UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Statistical Modelling and Design of Experiments

First assignment : Report

Submitted by :

- ***Henry Sparrow***
- ***Adrian Gruber***
- ***Othman Benmoussa***

First Assignment SMDE

Contents

1 Question 1

1.1 First question

```
library(FactoMineR)
data(decathlon)
head(decathlon)
```

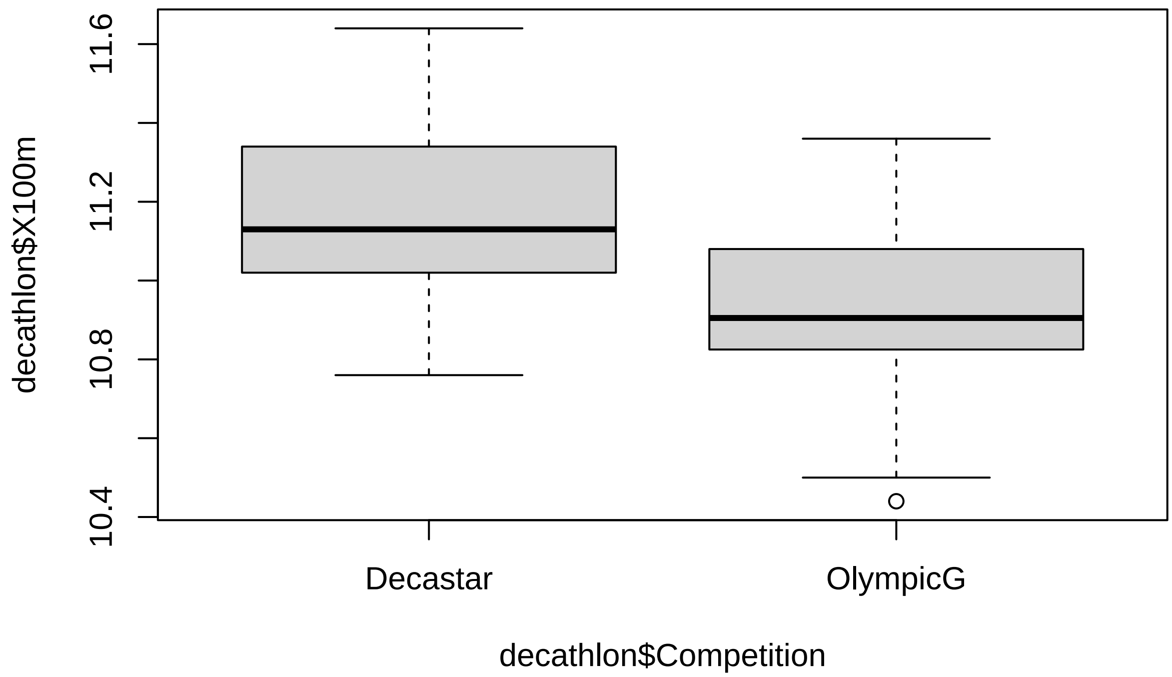
```
##          100m Long.jump Shot.put High.jump  400m 110m.hurdle Discus Pole.vault
## SEBRLE  11.04      7.58    14.83      2.07 49.81       14.69  43.75       5.02
## CLAY    10.76      7.40    14.26      1.86 49.37       14.05  50.72       4.92
## KARPOV  11.02      7.30    14.77      2.04 48.37       14.09  48.95       4.92
## BERNARD 11.02      7.23    14.25      1.92 48.93       14.99  40.87       5.32
## YURKOV  11.34      7.09    15.19      2.10 50.42       15.31  46.26       4.72
## WARNERS 11.11      7.60    14.31      1.98 48.68       14.23  41.10       4.92
##         Javeline 1500m Rank Points Competition
## SEBRLE     63.19 291.7    1   8217    Decastar
## CLAY       60.15 301.5    2   8122    Decastar
```

```
## KARPOV     50.31 300.2    3    8099     Decastar
## BERNARD    62.77 280.1    4    8067     Decastar
## YURKOV     63.44 276.4    5    8036     Decastar
## WARNERS    51.77 278.1    6    8030     Decastar
```

```
summary(decathlon)
```

```
##       100m          Long.jump       Shot.put       High.jump          400m      
##  Min.   :10.44   Min.   :6.61   Min.   :12.68   Min.   :1.850   Min.   :46.81  
##  1st Qu.:10.85   1st Qu.:7.03   1st Qu.:13.88   1st Qu.:1.920   1st Qu.:48.93  
##  Median :10.98   Median :7.30   Median :14.57   Median :1.950   Median :49.40  
##  Mean   :11.00   Mean   :7.26   Mean   :14.48   Mean   :1.977   Mean   :49.62  
##  3rd Qu.:11.14   3rd Qu.:7.48   3rd Qu.:14.97   3rd Qu.:2.040   3rd Qu.:50.30  
##  Max.   :11.64   Max.   :7.96   Max.   :16.36   Max.   :2.150   Max.   :53.20  
##   110m.hurdle        Discus        Pole.vault       Javeline    
##  Min.   :13.97   Min.   :37.92   Min.   :4.200   Min.   :50.31  
##  1st Qu.:14.21   1st Qu.:41.90   1st Qu.:4.500   1st Qu.:55.27  
##  Median :14.48   Median :44.41   Median :4.800   Median :58.36  
##  Mean   :14.61   Mean   :44.33   Mean   :4.762   Mean   :58.32  
##  3rd Qu.:14.98   3rd Qu.:46.07   3rd Qu.:4.920   3rd Qu.:60.89  
##  Max.   :15.67   Max.   :51.65   Max.   :5.400   Max.   :70.52  
##      1500m            Rank           Points       Competition
##  Min.   :262.1   Min.   : 1.00   Min.   :7313   Decastar:13  
##  1st Qu.:271.0   1st Qu.: 6.00   1st Qu.:7802   OlympicG:28  
##  Median :278.1   Median :11.00   Median :8021                
##  Mean   :279.0   Mean   :12.12   Mean   :8005                
##  3rd Qu.:285.1   3rd Qu.:18.00   3rd Qu.:8122                
##  Max.   :317.0   Max.   :28.00   Max.   :8893                
```

```
colnames(decathlon)[c(1,5,6,10)]<-c("X100m","X400m","X110m.hurdle","X1500m")
boxplot(decathlon$X100m ~ decathlon$Competition)
```



Here, we are comparing 100m times from the Decastar (the World Championships of Decathlon) and the Olympic Games 100m event. From this boxplot, we notice that the time needed to run 100m is higher for the Decastar competition than for the Olympics. Every metric (median, maximum, minimum, quartiles) is higher for the Decastar which means that the Olympic 100m sprinters are faster than the Decastar sprinters. This makes sense since, most 100m sprinters in the Olympics are 'specialized' in sprinting and will devote all their training time to sprinting whereas the decathlete must train for 9 other disciplines - some of which do not involve sprinting (example: Javelin).

1.2 Second question

```
decathlon$x100m_cat <- cut(decathlon$X100m,c(0,11,13), labels=c('<= 11s','> 11s'))
pt <- table(decathlon$x100m_cat,decathlon$Competition)
prop.table(pt)
```

```
##
##              Decastar   OlympicG
##   <= 11s 0.04878049 0.46341463
##   > 11s  0.26829268 0.21951220
```

```
row_marg <- margin.table(pt,1)
col_marg <- margin.table(pt,2)
sweep(pt,1,row_marg,'/')
```

```
##
```

```
##           Decastar  OlympicG
##   <= 11s 0.0952381 0.9047619
##   > 11s  0.5500000 0.4500000
```

```
chisq.test(pt)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  pt
## X-squared = 7.7962, df = 1, p-value = 0.005236
```

This in mind, we see that the time seems to have a lot of influence on the likelihood of being either at the Olympics or the Decastar. When the time is under 11s, there is a very strong chance that we at at the Olympics. When it's above 11s, it is a more evenly distributed: it is a slower time so the likelihood of being in the Decastar (where times are on average less impressive as demonstrated previously) is higher than before and is a now more or less the same as the one for the Olympics. Either way, the trend is not at all the same as the one observed for '<= 11s', which is a strong indication that the two variables are not independent. We confirm this by executing a Chi-Squared test, for which the null hypothesis is H0 (The variables "x100m_cat" (representing whether the clocked time is <= 11s or not) and the variable "Competition" (indicating in which event the time was clocked (Olympics or Decastar)) are independent.)

```
chisq.test(pt)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  pt
## X-squared = 7.7962, df = 1, p-value = 0.005236
```

After the Chi-squared test (results below), we get a value of p = 0.005236 which is inferior to 0.05. Therefore, we can reject H0, i.e.: "x100m_cat" and "Competition" are not independent.

1.3 Third question

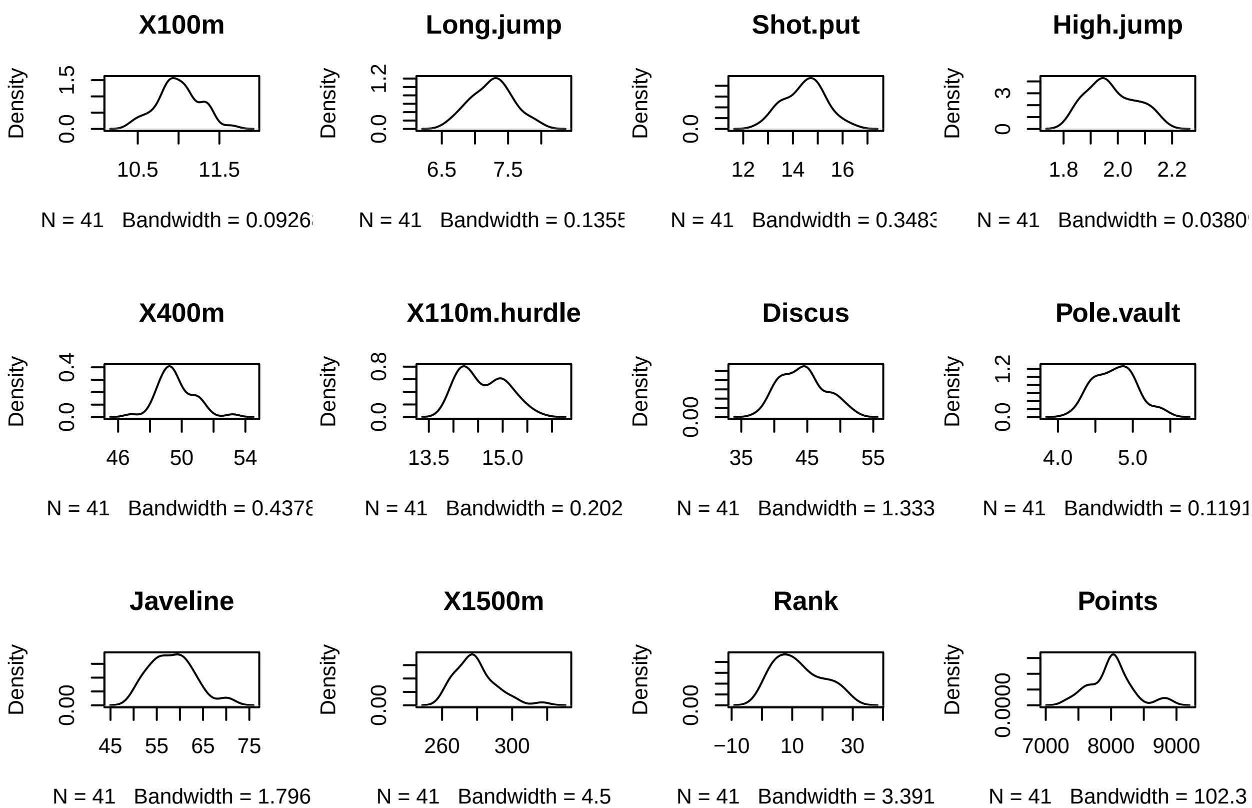
```
par(mfrow=c(3,4))
for(i in 1:ncol(decathlon)) {
  if (is.numeric(decathlon[ , i])) {
    print(colnames(decathlon)[i])
    plot(density(decathlon[ , i]), main = colnames(decathlon)[i] )
  }
}
```

```
## [1] "X100m"
```

```
## [1] "Long.jump"
```

```
## [1] "Shot.put"
```

```
## [1] "High.jump"

## [1] "X400m"

## [1] "X110m.hurdle"

## [1] "Discus"

## [1] "Pole.vault"

## [1] "Javeline"

## [1] "X1500m"

## [1] "Rank"

## [1] "Points"
```



Purely from shape observation, it is quite difficult to tell which variables follow a normal distribution. The allure of most of the distributions seems to be that of a normal distribution. However, we could hypothesise that: - "x1500m", "Rank", "Points" and "Javeline" do not since their distributions seem right skewed. - "x110m.hurdle" neither as its distribution seems to have two peaks.

We shall now confirm or deny these initial observations using a Shapiro test on each variable. For the Shapiro test, we have that the null hypothesis H0 is: "The variable being tested follows a normal distribution". Here is a table containing the results of these tests:

```
df_1 <- data.frame(matrix(ncol = 2, nrow = 12))
colnames(df_1)<-c("Variable","p-value")
rownames(df_1)<-colnames(decathlon)[c(1:12)]
df_1[1,]<-shapiro.test(decathlon$X100)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, 1, , value = structure(list(statistic = c(W
## = 0.981802994808368), : provided 4 variables to replace 2 variables
```

```
df_1[2,]<-shapiro.test(decathlon$Long.jump)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, 2, , value = structure(list(statistic = c(W
## = 0.987630123155191), : provided 4 variables to replace 2 variables
```

```
df_1[3,]<-shapiro.test(decathlon$Shot.put)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, 3, , value = structure(list(statistic = c(W
## = 0.988403679016286), : provided 4 variables to replace 2 variables
```

```
df_1[4,]<-shapiro.test(decathlon$High.jump)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, 4, , value = structure(list(statistic = c(W
## = 0.937342108573958), : provided 4 variables to replace 2 variables
```

```
df_1[5,]<-shapiro.test(decathlon$X400m)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, 5, , value = structure(list(statistic = c(W
## = 0.957138655047241), : provided 4 variables to replace 2 variables
```

```
df_1[6,]<-shapiro.test(decathlon$X110m.hurdle)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, 6, , value = structure(list(statistic = c(W
## = 0.930874272753335), : provided 4 variables to replace 2 variables
```

```
df_1[7,]<-shapiro.test(decathlon$Discus)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, 7, , value = structure(list(statistic = c(W
## = 0.969754995679605), : provided 4 variables to replace 2 variables
```

```
df_1[8,]<-shapiro.test(decathlon$Pole.vault)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, 8, , value = structure(list(statistic = c(W
## = 0.970030041574783), : provided 4 variables to replace 2 variables
```

```
df_1[9,]<-shapiro.test(decathlon$Javeline)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, 9, , value = structure(list(statistic = c(W
## = 0.971063098051053), : provided 4 variables to replace 2 variables
```

```
df_1[10,]<-shapiro.test(decathlon$X1500m)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, 10, , value = structure(list(statistic =
## c(W = 0.936522940822223), : provided 4 variables to replace 2 variables
```

```
df_1[11,]<-shapiro.test(decathlon$Rank)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, 11, , value = structure(list(statistic =
## c(W = 0.941883344774224), : provided 4 variables to replace 2 variables
```

```
df_1[12,]<-shapiro.test(decathlon$Points)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, 12, , value = structure(list(statistic =
## c(W = 0.955838549376261), : provided 4 variables to replace 2 variables
```

```
df_1
```

```
##                Variable    p-value
## X100m         0.9818030 0.74354650
## Long.jump     0.9876301 0.92892401
## Shot.put      0.9884037 0.94563452
## High.jump     0.9373421 0.02549591
## X400m         0.9571387 0.12481178
## X110m.hurdle  0.9308743 0.01544097
## Discus        0.9697550 0.33852835
## Pole.vault    0.9700300 0.34559926
## Javeline      0.9710631 0.37323396
## X1500m        0.9365229 0.02391230
## Rank          0.9418833 0.03648800
## Points        0.9558385 0.11232760
```

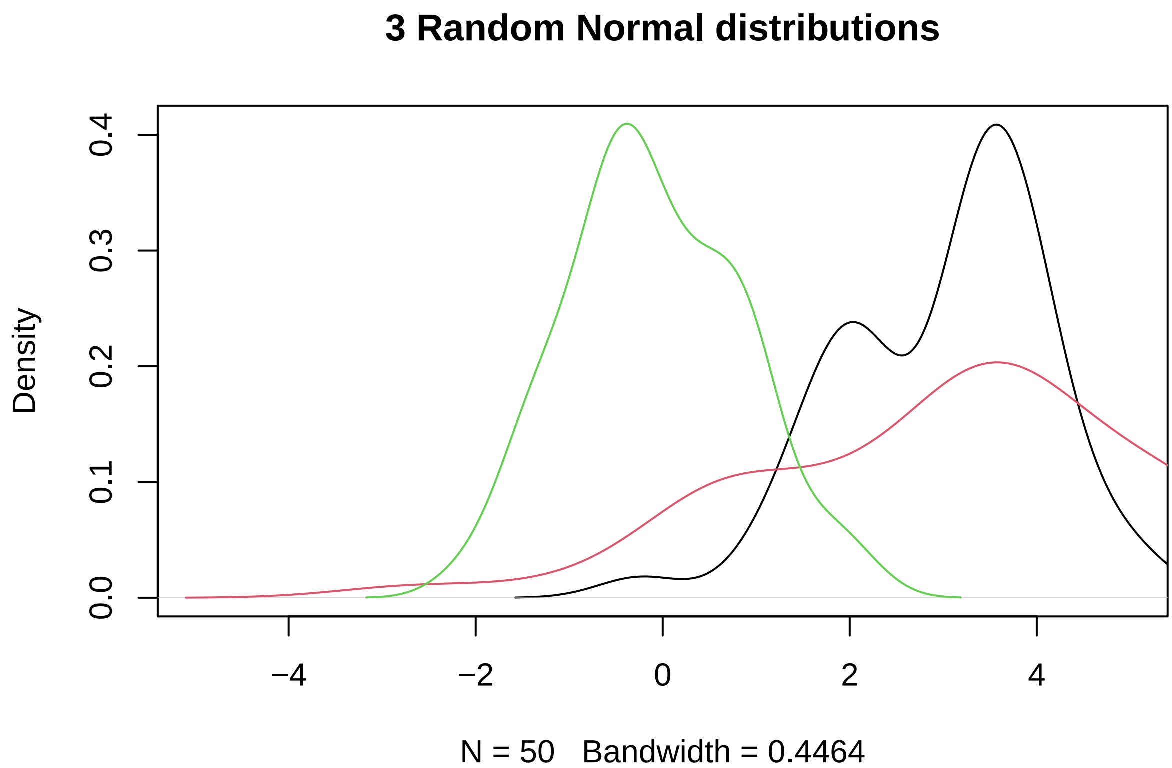
Following these results, we get that the variables that do not follow a normal distribution are "X1500m", "Rank"," X110m.hurdle" (which had all already suspected prior to the test) as well as "High.jump" which we did not originally suspect from the shape of its distribution.

1.4 Fourth question

```
v1=rnorm(50, mean=3, sd=1)
v2=rnorm(50, mean=3, sd=2)
v3=rnorm(50, mean=0, sd=1)
```

Below is a plot of 3 normal distributions such that two of them, v1 (black) and v2 (red), have the same mean (mean = 3), different standard deviations (sd = 1 and sd = 2) while the third one, v3 (yellow), has a different mean (mean = 0) but the same standard deviation with the first distribution (sd = 1):

```
plot(density(v1),xlim=c(-5,5),main="3 Random Normal distributions")
lines(density(v2),col=2)
lines(density(v3),col=3)
```



We will execute a T-test on these variables, for which the null hypothesis H0 is: "The two variables used for the test have the same mean". Before starting, based on the way we have generated the variables, we are expecting to accept the test for the pair (v1,v2), and reject the rest.

```
t.test(v1,v2)
```

```
##
##  Welch Two Sample t-test
##
## data:  v1 and v2
## t = -0.21965, df = 74.782, p-value = 0.8267
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.7210491  0.5778397
## sample estimates:
## mean of x mean of y
##  3.012750  3.084355
```

```
t.test(v2,v3)
```

```
##
##  Welch Two Sample t-test
##
## data:  v2 and v3
## t = 10.03, df = 68.695, p-value = 4.324e-15
## alternative hypothesis: true difference in means is not equal to 0
```

```
## 95 percent confidence interval:
##  2.542235 3.804709
## sample estimates:
##   mean of x   mean of y
##  3.08435489 -0.08911701
```

```
t.test(v1,v3)
```

```
##
##  Welch Two Sample t-test
##
## data:  v1 and v3
## t = 15.34, df = 95.82, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  2.700469 3.503266
## sample estimates:
##   mean of x   mean of y
##  3.01275021 -0.08911701
```

Moreover, the tests provide the actual values of the means. We get that: • mean(v1) = 3.028 • mean(v2) = 3.142 • mean(v3) = -0.084 As expected, we get that the test passes for (v1,v2) and rejects for the rest, meaning the only equality of means happens between the distributions of v1 and v2.

1.5 Fifth question

We execute two T-tests. The first is a T-test on the means the variable x100 split by the competition type, i.e. testing the x100 results obtained in the Olympics against those obtained in the Decastar. When we do this test, we get the following result :

```
t.test(decathlon$X100m ~ decathlon$Competition)
```

```
##
##  Welch Two Sample t-test
##
## data:  decathlon$X100m by decathlon$Competition
## t = 3.2037, df = 22.168, p-value = 0.00407
## alternative hypothesis: true difference in means between group Decastar and group OlympicG is not equ
## 95 percent confidence interval:
##  0.09164794 0.42769272
## sample estimates:
## mean in group Decastar mean in group OlympicG
##               11.17538               10.91571
```

The p-value has a value of 0.004 which is below 0.05 which leads us to reject the null hypothesis ("The mean of the Olympics x100 results is equal to the mean of the Decastar x100 results"), meaning that the means are not the same between the two competitions. The output of the test also informs us that, for the Olympics, the mean has the value 10.92s, whereas for the Decastar we have a mean of 11.18s. We can conclude that, based on the x100 data, the competitions are well and truly distinguishable, with the Olympians being faster.

We repeat the procedure, using this time the x400 data, leading us to the following results:

```
t.test(decathlon$X400m ~ decathlon$Competition)
```

```
##
##  Welch Two Sample t-test
##
## data:  decathlon$X400m by decathlon$Competition
## t = 0.05771, df = 32.106, p-value = 0.9543
## alternative hypothesis: true difference in means between group Decastar and group OlympicG is not eq
## 95 percent confidence interval:
##  -0.6858299  0.7258299
## sample estimates:
## mean in group Decastar mean in group OlympicG
##                  49.63                  49.61
```

```
t.test(decathlon$X400m,decathlon$X100m)
```

```
##
##  Welch Two Sample t-test
##
## data:  decathlon$X400m and decathlon$X100m
## t = 209.02, df = 44.149, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  38.24596 38.99062
## sample estimates:
## mean of x mean of y
##  49.61634  10.99805
```

This time, the p-value very high, being equal to 0.95. We now accept the null hypothesis and conclude that the means of the x400m results in the Olympics and the Decastar are equal (roughly both equal to 49.62s). Therefore, even though with the x100m event, we were able to effectively differentiate the two competitions, this is no longer the case if we take into consideration the x400m data. It seems like performance wise, the Olympians and the Decastart athletes are on a very similar level.

2 Question 2

2.1 First Part of the Q-2

2.1.1 Generating the data and Data treatement

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(rstatix)
```

```
##
## Attaching package: 'rstatix'

## The following object is masked from 'package:stats':
##
##     filter
```

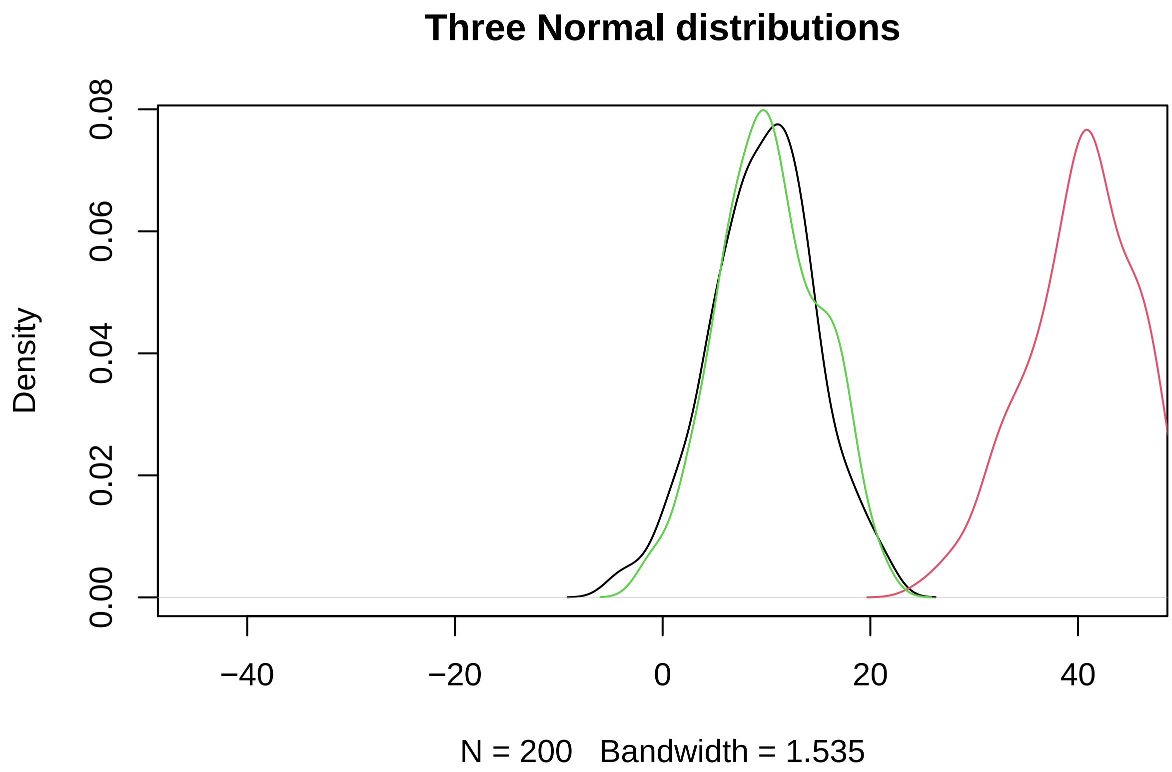
```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v stringr 1.4.0
## v tidyr   1.2.0     v forcats 0.5.1
## v readr   2.1.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x rstatix::filter() masks dplyr::filter(), stats::filter()
## x dplyr::lag()      masks stats::lag()
```

```
library(ggpubr)
library(rstatix)
```

```
v1=rnorm(200, mean=10, sd=5)
v2=rnorm(200, mean=40, sd=5)
v3=rnorm(200, mean=10, sd=5)
```

```
plot(density(v1),xlim=c(-45,45),main="Three Normal distributions")
lines(density(v2),col=2)
lines(density(v3),col=3)
```




```
v1n=data.frame(x1=v1, x2="v1")
v2n=data.frame(x1=v2, x2="v2")
v3n=data.frame(x1=v3, x2="v3")
```

```
library(RcmdrMisc)
```

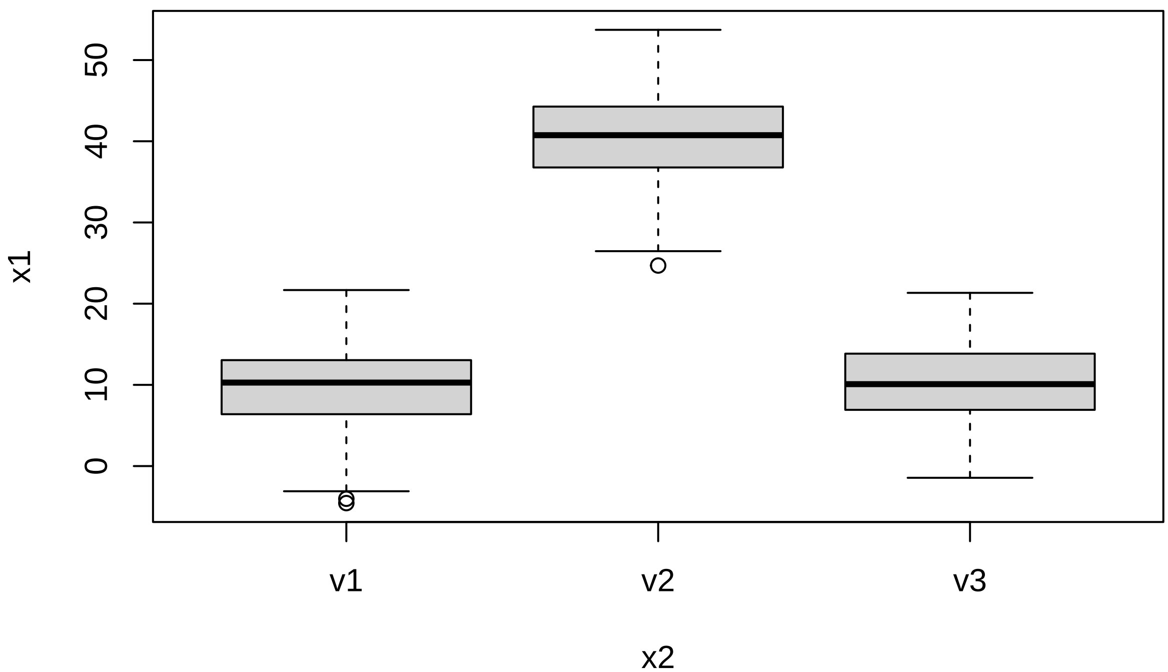
```
## Loading required package: car

## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:purrr':
##
##     some

## The following object is masked from 'package:dplyr':
##
##     recode

## Loading required package: sandwich
```

```
data=mergeRows(v1n, v2n, common.only=FALSE)
data=mergeRows(as.data.frame(data), v3n, common.only=FALSE)
head(data)
```

```
##          x1 x2
## 1  8.265924 v1
## 2  7.754375 v1
## 3 20.035471 v1
## 4 10.823744 v1
## 5  4.018235 v1
## 6 16.674774 v1
```

```
Boxplot(x1~x2,data=data,id=FALSE)
```

```
## Warning in Boxplot.default(mf[[response]], x, id = list(method = id.method, :
## NAs introduced by coercion
```



Assumptions of ANOVA

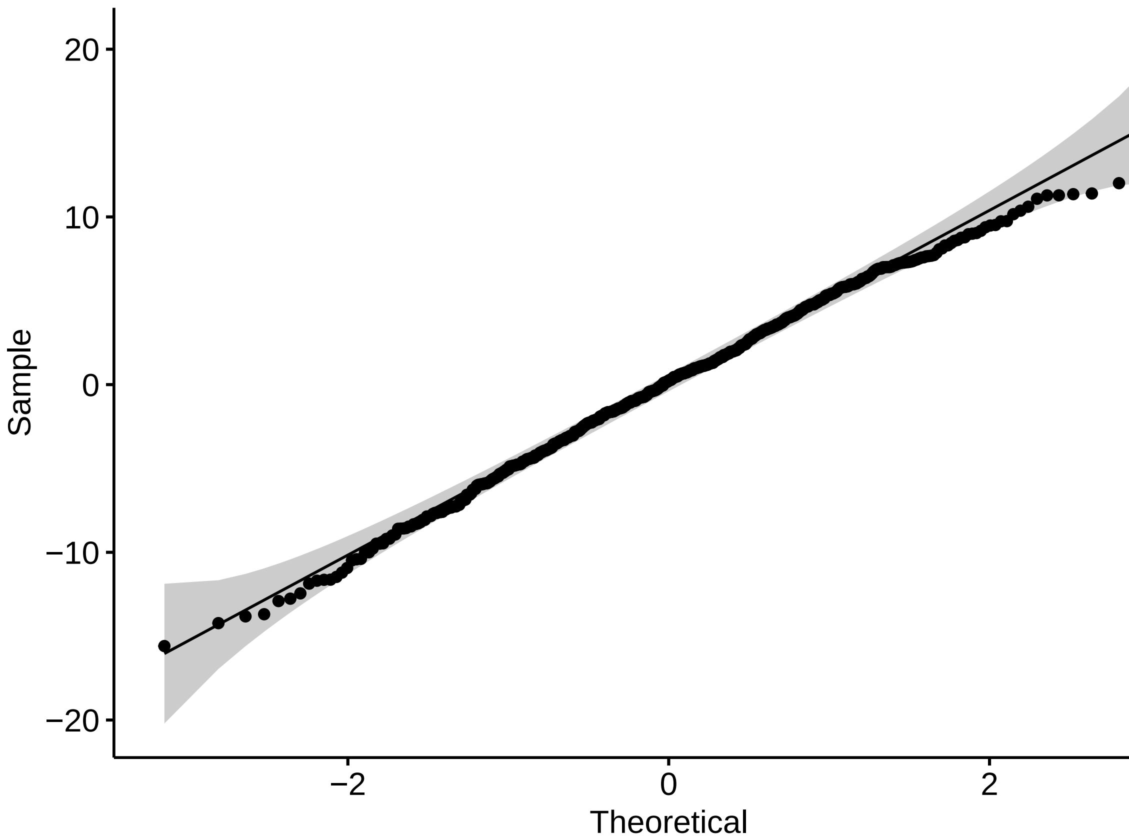
```
data %>%
  group_by(x2) %>%
  identify_outliers(x1)
```

2.1.1.1 Identifying outliers

```
## # A tibble: 3 x 4
##   x2       x1 is.outlier is.extreme
##   <chr> <dbl> <lgl>      <lgl>
## 1 v1    -4.03 TRUE       FALSE
## 2 v1    -4.56 TRUE       FALSE
## 3 v2    24.7  TRUE       FALSE
```

There are too few outliers, we will consider that they don't have an impact on this anova treatment

```
#implementing the linear model
model  <- lm(x1 ~ x2,
             data = data)
# Creating a QQ plot of the residuals
ggqqplot(residuals(model))
```



2.1.1.2 Shapiro test

```
#The populations from which the samples are selected must be normal.
#Shapiro test
shapiro.test(residuals(model))
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  residuals(model)
## W = 0.99593, p-value = 0.1231
```

The shapiro test is respected as the p-value is greater than 0.05, the condition is satisfied

```
data %>%
  levene_test(x1 ~ x2)
```

2.1.1.3 The homogeneity of variances

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.
```

```
## # A tibble: 1 x 4
##     df1   df2 statistic     p
##   <int> <int>     <dbl> <dbl>
## 1     2   597     0.498 0.608
```

The p-value is greater than 0.05, the condition is satisfied

```
#data(data=data, package="FactoMineR")

pairwise.t.test(data$x1, data$x2, p.adj="none")
```

2.1.1.4 Pairwise t test without correction

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  data$x1 and data$x2
##
##    v1     v2
## v2 <2e-16 -
## v3 0.25   <2e-16
##
## P value adjustment method: none
```

We see that v1 and v2 have the same mean as the p-value is greater than 0.05, the couples (v1 and v3) and (v2 and v3) don't have the same mean

2.1.2 ANOVA assumptions test

```
res.aov <- data %>% anova_test(x1 ~ x2)
```

```
## Coefficient covariances computed by hccm()
```

```
res.aov
```

```
## ANOVA Table (type II tests)
##
##   Effect DFn DFd        F         p p<.05   ges
## 1     x2   2 597 2367.037 1.49e-284     * 0.888
```

the p value is extremely small, thee H0 hypothesis is not respected and the mean depends on the category

2.1.3 Post hoc test

```
pwc <- data %>% tukey_hsd(x1 ~ x2)
pwc
```

```
## # A tibble: 3 x 9
##   term  group1 group2 null.value estimate conf.low conf.high    p.adj
## * <chr> <chr>  <chr>       <dbl>    <dbl>    <dbl>     <dbl>    <dbl>
## 1 x2    v1     v2              0   30.6     29.4       31.8  4.65e-10
## 2 x2    v1     v3              0    0.584   -0.612      1.78 4.85e- 1
## 3 x2    v2     v3              0  -30.0    -31.2      -28.8  4.65e-10
## # ... with 1 more variable: p.adj.signif <chr>
```

The difference between (v2 and v3) and (v1 and v3) is the most significant difference as the p-value is a lot less than 0.05 and the H0 hypothesis is not respected

2.2 Treatement of the diabete data

```
library(readr)
diabetes <- read_csv("~/Desktop/diabetes.csv")
```

```
## Rows: 768 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (9): Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, D...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
sel<-which(diabetes$Age<30)
sel
```

```
##   [1]   4   7   8  21  24  28  32  33  34  39  41  46  47  48  50  51  52  56
##  [19]  60  61  64  69  70  71  72  74  75  76  78  79  80  81  82  84  86  88
##  [37]  90  91  95  97  98  99 102 103 104 105 106 107 109 110 111 113 114 118
##  [55] 119 120 121 122 123 125 126 128 135 137 138 139 140 143 145 146 150 151
##  [73] 154 157 158 159 163 164 167 169 170 172 173 174 178 182 183 184 190 191
##  [91] 196 197 198 199 200 201 202 204 206 209 211 212 214 217 221 225 226 227
## [109] 228 230 231 233 234 235 236 238 240 241 242 243 245 248 250 252 253 254
## [127] 256 258 259 262 263 267 268 269 270 272 274 276 277 278 280 281 288 289
## [145] 291 292 294 296 297 298 302 304 306 308 309 312 313 314 316 318 319 322
## [163] 325 326 329 332 335 336 341 343 347 348 349 351 354 355 357 360 361 367
## [181] 368 369 371 372 373 374 375 377 378 381 382 383 384 385 386 390 392 393
## [199] 396 398 399 400 406 408 410 411 412 413 414 415 416 417 419 420 421 422
## [217] 423 424 427 429 431 433 434 436 438 439 442 443 446 447 448 449 450 451
## [235] 452 453 455 458 462 466 467 468 470 471 472 473 475 477 482 483 484 486
## [253] 487 489 491 495 498 501 502 508 509 512 514 515 516 521 522 523 525 526
## [271] 527 528 529 531 532 533 535 536 539 542 544 545 548 551 552 554 555 562
## [289] 563 565 566 567 572 573 574 575 576 578 581 582 586 588 590 592 594 596
## [307] 598 600 601 602 606 607 608 609 610 611 614 616 618 620 621 622 624 625
## [325] 626 627 628 630 632 633 634 638 640 641 642 645 648 650 651 652 653 654
## [343] 655 656 657 660 662 666 672 674 678 679 680 681 682 683 684 686 687 688
## [361] 689 693 695 698 699 700 701 705 706 708 710 711 714 719 722 727 728 729
## [379] 730 732 733 734 736 737 739 742 743 747 751 752 753 754 759 761 765 768
```

```r
under30<-diabetes[diabetes$Age<30,]
under30
```

```
## # A tibble: 396 x 9
##    Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI
##          <dbl>   <dbl>         <dbl>         <dbl>   <dbl> <dbl>
##  1           1      89            66            23      94  28.1
##  2           3      78            50            32      88  31
##  3          10     115             0             0       0  35.3
##  4           3     126            88            41     235  39.3
##  5           9     119            80            35       0  29
##  6           1      97            66            15     140  23.2
##  7           3     158            76            36     245  31.6
##  8           3      88            58            11      54  24.8
##  9           6      92            92             0       0  19.9
## 10           2      90            68            42       0  38.2
## # ... with 386 more rows, and 3 more variables: DiabetesPedigreeFunction <dbl>,
## #   Age <dbl>, Outcome <dbl>
```

```r
mid_age<-diabetes[diabetes$Age>30 & diabetes$Age<50, ]
over_50<-diabetes[diabetes$Age>50,]
```

2.2.1 First question

```r
positive_under30<-under30[which(under30[,"Outcome"]==1),]
positive_under30
```

```
## # A tibble: 84 x 9
```

```
##    Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI
##          <dbl>   <dbl>         <dbl>         <dbl>   <dbl> <dbl>
##  1           3      78            50            32      88  31
##  2           9     119            80            35       0  29
##  3           3     158            76            36     245  31.6
##  4           2      90            68            42       0  38.2
##  5           0     180            66            39       0  42
##  6           2     100            66            20      90  32.9
##  7           0     131             0             0       0  43.2
##  8           0      95            85            25      36  37.4
##  9           3     171            72            33     135  33.3
## 10           0     162            76            56     100  53.2
## # ... with 74 more rows, and 3 more variables: DiabetesPedigreeFunction <dbl>,
## #   Age <dbl>, Outcome <dbl>
```

```
percentage_young_positive<-(nrow(positive_under30)/nrow(under30))*100
percentage_young_positive
```

```
## [1] 21.21212
```

21% of the young people have diabete

```
positive_midage<-under30[which(mid_age[,"Outcome"]==1),]
positive_midage
```

```
## # A tibble: 135 x 9
##    Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI
##          <dbl>   <dbl>         <dbl>         <dbl>   <dbl> <dbl>
##  1           3      78            50            32      88  31
##  2          10     115             0             0       0  35.3
##  3           3     126            88            41     235  39.3
##  4           9     119            80            35       0  29
##  5           1      97            66            15     140  23.2
##  6           3     158            76            36     245  31.6
##  7           6      92            92             0       0  19.9
##  8           2      90            68            42       0  38.2
##  9           3     180            64            25      70  34
## 10           0     180            66            39       0  42
## # ... with 125 more rows, and 3 more variables: DiabetesPedigreeFunction <dbl>,
## #   Age <dbl>, Outcome <dbl>
```

```
percentage_midage_positive<-(nrow(positive_midage)/nrow(mid_age))*100
percentage_midage_positive
```

```
## [1] 51.52672
```

51% of the mid aged people have diabete

```
positive_overage<-over_50[which(over_50[,"Outcome"]==1),]
positive_overage
```

```
## # A tibble: 38 x 9
##    Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI
##          <dbl>   <dbl>         <dbl>         <dbl>   <dbl> <dbl>
##  1           2     197            70            45     543  30.5
##  2           8     125            96             0       0   0
##  3           1     189            60            23     846  30.1
##  4           5     166            72            19     175  25.8
##  5          11     143            94            33     146  36.6
##  6           4     111            72            47     207  37.1
##  7           9     171           110            24     240  45.4
##  8           8     176            90            34     300  33.7
##  9           4     134            72             0       0  23.8
## 10           4     146            92             0       0  31.2
## # ... with 28 more rows, and 3 more variables: DiabetesPedigreeFunction <dbl>,
## #   Age <dbl>, Outcome <dbl>
```

```r
percentage_overage_positive<-(nrow(positive_overage)/nrow(over_50))*100
percentage_overage_positive
```

```
## [1] 46.91358
```

47% of the aged people have diabete

2.2.2 Second question

```r
res <- cor(diabetes, use = "complete.obs")
res
```

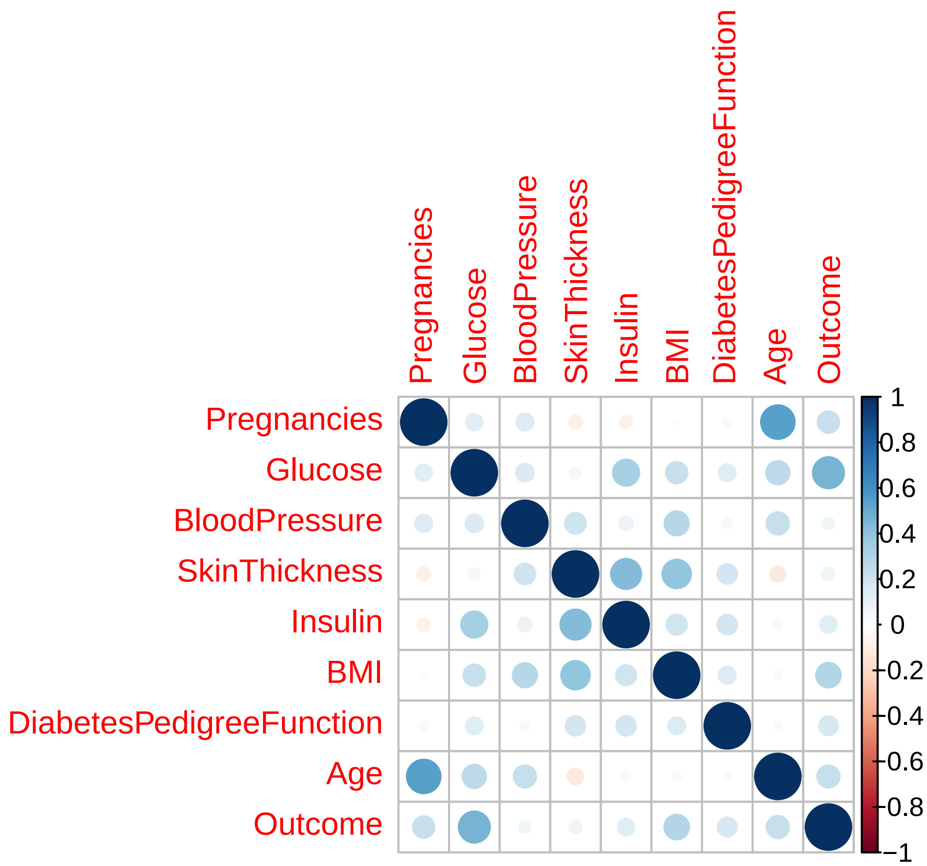
```
##                          Pregnancies    Glucose BloodPressure SkinThickness
## Pregnancies               1.00000000 0.12945867    0.14128198   -0.08167177
## Glucose                   0.12945867 1.00000000    0.15258959    0.05732789
## BloodPressure             0.14128198 0.15258959    1.00000000    0.20737054
## SkinThickness            -0.08167177 0.05732789    0.20737054    1.00000000
## Insulin                  -0.07353461 0.33135711    0.08893338    0.43678257
## BMI                       0.01768309 0.22107107    0.28180529    0.39257320
## DiabetesPedigreeFunction -0.03352267 0.13733730    0.04126495    0.18392757
## Age                       0.54434123 0.26351432    0.23952795   -0.11397026
## Outcome                   0.22189815 0.46658140    0.06506836    0.07475223
##                              Insulin        BMI DiabetesPedigreeFunction
## Pregnancies              -0.07353461 0.01768309              -0.03352267
## Glucose                   0.33135711 0.22107107               0.13733730
## BloodPressure             0.08893338 0.28180529               0.04126495
## SkinThickness             0.43678257 0.39257320               0.18392757
## Insulin                   1.00000000 0.19785906               0.18507093
## BMI                       0.19785906 1.00000000               0.14064695
## DiabetesPedigreeFunction  0.18507093 0.14064695               1.00000000
## Age                      -0.04216295 0.03624187               0.03356131
## Outcome                   0.13054795 0.29269466               0.17384407
##                                  Age    Outcome
## Pregnancies               0.54434123 0.22189815
## Glucose                   0.26351432 0.46658140
```

```
## BloodPressure              0.23952795 0.06506836
## SkinThickness             -0.11397026 0.07475223
## Insulin                   -0.04216295 0.13054795
## BMI                        0.03624187 0.29269466
## DiabetesPedigreeFunction   0.03356131 0.17384407
## Age                        1.00000000 0.23835598
## Outcome                    0.23835598 1.00000000
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
corrplot(res)
```



We see that the outcome is highly related with the Glucose rate in the body. It's also related with the body mass index and age

2.2.3 Third question

We will add a new variable giving us the age category

```
diabetes$age_category<-0
sel<-which(diabetes$Age<30)
diabetes[sel,"age_category"]<-1
sout<-which(diabetes$Age>=30 & diabetes$Age<50)
```

```r
diabetes[sout,"age_category"]<-2
sin<-which(diabetes$Age>=50)
diabetes[sin,"age_category"]<-3
```

```r
diabetes
```

```
## # A tibble: 768 x 10
##    Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI
##          <dbl>   <dbl>         <dbl>         <dbl>   <dbl> <dbl>
##  1           6     148            72            35       0  33.6
##  2           1      85            66            29       0  26.6
##  3           8     183            64             0       0  23.3
##  4           1      89            66            23      94  28.1
##  5           0     137            40            35     168  43.1
##  6           5     116            74             0       0  25.6
##  7           3      78            50            32      88  31
##  8          10     115             0             0       0  35.3
##  9           2     197            70            45     543  30.5
## 10           8     125            96             0       0   0
## # ... with 758 more rows, and 4 more variables: DiabetesPedigreeFunction <dbl>,
## #   Age <dbl>, Outcome <dbl>, age_category <dbl>
```

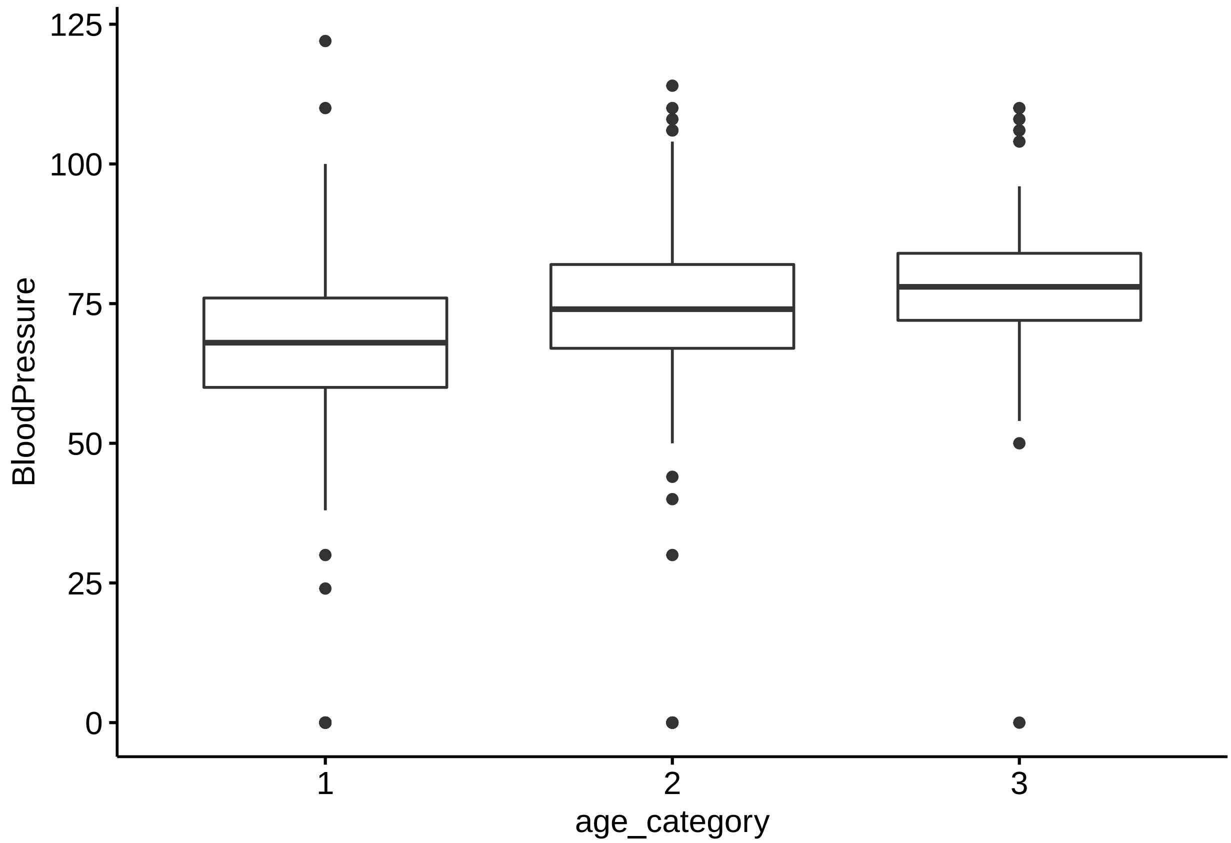
```r
diabetes %>%
  group_by(age_category, Outcome) %>%
  get_summary_stats(BloodPressure, type = "mean_sd")
```

```
## # A tibble: 6 x 6
##   Outcome age_category variable          n  mean    sd
##     <dbl>        <dbl> <chr>         <dbl> <dbl> <dbl>
## 1       0            1 BloodPressure   312  65.3  17.9
## 2       1            1 BloodPressure    84  65.4  23.4
## 3       0            2 BloodPressure   142  71.8  17.8
## 4       1            2 BloodPressure   141  71.1  21.9
## 5       0            3 BloodPressure    46  76.3  16.0
## 6       1            3 BloodPressure    43  80.6  10.1
```

Diabete: - doesn't change bloodPressure for young people, - doesn't change bloodPressure for mid aged people - increases bloodPressure for aged people

Age: -increases bloodpressure overall - Increases bloodpressure for diabetic peopl, this increase gets biggers with age, especially after 45 years - Increases bloodpressure for non diabetic people but in a less agressive manner than for the >45 years diabetic people

```r
bxp <- ggboxplot(diabetes,x = "age_category", y = "BloodPressure", color = "Outcome", palette = "jco")
bxp
```

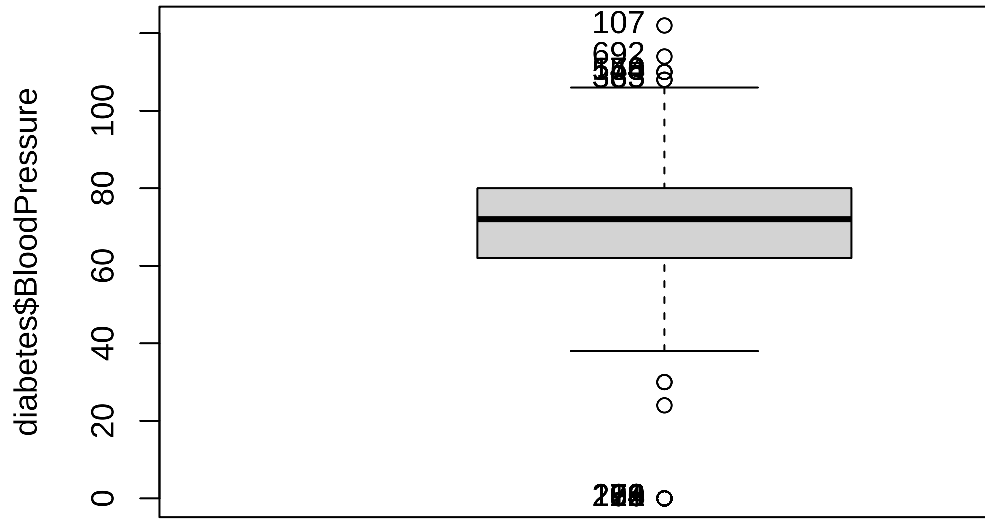



```
diabetes %>%
  group_by(Outcome, age_category) %>%
  identify_outliers(BloodPressure)
```

```
## # A tibble: 53 x 12
##    Outcome age_category Pregnancies Glucose BloodPressure SkinThickness Insulin
##      <dbl>        <dbl>       <dbl>   <dbl>         <dbl>         <dbl>   <dbl>
##  1       0            1          10     115             0             0       0
##  2       0            1           7     105             0             0       0
##  3       0            1           2      84             0             0       0
##  4       0            1           2      74             0             0       0
##  5       0            1           1      96           122             0       0
##  6       0            1           2      87             0            23       0
##  7       0            1           3     116             0             0       0
##  8       0            1           0      94             0             0       0
##  9       0            1           2      99             0             0       0
## 10       0            1           3      80             0             0       0
## # ... with 43 more rows, and 5 more variables: BMI <dbl>,
## #   DiabetesPedigreeFunction <dbl>, Age <dbl>, is.outlier <lgl>,
## #   is.extreme <lgl>
```

2.2.3.1 Treatement of the hypothesis and conditions

```r
Boxplot(diabetes$BloodPressure)
```



2.2.3.1.1 Removing the outliers

```
##  [1]   8  16  50  61  79  82 173 194 223 262  44  85 107 178 363 550 692
```

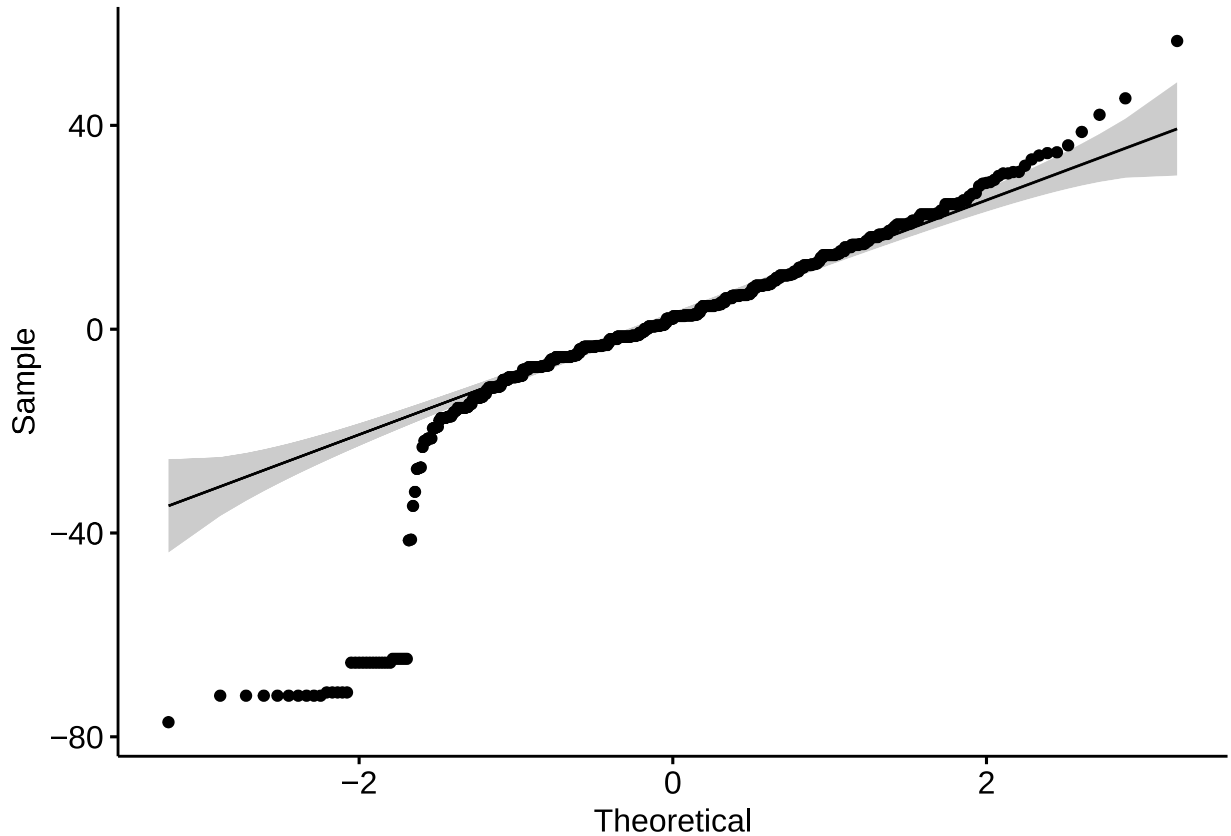
```r
boxplot(diabetes$BloodPressure, plot=FALSE)$out
```

```
##  [1]   0   0  30 110   0   0   0   0 108 122  30   0 110   0   0   0   0   0   0
## [20]   0   0   0   0 108   0   0   0   0   0   0   0   0   0   0 110   0  24   0
## [39]   0   0   0 114   0   0   0
```

```r
outliers<-boxplot(diabetes$BloodPressure, plot=FALSE)$out
diabetes_corrected<- diabetes[-which(diabetes$BloodPressure %in% outliers),]
```

‘ ##### Verifying the normality hypothesis

```r
#implementing the linear model
model  <- lm(BloodPressure ~ Outcome*age_category,
             data = diabetes)
# Creating a QQ plot of the residuals
ggqqplot(residuals(model))
```



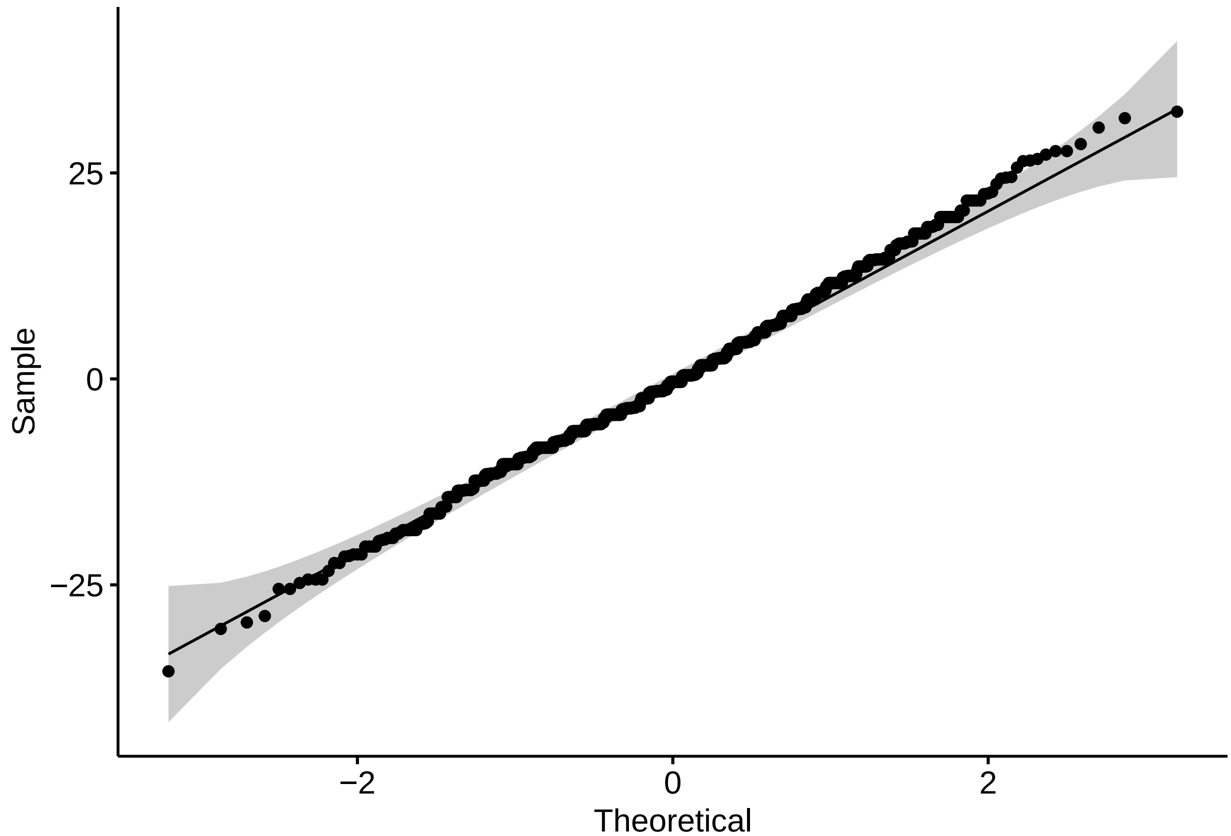

```
shapiro_test(residuals(model))
```

```
## # A tibble: 1 x 3
##   variable         statistic  p.value
##   <chr>                <dbl>    <dbl>
## 1 residuals(model)     0.810 4.10e-29
```

the p value is really low so the original data with the initial outliers doesn't follow a normal distribution.

We will check if our outliers correction was effective in that sense.

```
#implementing the linear model
model_1  <- lm(BloodPressure ~ Outcome*age_category,
             data = diabetes_corrected)
# Creating a QQ plot of the residuals
ggqqplot(residuals(model_1))
```



We can see that all our points are now in the normal area.

We check with the shapiro test:

```
shapiro_test(residuals(model_1))
```

```
## # A tibble: 1 x 3
##   variable          statistic p.value
##   <chr>                 <dbl>   <dbl>
## 1 residuals(model_1)    0.997   0.126
```

Our p-value is greater than 0.05, the normality condition is thus verified.

```
diabetes_corrected$Outcome<-factor((diabetes_corrected$Outcome))
diabetes_corrected$age_category<-factor(diabetes_corrected$age_category)
#As the levene test doesn't accept quantitative variables, we enter Outcome an age category as qualitat
diabetes_corrected %>%
  levene_test(BloodPressure ~ Outcome*age_category)
```

2.2.3.1.2 Homogenity of variances:

```
## # A tibble: 1 x 4
##     df1   df2 statistic     p
##   <int> <int>     <dbl> <dbl>
## 1     5   717      1.01 0.409
```

We see that p>0.05, this condition is thus respected.

```
res.aov <- diabetes_corrected %>%
  anova_test(BloodPressure ~ Outcome * age_category)
```

2.2.3.2 Anova test

```
## Coefficient covariances computed by hccm()
```

```
res.aov
```

```
## ANOVA Table (type II tests)
##
##                Effect DFn DFd      F        p p<.05   ges
## 1             Outcome   1 717  5.433 2.00e-02     * 0.008
## 2        age_category   2 717 32.331 3.60e-14     * 0.083
## 3 Outcome:age_category   2 717  0.809 4.46e-01       0.002
```

The H0 hypothesis denotes that the blood pressure is the same for all the categories. Here the p-value 0.046 is smaller than 0.05, the H0 hypothesis is thus not respected BloodPressure thus depends on Diabete and age category.

```
model <- lm(BloodPressure ~ Outcome * age_category, data = diabetes_corrected)
diabetes_corrected %>%
  group_by(age_category) %>%
  anova_test(BloodPressure ~ Outcome, error = model)
```

2.2.3.3 Analysing this dependance

```
## Coefficient covariances computed by hccm()
## Coefficient covariances computed by hccm()
## Coefficient covariances computed by hccm()
```

```
## # A tibble: 3 x 8
##   age_category Effect    DFn   DFd     F     p `p<.05`      ges
## * <fct>        <chr>   <dbl> <dbl> <dbl> <dbl> <chr>      <dbl>
## 1 1            Outcome     1   717 5.45  0.02  "*"     0.008
## 2 2            Outcome     1   717 0.406 0.524 ""      0.000566
## 3 3            Outcome     1   717 1.20  0.275 ""      0.002
```

Diabete's effect on bloodpressure is significant for the youngest category while it doesn't have a big dependance for the other two age categories. We think that Bloodpressure is high in all cases for the old people.

```r
model <- lm(BloodPressure ~ Outcome * age_category, data = diabetes_corrected)
diabetes_corrected %>%
  group_by(Outcome) %>%
  anova_test(BloodPressure ~ age_category, error = model)
```

```
## Coefficient covariances computed by hccm()
## Coefficient covariances computed by hccm()

## # A tibble: 2 x 8
##   Outcome Effect         DFn   DFd     F        p `p<.05`   ges
## * <fct>   <chr>        <dbl> <dbl> <dbl>    <dbl> <chr>   <dbl>
## 1 0       age_category     2   717 24.7  4.04e-11 *       0.065
## 2 1       age_category     2   717  8.39 2.5 e- 4 *       0.023
```

Age has a significative effect on bloodpressure for either diabetic or non diabetic people.

2.2.4 Fourth question

2.2.4.1 Trying with the Outcome variable

2.2.4.1.1 Outliers We first suspected the outcome variable to be the best suited for an ANOVA analysis

```r
diabetes %>%
  group_by(Outcome) %>%
  identify_outliers(Insulin)
```

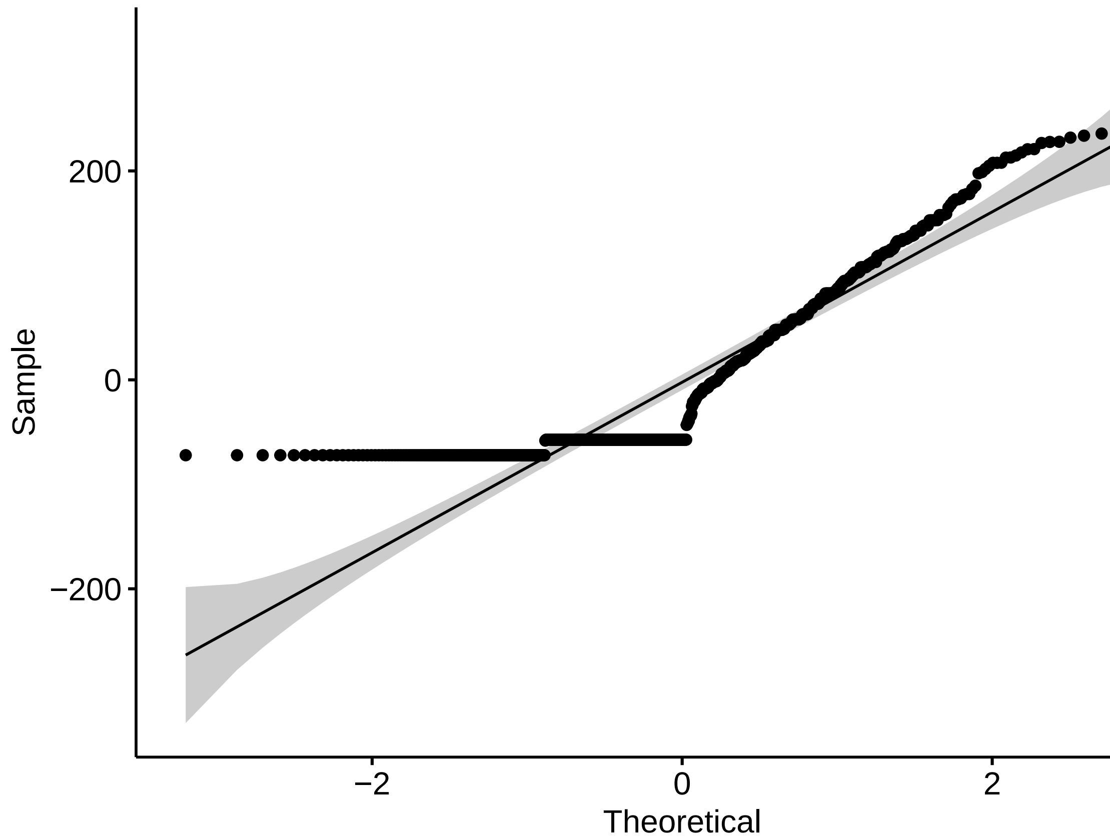
```
## # A tibble: 38 x 12
##    Outcome Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI
##      <dbl>       <dbl>   <dbl>         <dbl>         <dbl>   <dbl> <dbl>
##  1       0           7     150            66            42     342  34.7
##  2       0           4     129            86            20     270  35.1
##  3       0           5     105            72            29     325  36.9
##  4       0           4     154            62            31     284  32.8
##  5       0           1     153            82            42     485  40.6
##  6       0           0     114            80            34     285  44.2
##  7       0           4     197            70            39     744  36.7
##  8       0           0     165            90            33     680  52.3
##  9       0           9     124            70            33     402  35.4
## 10       0           1     193            50            16     375  25.9
## # ... with 28 more rows, and 5 more variables: DiabetesPedigreeFunction <dbl>,
## #   Age <dbl>, age_category <dbl>, is.outlier <lgl>, is.extreme <lgl>
```

```r
boxplot(diabetes$Insulin, plot=FALSE)$out
```

```
##  [1] 543 846 342 495 325 485 495 478 744 370 680 402 375 545 360 325 465 325 415
## [20] 579 474 328 480 326 330 600 321 440 540 480 335 387 392 510
```

```r
outliers<-boxplot(diabetes$Insulin, plot=FALSE)$out
diabetes_corrected_2<- diabetes[-which(diabetes$Insulin %in% outliers),]
```

```
#implementing the linear model
model  <- lm(Insulin ~ Outcome,
             data = diabetes_corrected_2)
# Creating a QQ plot of the residuals
ggqqplot(residuals(model))
```



2.2.4.1.2 Shapiro test

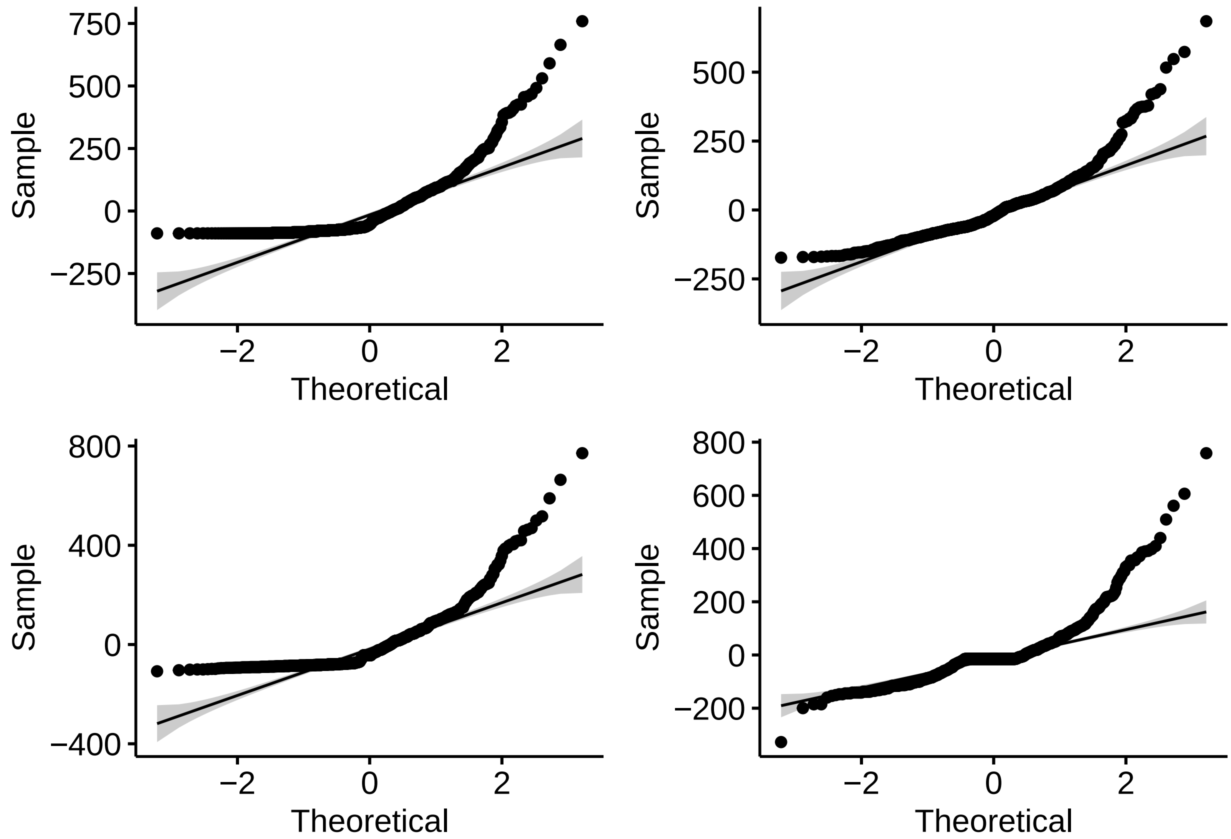
We can see that, even after removing the outliers,we are still really far from a normal distribution. the Outcome variable is thus not the best suited for an ANOVA analysis.

```
#implementing the linear model
model_1  <- lm(Insulin ~ Pregnancies,data = diabetes)
model_2  <- lm(Insulin ~ Glucose,data = diabetes)
model_3  <- lm(Insulin ~ BloodPressure,data = diabetes)
model_4  <- lm(Insulin ~ SkinThickness,data = diabetes)
# Creating a QQ plot of the residuals
par(mfrow=c(2,2))
bp<-ggqqplot(residuals(model_1))
dp<-ggqqplot(residuals(model_2))
vp<-ggqqplot(residuals(model_3))
sc<-ggqqplot(residuals(model_4))
library(gridExtra)
```

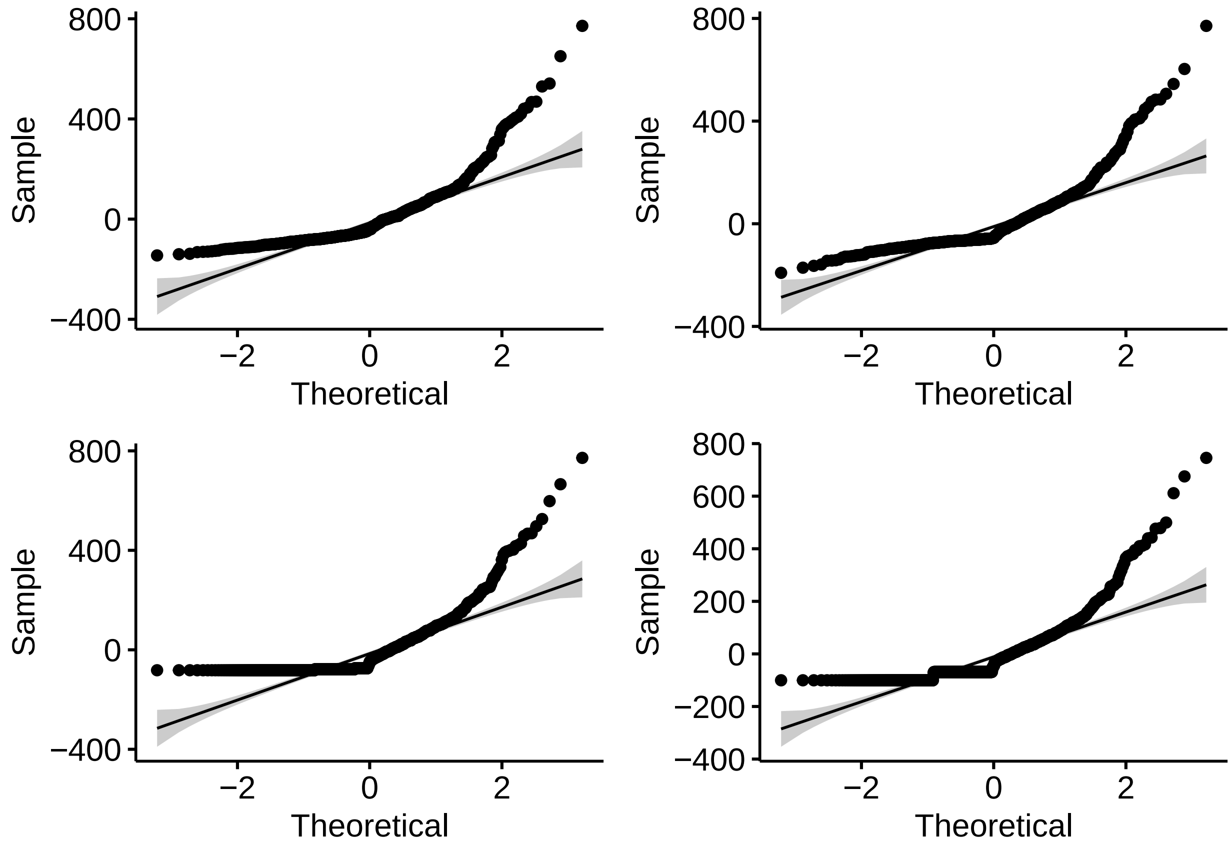
2.2.4.2 Trying with the other variables

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
grid.arrange(bp, dp, vp, sc, ncol=2, nrow = 2)
```




```r
#implementing the linear model
model_5  <- lm(Insulin ~ BMI,data = diabetes)
model_6  <- lm(Insulin ~ DiabetesPedigreeFunction,data = diabetes)
model_7  <- lm(Insulin ~ age_category,data = diabetes)
model_8  <- lm(Insulin ~ Outcome,data = diabetes)
# Creating a QQ plot of the residuals
par(mfrow=c(2,2))
bp<-ggqqplot(residuals(model_5))
dp<-ggqqplot(residuals(model_6))
vp<-ggqqplot(residuals(model_7))
sc<-ggqqplot(residuals(model_8))
library(gridExtra)
grid.arrange(bp, dp, vp, sc, ncol=2, nrow = 2)
```



We can see that the second model is the most close to a normal distribution , Glucose can be a good variable to use for an ANOVA analysis of Insulin. SkinThickeness is also a good candidate for an ANOVA analysis However this analysis is not of a great accuracy. We will thus calculate the p-values.

```
#The populations from which the samples are selected must be normal.
#Shapiro test
df <- data.frame(matrix(ncol = 3, nrow = 8))
colnames(df)<-c("model","statistic","p-value")

df[1,]<-shapiro_test(residuals(model_1))
df[2,]<-shapiro_test(residuals(model_2))
df[3,]<-shapiro_test(residuals(model_3))
df[4,]<-shapiro_test(residuals(model_4))
df[5,]<-shapiro_test(residuals(model_5))
df[6,]<-shapiro_test(residuals(model_6))
df[7,]<-shapiro_test(residuals(model_7))
df[8,]<-shapiro_test(residuals(model_8))
df
```

```
##                 model statistic      p-value
## 1 residuals(model_1) 0.7479398 1.415599e-32
## 2 residuals(model_2) 0.8823746 1.138339e-23
## 3 residuals(model_3) 0.7587578 5.060836e-32
## 4 residuals(model_4) 0.8120200 5.702389e-29
## 5 residuals(model_5) 0.8033310 1.641354e-29
## 6 residuals(model_6) 0.7915956 3.264020e-30
## 7 residuals(model_7) 0.7290961 1.701320e-33
```

```
## 8 residuals(model_8) 0.7777604 5.317261e-31
```

We can see that given the p-values calculated, Insulin and SkinThickness are the two variables that would be the best suited for an Anova analysis.

3 Question 3

```
#install.packages(RcmdrPlugin.FactoMineR)
#install.packages("Rcmdr",dependencies=TRUE)
#install.packages("FactoMineR",dependencies=TRUE)
#install.packages("weatherData",repos = "http://cran.us.r-project.org")

library(FactoMineR)
data(decathlon)
data(decathlon)
colnames(decathlon)
```

```
##  [1] "100m"        "Long.jump"   "Shot.put"    "High.jump"   "400m"
##  [6] "110m.hurdle" "Discus"      "Pole.vault"  "Javeline"    "1500m"
## [11] "Rank"        "Points"      "Competition"
```

```
head(decathlon)
```

```
##          100m Long.jump Shot.put High.jump  400m 110m.hurdle Discus Pole.vault
## SEBRLE  11.04      7.58    14.83      2.07 49.81       14.69  43.75       5.02
## CLAY    10.76      7.40    14.26      1.86 49.37       14.05  50.72       4.92
## KARPOV  11.02      7.30    14.77      2.04 48.37       14.09  48.95       4.92
## BERNARD 11.02      7.23    14.25      1.92 48.93       14.99  40.87       5.32
## YURKOV  11.34      7.09    15.19      2.10 50.42       15.31  46.26       4.72
## WARNERS 11.11      7.60    14.31      1.98 48.68       14.23  41.10       4.92
##         Javeline 1500m Rank Points Competition
## SEBRLE     63.19 291.7    1   8217    Decastar
## CLAY       60.15 301.5    2   8122    Decastar
## KARPOV     50.31 300.2    3   8099    Decastar
## BERNARD    62.77 280.1    4   8067    Decastar
## YURKOV     63.44 276.4    5   8036    Decastar
## WARNERS    51.77 278.1    6   8030    Decastar
```

```
summary(decathlon)
```

```
##       100m         Long.jump       Shot.put       High.jump         400m
##  Min.   :10.44   Min.   :6.61   Min.   :12.68   Min.   :1.850   Min.   :46.81
##  1st Qu.:10.85   1st Qu.:7.03   1st Qu.:13.88   1st Qu.:1.920   1st Qu.:48.93
##  Median :10.98   Median :7.30   Median :14.57   Median :1.950   Median :49.40
##  Mean   :11.00   Mean   :7.26   Mean   :14.48   Mean   :1.977   Mean   :49.62
##  3rd Qu.:11.14   3rd Qu.:7.48   3rd Qu.:14.97   3rd Qu.:2.040   3rd Qu.:50.30
##  Max.   :11.64   Max.   :7.96   Max.   :16.36   Max.   :2.150   Max.   :53.20
##   110m.hurdle        Discus        Pole.vault       Javeline
##  Min.   :13.97   Min.   :37.92   Min.   :4.200   Min.   :50.31
```

```
##  1st Qu.:14.21   1st Qu.:41.90   1st Qu.:4.500   1st Qu.:55.27
##  Median :14.48   Median :44.41   Median :4.800   Median :58.36
##  Mean   :14.61   Mean   :44.33   Mean   :4.762   Mean   :58.32
##  3rd Qu.:14.98   3rd Qu.:46.07   3rd Qu.:4.920   3rd Qu.:60.89
##  Max.   :15.67   Max.   :51.65   Max.   :5.400   Max.   :70.52
##      1500m            Rank           Points       Competition
##  Min.   :262.1   Min.   : 1.00   Min.   :7313   Decastar:13
##  1st Qu.:271.0   1st Qu.: 6.00   1st Qu.:7802   OlympicG:28
##  Median :278.1   Median :11.00   Median :8021
##  Mean   :279.0   Mean   :12.12   Mean   :8005
##  3rd Qu.:285.1   3rd Qu.:18.00   3rd Qu.:8122
##  Max.   :317.0   Max.   :28.00   Max.   :8893
```

3.1 Most correlated variable with X1500m

3.1.1 Correlation matrix

To start, we must figure out which of the variables of the data have the most correlation with X1500m in order to be a good predictor for the linear expression. Instead of comparing all variables one by one with X1500m, we used a correlation matrix to view all combinations and their correlation value. We visualized the results with the corrplot method from the corrplot library.

We will Add X in front of the variables 100m, 400m, 110m.hurdle and 1500m. Otherwise,in the lm function they are not read since they begin with a number.

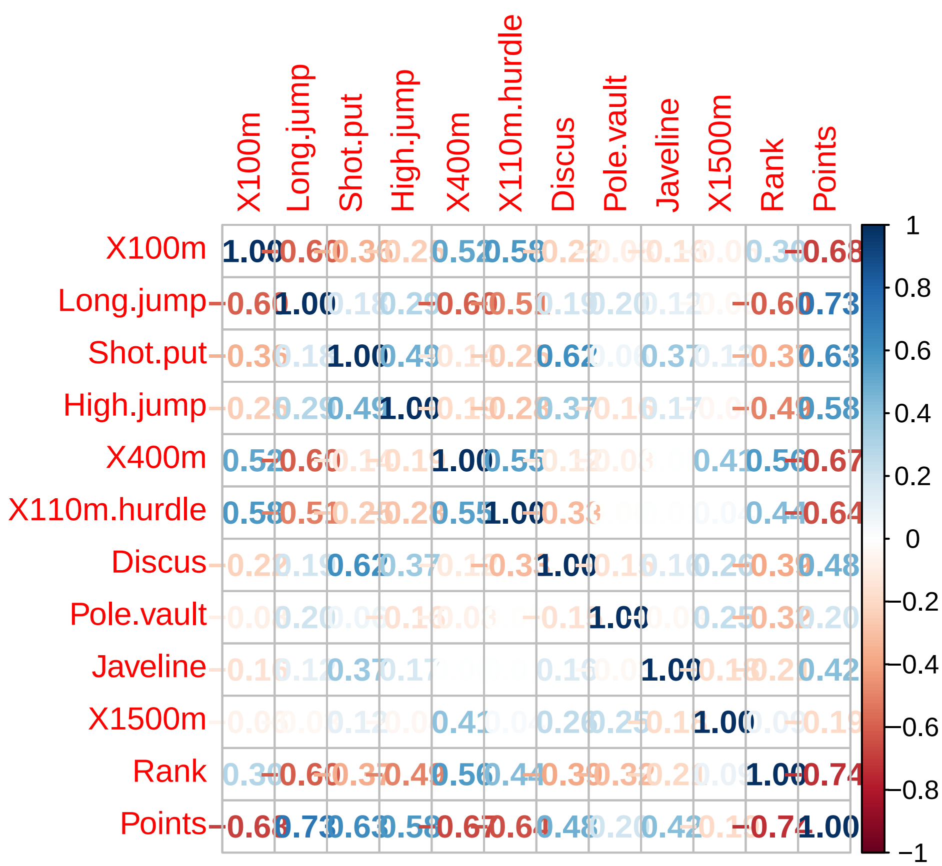
```
colnames(decathlon)[c(1,5,6,10)]<-c("X100m","X400m","X110m.hurdle","X1500m")
```

```
cor(decathlon[1:12])#last column is not numerical
```

```
##                    X100m   Long.jump    Shot.put   High.jump        X400m
## X100m         1.00000000 -0.59867767 -0.35648227 -0.24625292  0.520298155
## Long.jump    -0.59867767  1.00000000  0.18330436  0.29464444 -0.602062618
## Shot.put     -0.35648227  0.18330436  1.00000000  0.48921153 -0.138432919
## High.jump    -0.24625292  0.29464444  0.48921153  1.00000000 -0.187956928
## X400m         0.52029815 -0.60206262 -0.13843292 -0.18795693  1.000000000
## X110m.hurdle  0.57988893 -0.50541009 -0.25161571 -0.28328909  0.547987756
## Discus       -0.22170757  0.19431009  0.61576810  0.36921834 -0.117879365
## Pole.vault   -0.08253683  0.20401411  0.06118185 -0.15618074 -0.079292469
## Javeline     -0.15774645  0.11975893  0.37495551  0.17188009  0.004232096
## X1500m       -0.06054645 -0.03368613  0.11580306 -0.04490252  0.408106432
## Rank          0.29670366 -0.60405452 -0.36996958 -0.49276873  0.562118543
## Points       -0.68427243  0.72513490  0.62738936  0.57670316 -0.666939955
##              X110m.hurdle     Discus   Pole.vault     Javeline      X1500m
## X100m         0.579888931 -0.2217076 -0.082536834 -0.157746452 -0.06054645
## Long.jump    -0.505410086  0.1943101  0.204014112  0.119758933 -0.03368613
## Shot.put     -0.251615714  0.6157681  0.061181853  0.374955509  0.11580306
## High.jump    -0.283289090  0.3692183 -0.156180742  0.171880092 -0.04490252
## X400m         0.547987756 -0.1178794 -0.079292469  0.004232096  0.40810643
## X110m.hurdle  1.000000000 -0.3262010 -0.002703885  0.008743251  0.03754024
## Discus       -0.326200961  1.0000000 -0.150072400  0.157889799  0.25817510
## Pole.vault   -0.002703885 -0.1500724  1.000000000 -0.030000603  0.24744778
## Javeline      0.008743251  0.1578898 -0.030000603  1.000000000 -0.18039313
## X1500m        0.037540240  0.2581751  0.247447780 -0.180393128  1.00000000
```

```
## Rank          0.439102281 -0.3891251 -0.320379567 -0.208094646  0.08989781
## Points       -0.644460200  0.4841830  0.197436342  0.422393176 -0.19434860
##                    Rank     Points
## X100m         0.29670366 -0.6842724
## Long.jump    -0.60405452  0.7251349
## Shot.put     -0.36996958  0.6273894
## High.jump    -0.49276873  0.5767032
## X400m         0.56211854 -0.6669400
## X110m.hurdle  0.43910228 -0.6444602
## Discus       -0.38912515  0.4841830
## Pole.vault   -0.32037957  0.1974363
## Javeline     -0.20809465  0.4223932
## X1500m        0.08989781 -0.1943486
## Rank          1.00000000 -0.7391835
## Points       -0.73918347  1.0000000
```

```
M = cor(decathlon[1:12])
corrplot(M, method = 'number')
```



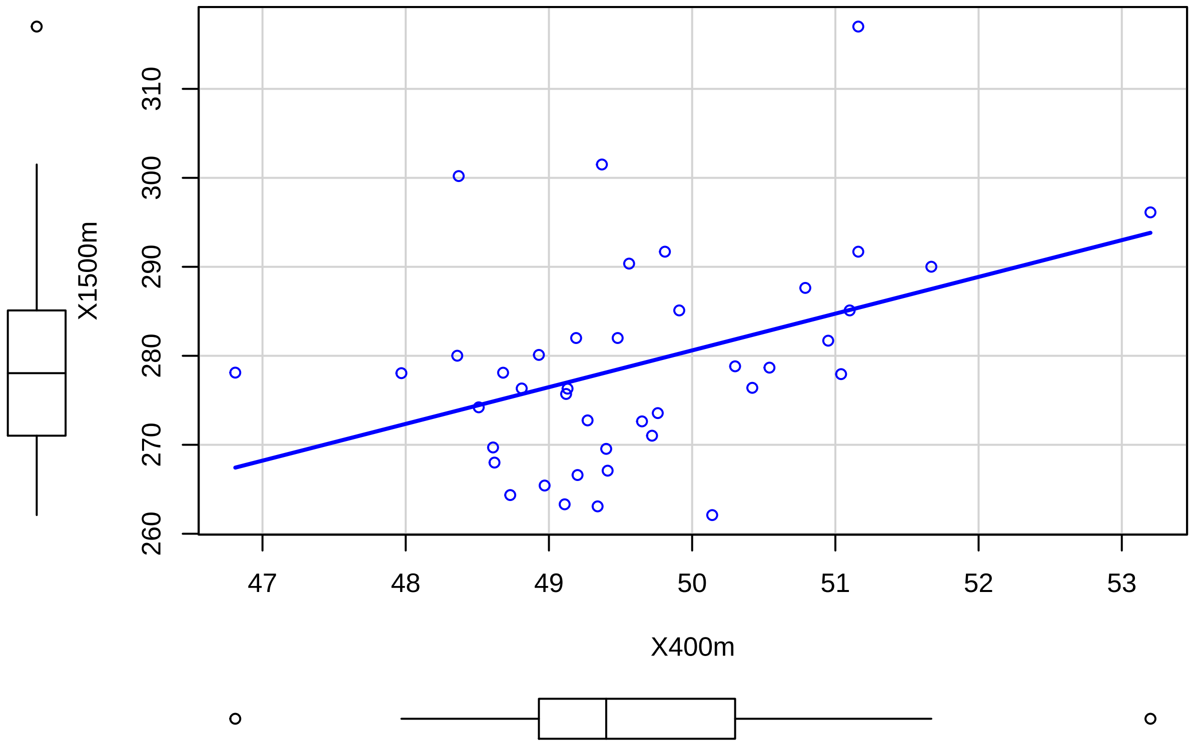
The most positive correlations from independent variables are X400m (0.41), Discus (0.26), and Pole.vault (0.25), whereas X400m has the highest correlation with X1500m.

To further determine the strength of the linear relationship between those variables and X1500m, we created scatterplots to visualize the data.

3.1.2 Scatterplots for X400m, Discus, Pole.vault

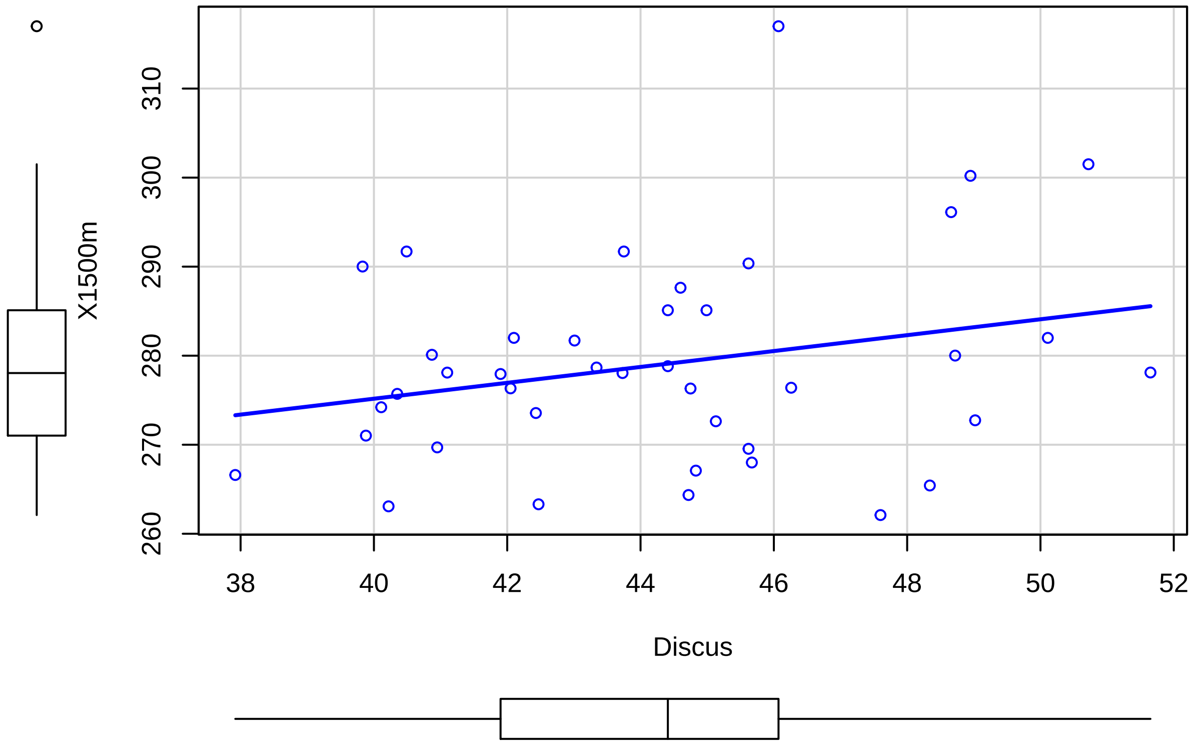
```
par(mfrow=c(1,3))
scatterplot(X1500m~X400m, regLine=lm, smooth=FALSE, data=decathlon)
```

```
## Warning in applyDefaults(regLine, defaults = list(method = lm, lty = 1, :
## unnamed arguments, will be ignored
```



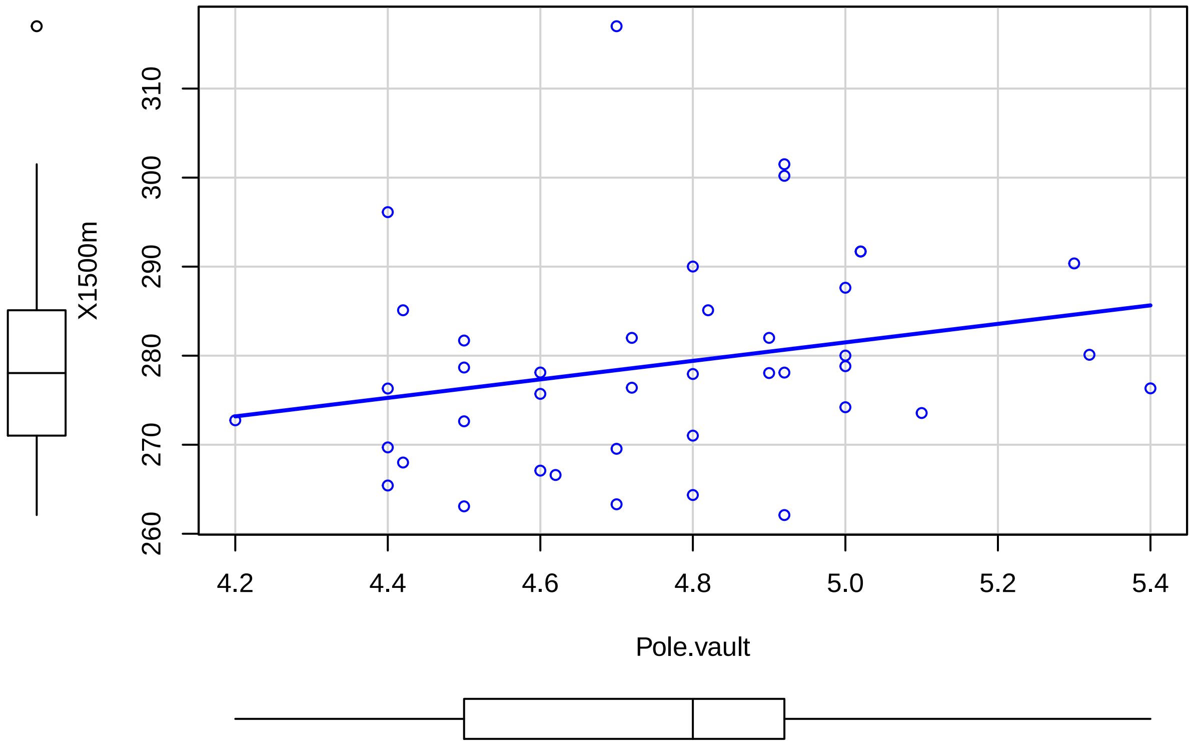

```
scatterplot(X1500m~Discus, regLine=lm, smooth=FALSE, data=decathlon)
```

```
## Warning in applyDefaults(regLine, defaults = list(method = lm, lty = 1, :
## unnamed arguments, will be ignored
```




```
scatterplot(X1500m~Pole.vault, regLine=lm, smooth=FALSE, data=decathlon)
```

```
## Warning in applyDefaults(regLine, defaults = list(method = lm, lty = 1, :
## unnamed arguments, will be ignored
```



In these scatterplots, we can observe two numerical variables while the line indicates the level of the linear relationship, which can be moderate, strong or weak. When the points are closer to the line, the linear relationship is stronger and when they are further away from the line, the linear relationship is weaker.

We can state that there is a positive moderate linear relationship between X400m and X1500m. The other variables shows a weaker linear relationship.

3.1.3 Testing regression assumptions

The next step is to test the regressions assumptions (normality of the error term, homogeneity of variance, independence of errors) on the defined linear models.

3.1.3.1 Regression 1 In the following, we can see how to perform the assumption tests for regression model Reg1.

```
# Reg1
Reg1 <-lm(X1500m ~ X400m, data=decathlon)
summary(Reg1)
```

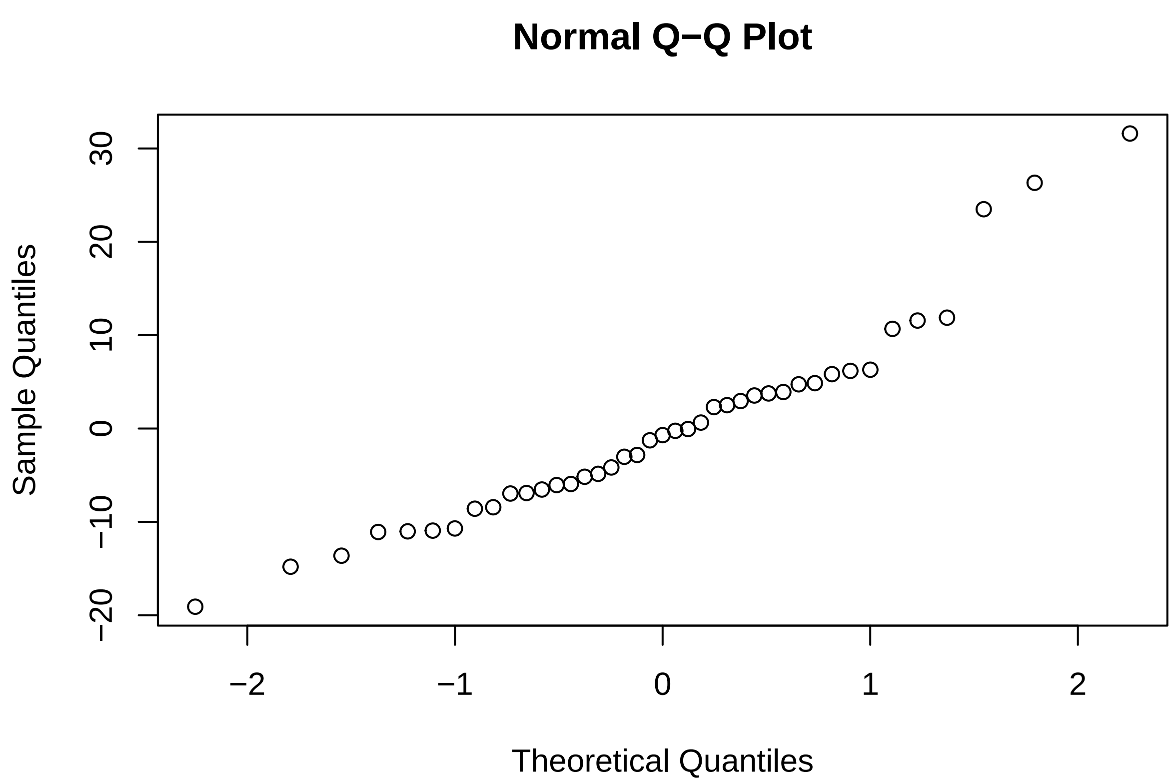
```
##
## Call:
## lm(formula = X1500m ~ X400m, data = decathlon)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -19.0877  -6.9098  -0.7062   4.7360  31.5996
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   74.102     73.424   1.009  0.31909
## X400m          4.130      1.479   2.792  0.00808 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.79 on 39 degrees of freedom
## Multiple R-squared:  0.1666, Adjusted R-squared:  0.1452
## F-statistic: 7.793 on 1 and 39 DF,  p-value: 0.008078
```

3.1.3.1.1 1. Normality of the Error Term

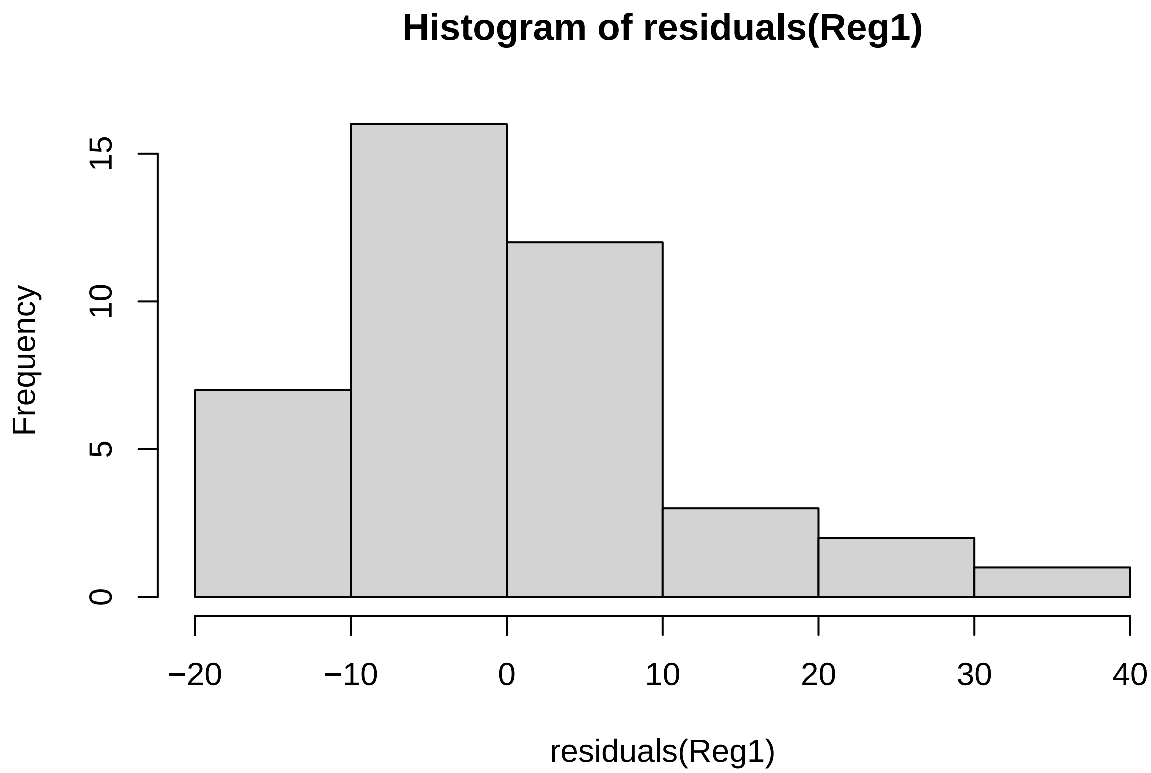
Using QQ plot

```
qqnorm(residuals(Reg1))
```



Using Histogram

```
hist(residuals(Reg1))
```



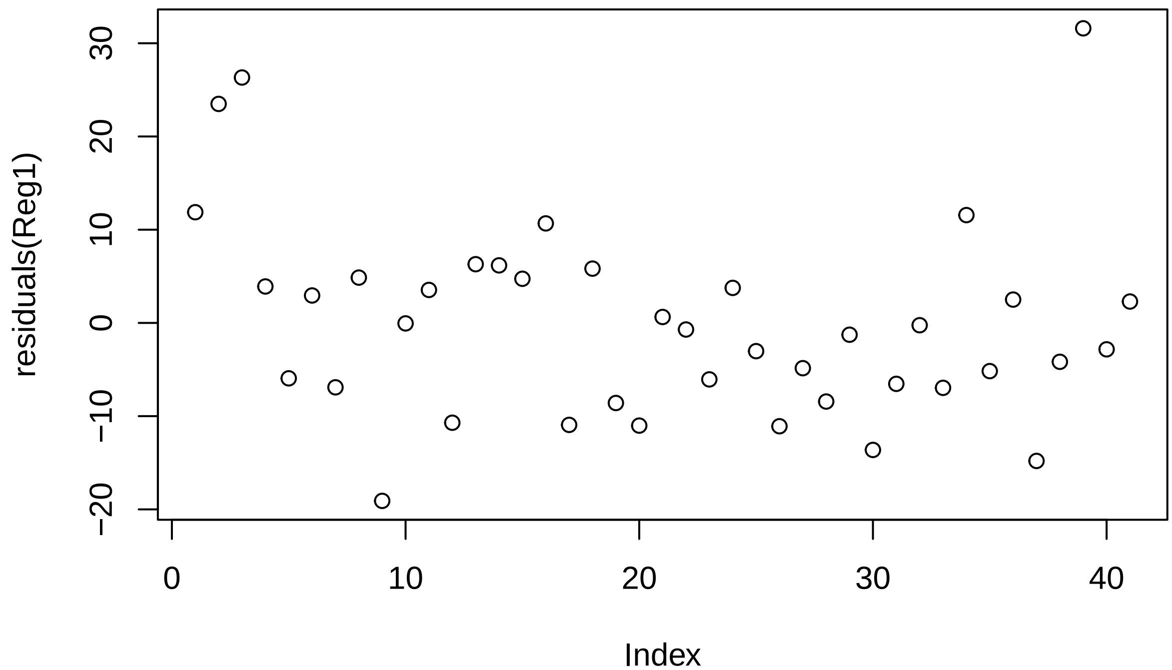
Shapiro Wilks Test

```
shapiro.test(residuals(Reg1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(Reg1)
## W = 0.93244, p-value = 0.01742
```

3.1.3.1.2 2. Homogenity of Variance Residual Analysis

```
plot(residuals(Reg1))
```



Breusch Pagan Test

```
library(lmtest)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
bptest(Reg1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Reg1
## BP = 0.0010727, df = 1, p-value = 0.9739
```

```
dwtest(Reg1, alternative = "two.sided")
```

3.1.3.1.3 3. The independence of errors

```
##
##  Durbin-Watson test
##
## data:  Reg1
## DW = 1.7274, p-value = 0.3458
## alternative hypothesis: true autocorrelation is not 0
```

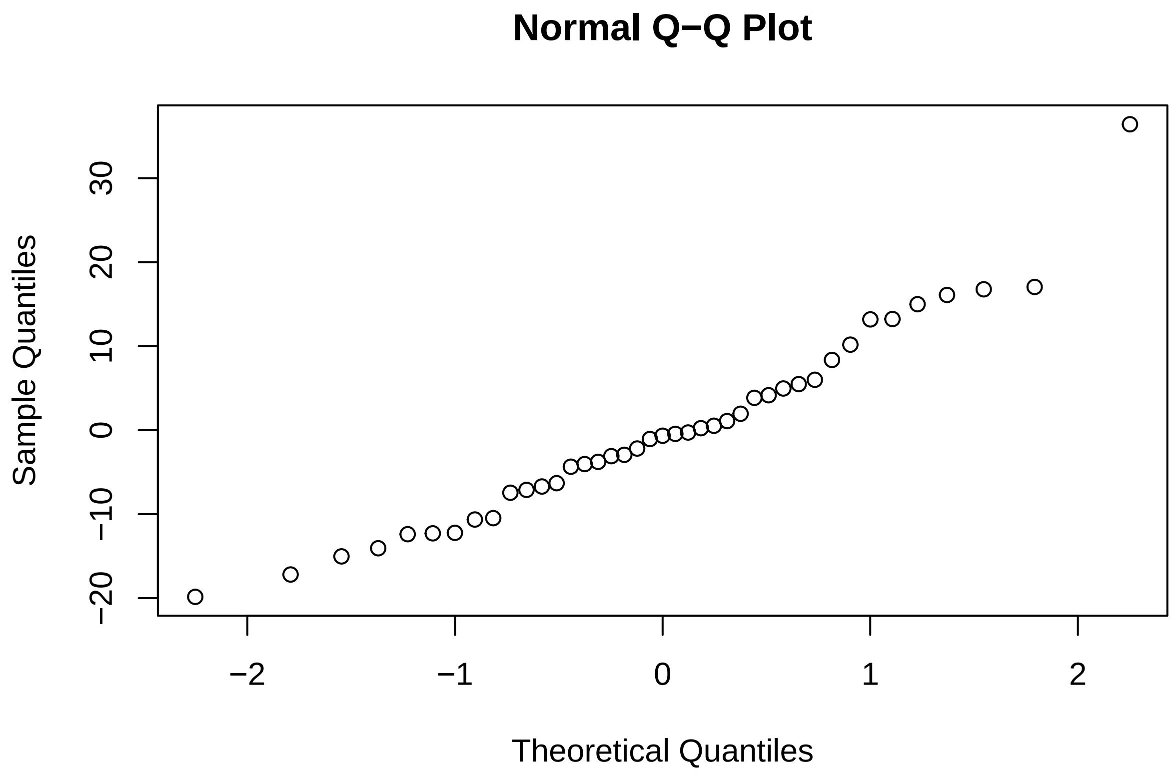
3.1.3.2 Regression 2 In the following, we can see how to perform the assumption tests for regression model Reg1.

```
# Reg2
Reg2 <-lm(X1500m ~ Discus, data=decathlon)
summary(Reg2)
```

```
##
## Call:
## lm(formula = X1500m ~ Discus, data = decathlon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.846  -7.113  -0.665   5.482  36.419
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 239.4772    23.7642  10.077 2.06e-12 ***
## Discus        0.8922     0.5346   1.669    0.103
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.42 on 39 degrees of freedom
## Multiple R-squared:  0.06665,    Adjusted R-squared:  0.04272
## F-statistic: 2.785 on 1 and 39 DF,  p-value: 0.1032
```

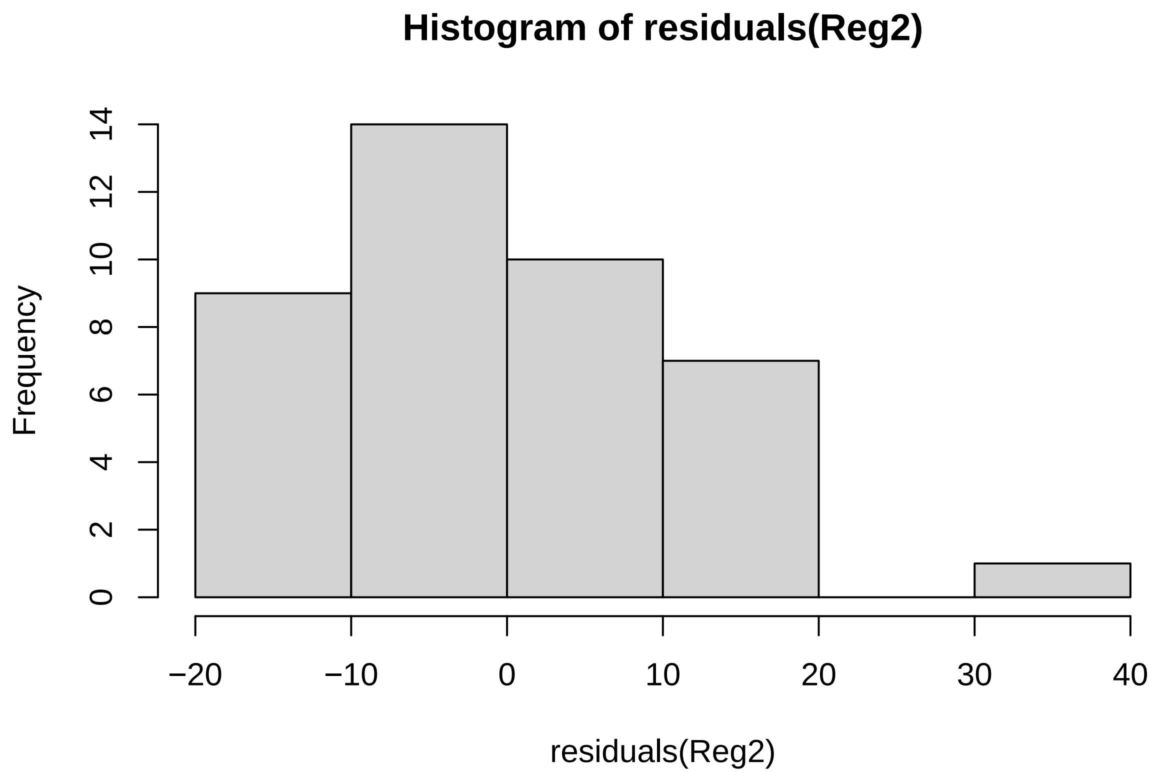
3.1.3.2.1 1. Normality of the Error Term Using QQ plot

```
qqnorm(residuals(Reg2))
```



Using Histogram

```
hist(residuals(Reg2))
```



Shapiro Wilks Test

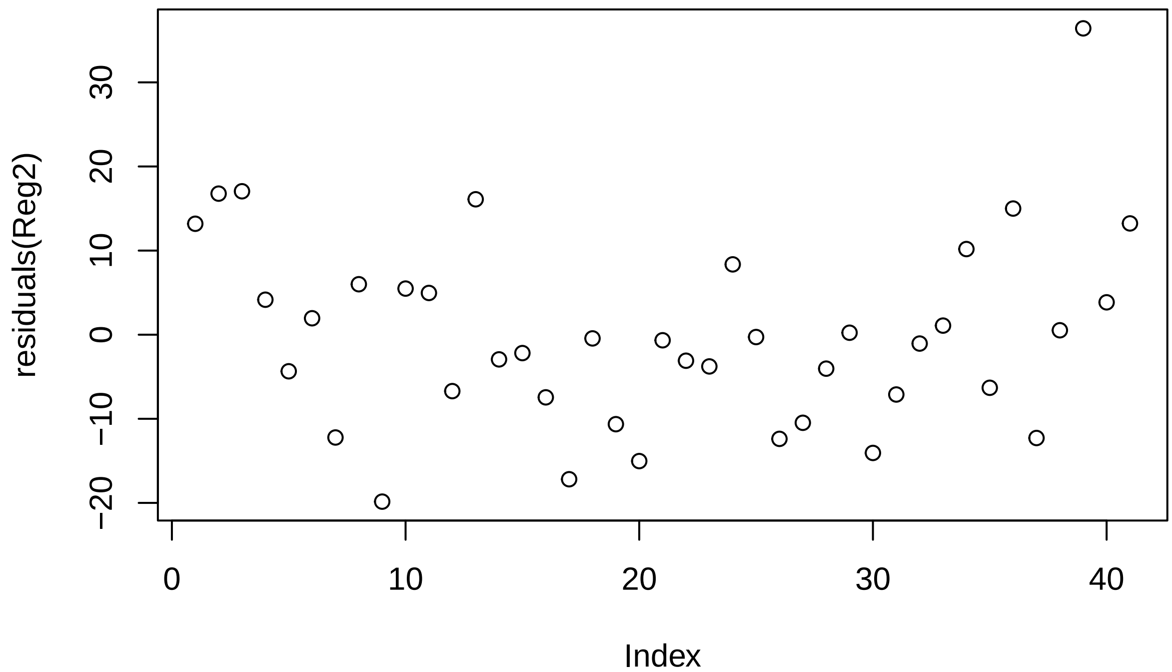
```
shapiro.test(residuals(Reg2))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(Reg2)
## W = 0.95789, p-value = 0.1327
```

3.1.3.2.2 2. Homogenity of Variance

Residual Analysis

```
plot(residuals(Reg2))
```



Breusch Pagan Test

```
bptest(Reg2)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Reg2
## BP = 2.0819, df = 1, p-value = 0.1491
```

```
dwtest(Reg2, alternative = "two.sided")
```

3.1.3.2.3 3. The independence of errors

```
##
##  Durbin-Watson test
##
## data:  Reg2
## DW = 1.7242, p-value = 0.3434
## alternative hypothesis: true autocorrelation is not 0
```

3.1.3.3 Regression 3

In the following, we can see how to perform the assumption tests for regression model Reg1.

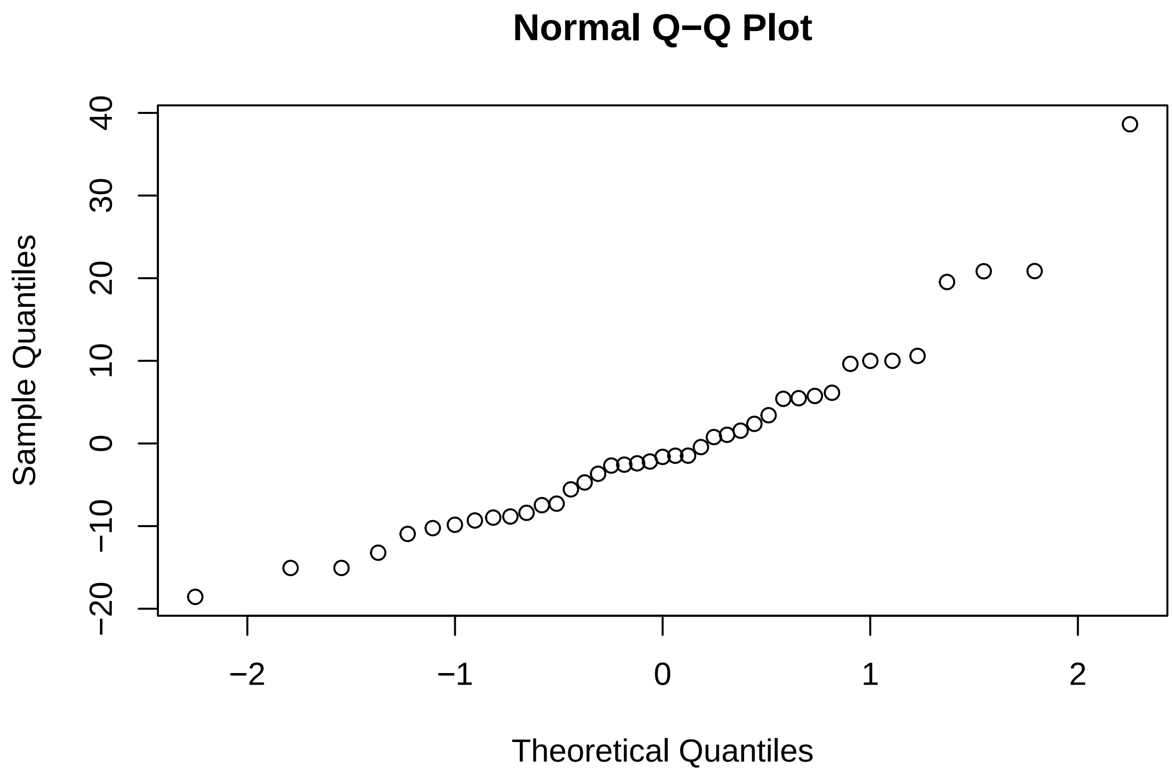
```
# Reg3
Reg3 <-lm(X1500m ~ Pole.vault, data=decathlon)
summary(Reg3)
```

```
##
## Call:
## lm(formula = X1500m ~ Pole.vault, data = decathlon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.562  -8.395  -1.627   5.477  38.624
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  229.541     31.077   7.386 6.36e-09 ***
## Pole.vault    10.390      6.515   1.595    0.119
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.45 on 39 degrees of freedom
## Multiple R-squared:  0.06123,    Adjusted R-squared:  0.03716
## F-statistic: 2.544 on 1 and 39 DF,  p-value: 0.1188
```

3.1.3.3.1 1. Normality of the Error Term

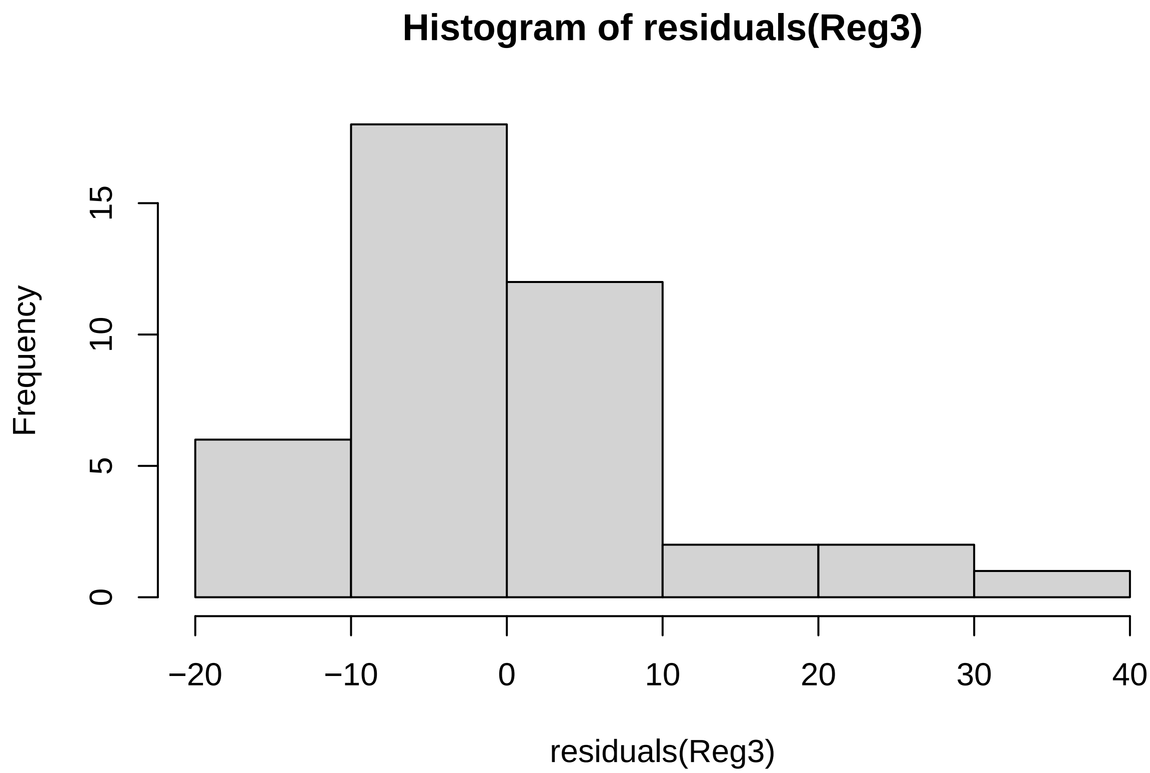
Using QQ plot

```
qqnorm(residuals(Reg3))
```



Using Histogram

```
hist(residuals(Reg3))
```



Shapiro Wilks Test

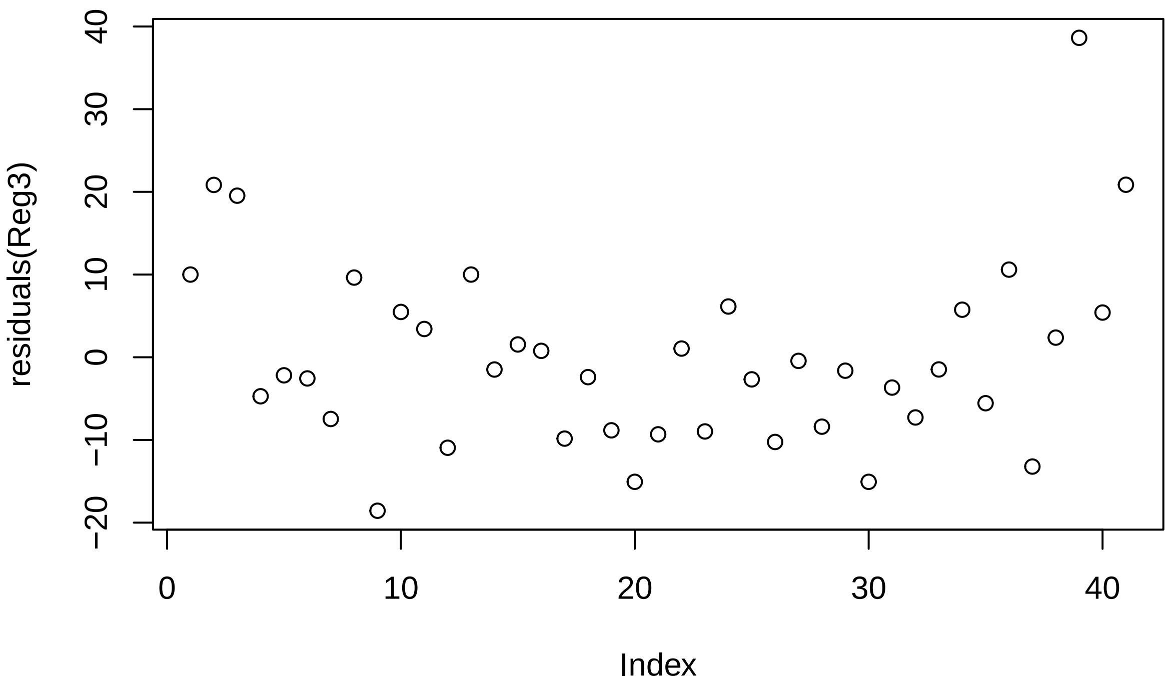
```
shapiro.test(residuals(Reg3))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(Reg3)
## W = 0.92665, p-value = 0.0112
```

3.1.3.3.2 2. Homogenity of Variance

Residual Analysis

```
plot(residuals(Reg3))
```



Breusch Pagan Test

```
bptest(Reg3)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Reg3
## BP = 0.027189, df = 1, p-value = 0.869
```

```
dwtest(Reg3, alternative = "two.sided")
```

3.1.3.3.3 3. The independence of errors

```
##
##  Durbin-Watson test
##
## data:  Reg3
## DW = 1.6645, p-value = 0.2709
## alternative hypothesis: true autocorrelation is not 0
```

3.1.3.4 Regression 4

In the following, we can see how to perform the assumption tests for regression model Reg4.

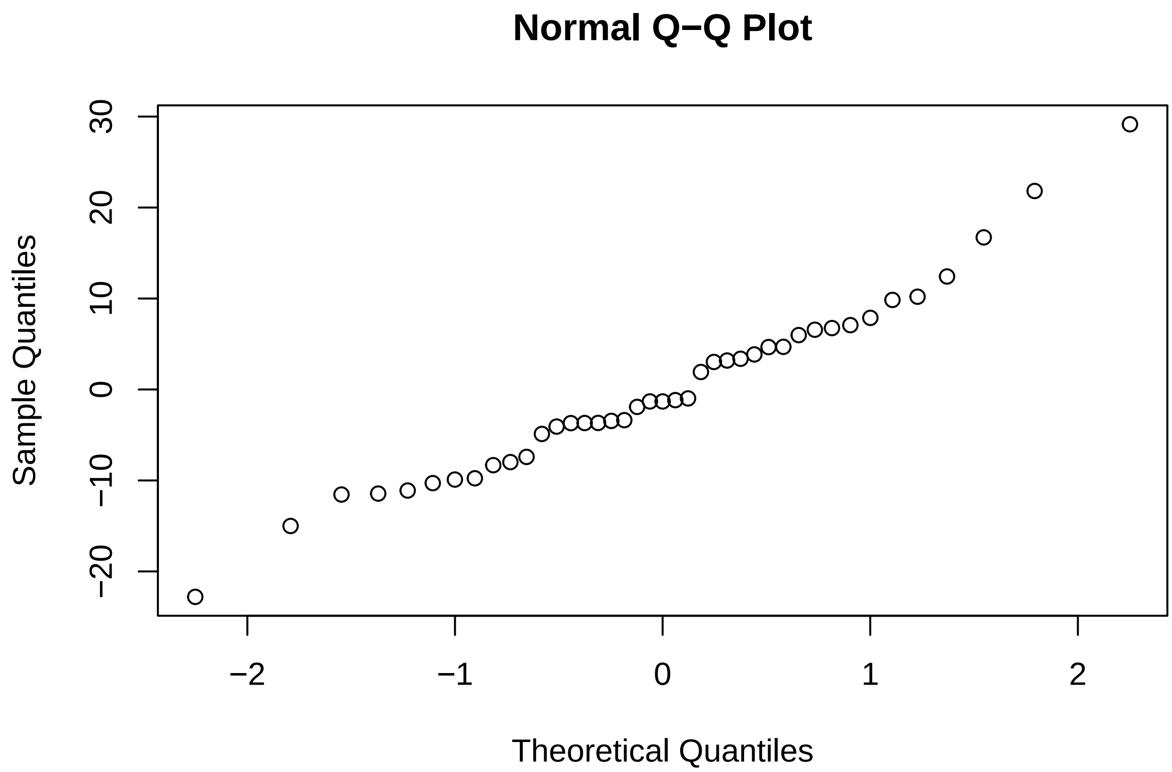
```
# Reg4
Reg4 <-lm(X1500m ~ X400m+Discus, data=decathlon)
summary(Reg4)
```

```
##
## Call:
## lm(formula = X1500m ~ X400m + Discus, data = decathlon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.796  -7.410  -1.315   5.978  29.155
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.1393    76.0927   0.107  0.91538
## X400m         4.5007     1.4206   3.168  0.00302 **
## Discus        1.0734     0.4851   2.213  0.03300 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.29 on 38 degrees of freedom
## Multiple R-squared:  0.2617, Adjusted R-squared:  0.2228
## F-statistic: 6.734 on 2 and 38 DF,  p-value: 0.003138
```

3.1.3.4.1 1. Normality of the Error Term

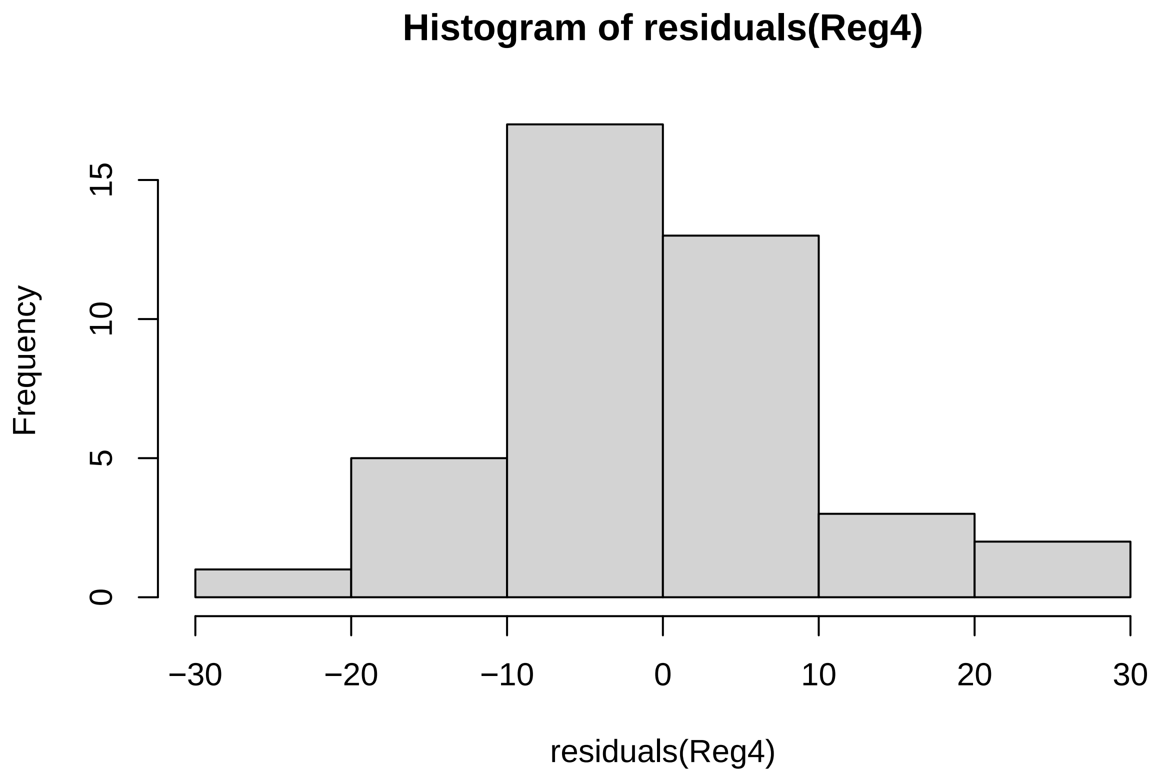
Using QQ plot

```
qqnorm(residuals(Reg4))
```



Using Histogram

```
hist(residuals(Reg4))
```



Shapiro Wilks Test

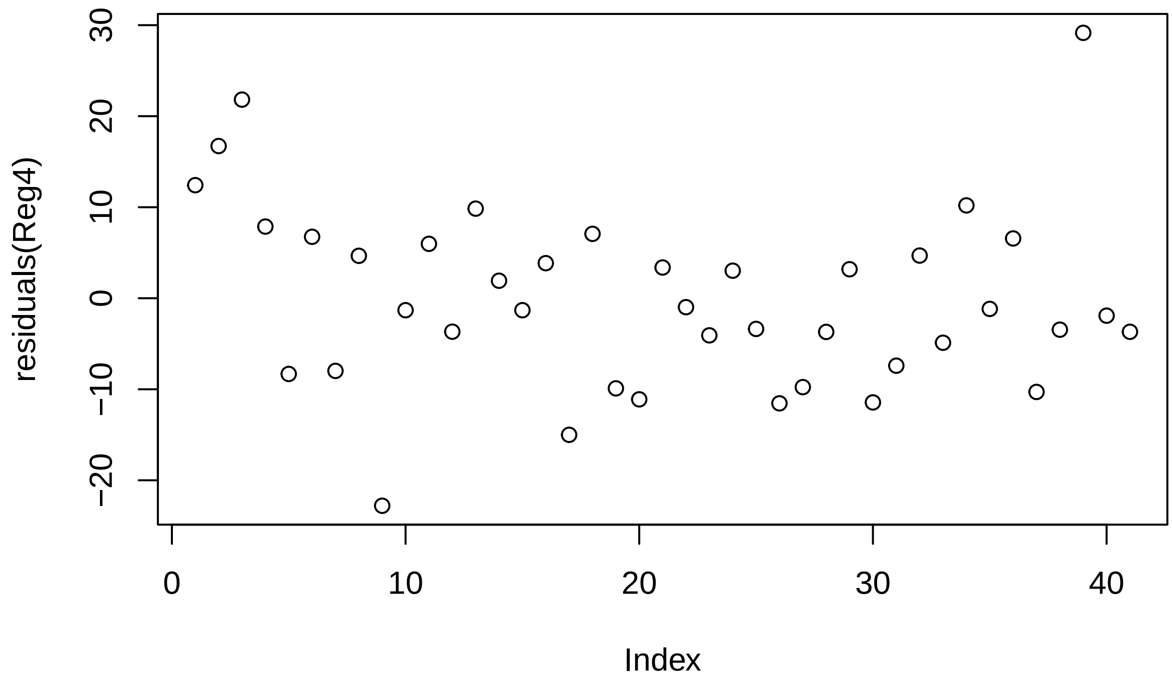
```
shapiro.test(residuals(Reg4))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(Reg4)
## W = 0.97243, p-value = 0.4124
```

3.1.3.4.2 2. Homogenity of Variance

Residual Analysis

```
plot(residuals(Reg4))
```



Breusch Pagan Test

```
bptest(Reg4)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Reg4
## BP = 4.7275, df = 2, p-value = 0.09407
```

```
dwtest(Reg4, alternative = "two.sided")
```

3.1.3.4.3 3. The independence of errors

```
##
##  Durbin-Watson test
##
## data:  Reg4
## DW = 1.8923, p-value = 0.6491
## alternative hypothesis: true autocorrelation is not 0
```

3.1.3.5 Regression 5

In the following, we can see how to perform the assumption tests for regression model Reg5.

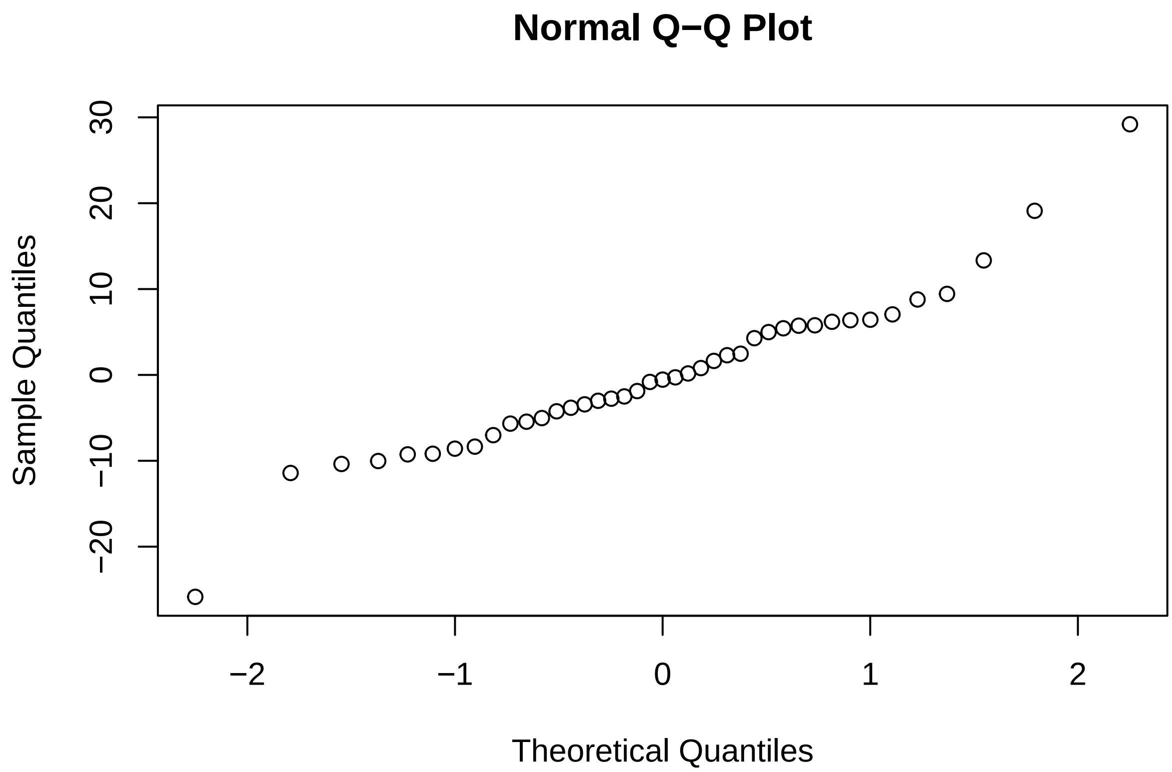
```
# Reg5
Reg5 <-lm(X1500m ~ X400m + Discus + Pole.vault,, data=decathlon)
summary(Reg5)
```

```
##
## Call:
## lm(formula = X1500m ~ X400m + Discus + Pole.vault, data = decathlon)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -25.8471  -5.4464  -0.5458   5.7325  29.1929
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -85.1261    79.7382  -1.068 0.292631
## X400m         4.8393     1.3324   3.632 0.000847 ***
## Discus        1.2635     0.4588   2.754 0.009071 **
## Pole.vault   14.2863     5.5527   2.573 0.014231 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.605 on 37 degrees of freedom
## Multiple R-squared:  0.3737, Adjusted R-squared:  0.3229
## F-statistic:  7.36 on 3 and 37 DF,  p-value: 0.0005479
```

3.1.3.5.1 1. Normality of the Error Term

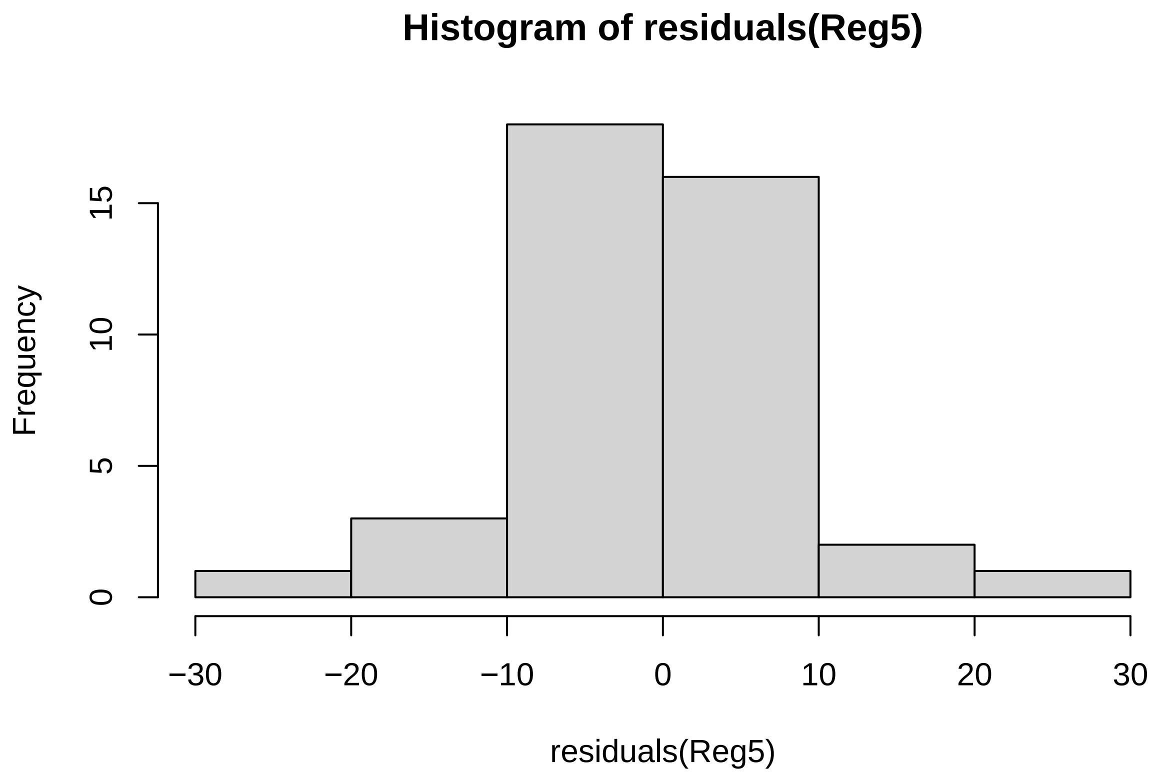
Using QQ plot

```
qqnorm(residuals(Reg5))
```



Using Histogram

```
hist(residuals(Reg5))
```



Shapiro Wilks Test

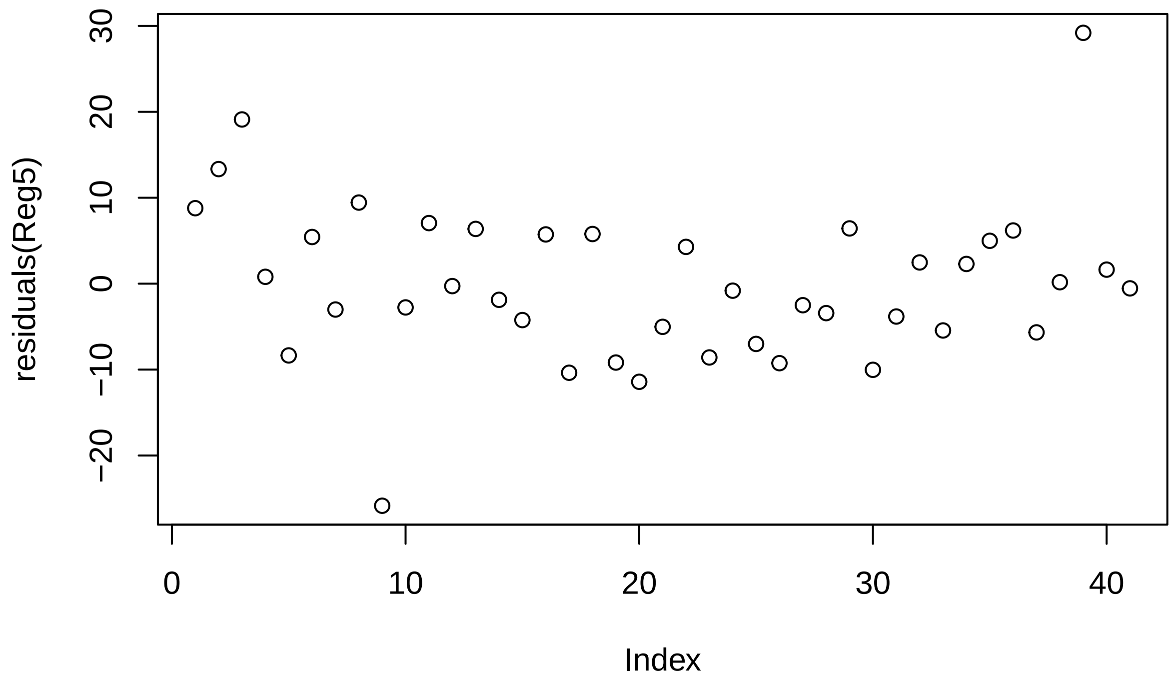
```
shapiro.test(residuals(Reg5))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(Reg5)
## W = 0.95122, p-value = 0.07727
```

3.1.3.5.2 2. Homogenity of Variance

Residual Analysis

```
plot(residuals(Reg5))
```



Breusch Pagan Test

```
bptest(Reg5)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Reg5
## BP = 4.4675, df = 3, p-value = 0.2152
```

```
dwtest(Reg5, alternative = "two.sided")
```

3.1.3.5.3 3. The independence of errors

```
##
##  Durbin-Watson test
##
## data:  Reg5
## DW = 1.9294, p-value = 0.7275
## alternative hypothesis: true autocorrelation is not 0
```

When comparing the results from the different regression models, we can determine that model Reg5 is the best of the defined models, because it is able to explain 37.4% of the variance of the X1500m and it has the least residual error (9.605) out of all the models.

Afterwards we want to use our linear expression to predict the behavior of an athlete, but we will check first if the model is accurate. We can look at the confidence intervals of each athlete and compare it to the real value of the X1500m.

3.1.4 Confidence intervals for the parameters of the model

```
confint(Reg5)
```

```
##                   2.5 %    97.5 %
## (Intercept) -246.6910613 76.438783
## X400m          2.1395872  7.539068
## Discus         0.3339384  2.192976
## Pole.vault     3.0355029 25.537157
```

###Predicted Values of the Response

```
yhat<-Reg5$fitted.values
yhat
```

```
##      SEBRLE        CLAY      KARPOV     BERNARD      YURKOV     WARNERS
##    282.9144    288.1628    281.0871    279.3029    284.7518    272.6692
##   ZSIVOCZKY    McMULLEN   MARTINEAU       HERNU      BARRAS        NOOL
##    271.0096    275.6604    287.9471    287.8666    274.9468    266.8819
## BOURGUIGNON      Sebrle        Clay      Karpov       Macey     Warners
##    285.3286    281.8910    286.2352    272.3775    275.7911    272.2704
##   Zsivoczky       Hernu        Nool     Bernard    Schwarzl   Pogorelov
##    278.7213    275.7705    281.3560    272.0296    282.1476    288.4452
##  Schoenbeck      Barras       Smith   Averyanov    Ojaniemi     Smirnov
##    285.8338    276.3429    275.2448    274.4463    269.2792    273.3380
##          Qi       Drews Parkhomenko       Terek       Gomez        Turi
##    276.4548    271.7386    283.3864    288.0674    264.7120    283.8198
##     Lorenzo   Karlivans Korkizoglou       Uldal     Casarsa
##    268.7510    278.5002    287.8071    280.0674    296.6658
```

3.1.5 Predicted Values of the Response with Their confidence Levels

```
predict.lm(Reg5,interval="confidence")
```

```
##                  fit      lwr      upr
## SEBRLE      282.9144 278.6811 287.1477
## CLAY        288.1628 281.0726 295.2529
## KARPOV      281.0871 274.7906 287.3836
## BERNARD     279.3029 271.8786 286.7273
## YURKOV      284.7518 280.5147 288.9888
## WARNERS     272.6692 267.4577 277.8806
## ZSIVOCZKY   271.0096 265.3096 276.7096
## McMULLEN    275.6604 270.7582 280.5626
## MARTINEAU   287.9471 282.7500 293.1441
```

```
## HERNU       287.8666 282.6344 293.0987
## BARRAS      274.9468 271.1475 278.7461
## NOOL        266.8819 259.5530 274.2108
## BOURGUIGNON 285.3286 278.7819 291.8753
## Sebrle      281.8910 275.3605 288.4215
## Clay        286.2352 279.7110 292.7595
## Karpov      272.3775 262.3040 282.4509
## Macey       275.7911 269.6572 281.9250
## Warners     272.2704 266.7289 277.8119
## Zsivoczky   278.7213 275.3830 282.0596
## Hernu       275.7705 271.9109 279.6300
## Nool        281.3560 273.4096 289.3024
## Bernard     272.0296 266.7240 277.3351
## Schwarzl    282.1476 277.1603 287.1348
## Pogorelov   288.4452 283.0980 293.7923
## Schoenbeck  285.8338 281.2649 290.4027
## Barras      276.3429 272.7423 279.9436
## Smith       275.2448 267.4733 283.0163
## Averyanov   274.4463 269.3741 279.5185
## Ojaniemi    269.2792 263.6230 274.9355
## Smirnov     273.3380 269.3661 277.3099
## Qi          276.4548 272.2385 280.6711
## Drews       271.7386 265.5105 277.9667
## Parkhomenko 283.3864 278.1898 288.5830
## Terek       288.0674 281.0388 295.0960
## Gomez       264.7120 257.5254 271.8986
## Turi        283.8198 276.6497 290.9899
## Lorenzo     268.7510 262.5955 274.9066
## Karlivans   278.5002 273.6187 283.3817
## Korkizoglou 287.8071 282.2787 293.3355
## Uldal       280.0674 274.5611 285.5736
## Casarsa     296.6658 285.1383 308.1932
```

We can also compare the Root Mean Squared Error (RMSE) of the training and the test data

3.1.6 Model Validation

```r
n <- nrow(decathlon)
train.sample <- sample(1:n, round(0.67*n))
train.set <- decathlon[train.sample, ]
test.set <- decathlon[-train.sample, ]

train.Reg5 <- lm(X1500m ~ X400m + Discus + Pole.vault, data=decathlon)
summary(train.Reg5)
```

3.1.6.1 Test-Train Models

```
##
## Call:
## lm(formula = X1500m ~ X400m + Discus + Pole.vault, data = decathlon)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -25.8471  -5.4464  -0.5458   5.7325  29.1929
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -85.1261    79.7382  -1.068 0.292631
## X400m         4.8393     1.3324   3.632 0.000847 ***
## Discus        1.2635     0.4588   2.754 0.009071 **
## Pole.vault   14.2863     5.5527   2.573 0.014231 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.605 on 37 degrees of freedom
## Multiple R-squared:  0.3737, Adjusted R-squared:  0.3229
## F-statistic:  7.36 on 3 and 37 DF,  p-value: 0.0005479
```

```r
yhat<-predict(train.Reg5, test.set, interval="prediction")
yhat
```

```
##                  fit      lwr      upr
## SEBRLE      282.9144 262.9975 302.8313
## CLAY        288.1628 267.4497 308.8759
## WARNERS     272.6692 252.5217 292.8166
## BOURGUIGNON 285.3286 264.7952 305.8620
## Clay        286.2352 265.7090 306.7615
## Karpov      272.3775 250.4632 294.2918
## Warners     272.2704 252.0350 292.5058
## Hernu       275.7705 255.9297 295.6113
## Nool        281.3560 260.3344 302.3776
## Bernard     272.0296 251.8576 292.2016
## Schwarzl    282.1476 262.0569 302.2382
## Pogorelov   288.4452 268.2621 308.6282
## Qi          276.4548 256.5415 296.3681
## Parkhomenko 283.3864 263.2427 303.5300
```

```r
y<-test.set$X1500m
error<-cbind(yhat[,1,drop=FALSE],y,(y-yhat[,1])^2)
sqr_err<-error[,3]
mse<-mean(sqr_err)
```

3.1.7 Root Mean Square Error

```r
RMSE<-sqrt(mse/(nrow(test.set)))
RMSE
```

```
## [1] 1.892619
```

```r
names(train.Reg5)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```
RMSE_train<- sqrt(mean((train.Reg5$residuals)^2)/nrow(train.set))
RMSE_train
```

```
## [1] 1.75602
```

In both cases, we see that the test data does not deviate much from the predicted values, so we can say that Reg5 is accurate.

4 Question 4

```
#install.packages('HSAUR2')
#install.packages("FactoMineR")
library(FactoMineR)
library(lmtest)

data("heptathlon", package = "HSAUR2")
summary(heptathlon)
```

```
##     hurdles         highjump          shot          run200m
##  Min.   :12.69   Min.   :1.500   Min.   :10.00   Min.   :22.56
##  1st Qu.:13.47   1st Qu.:1.770   1st Qu.:12.32   1st Qu.:23.92
##  Median :13.75   Median :1.800   Median :12.88   Median :24.83
##  Mean   :13.84   Mean   :1.782   Mean   :13.12   Mean   :24.65
##  3rd Qu.:14.07   3rd Qu.:1.830   3rd Qu.:14.20   3rd Qu.:25.23
##  Max.   :16.42   Max.   :1.860   Max.   :16.23   Max.   :26.61
##     longjump        javelin         run800m          score
##  Min.   :4.880   Min.   :35.68   Min.   :124.2   Min.   :4566
##  1st Qu.:6.050   1st Qu.:39.06   1st Qu.:132.2   1st Qu.:5746
##  Median :6.250   Median :40.28   Median :134.7   Median :6137
##  Mean   :6.152   Mean   :41.48   Mean   :136.1   Mean   :6091
##  3rd Qu.:6.370   3rd Qu.:44.54   3rd Qu.:138.5   3rd Qu.:6351
##  Max.   :7.270   Max.   :47.50   Max.   :163.4   Max.   :7291
```

4.1 Heptathlon dataset

```
res_PCA<-PCA(heptathlon,scale=TRUE, graph=FALSE) # by default scale=TRUE
res_PCA$eig
```

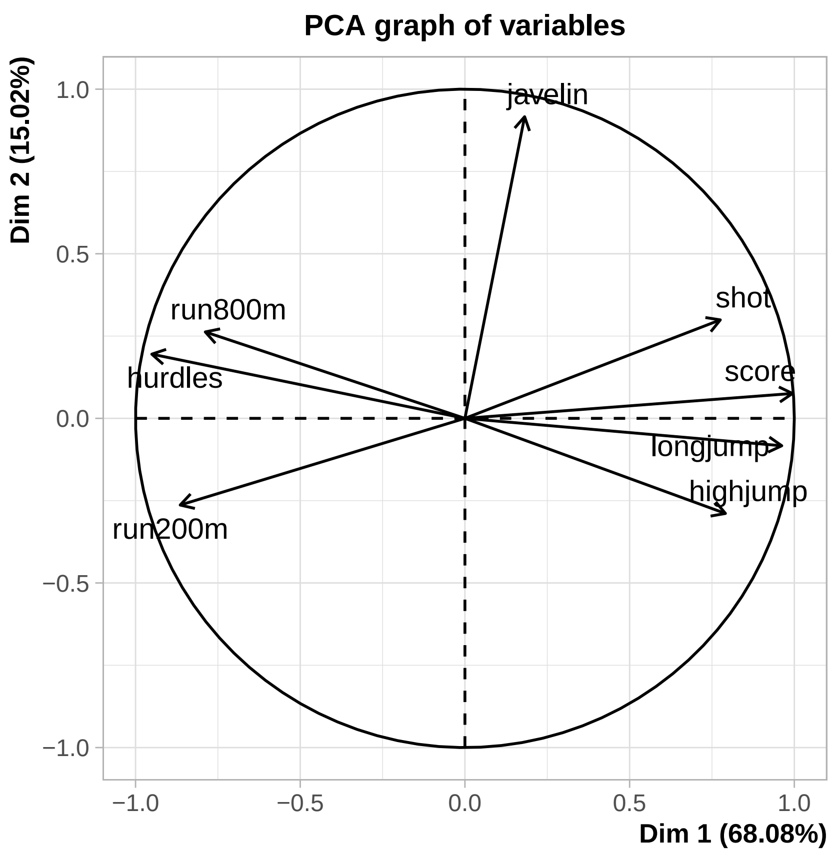
```
##        eigenvalue percentage of variance cumulative percentage of variance
## comp 1 5.446267395            68.07834244                          68.07834
## comp 2 1.201724419            15.02155524                          83.09990
## comp 3 0.521035564             6.51294454                          89.61284
## comp 4 0.457188388             5.71485485                          95.32770
```

```
## comp 5 0.246988048              3.08735060                       98.41505
## comp 6 0.073754918              0.92193647                       99.33698
## comp 7 0.049036966              0.61296208                       99.94995
## comp 8 0.004004302              0.05005378                      100.00000
```

We instantly see that in order to surpass 70% in cumulative percentage of variance, we should retain the two first components. By retaining the two first directions ("comp1" and "comp2"), we explain 83% of the variance of the data, which is satisfactory. Although, it's important t onotice that the first component already explains 68% (almost 70%) of the variance; so we could almost just retain "comp1".

We now project each variable onto these two components. By doing so, we obtain the following Variable Chart :

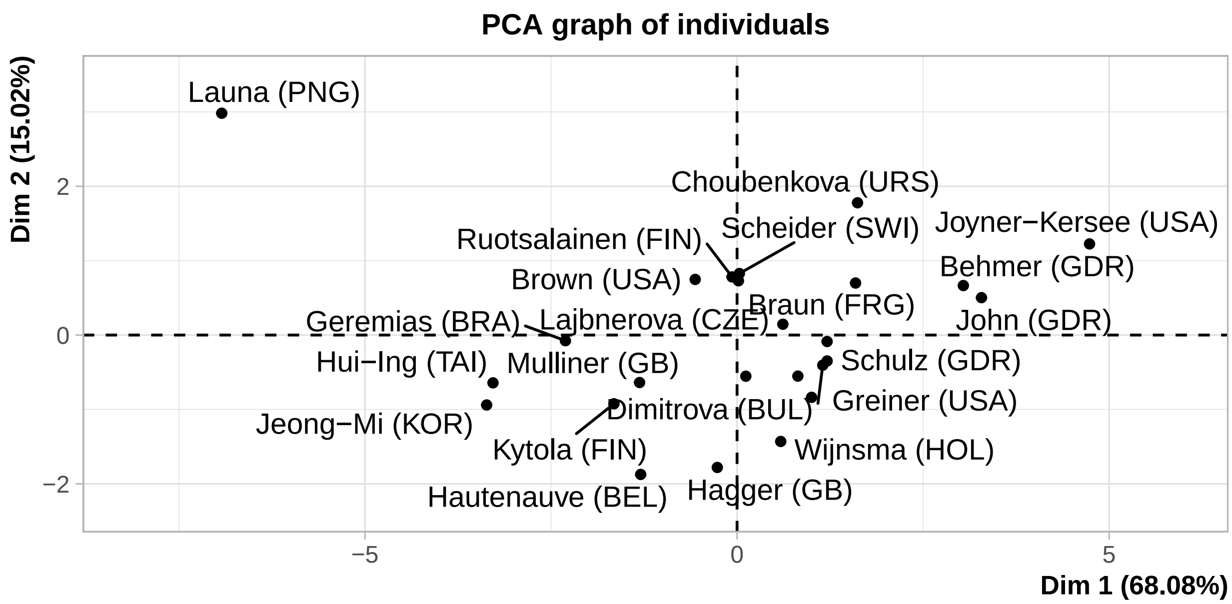
```
#res_PCA$var$coord
#res_PCA$var$cor
plot(res_PCA,choix="var")
```



By observing this chart, we can see that our model is not very efficient. The only variable that seems to contribute the the second dimension is "javelin". All the other variables have a low coordinate along this axis and hence hardly contribute to it. The whole point of PCA being to summarize data, here our second dimension is essentially explaining "javelin" only. This is not a good model. The Individual Chart seems to confirm this observation :

```
#res_PCA$ind$coord
plot(res_PCA,choix = "ind")
```

```
## Warning: ggrepel: 4 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



We see that Laura (PNG) and Choubenkova (URS) are the two individuals with the highest coordinate along Dim 2, and Hautenauve (BEL) and Hagger (GB) the two with the lowest. If we look at the javelin results in the data set, Laura and Choubenkova are the top performers, Hautenauve and Hagger are the worst. This confirms that Dim2 is esentially a direct reflection of the javelin variable. Moreover, if we take a look at the first dimension, all the variables (except "javelin") seem to contribute equally to it : in magnitude, their coordinate along Dim 1 is approximately 0.8, which is quite high. It is quite hard therefore to identify a possible interpretation of this component, hence once again the model is not very functional.

4.2 Principal Component Regression

We shall now construct a regression of the variable "score" using the two components we previously retained. We do so using the following R script :

```r
heptathlon$PC1<-res_PCA$ind$coord[,1]
heptathlon$PC2<-res_PCA$ind$coord[,2]
hepta_reg<-lm(score~PC1 + PC2, data=heptathlon)
summary(hepta_reg)
```

```
##
## Call:
## lm(formula = score ~ PC1 + PC2, data = heptathlon)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -177.282  -10.346    4.355   22.759   44.642
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6090.600      8.689 700.967  < 2e-16 ***
## PC1          237.343      3.723  63.747  < 2e-16 ***
## PC2           38.420      7.926   4.847 7.63e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.44 on 22 degrees of freedom
## Multiple R-squared:  0.9946, Adjusted R-squared:  0.9942
## F-statistic:  2044 on 2 and 22 DF,  p-value: < 2.2e-16
```

Let's also make sure we are actually in a position to execute such a regression. To do so, let's check the assumptions that are : • Normality of the residuals (checked via a Shapiro test) • Homogeneity of the residuals variance, a.k.a homoscedasticity (checked via a Breusch-Pagan test) • Independence of residuals error terms (checked via a Durbin-Watson test)

4.2.1 Normality

```
shapiro.test(residuals(hepta_reg))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(hepta_reg)
## W = 0.64854, p-value = 1.592e-06
```

The p-value being inferior to 0.05, we reject H0 and therefore conclude that the residuals are not normally distributed, which is problematic.

###Homogeneity

```
bptest(hepta_reg)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  hepta_reg
## BP = 0.48675, df = 2, p-value = 0.784
```

The p-value being superior to 0.05, we accept H0 and therefore conclude we do indeed have homoscedasticity.

4.2.2 Homogeneity

```r
dwtest(hepta_reg)
```

```
##
##  Durbin-Watson test
##
## data:  hepta_reg
## DW = 2.3213, p-value = 0.7002
## alternative hypothesis: true autocorrelation is greater than 0
```

The p-value being again superior to 0.05, we accept H0 and conclude that the residuals are independent.

4.3 Prediction

The data associated with this regression is the following :

```r
n <- nrow(heptathlon)
train.sample1 <- sample(1:n, round(0.67*n))
train.set1 <- heptathlon[train.sample1, ]
test.set1 <- heptathlon[-train.sample1, ]

train.model <- lm(score ~ PC1+PC2 , data = heptathlon[train.sample1,])
summary(train.model)
```

```
##
## Call:
## lm(formula = score ~ PC1 + PC2, data = heptathlon[train.sample1,
##     ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -169.904   -6.003    8.596   27.413   51.243
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6085.486     12.693 479.446  < 2e-16 ***
## PC1          235.835      5.289  44.590  < 2e-16 ***
## PC2           39.725     12.955   3.066  0.00837 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 51.87 on 14 degrees of freedom
## Multiple R-squared:  0.9933, Adjusted R-squared:  0.9923
## F-statistic:  1038 on 2 and 14 DF,  p-value: 6.067e-16
```

We see that the R^2 value for this model is 99% (0.9984) which is very high and hence suggests that the model is very efficient in predicting. Our predictors, PC1 and PC2 are capable of explaining 99% of the variation in scores.

```r
yhat<-predict(train.model, heptathlon[-train.sample1,], interval="prediction")
yhat
```

```
##                          fit      lwr      upr
## Joyner-Kersee (USA) 7251.185 7115.433 7386.937
## Choubenkova (URS)   6537.893 6409.773 6666.013
## Schulz (GDR)        6367.349 6251.656 6483.042
## Bouraga (URS)       6256.194 6140.229 6372.158
## Ruotsalainen (FIN)  6100.615 5984.260 6216.970
## Hagger (GB)         5952.116 5826.723 6077.509
## Mulliner (GB)       5751.175 5633.980 5868.369
## Hautenauve (BEL)    5705.635 5577.342 5833.928
```

We finally compute the residual mean squared error to confirm yhe choice of this model

```r
y<-test.set1$score

error<-cbind(yhat[,1,drop=FALSE],y,(yhat[,1]-y)^2)
sqr_err<-error[,3]
mse<-mean(sqr_err)
RMSE<-sqrt(mse/(nrow(test.set1)))
RMSE
```

```
## [1] 8.721258
```

Moreover, when we compute the RMSE (see R script), we get that a low value , which also seems very low since it corresponds to the average deviation between the predicted score and the real score, and the value of a score is usually in the thousands (6137,5746 etc...). These two metrics being analyzed, we can conclude that the model seems efficient in predicting a score.