Min.   :10.44    Min.   :6.61    Min.   :12.68
## Class :character 1st Qu.:10.85    1st Qu.:7.03    1st Qu.:13.88
## Mode  :character Median :10.98    Median :7.30    Median :14.57
##                Mean  :11.00    Mean  :7.26    Mean  :14.48
##                3rd Qu.:11.14    3rd Qu.:7.48    3rd Qu.:14.97
##                Max.   :11.64    Max.   :7.96    Max.   :16.36
## High.jump       X400m       X110m.H       Discus
## Min.   :1.850    Min.   :46.81    Min.   :13.97    Min.   :37.92
## 1st Qu.:1.920    1st Qu.:48.93    1st Qu.:14.21    1st Qu.:41.90
## Median :1.950    Median :49.40    Median :14.48    Median :44.41
## Mean   :1.977    Mean   :49.62    Mean   :14.61    Mean   :44.33
## 3rd Qu.:2.040    3rd Qu.:50.30    3rd Qu.:14.98    3rd Qu.:46.07
## Max.   :2.150    Max.   :53.20    Max.   :15.67    Max.   :51.65
## Pole.vault      Javeline       X1500m       Rank       Point
s
```

```
## Min. :4.200 Min. :50.31 Min. :262.1 Min. : 1.00 Min. :7
313
## 1st Qu.:4.500 1st Qu.:55.27 1st Qu.:271.0 1st Qu.: 6.00 1st Qu.:7
802
## Median :4.800 Median :58.36 Median :278.1 Median :11.00 Median :8
021
## Mean :4.762 Mean :58.32 Mean :279.0 Mean :12.12 Mean :8
005
## 3rd Qu.:4.920 3rd Qu.:60.89 3rd Qu.:285.1 3rd Qu.:18.00 3rd Qu.:8
122
## Max. :5.400 Max. :70.52 Max. :317.0 Max. :28.00 Max. :8
893
```

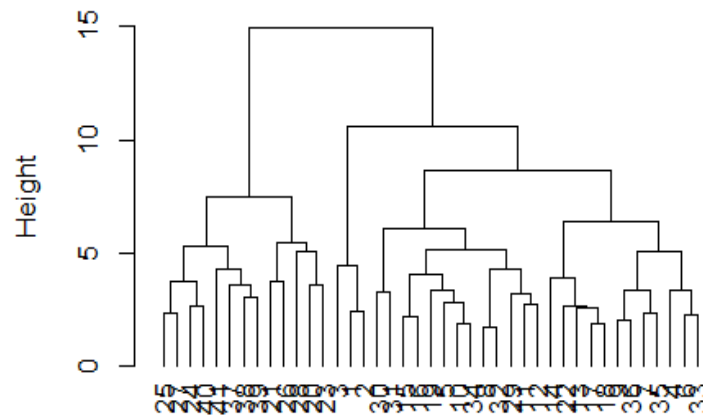
As a next step, we create a dendrogram of the dataset. To do this, we had to exclude columns 1 and 14 which are non-numeric columns. Column 1 refers to the names of the athletes and column 14 refers to the name of the competition "Olympic games". After that, we normalize the data because we want to perform a reliable analysis. If we do not normalize the data, the algorithm will be dominated by the variables that use a larger scale, adversely affecting model performance.

Therefore, we decided it is imperative to normalize the data. After normalizing the data, we calculated the Euclidean distances of the values of the data and stored it in a table. Then we performed hierarchical clustering on the distances of the data. After that, we converted the hierarchical clustering into a dendrogram and plot it by using the function plot.

```
# exclude columns 1 and 14 as they are non-numeric columns
mydata <- mydata[,-c(1,14)]

# compute euclidean distances and hierarchical clustering
dd <- dist(scale(mydata), method = "euclidean")
hc <- hclust(dd, method = "ward.D2")

# Convert hierarchical clustering into a dendrogram and plot
hcd <- as.dendrogram(hc)
# Default plot
plot(hcd, type = "rectangle", ylab = "Height")
```



Then, we implemented the k-means clustering algorithm. Initially, we experimented with different k values and displayed the cluster centers and the data points in each cluster. Different values for k were tested (2 ~ 40).

```
# set the seed for reproducibility
set.seed(76964057)

# initialize k-means and experiment with the number of clusters
k <- kmeans(scale(mydata), centers=5)

# display cluster centers
# display averages for each numeric variable
k$centers
```

	X100m	Long.jump	Shot.put	High.jump	X400m	X110m.H
## 1	-0.4195697	0.4278885	-0.04683446	-0.2929723	-0.29559721	-0.4395866
## 2	-0.1779266	-0.1145696	0.35530925	1.0614972	-0.26883805	-0.2773563
## 3	0.8710040	-0.6998338	-0.57000784	-0.6228260	1.54389019	0.9595767
## 4	-1.5260343	1.9279293	1.65317910	1.2722886	-1.29727375	-1.1781827
## 5	0.5358893	-0.5530945	-0.32031085	-0.4140422	-0.09219419	0.4750987

```
##      Discus Pole.vault   Javeline   X1500m      Rank      Points
## 1 -0.1824770  1.0205576 -0.4125285  0.3575341 -0.3942381  0.2575246
## 2  0.5249324 -0.9170474  0.1322640 -0.5818756 -0.5836571  0.4300979
## 3 -0.1644865 -0.3068209 -0.3674734  1.0662458  1.2113141 -1.3470882
## 4  1.7272523  0.2550157  1.4378164  0.0869614 -1.2781937  2.3578734
## 5 -0.5857610 -0.4548169  0.2563623 -0.7717542  0.5149735 -0.4432606

# give a count of data points in each cluster
table(k$cluster)
```

	1	2	3	4	5
##	13	8	7	3	10

As a next step, we implemented an iteration with different values of k to find the best value. We tried for k values from 2 to 30 with 100 iterations for each k value. In each iteration we store the total sum of squares of the distance from each data point to the cluster center. After the 100 iterations we calculate the mean of the total sum of squares. Then, we plot the average total sum of squares vs k value.

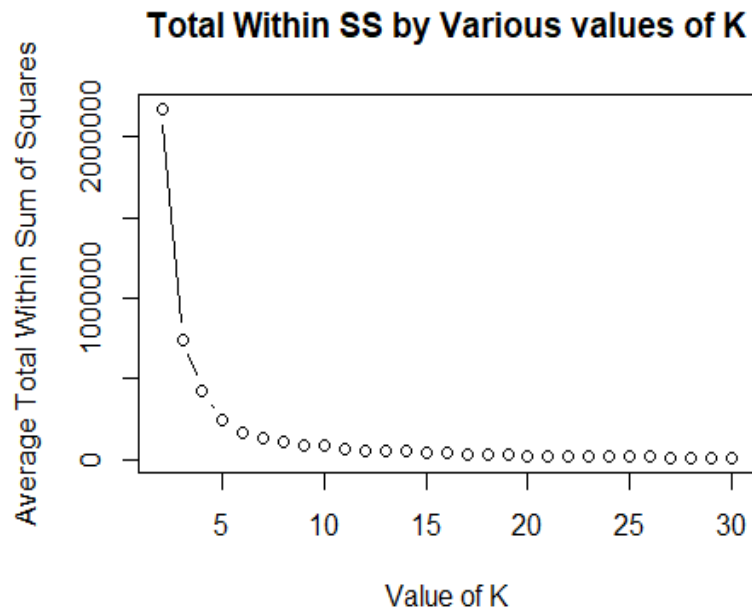
```
# K from 2 to 30
rng <- 2:30

# iterate k-means algorithm for 100 times
tries <- 100

# set up an empty vector to hold all of points
avg.totw.ss<-integer(length(rng))

# iterate for each value of the range variable
for(v in rng){
  # set up an empty vector to store the 100 tries
  v.totw.ss <- integer(tries)
  for(i in 1:tries){
    #Run kmeans
    k.temp <- kmeans(mydata, centers=v)
    #Store the total sum of squares of the distance from each data point to the cluster center
    v.totw.ss[i]<-k.temp$tot.withinss
  }
  #Average the 100 total sum of squares of the distance from each data point to the cluster center
  avg.totw.ss[v-1]<-mean(v.totw.ss)
}

# plot the average total sum of squares vs k value
plot(rng,avg.totw.ss,type="b", main="Total Within SS by Various values of K",
      ylab="Average Total Within Sum of Squares",
      xlab="Value of K")
```



Viewing the plot we can observe the “elbow” point being in value $k=5$. So we assume that a value of $k=5$ is the best number of clusters for k-means in our example.

1.4 Exercise 4

We load the following dataset and we fit a simple regression model with the following commands:

```

cpi <- c(163.3, 165.7, 166.5, 168, 168.4, 169,
         167.5, 170.1, 173, 172.1, 174.5, 174)

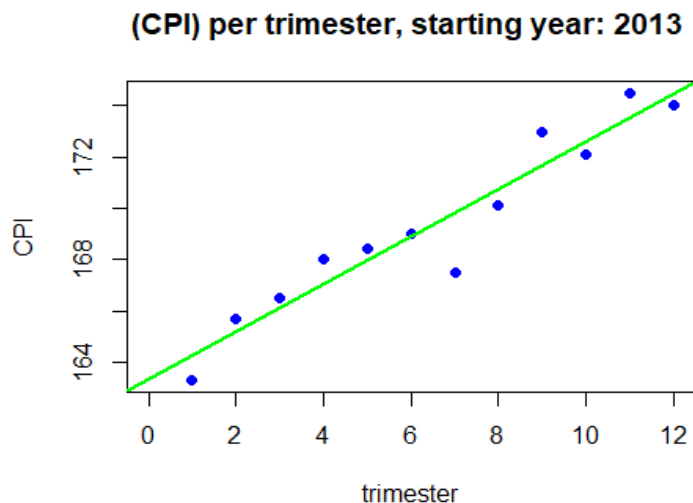
trimester <- seq(1, 12 )

# Apply the lm() function.
relation <- lm(cpi~ trimester)

print(relation)

```

```
##
## Call:
## lm(formula = cpi ~ trimester)
##
## Coefficients:
## (Intercept)      trimester
##    163.3258         0.9255
```



To predict the values for the 4 trimesters of 2016:

```
trimester2016 <- data.frame(trimester = c(13,14,15,16))
predCPI <- predict(relation, trimester2016)
print(predCPI)

##          1          2          3          4
## 175.3576 176.2831 177.2086 178.1341
```

For the next part of the exercise, we use the dataset **Census.txt**. We fit a multiple linear regression model, and then we identify the regression line. Finally, we compute the ANOVA table.

```
Census <- read.delim("Datasets/Census.txt",
                     header=TRUE,
                     sep=" ",
                     dec = ".")

lmCensus <- lm(LIFE~MALE + BIRTH + DIVO + BEDS + EDUC + INCO, data = Census)
summary(lmCensus)
```

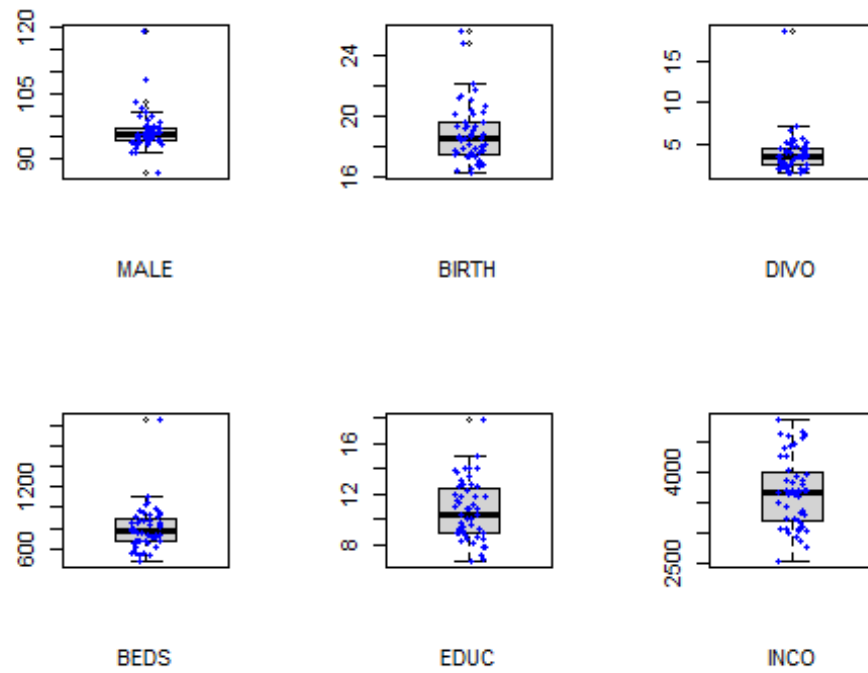
```
##
## Call:
## lm(formula = LIFE ~ MALE + BIRTH + DIVO + BEDS + EDUC + INCO,
##     data = Census)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.5563 -0.6629  0.0755  0.6983  3.3215
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  70.5577813   4.2897471   16.448 < 2e-16 ***
## MALE         0.1261019   0.0472318    2.670  0.01059 *
## BIRTH       -0.5160558   0.1172775   -4.400 6.78e-05 ***
## DIVO        -0.1965375   0.0739533   -2.658  0.01093 *
## BEDS        -0.0033392   0.0009795   -3.409  0.00141 **
## EDUC         0.2368223   0.1110225    2.133  0.03853 *
## INCO        -0.0003612   0.0004598   -0.786  0.43633
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.176 on 44 degrees of freedom
## Multiple R-squared:  0.4685, Adjusted R-squared:  0.396
## F-statistic: 6.464 on 6 and 44 DF, p-value: 6.112e-05

summary(aov(lmCensus))

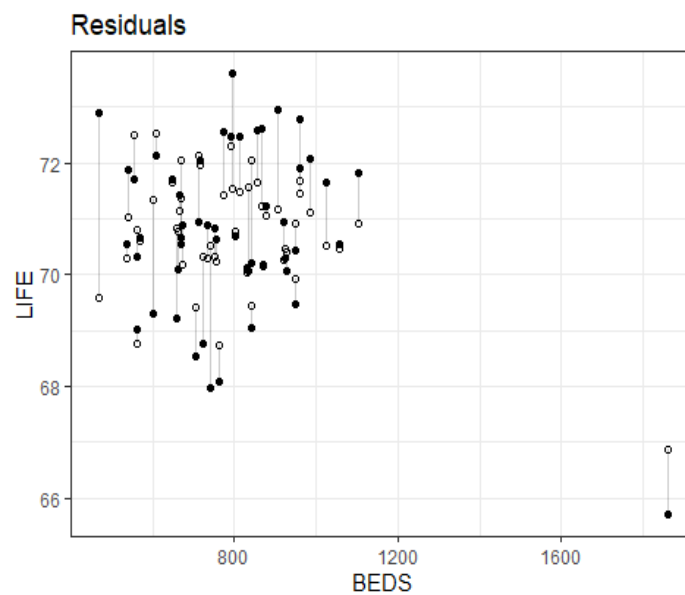
##              Df Sum Sq Mean Sq F value    Pr(>F)
## MALE           1   4.56   4.563    3.302 0.076003 .
## BIRTH           1  24.41  24.410   17.664 0.000127 ***
## DIVO           1   4.67   4.673    3.382 0.072683 .
## BEDS           1  12.24  12.240    8.857 0.004729 **
## EDUC           1   6.86   6.856    4.961 0.031087 *
## INCO           1   0.85   0.853    0.617 0.436329
## Residuals     44  60.80   1.382
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For the ANOVA table, we conclude that BIRTH, BEDS and EDUC are statistically significant when prediction the average lifespan of 50 USA states. Not the above fitted model is called additive model. It assumes that the two factor variables are independent.

The next step is to plot the boxplots for each variable:



The next step is to plot the residuals for the BED variable:



1.5 Exercise 5

In the first part of this exercise, we present the theoretical background on Principal Component Analysis and then perform PCA on the **iris** dataset.

Principal Component Analysis

PCA is a statistical method that uses an orthogonal transformation of the data to construct a new dataset consisting of linear independent variables called Principal Components. It is a non-parametric method and can be applied to any dataset and return results if the dataset is accordingly preprocessed. It is worth noting that if the initial dataset contains higher order relationships between the variables, the PCA method will fail.

The purpose of PCA is to find a linear transformation **P** of the dataset **X**

$$Y = PX$$

The steps of performing PCA are the following:

Subtract the mean

For PCA to work properly, you must subtract the mean from each of the data dimensions. The mean subtracted is the average across each dimension.

Compute the covariance matrix

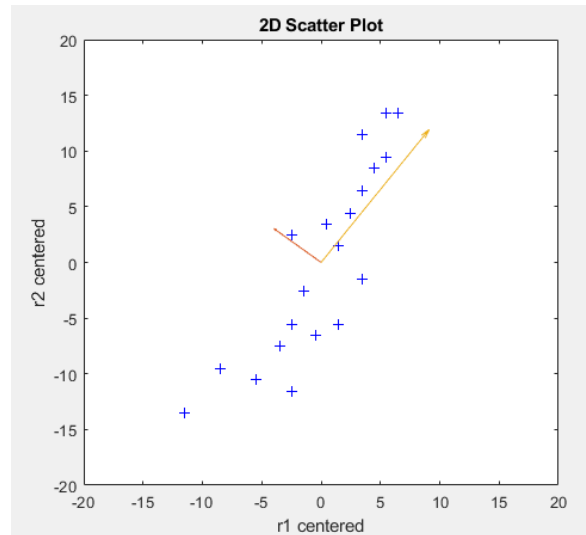
The covariance matrix is computed as follows:

$$C_X = \frac{1}{m-1} X_{cent} X_{cent}^T$$

Calculate the eigenvectors and eigenvalues of the covariance matrix

The next step is to compute the eigenvectors of the covariance matrix. In the following image, the eigenvectors for a 2dimensional datasets are plotted.

We observe that they are perpendicular to each other and they provide information about the patterns in the data. Each eigenvector is showing us how the two variables are related along each line. So, by computing the eigenvectors of the covariance matrix, we have extracted lines that characterize the data



Choosing components and forming a feature vector

In the final step, we choose which principal component to keep. The eigenvectors with the highest eigenvalues represent most of the variance in the dataset. So, we order the eigenvectors based on the eigenvalues from highest to lowest and then we decide to filter out the components of lesser significance. If we originally have n dimensions, we will compute n eigenvectors. If we decide to keep p eigenvectors, then we will reduce the number of dimensions of the final dataset ($p < n$).

We then form a feature vector from the final eigenvectors:

$$FeatureVector = (PC1 \ PC2 \ .. \ PCn)$$

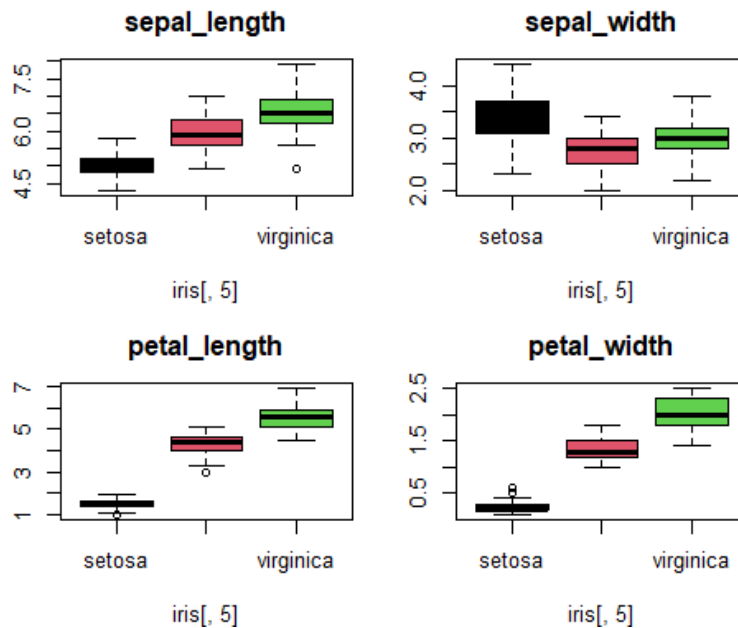
Transform the initial dataset

The final step is to use the Feature Vector to transform the initial dataset in the new feature space. So, the data are now expressed in the axis described by the eigenvectors.

Iris Dataset – PCA Example

The following example in R uses the function **princomp** to perform PCA on the iris dataset. We also use the library **factoextra** for better visualization of PCA variables.

The dataset consists of 4 variables describing sepal length and width and petal length and width. Also, a categorical variable describing the species of the flower. The boxplots are the following:



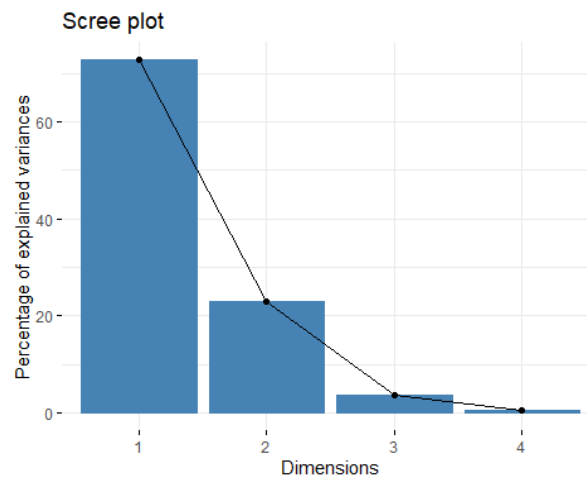
The next step is to perform PCA using the following commands. We set `cor=TRUE` so the data will be centered and scaled before the analysis. Also, we set `score=TRUE` so that the coordinates on each principal component are calculated.

```
iris_pca <- princomp(iris[,c('sepal_length', 'sepal_width', 'petal_length', 'petal_width')], scores=TRUE, cor=TRUE)
fviz_eig(iris_pca)
```

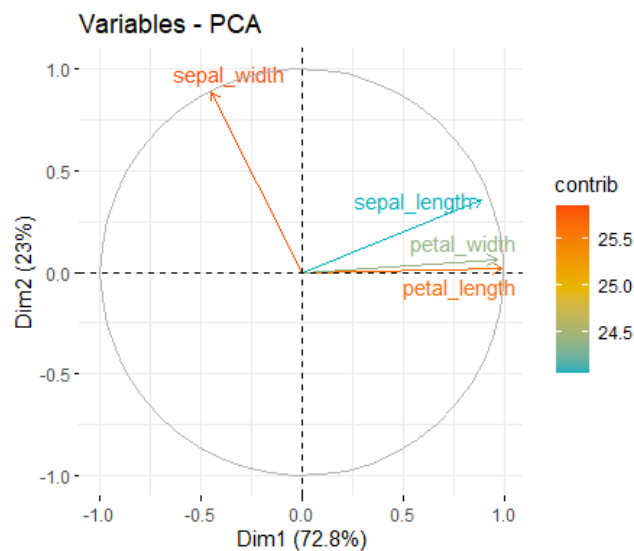
```
summary(iris_pca)
```

```
## Importance of components:
##               Comp.1   Comp.2   Comp.3   Comp.4
## Standard deviation  1.7061120 0.9598025 0.38386622 0.143553848
## Proportion of Variance 0.7277045 0.2303052 0.03683832 0.005151927
## Cumulative Proportion 0.7277045 0.9580098 0.99484807 1.000000000
```

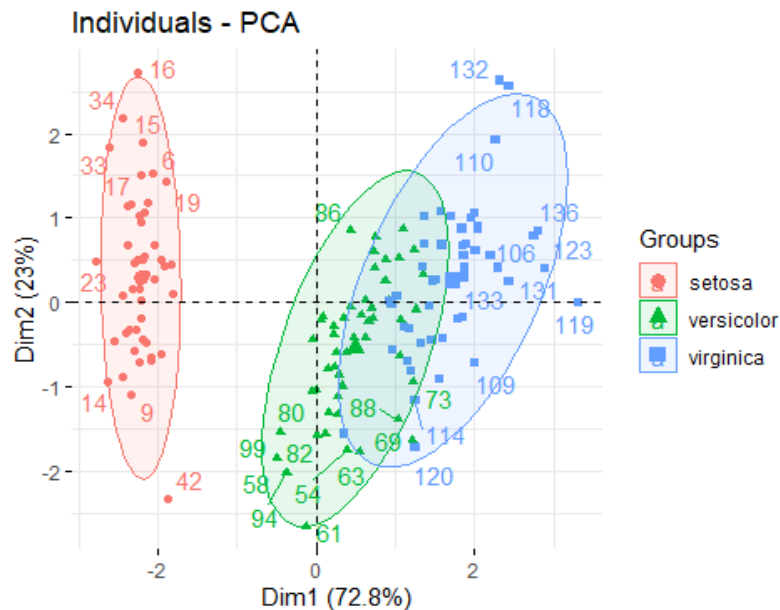
From the summary, first PC explains about 72% of total variables.
By the time we reach the second PC 95% of total variance is explained.



Next, we plot the principal components. We observe that **sepal_length**, **petal_width** and **petal_length** all contribute to PC1 that explains 73% of the variance



Finally, we plot the examples in our dataset projected to the first two principal components. We can see that the setosa group already form a distinct cluster to the left. We can also observe the samples that contribute the most to each principal component – for example point 119 contributes most to PC1 etc.



Exercise 5 – Part B

In this exercise we analyzed the dataset "ask1mf.xls" which refer to a sample of 32 observations of high school students. The variables in this dataset are GPA (mean grade), TUCE (exams grade in finance), PSI (student relation with finance). We were called to perform a scatter plot between the variables, find the sample mean and sample variance, find the correlation coefficient, implement linear regression, and calculate the anova matrix.

At start, we load the required libraries. Then, we read the dataset with the read_excel function which puts the data into a data-frame. Then, we view the column names and the first few rows to get an idea on how the dataset looks like. The summary function gives us some basic descriptives of the data (min value, max value, median, mean, 1st Qu, 3rd Qu). After that, we do a scatter plot of the variable GPA vs variable TUCE.

```
# read data
mydata <- read_excel("ask1mf.XLS")

# view what is on the data frame
names(mydata)

## [1] "GPA" "TUCE" "PSI"

# view the first few rows of the data
head(mydata)

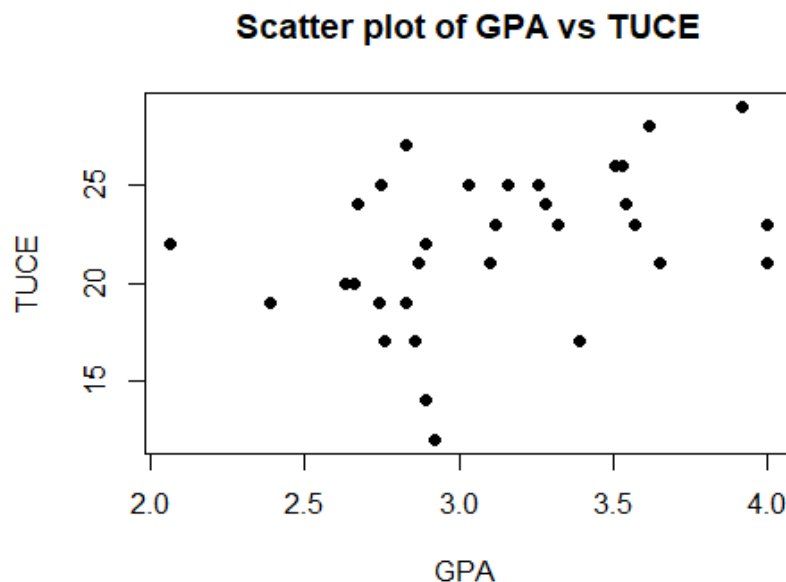
## # A tibble: 6 x 3
##   GPA  TUCE  PSI
##   <dbl> <dbl> <dbl>
## 1  2.66   20    0
## 2  2.89   22    0
```

```
## 3  3.28    24    0
## 4  2.92    12    0
## 5  4       21    0
## 6  2.86    17    0

# view basic descriptives of the data
summary(mydata)

##           GPA           TUCE           PSI
##  Min.      :2.060   Min.      :12.00   Min.      :0.0000
## 1st Qu.:2.812   1st Qu.:19.75   1st Qu.:0.0000
##  Median :3.065   Median :22.50   Median :0.0000
##   Mean   :3.117   Mean   :21.94   Mean   :0.4375
## 3rd Qu.:3.515   3rd Qu.:25.00   3rd Qu.:1.0000
##   Max.   :4.000   Max.   :29.00   Max.   :1.0000

# scatter plot of GPA vs TUCE
plot(mydata$GPA, mydata$TUCE, main="Scatter plot of GPA vs TUCE",
      xlab="GPA ", ylab="TUCE", pch=19)
```

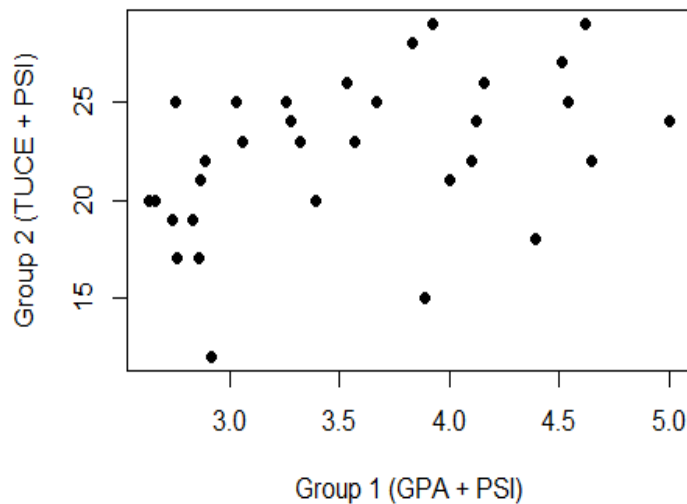


From the scatter plot output above we don't see a pattern/relationship between the variables GPA and TUCE. There is no ascending or descending relationship between the variables as we go from right to left on the plot.

As a next step, make two groups. Group 1 involves the variables GPA+PSI and Group 2 involves the variables TUCE+PSI. Then we implement a scatter plot between Group 1 (GPA+PSI) and Group 2 (TUCE+PSI).

```
# scatter plot of Group 1 (GPA + PSI) vs Group 2 (TUCE + PSI)
plot(mydata$GPA + mydata$PSI, mydata$TUCE + mydata$PSI, main="Scatter plot of
Group 1 (GPA + PSI) vs Group 2 (TUCE + PSI)",
     xlab="Group 1 (GPA + PSI)", ylab="Group 2 (TUCE + PSI)", pch=19)
```

catter plot of Group 1 (GPA + PSI) vs Group 2 (TUCE



Now the scatter plot presents an ascending relationship between the two groups. As we go from left to right the two groups increase together.

Furthermore, we calculate the sample mean and sample variance for each variable. Function `summary` shows us the sample mean for each variable and function `var` presents the sample variance for each variable.

```
# view sample mean
summary(mydata)

##           GPA           TUCE           PSI
##  Min.    :2.060   Min.    :12.00   Min.    :0.0000
## 1st Qu.:2.812   1st Qu.:19.75   1st Qu.:0.0000
##  Median :3.065   Median :22.50   Median :0.0000
##   Mean  :3.117   Mean    :21.94   Mean    :0.4375
## 3rd Qu.:3.515   3rd Qu.:25.00   3rd Qu.:1.0000
##   Max.  :4.000   Max.    :29.00   Max.    :1.0000

# view sample variance
var_GPA <- var(mydata$GPA)
var_GPA

## [1] 0.2178209
```



```
var_TUCE <- var(mydata$TUCE)
var_TUCE
```

```
## [1] 15.22177
```

```
var_PSI <- var(mydata$PSI)
var_PSI
```

```
## [1] 0.2540323
```

Additionally, we calculate the correlation coefficient between variables GPA and TUCE.

```
# correlation coefficient between variables GPA and TUCE
```

```
cor(mydata$GPA, mydata$TUCE, method = "kendall")
```

```
## [1] 0.2649643
```

In this step we calculate the linear regression between the variables TUCE AND GPA. TUCE is the dependent variable whereas GPA is the independent one.

```
# linear regression
```

```
# GPA is the independent variable whereas TUCE is the dependent variable
```

```
lmmodel <- lm(mydata$TUCE ~ mydata$GPA, data = mydata)
```

```
print(lmmodel)
```

```
##
```

```
## Call:
```

```
## lm(formula = mydata$TUCE ~ mydata$GPA, data = mydata)
```

```
##
```

```
## Coefficients:
```

```
## (Intercept)    mydata$GPA
```

```
##      11.853      3.235
```

```
# view the statistical summary of the linear regression model
```

```
summary(lmmodel)
```

```
##
```

```
## Call:
```

```
## lm(formula = mydata$TUCE ~ mydata$GPA, data = mydata)
```

```
##
```

```
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
```

```
## -9.2996 -1.8472  0.1343  2.8248  5.9916
```

```
##
```

```
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)   11.853      4.434   2.673  0.0120 *
```

```
## mydata$GPA     3.235      1.407   2.299  0.0287 *
```

```
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
```

```
## Residual standard error: 3.657 on 30 degrees of freedom
```

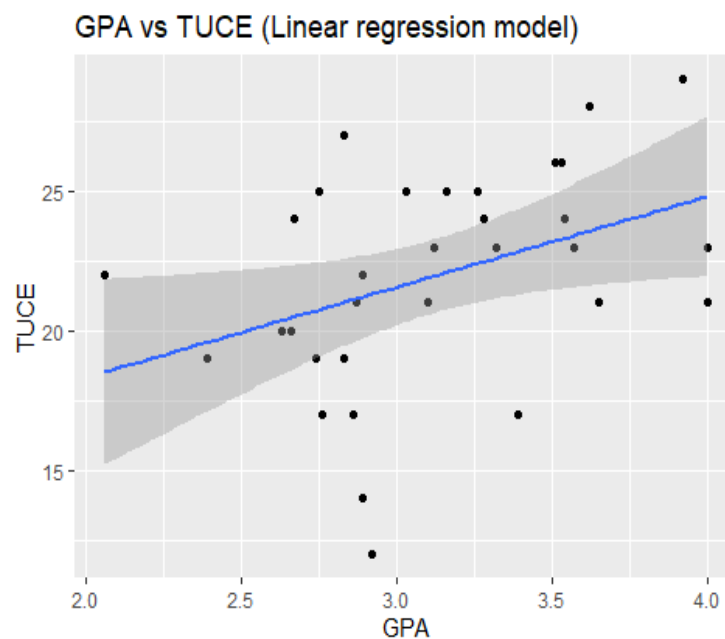
```
## Multiple R-squared:  0.1498, Adjusted R-squared:  0.1214
## F-statistic: 5.284 on 1 and 30 DF,  p-value: 0.02866

# correlation coefficient between GPA and TUCE
cor(mydata$GPA, mydata$TUCE)

## [1] 0.3869863

# scatter plot of the linear regression line
ggplot(mydata, aes(GPA, TUCE)) + geom_point() + stat_smooth(method = lm) + gg
title("GPA vs TUCE (Linear regression model)")

## `geom_smooth()` using formula 'y ~ x'
```



The linear model coefficients intercept is $w_0 = 11.853$ and slope is $w_1 = 3.235$. The multiple R-squared is 14.98%. We have a low R-squared value which means that we have a high bias on our data-set. The linear model scatter plot verifies the scatter plot assumptions we did in earlier step between variables TUCE and GPA. There is no obvious relationship between these two variables. As a last step, we calculate the anova matrix between the variables TUCE and GPA.

```

# anova matrix
fit <- aov(TUCE ~ GPA, data=mydata)
fit

## Call:
## aov(formula = TUCE ~ GPA, data = mydata)
##
## Terms:
##              GPA Residuals
## Sum of Squares  70.6672  401.2078
## Deg. of Freedom      1      30
##
## Residual standard error: 3.656992
## Estimated effects may be unbalanced

```

2 Exercises Part 2

In this part, the exercises from the following file are presented:

AI_mf_For ALL_2020_PartII.pdf

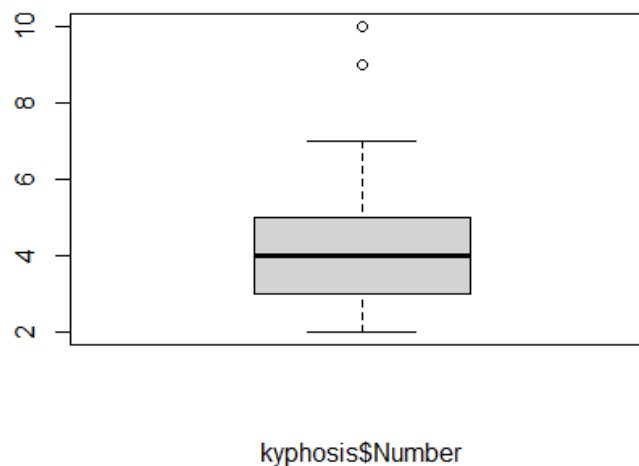
2.1 Exercise 1

We load the package **rpart** and then the dataset **kyphosis** using the following commands.

```
data("kyphosis", package = 'rpart')
```

Next, we create the boxplot, and we find the outliers along with the corresponding indices, as follows:

```
values <- boxplot(kyphosis$Number, xlab="kyphosis$Number")
outliers <- values$out
outliers
## [1] 9 10
which(kyphosis$Number %in% outliers)
## [1] 43 53
```



We can also use the **identify** function to identify the outlier's indices. Identify reads the position of the graphics pointer when the (first) mouse button is pressed. It then searches the coordinates given in x and y for the point closest to the pointer. If this point is close enough to the pointer, its index will be returned as part of the value of the call. In our example, we run identify and select the two points at the top of the diagram. Then by clicking Finish, the indices appear on the plot. They are the same as before.

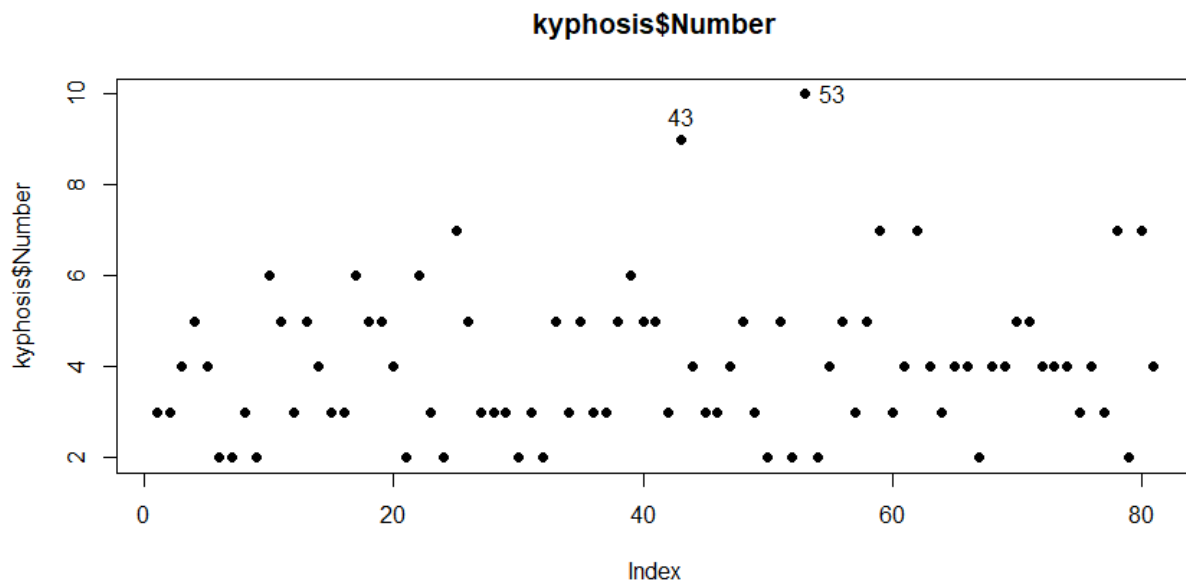
```
x <- seq_along(kyphosis$Number)
y <- kyphosis$Number

plot(x, y, main="kyphosis$Number",
     xlab="Index", ylab="kyphosis$Number", pch=19)

identify(x, y, plot=TRUE)

kyphosis$Number[43]
## [1] 9

kyphosis$Number[53]
## [1] 10
```



2.2 Exercise 2

We load the dataset **capital.csv** and create the plots of Balance per Gender. To be able to express their relationship, we compute the relative frequency table and then the corresponding bar charts. We split the variable balance to the following three levels:

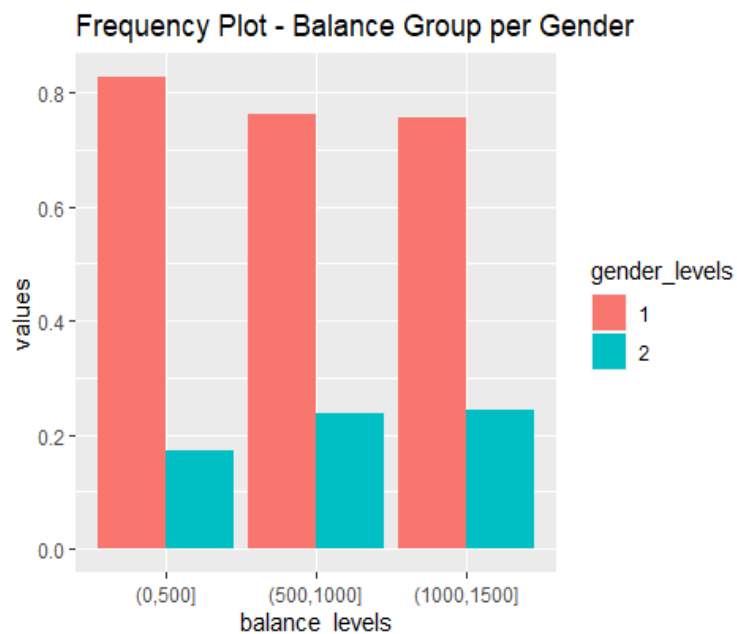
- (0, 500]
- (500,1000]
- (1000,1500]

```
balance <- Capital$balance
bins <- seq(0,1500,by=500)
balance_bins <- cut(balance, bins,dig.lab=10)

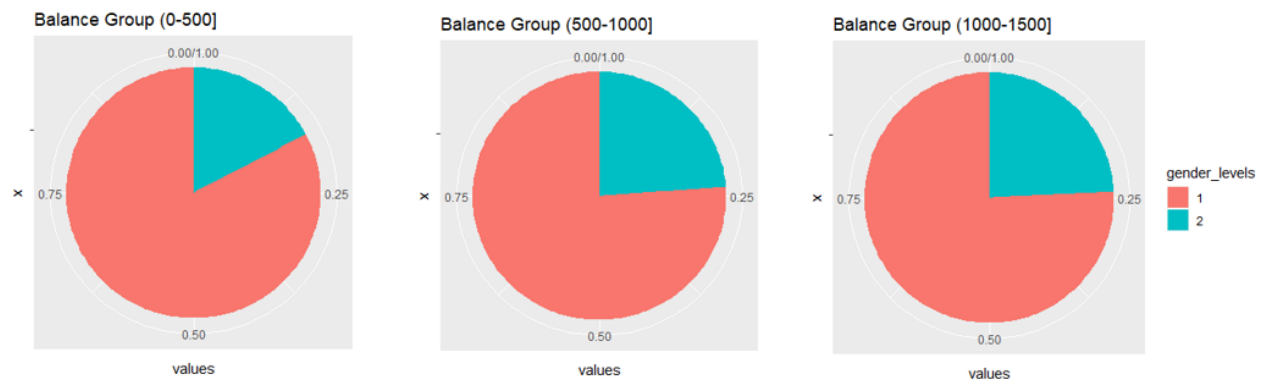
balance_gender_freq <- table(balance_bins, Capital$gender, dnn = c("cBalance",
" cGender"))

balance_gender_relfreq <- prop.table(balance_gender_freq,1)
balance_gender_relfreq
```

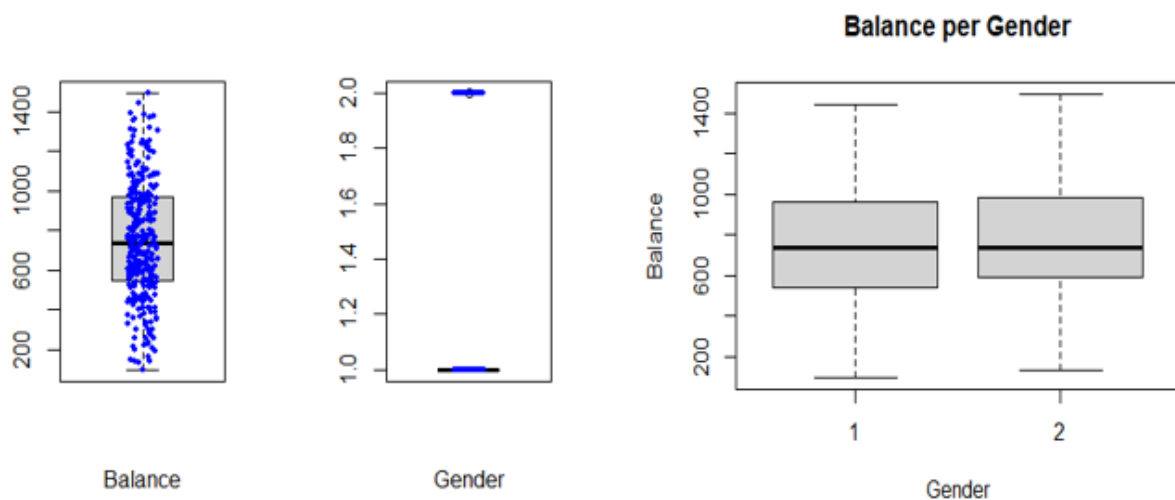
##	cBalance	cGender	1	2
##	(0,500]		0.8275862	0.1724138
##	(500,1000]		0.7613636	0.2386364
##	(1000,1500]		0.7575758	0.2424242



Next, by using the relative frequency table, we create one pie chart per level of balance per gender:



We continue the Exploratory Data Analysis of our dataset by creating boxplots for each variable and a boxplot of balance per gender.



We compute the summary of the dataset (mean, standard deviation etc.)

```
# Summary of datasets
summary(Capital)
```

```
##      balance      gender
## Min.   : 99.0   Min.   :1.000
## 1st Qu.: 550.8   1st Qu.:1.000
## Median : 737.0   Median :1.000
## Mean   : 753.7   Mean    :1.227
## 3rd Qu.: 964.2   3rd Qu.:1.000
## Max.   :1493.0   Max.    :2.000

sd(Capital$balance)

## [1] 294.4837

sd(Capital$gender)

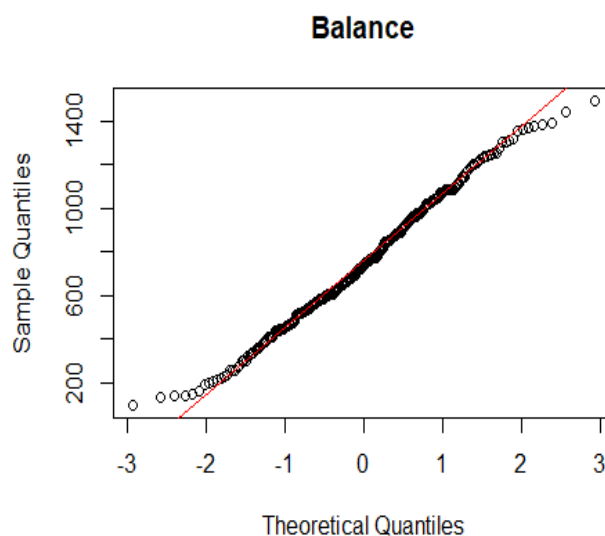
## [1] 0.4193747
```

Finally, we perform a normality test for the balance variable. We use the command **qqnorm** and then we perform a Shapiro-Wilk normality test. By looking at the plot, we observe that it is possible that the variable follows a normal distribution.

We confirm our assumption with the Shapiro test, as the p-value $0.089 > 0.05$, so the distribution of the data is not significantly different from normal distribution. We can assume normality.

```
# Check normal distribution
qqnorm(Capital$balance, main="Weight")
qqline(Capital$balance, col="red")

shapiro.test(Capital$balance)
## Shapiro-Wilk normality test
##
## data: Capital$balance
## W = 0.99167, p-value = 0.08909
```



2.3 Exercise 3

In this exercise, we will check a company's statement, in significance level 0.05. We construct the following hypothesis:

- $H_0 : \mu = 10000$
- $H_1 : \mu > 10000$

The problem's parameters are the following:

```
mu <- 10000
sdev <- 120
n <- 30
x_sample <- 9900
a <- 0.05

Z <- ((x_sample - mu)*sqrt(n))/sdev
Z

## [1] -4.564355

Z_a <- qnorm(1-a)
Z_a

## [1] 1.644854

Z > Z_a

## [1] FALSE
```

We cannot reject the H_0 hypothesis, so we reject the company's statement.

2.4 Exercise 4

In this exercise we analyzed the data-set **mtcars**. The mtcars data-set is a build-in data-set in R. We were called to find the confidence level with a factor of 0.95 for the difference of mean values corresponding to fuel consumption with manual gearing vs fuel consumption with automatic gearing.

At first, we load the data-set mtcars. Then, we view the column names and the first few rows to get an idea on how the data-set looks like. The summary function gives us some basic descriptives of the data (min value, max value, median, mean, 1st Qu, 3rd Qu).

```

# read data
mydata <- mtcars

# view what is on the data frame
names(mydata)

## [1] "mpg" "cyl" "disp" "hp" "drat" "wt" "qsec" "vs" "am" "gear"
## [11] "carb"

# view the first few rows of the data
head(mydata)

##           mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4      21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag  21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710      22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive  21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant         18.1   6  225 105 2.76 3.460 20.22  1  0    3    1

```

As a next step, we divide the data-set depending on if the car is automatic or has manual gearing and then we take the miles per gallon column. In the end, we have two vectors, the miles per gallon for automatic gearing and the miles per gallon for manual gearing. Then, we perform the t.test on these two vectors.

```

# miles per gallon for manual gearing
x <- mydata[ which(mydata$am == 0),]
x <- x$mpg
x

## [1] 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4 10.4 14.7
## [16] 15.5 15.2 13.3 19.2

# miles per gallon for automatic gearing
y <- mydata[ which(mydata$am == 1),]
y <- y$mpg
y

## [1] 21.0 21.0 22.8 32.4 30.4 33.9 27.3 26.0 30.4 15.8 19.7 15.0 21.4

# t.test function
# miles per gallon for automatic vs miles per gallon for manual gearing
t.test(x,y)

##
## Welch Two Sample t-test
##
## data: x and y

```

```
## t = -3.7671, df = 18.332, p-value = 0.001374
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -11.280194  -3.209684
## sample estimates:
## mean of x mean of y
##  17.14737  24.39231
```

The p-value=0.001374 means that the data is statistically significantly associated. The 95% confidence interval reported back by t.test function is:

-11.280194 -3.209684

2.5 Exercise 5

We load the dataset **OctopusF.txt** and compute the summary(mean, standard deviation etc.). Then we plot the histogram. The commands used are:

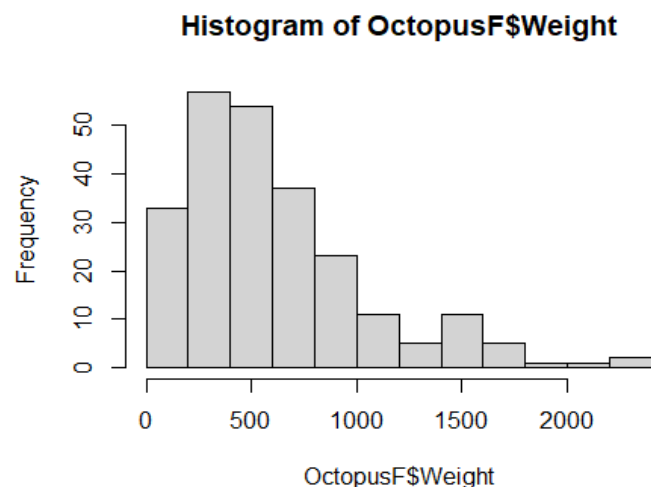
```
summary(OctopusF)

##      Weight
## Min.   : 40.0
## 1st Qu.: 300.0
## Median : 545.0
## Mean   : 639.6
## 3rd Qu.: 800.0
## Max.   :2400.0

sd(OctopusF$Weight)

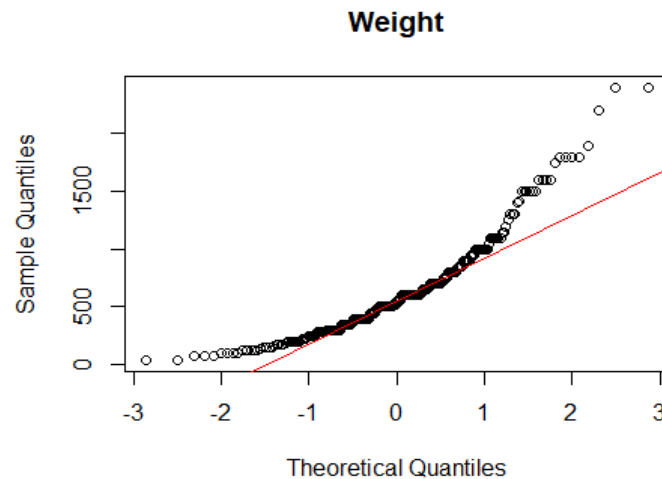
## [1] 445.8965

hist(OctopusF$Weight)
```



We then check if the dataset follows the normal distribution using qqplot and Shapiro test:

```
# Check normal distribution
qqnorm(OctopusF$Weight, main="Weight")
qqline(OctopusF$Weight, col="red")
```



We observe that the variable Weight is not likely to follow the normal distribution. We confirm our observation from the Shapiro test:

```
shapiro.test(OctopusF$Weight)

##
##  Shapiro-Wilk normality test
##
## data:  OctopusF$Weight
## W = 0.88612, p-value = 1.863e-12 < 0.05 We cannot assume normality.
```

2.6 Exercise 6

In this exercise we analyzed a data-set which consists of hair color among men and women. Then we are called to check if the hair color is independent from the gender for significance level of 0.05. The null hypothesis H_0 is "The hair color is independent from the gender" and we check for a significance level of 0.05.

First, we create the data vectors and then put all the vectors in a data-frame. Then, we view the column names and the first few rows to get an idea on how the data-set looks like. The summary function gives us some basic descriptives of the data (min value, max value, median, mean, 1st Qu, 3rd Qu).

```

# create vectors
gender <- c("Agoria", "Koritsia")
fair <- c(592, 544)
red <- c(119, 97)
medium <- c(849, 677)
dark <- c(504, 451)
jetblack <- c(36, 14)

# create data frame
mydata <- data.frame(gender, fair, red, medium, dark, jetblack)

# view what is on the data frame
names(mydata)

## [1] "gender"    "fair"      "red"       "medium"    "dark"      "jetblack"

# view the first few rows of the data
head(mydata)

##      gender fair red medium dark jetblack
## 1   Agoria 592 119   849  504     36
## 2 Koritsia 544  97   677  451     14

```

After that, we exclude the column gender because it is non-numeric. Then we convert the data as a table and create a balloon plot to visualize the table data.

```

# exclude gender column
mydata <- mydata[, -c(1)]











# view the first few rows of the data again
head(mydata)

##    fair red medium dark jetblack
## 1  592 119   849  504     36
## 2  544  97   677  451     14

# convert the data as a table
dt <- as.table(as.matrix(mydata))
# Graph the table
balloonplot(t(dt), main = "data table visualization", xlab = "hair color", ylab = "gender",
            label = FALSE, show.margins = FALSE)

```

data table visualization

		hair color				
		fair	red	medium	dark	jetblack
gender	A					
	B					

Furthermore, we implement the chi-squared test to check the significance association between the rows and the columns.

```
# Chi-square statistic test
# check significance association between rows and columns
chisq <- chisq.test(mydata)
chisq

##
##  Pearson's Chi-squared test
##
## data:  mydata
## X-squared = 10.467, df = 4, p-value = 0.03325

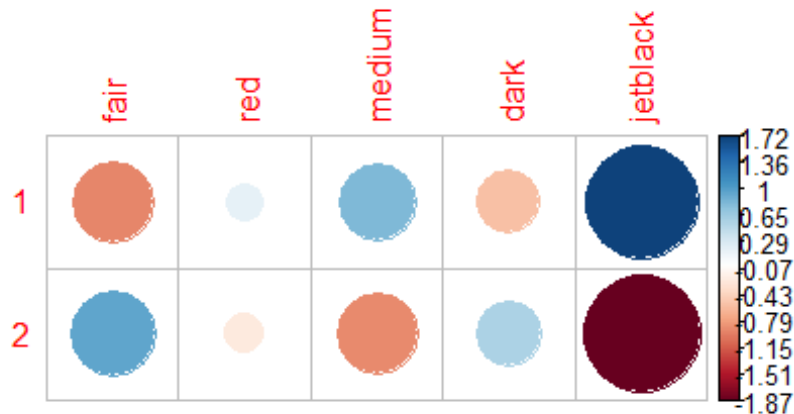
chisq$observed

##      fair red medium dark jetblack
## [1,] 592 119   849  504     36
## [2,] 544  97   677  451     14

round(chisq$expected,2)

##      fair    red medium  dark jetblack
## [1,] 614.37 116.82 825.29 516.48   27.04
## [2,] 521.63  99.18 700.71 438.52   22.96

# view Pearson residuals
corrplot(chisq$residuals, is.cor = FALSE)
```



The chi-squared test reported back a p-value = 0.03325. This means that the row and the column variables are statistically significantly associated. The significance level for this analysis is 0.05. Therefore, we reject the null hypothesis H_0 "The hair color is independent from the gender".

2.7 Exercise 7.1

We load the dataset `Concrete_Data.xls`. We will construct a neural network using the package **neuralnet**. We load the dataset and then perform normalization. Then we create the train and test datasets by splitting the initial dataset. We choose 75% for the train dataset.

```
preproc1 <- preProcess(data, method=c("range"))
norm_data <- predict(preproc1, data)
summary(norm_data)

## 75% of the sample size
sample_size <- floor(0.75 * nrow(norm_data))

set.seed(123)
train_ind <- sample(seq_len(nrow(norm_data)), size = sample_size)

train <- norm_data[train_ind, ]
test <- norm_data[-train_ind, ]
```

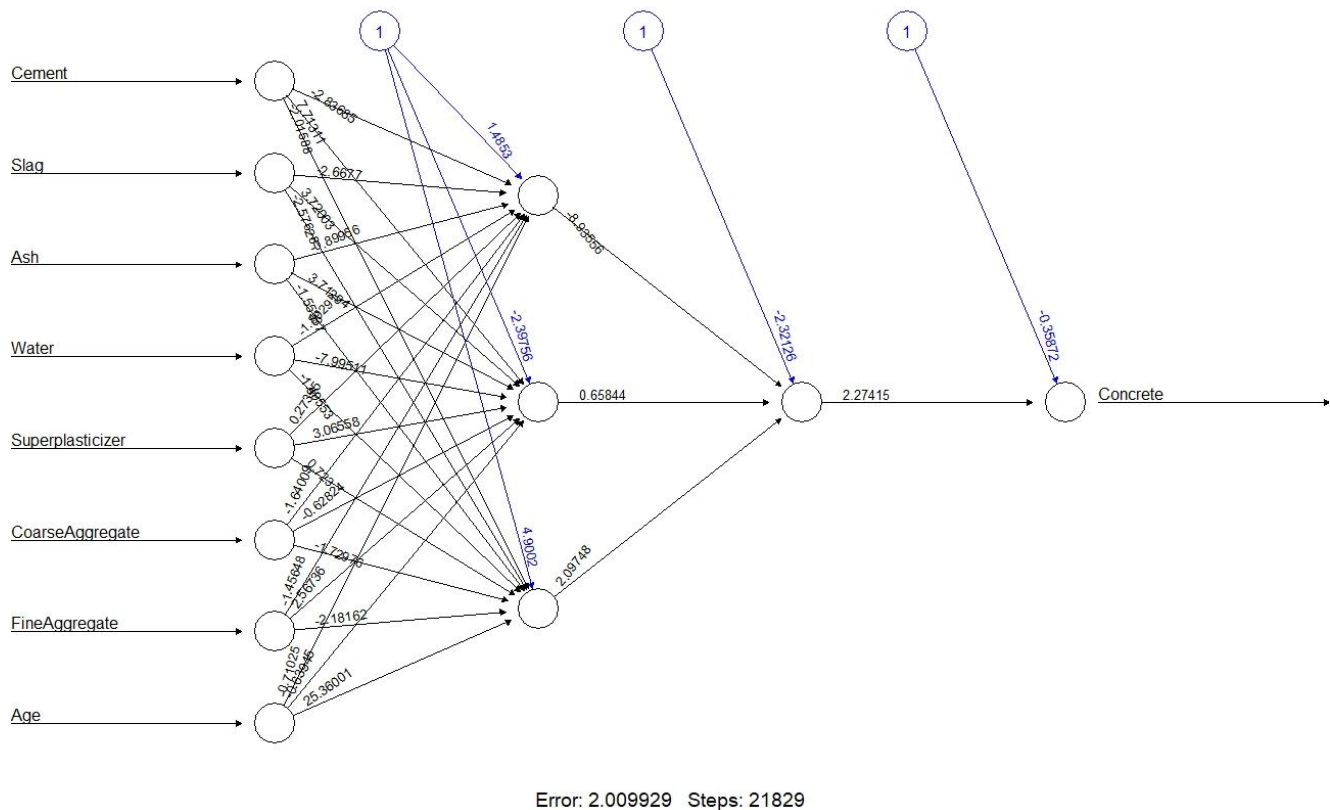
We use the library `neuralnet` to train our model, using all the variables from the dataset. The neural network will have a linear output. The number of hidden layers is (3,1) and the threshold is set to 0.01. If the change in error during an iteration is less than 1%, then no further optimization will be carried out by the model.

```
library(neuralnet)

## Warning: package 'neuralnet' was built under R version 4.0.4

nn <- neuralnet(Concrete ~ Cement + Slag + Ash + Water + Superplasticizer +
  CoarseAggregate + FineAggregate + Age,
  data=train,
  hidden=c(3,1),
  linear.output=TRUE,
  threshold=0.01)
```

We plot the neural network:



If we use compute function, we get the output of each hidden layer as well as the final output. Predict only returns the final output. To evaluate the model, we use the RMSE metric. We observe that RMSE is low enough, our model performs pretty well on the test dataset.

```
## Prediction using neural network
Predict <- compute(nn,test)
Predict2 <- predict(nn,test)

RMSE <- (sum((test$Concrete - Predict$net.result)^2) / nrow(test)) ^ 0.5
RMSE

## [1] 0.07796686
```

We also compute the accuracy on the test dataset:

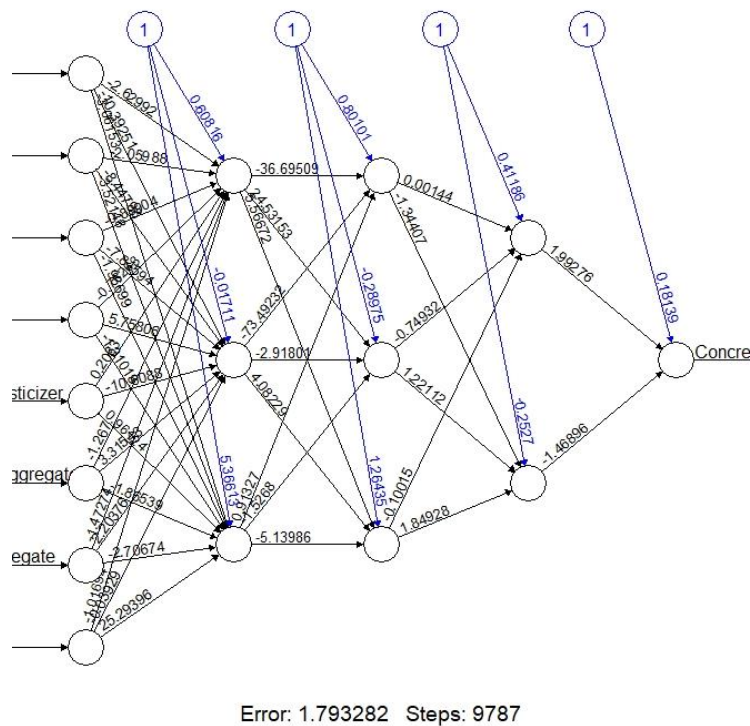
```
deviation=((test$Concrete-Predict$net.result)/test$Concrete)
deviation[which(!is.finite(deviation))] <- 0
accuracy=1-abs(mean(deviation))
accuracy

## [1] 0.9617394
```

Function cor() computes the covariances (or correlations) between the columns of x and the columns of y, of the given matrix. If we apply this function to a dataframe, we get the correlation matrix for all variables. We can omit variables if they have a high correlation, to improve the performance of the model.

Instead, we choose to create a neural network with more hidden layers, as follows:

```
nn2 <- neuralnet(Concrete ~ Cement + Slag + Ash + Water + Superplasticizer +
                  CoarseAggregate + FineAggregate + Age,
                  data=train,
                  hidden=c(3,3,2),
                  linear.output=TRUE,
                  threshold=0.01)
```



The performance of the model with more hidden layers is slightly better in our case. We observe that the error decreased, and the accuracy increased.

2.8 Exercise 7.2

In this exercise we analyzed a data-set "insurance.csv" which includes 1338 samples on variables: age, sex, bmi, children, smoker, region and charges. We are called to investigate how these variables affect the charges variable.

First we read the data with read.csv function and put it in a data-frame. Then, we view the column names and the first few rows to get an idea on how the data-set looks like. The summary function gives us some basic descriptives of the data (min value, max value, median, mean, 1st Qu, 3rd Qu).

```

# read data
mydata <- read.csv("insurance.csv", header=TRUE)

# view what is on the data frame
names(mydata)

## [1] "age"      "sex"      "bmi"      "children" "smoker"   "region"   "charges"

# view the first few rows of the data
head(mydata)

##   age    sex    bmi children smoker   region  charges
## 1  19 female 27.900         0    yes southwest 16884.924
## 2  18  male 33.770         1    no  southeast  1725.552
## 3  28  male 33.000         3    no  southeast  4449.462
## 4  33  male 22.705         0    no northwest 21984.471
## 5  32  male 28.880         0    no northwest  3866.855
## 6  31 female 25.740         0    no  southeast  3756.622

```

Then we make a check of the variables with the “psych” package. We extract the geometric mean and also the harmonic mean for each of the variables.

```

# geometric.mean function
geometric.mean(mydata[c(1,3,4,7)],na.rm=TRUE)

##      age      bmi  children  charges
## 36.49672 30.05250  0.00000 8943.28928

# harmonic.mean function
harmonic.mean(mydata[c(1,3,4,7)],na.rm=TRUE)

##      age      bmi  children  charges
## 33.73384 29.43301  0.00000 5907.39090

```

After that, we perform the t.test function to check how each numeric variable affects the charges variable.

```

age <- mydata$age
bmi <- mydata$bmi
children <- mydata$children
charges <- mydata$charges

# two sample t-test to check how each variable affects charges
t.test(age,charges)

```

```

##
## Welch Two Sample t-test
##
## data: age and charges
## t = -39.965, df = 1337, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -13880.68 -12581.75
## sample estimates:
## mean of x mean of y
## 39.20703 13270.42227

t.test(bmi,charges)

##
## Welch Two Sample t-test
##
## data: bmi and charges
## t = -39.991, df = 1337, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -13889.23 -12590.29
## sample estimates:
## mean of x mean of y
## 30.6634 13270.4223

t.test(children,charges)

##
## Welch Two Sample t-test
##
## data: children and charges
## t = -40.08, df = 1337, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -13918.80 -12619.86
## sample estimates:
## mean of x mean of y
## 1.094918 13270.422265

```

In every t.test analysis the p-value is 0 which means that all variables are **statistically significantly associated** with each other. Therefore, all variables affect the charges variable.

2.9 Exercise 8

In this exercise we will analyze the data "mf.xls". We are called to implement correlation tables between the variables, check the relationship between the complaint code variable and the manufacturing plant variable for significance level 0.01 and then check the difference between the dollar claim amount variable and the manufacturing plant variable for a significance level of 0.02.

First we read the data with read.csv function and put it in a data-frame. Then, we view the column names and the first few rows to get an idea on how the data-set looks like. The summary function gives us some basic descriptives of the data (min value, max value, median, mean, 1st Qu, 3rd Qu).

```
# read data
mydata <- read_excel("mf.xls")

# view what is on the data frame
names(mydata)

## [1] "Dollar Claim Amount" "Shift"          "Complaint Code"
## [4] "Manufacturing Plant"

# view the first few rows of the data
head(mydata)

## # A tibble: 6 x 4
##   `Dollar Claim Amount` Shift `Complaint Code` `Manufacturing Plant`
##           <dbl> <dbl>           <dbl>           <dbl>
## 1             259     2             2             1
## 2             211     1             2             1
## 3             287     2             3             3
## 4             338     3             2             2
## 5             334     1             2             1
## 6             361     1             1             1

# view basic descriptives of the data
summary(mydata)

## Dollar Claim Amount      Shift      Complaint Code      Manufacturing Plant
## Min.   :146.0      Min.   :1.000      Min.   :1.000      Min.   :1.000
## 1st Qu.:236.5      1st Qu.:1.000      1st Qu.:1.000      1st Qu.:1.000
## Median :270.0      Median :1.000      Median :2.000      Median :1.000
## Mean   :272.4      Mean   :1.364      Mean   :2.018      Mean   :1.364
## 3rd Qu.:309.8      3rd Qu.:2.000      3rd Qu.:3.000      3rd Qu.:2.000
## Max.   :393.0      Max.   :3.000      Max.   :4.000      Max.   :3.000
```

After that, we implement the correlation between the variables in two groups. Group 1 is the correlation between dollar claim amount variable and shift. Group 2 is the correlation between complaint code variable and manufacturing plant.

```
# correlation table of dollar claim amount vs shift
res <- cor(mydata[-c(3,4)])
round(res, 2)

##                Dollar Claim Amount Shift
## Dollar Claim Amount          1.00 -0.02
## Shift                        -0.02  1.00

# correlation table of complaint code vs manufacturing plant
res <- cor(mydata[-c(1,2)])
round(res, 2)

##                Complaint Code Manufacturing Plant
## Complaint Code          1.00                0.21
## Manufacturing Plant      0.21                1.00
```

The correlation coefficient for Group 1 (Dollar claim amount, shift) is -0.02 and shows that there is no linear relationship between the movement of the two variables. The correlation coefficient for Group 2 (Complaint code, manufacturing plant) is 0.21 and shows that there is a positive correlation between two variables, but it is weak and likely unimportant.

As a next step, we calculated the expected values for each variable. We make a null hypothesis H_0 "There is a relation between complaint codes and manufacturing plants" and check for a significance level of 0.01. To do this, we check with the t.test function between complaint code and manufacturing plant variables.

```
# t.test function between complaint code variable and manufacturing plant variable
t.test(mydata$`Complaint Code`, mydata$`Manufacturing Plant`)

##
##  Welch Two Sample t-test
##
## data:  mydata$`Complaint Code` and mydata$`Manufacturing Plant`
## t = 6.3494, df = 194.29, p-value = 1.495e-09
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.4512312 0.8578597
## sample estimates:
## mean of x mean of y
##  2.018182  1.363636
```

The t.test result returned back a p-value=0.000000001. This means that, the variables are statistically significantly associated. The significance level for this analysis is 0.01. Therefore, we reject the null hypothesis H_0 "There is a relation between complaint codes and manufacturing plants". The manager's suspicion is wrong on this matter.

After that, we check the relationship between the dollar claim amount and the manufacturing plants (Boise and Salt Lake city) for a significance level $\alpha=0.02$.

```
# select data based on manufacturing plant observations 1 and 2
data_mod <- mydata[ which(mydata$`Manufacturing Plant` < 3),]
data_mod <- data_mod[c(1,4)]
data_mod

## # A tibble: 102 x 2
##   `Dollar Claim Amount` `Manufacturing Plant`
##           <dbl>           <dbl>
## 1             259             1
## 2             211             1
## 3             338             2
## 4             334             1
## 5             361             1
## 6             165             1
## 7             263             1
## 8             329             1
## 9             220             1
## 10            240             1
## # ... with 92 more rows

# Chi-square statistic test
# check significance association between rows and columns
chisq <- chisq.test(data_mod, simulate.p.value = TRUE)
chisq

##
## Pearson's Chi-squared test with simulated p-value (based on 2000
## replicates)
##
## data:  data_mod
## X-squared = 20.106, df = NA, p-value = 1
```

The chi-squared test returns a p-value=1. This means that, the complaint code and manufacturing plant variables are not statistically significantly associated. That means that, there is no statistical significance to reject the manager's hypothesis about the dollar claim amount and the manufacturing plant.

3 Exercises Part 3

In this part, the exercises from the following file are presented:

AI_MEROS_III_mf2020.pdf

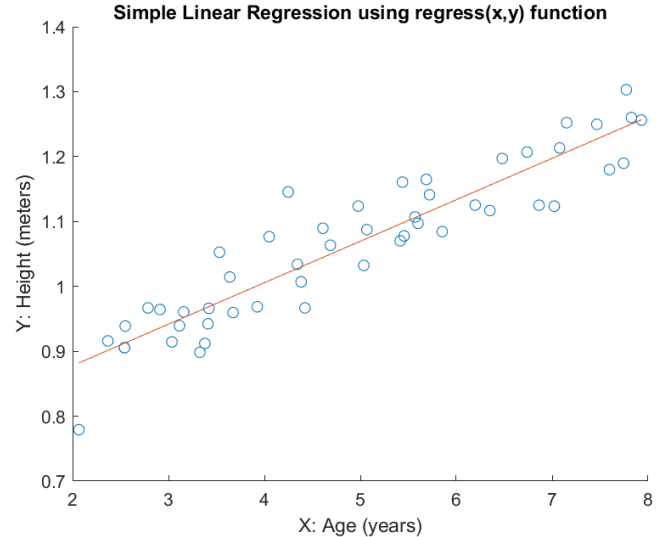
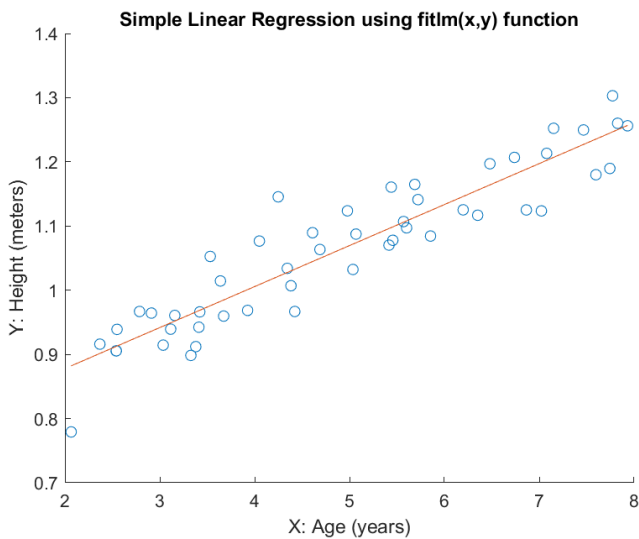
3.1 Exercise 1

In this exercise, we use the file **simple_linear_regression.m**. Firstly, we use the `regress(x, y)` of Statistics and Machine Learning Toolbox.

```
a2 = regress(y,[ones(size(x)) x])
```

We add the ones vector before x variable and run the regress function. The fitted coefficients are the same as with the `fitlm` function:

```
a2 = [0.7502, 0.0639]
```



We observe that the two regression lines are the same as expected.

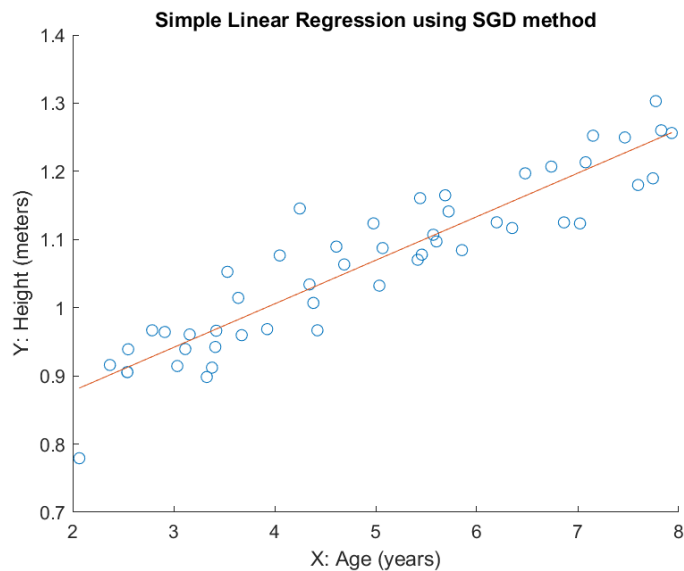
The next step is to implement Gradient Descent from scratch. We implement in Matlab code the gradient descent update condition:

$$\theta_i := \theta_i - a \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})x_j^{(i)}$$

```
% 3. Using Gradient Descent
theta = zeros(2, 1);
iterations = 1500;
alpha = 0.07;

x2 = [ones(size(x)) x];

for num_iterations = 1:iterations
    grad = (alpha/m)*sum((x2*theta) - y).*x2);
    theta = theta - grad';
end
```



The coefficients are again the same as the previous cases and as a result the regression line also.

Multivariate Linear Regression

We use the following file: **multivariate_linear_regression.m**. The dataset consists of two variables. We implement the Gradient Descent update as follows:

```
for num_iterations = 1:iters
    % Calculate the J term
    J(num_iterations) = (1/(2*m)) * (x*theta - y)' * (x*theta - y);

    temp = theta;

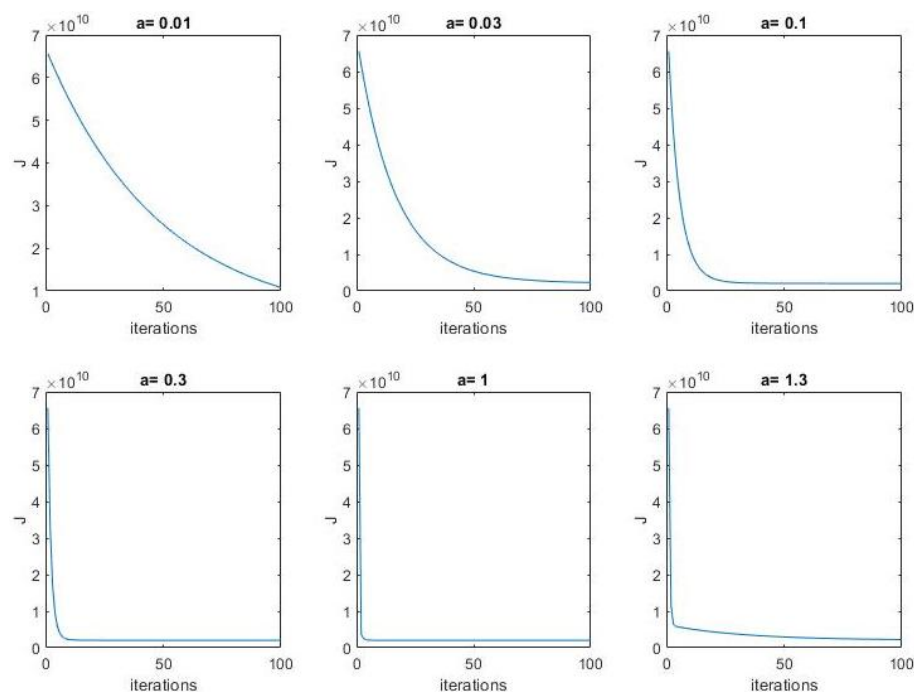
    for j=1:size(x,2)
        temp(j) = theta(j) - (alpha(i)/m)*sum((x*theta) - y).*x(:,j)) ;
    end

    for j=1:size(x,2)
        theta(j) = temp(j) ;
    end
end
```

The theta values are updated simultaneously, after the calculation of the gradient. We also keep the values of the cost function J. We choose the following values for alpha and for iterations:

```
alpha = [0.01, 0.03, 0.1, 0.3, 1, 1.3];
iters = 100;
```

We observe that alpha=1 is the best value in our case because the algorithm converges faster.



3.2 Exercise 2

In the next part, we implement logistic regression to predict if a student gets accepted into the university. We use the file **logistic_regression.m**

The goal is to implement Gradient Descent (same implementation as the previous exercise) and then to use the Newton method to estimate the theta parameters.

Gradient Descent

We implement the function **cost_function_lr** which computes the cost function value and gradient for the given theta values. In this case we observe that gradient is too sensitive to the alpha values and the iterations used.

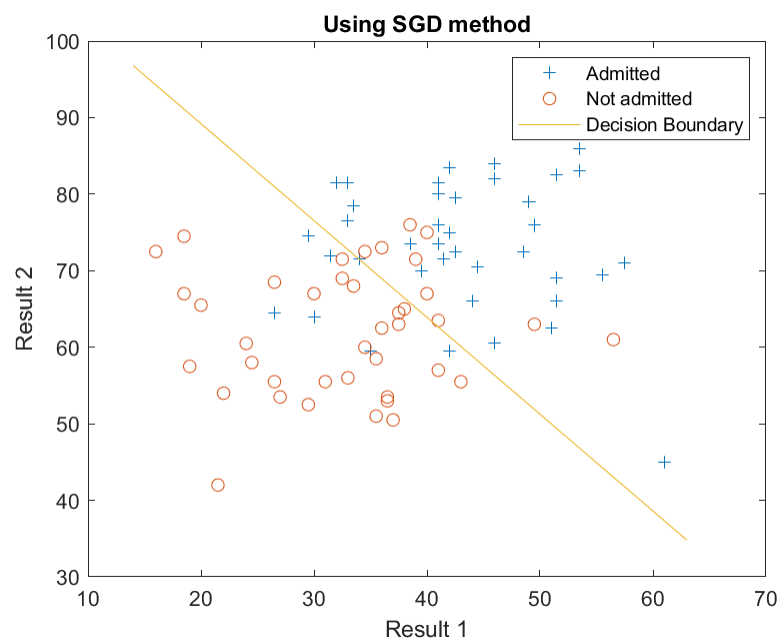
The algorithm converges with the following settings after several tests:

```
iters = 700000;  
alpha = 0.001;
```

The following values for theta are returned, and the decision boundary is the following:

```
theta = [-8.17, 0.09, 0.07]
```

We observe that the algorithm can separate the two classes.



Newton Method

In the next part we implement the Newton method to estimate the theta values. The method is implemented as follows:

```
% 3. Using Newton's method
iters = 15;

% Initialize fitting parameters
theta = zeros(3,1);

% Define the sigmoid function
g = @(z) (1.0 ./ (1.0 + exp (-z)));

theta_old = ones (n + 1, 1);
% Newton's method

for i = 1:iters
    theta_old = theta;
    % Calculate the hypothesis function
    z = x * theta;
    h = g (z);

    % Calculate gradient and hessian.
    grad = x' * (h-y);
    H = x' * diag (h) * diag (1-h) * x;

    % Calculate J (for testing convergence)
    [J(i), ] = cost_function_lr(theta,x,y);

    theta = theta-H\grad;
end
```

The method converges with only 15 iterations and the theta parameters returned are the following:

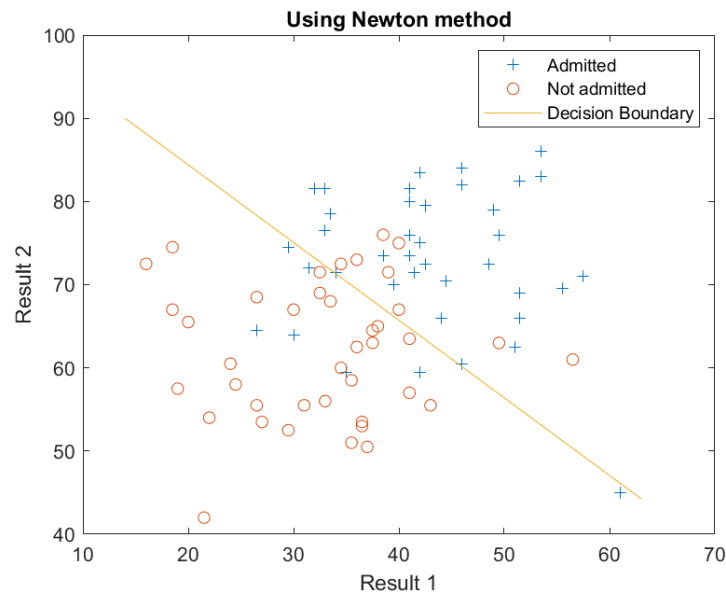
```
theta = [-16.37, 0.14, 0.15]
```

We observe that the theta values are different from the values calculated with Gradient Descent algorithm. Furthermore, the values are the same with the values returned from MATLAB's function fitglm:

```
B = fitglm(x,y,'linear', 'distr', 'binomial' )
coeffs=B.Coefficients.Estimate;

coeffs = [-16.37, 0.14, 0.15]
```

The decision boundary for Newton's method:



3.3 Exercise 3

In this part, we implement k-means, an unsupervised learning algorithm. The following files are used:

- `k_means.m`
- `find_closest_centroids.m`
- `compute_centroids.m`

The way kmeans algorithm works is as follows:

1. Specify number of clusters K .
2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids.
 - Compute the sum of the squared distance between data points and all centroids.
 - Assign each data point to the closest cluster (centroid). `find_closest_centroids`
 - Compute the centroids for the clusters by taking the average of all data points that belong to each cluster. `compute_centroids`

The results are the following for different number of centroids:

