“TYPE OF GLASS” REPORT

Data Analysis

PROF: ANTONIO PUNZO

SIMONE ALESSIA

1000037243

Index

Introduction

Used libraries

I suggest installing and uploading the following libraries which have been used in this report before starting the analysis:

```
library(ggplot2)
library(gamlss)
library(labstatR)
library(gamlss.mx)
library(vcd)
library(gridExtra)
library(factoextra)
library(clustertend)
library(KernSmooth)
library(rgl)
library(stats)
library(NbClust)
library(fpc)
library(pvclust)
library(cluster)
library(mclust)
library(clValid)
library(FactoMineR)
```

Description

The aim of this report is the analysis of fragments of glass collected in forensic work. The dataset comes from the UCI Machine Learning Repository[1] and is also present in the library 'MASS' as 'fgl'.

```
str(glass)
```

```
## 'data.frame':    214 obs. of  10 variables:
##  $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
##  $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
##  $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
##  $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
##  $ Si  : num  71.8 72.7 73 72.6 73.1 ...
##  $ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
##  $ Ca  : num  8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...
##  $ Ba  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
##  $ Type: int  1 1 1 1 1 1 1 1 1 1 ...
```

[1] Glass Classification | Kaggle

This dataset contains 214 observations of 9 continuous variables and 1 categorical variable, the variable ID has been set as row names:

- **RI:** Refractive index which measures the bending of a light ray when passing from one medium to another.
- **Na:** Sodium.
- **Mg:** Magnesium.
- **Al:** Aluminium.
- **Si:** Silicon.
- **K:** Potassium.
- **Ca:** Calcium.
- **Ba:** Barium.
- **Fe:** Iron.
- **Type:** Type of glass of each observation ordered from 1 to 7 (the 4th type is not present in this dataset) which correspond to building windows float processed, building windows non-float processed, vehicle windows float processed, vehicle windows non-float processed, containers, tableware, headlamps.

The chemical compositions are measured as the weight percentage in the corresponding oxide. FP refers to the float process used to make the glass.

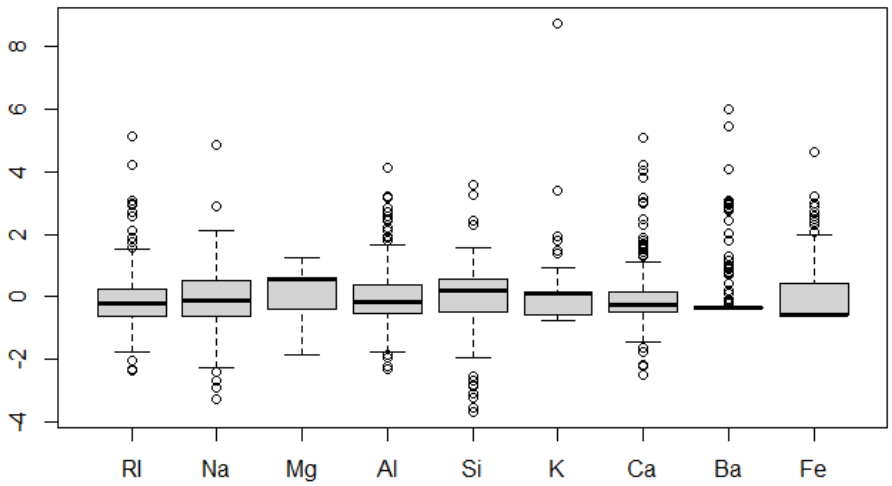
Throw the head() function I show the first six rows of the dataset to have a quick view of it:

```
head(glass)
```

```
##        RI    Na   Mg   Al    Si    K   Ca Ba   Fe Type
## 1 1.52101 13.64 4.49 1.10 71.78 0.06 8.75  0 0.00    1
## 2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83  0 0.00    1
## 3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78  0 0.00    1
## 4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22  0 0.00    1
## 5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07  0 0.00    1
## 6 1.51596 12.79 3.61 1.62 72.97 0.64 8.07  0 0.26    1
```

Throw the following code I show the graphical view of descriptive statistics as better understand all the value in a bi-dimensional space throws the boxplot:

```
scaled_glass <- apply(glass[-10],2,scale)
boxplot(scaled_glass)
```



The standardized dataset allows me to compare all the observations together: the black lines within the boxes represent the median, the top and bottom horizontal lines of the boxes represent the first and third quartiles, and the black dots are the outliers. The "Ba" variable has a lot of outliers, instead of the variable "Mg" which has no outliers.

In the next chapter, I analyze each variable one by one throw the univariate analysis. To have easy access to the variable I used attach() function and, to ensure the same random generation computed in this report, I recommend setting the seed:

```
attach(glass)
set.seed(1234)
```

Univariate Analysis

RI

It is a continuous quantitative variable that assumes values within the range [1.511,1.534].

```
summary(RI)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.511   1.517   1.518   1.518   1.519   1.534
```

According to the position indices, the observations have the smallest value of the refractive index of 1.511 and the greatest value of 1.534, at least 25% has an index of 1.517, at least 50% has a refractive index value of 1.518 and at least 75% has a refractive index value of 1.519. The whole observations' refractive index value is above 1.518 and the interquartile distance is 0.002.

In detail, the outliers are 17 out of 214 and they are:

```
boxplot.stats(RI)$out

##  [1] 1.52667 1.52320 1.51215 1.52725 1.52410 1.52475 1.53125 1.53393 1.52664
## [10] 1.52739 1.52777 1.52614 1.52369 1.51115 1.51131 1.52315 1.52365
```

By computing the shape and variability indices I can predict the shape of the curve:

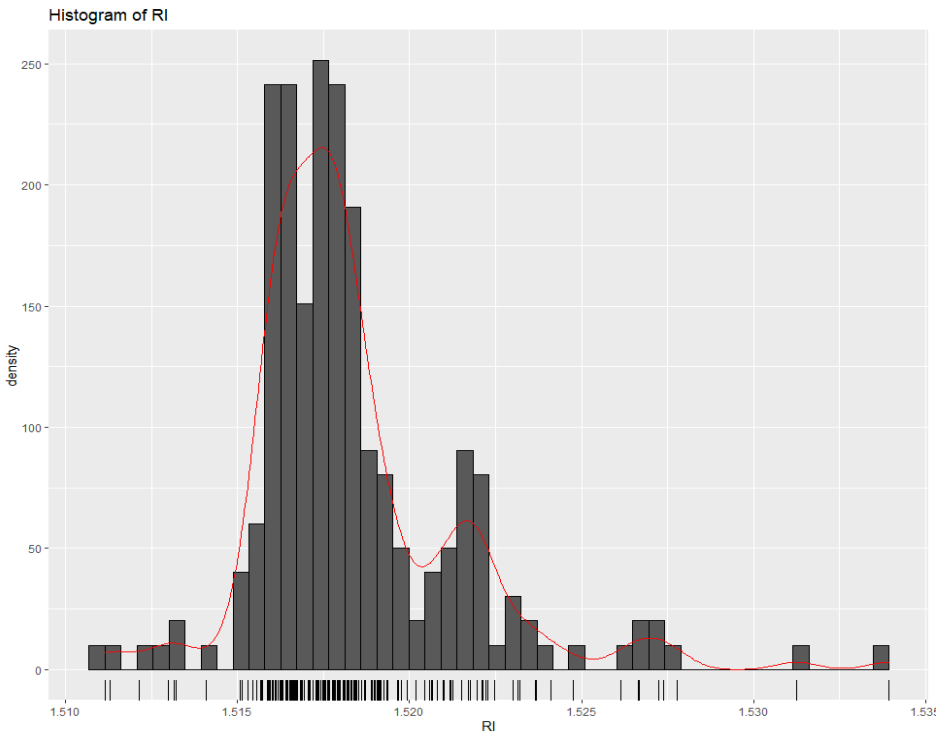
```
labstatR:::skew(RI)

## [1] 1.614015

labstatR:::kurt(RI)

## [1] 7.789354

sd(RI)

## [1] 0.003036864
```

The coefficient of skewness is positive, so the distribution is right-skewed, this means that the graph is asymmetric, and the values are concentrated on the left side of the graph.

The Kurtosis index value is greater than 3, so we can assume that the distribution is leptokurtic with a higher shape than a normal distribution.

The standard deviation is very close to 0, this means that the values are concentrated near the mean. This is confirmed by plotting the histogram of the refractive index and highlighting the density function:

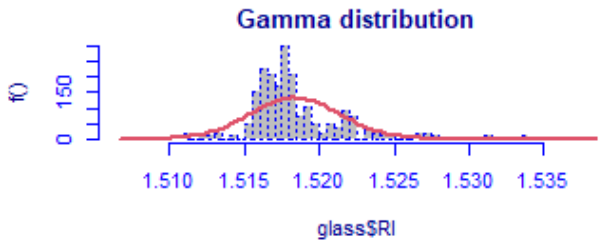
```
ggplot2:::ggplot(glass, aes(x=RI))+
  geom_histogram(aes(y=..density..), color='black', bins=50)+
  geom_density(color='red')+
  geom_rug()+
  ggtitle('Histogram of RI')
```

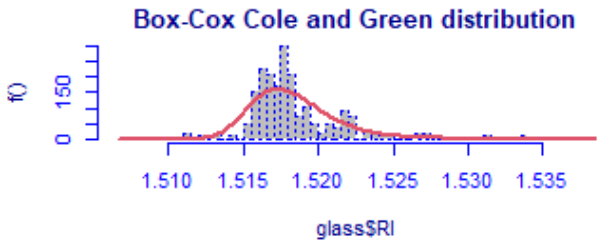


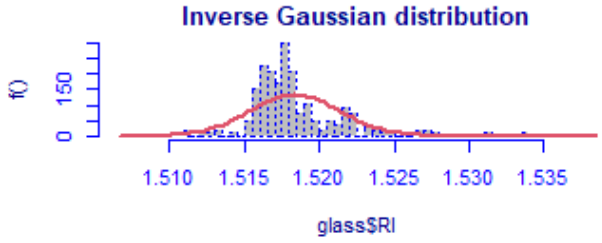
Single parametric distributions

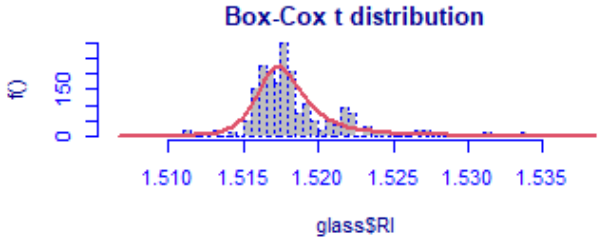
Throw the following code I will try to establish what is the best distribution which fits the variable "RI". I first introduce five possible distributions which handle the value of positive real line and for each, I will compute the value of the parameters which, in this case, are called mu, sigma, nu, and tau:

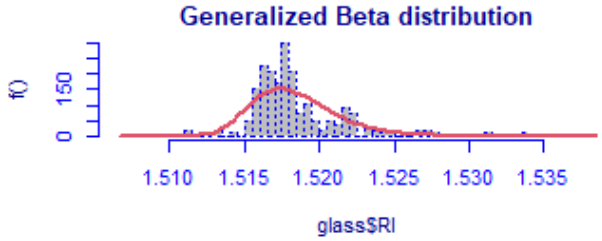
```
par(mfrow=c(3,2))
fit_ri.GA <- gamlss:::histDist(glass$RI, family=GA, nbins = 50, main="Gamma distribution")
fit_ri.BCCG <- gamlss:::histDist(glass$RI, family=BCCG, nbins = 50, main="Box-Cox Cole and
Green distribution")
fit_ri.IG <- gamlss:::histDist(glass$RI, family=IG, nbins = 50, main="Inverse Gaussian dis
tribution")
fit_ri.BCT <- gamlss:::histDist(glass$RI, family=BCT, nbins = 50, main="Box-Cox t distribu
tion")
fit_ri.GB2 <- gamlss:::histDist(glass$RI, family=GB2, nbins = 50, main="Generalized Beta d
istribution")
```












```
fit_ri.GA$df.fit
## [1] 2
fitted(fit_ri.GA, "mu")[1]
## [1] 1.518365
fitted(fit_ri.GA, "sigma")[1]
## [1] 0.001993293

fit_ri.BCCG$df.fit
## [1] 3
fitted(fit_ri.BCCG, "mu")[1]
## [1] 1.51794
fitted(fit_ri.BCCG, "sigma")[1]
## [1] 0.001687433
fitted(fit_ri.BCCG, "nu")[1]
## [1] -163.3554

fit_ri.IG$df.fit
## [1] 2
fitted(fit_ri.IG, "mu")[1]
## [1] 1.518365
fitted(fit_ri.IG, "sigma")[1]
## [1] 0.001623157

fit_ri.BCT$df.fit
## [1] 4
fitted(fit_ri.BCT, "mu")[1]
## [1] 1.517957
fitted(fit_ri.BCT, "sigma")[1]
```

```
## [1] 0.00128292
fitted(fit_ri.BCT, "nu")[1]
## [1] -424.8137
fitted(fit_ri.BCT, "tau")[1]
## [1] 2.562849

fit_ri.GB2$df.fit
## [1] 4
fitted(fit_ri.GB2, "mu")[1]
## [1] 1.469147
fitted(fit_ri.GB2, "sigma")[1]
## [1] 396.0879
fitted(fit_ri.GB2, "nu")[1]
## [1] 850752.5
fitted(fit_ri.GB2, "tau")[1]
## [1] 2.307737
```

I create a dataframe which show a ranking of AIC, BIC, and Log-likelihood values:

```
aic <- AIC(fit_ri.GA,fit_ri.IG,fit_ri.BCCG,fit_ri.BCT,fit_ri.GB2)
aic$BIC <- c(fit_ri.BCT$sbc,fit_ri.BCCG$sbc,fit_ri.GB2$sbc,fit_ri.IG$sbc,fit_ri.GA$sbc)
aic$LogLik <- c(logLik(fit_ri.BCT),logLik(fit_ri.BCCG),logLik(fit_ri.GB2),logLik(fit_ri.IG
),logLik(fit_ri.GA))
aic

##             df       AIC       BIC   LogLik
## fit_ri.BCT   4 -1951.670 -1938.206 979.8351
## fit_ri.BCCG  3 -1921.555 -1911.457 963.7774
## fit_ri.GB2   4 -1913.000 -1899.536 960.5000
## fit_ri.IG    2 -1871.461 -1864.729 937.7305
## fit_ri.GA    2 -1871.240 -1864.508 937.6200
```

The best distribution model that fits the "RI" variable is the Box-Cox-t distribution with 4 parameters, the minimum value of AIC and BIC, and the maximum value of Log-likelihood.

Goodness of fit

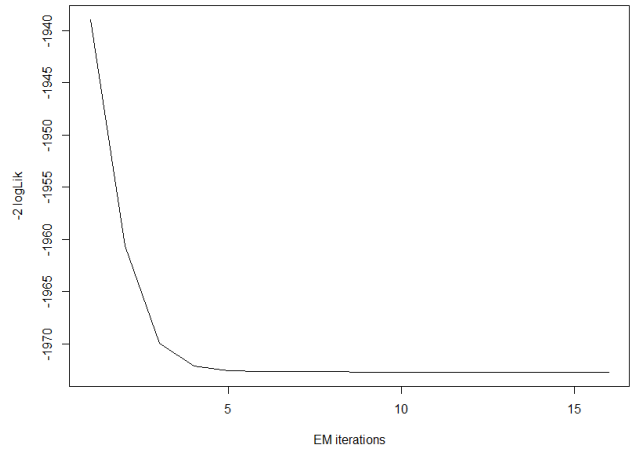
```
gamlss:::LR.test(fit_ri.IG,fit_ri.GB2)

##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
##        Null model: deviance= -1875.461 with  2 deg. of freedom
##  Altenative model: deviance= -1921 with  4 deg. of freedom
##
##  LRT = 45.53903 with 2 deg. of freedom and p-value= 1.292189e-10
```

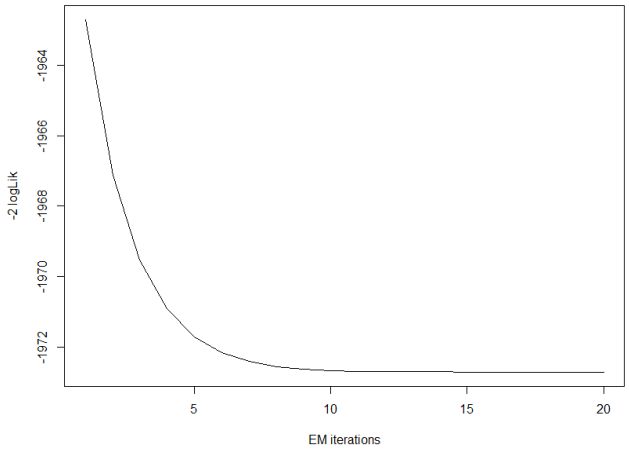
Throw the likelihood-ratio test, as the p-value is smaller than 0.05, I reject the Inverse Gaussian distribution and accept the Generalize Beta Type 2 distribution.

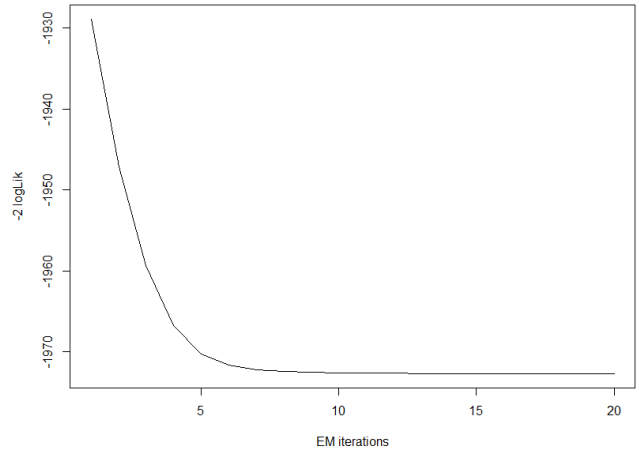
Mixture of distributions

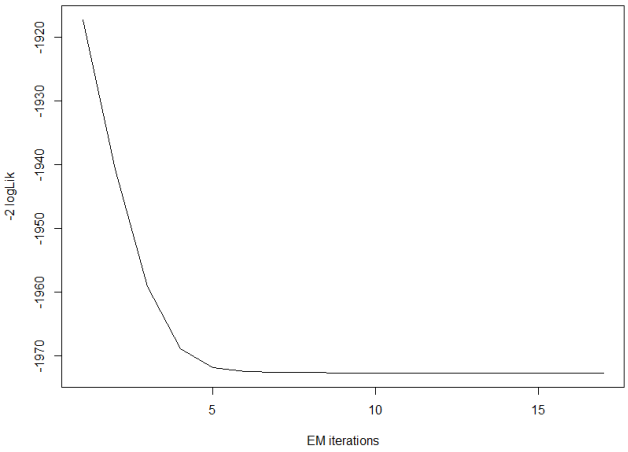
In the following code I compute a mixture of two Gamma distributions and, to find the best distribution, the algorithm is repeated 5 times:

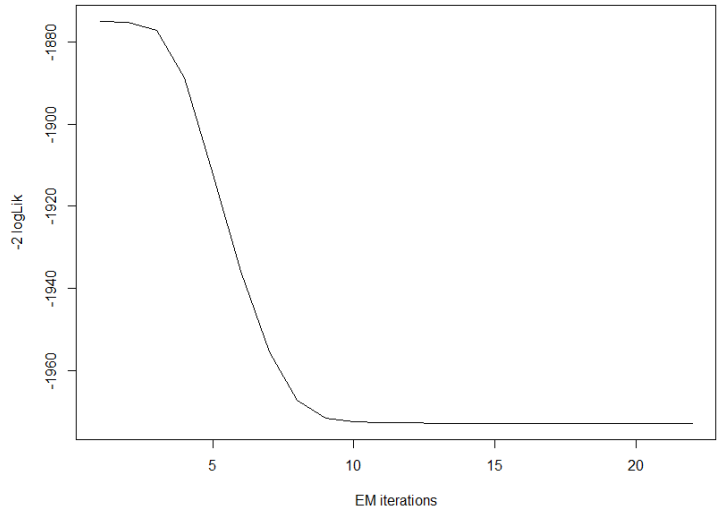
```
mix_ri.GA <- gamlss.mx:::gamlssMXfits(n = 5, glass$RI~1, family = GA, K = 2, data = NULL)
```



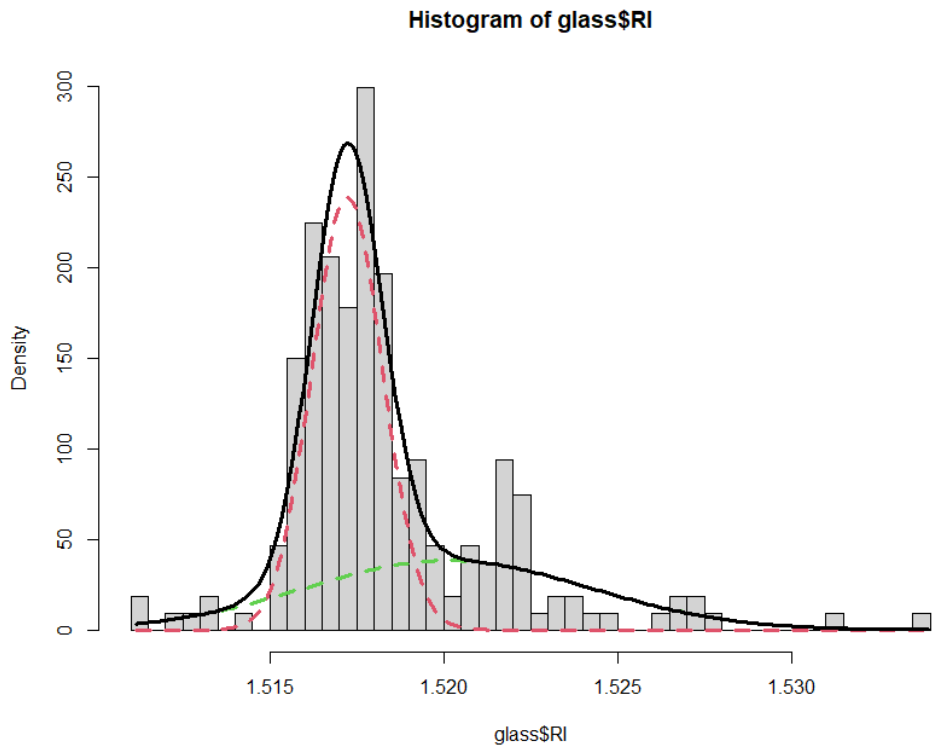









```
mu.hat1 <- exp(mix_ri.GA[["models"]][[1]][["mu.coefficients"]])
sigma.hat1 <- exp(mix_ri.GA[["models"]][[1]][["sigma.coefficients"]])
mu.hat2 <- exp(mix_ri.GA[["models"]][[2]][["mu.coefficients"]])
sigma.hat2 <- exp(mix_ri.GA[["models"]][[2]][["sigma.coefficients"]])

hist(glass$RI, breaks = 50,freq = FALSE)
lines(seq(min(glass$RI),max(glass$RI),length=length(glass$RI)),mix_ri.GA[["prob"]][1]*dGA(
seq(min(glass$RI),max(glass$RI),length=length(glass$RI)), mu = mu.hat1, sigma =
sigma.hat1),lty=2,lwd=3,col=2)
lines(seq(min(glass$RI),max(glass$RI),length=length(glass$RI)),mix_ri.GA[["prob"]][2]*dGA(
seq(min(glass$RI),max(glass$RI),length=length(glass$RI)), mu = mu.hat2, sigma =
sigma.hat2),lty=2,lwd=3,col=3)
lines(seq(min(glass$RI),max(glass$RI),length=length(glass$RI)),
mix_ri.GA[["prob"]][1]*dGA(seq(min(glass$RI),max(glass$RI),length=length(glass$RI)), mu =
mu.hat1, sigma = sigma.hat1) +
mix_ri.GA[["prob"]][2]*dGA(seq(min(glass$RI),max(glass$RI),length=length(glass$RI)), mu =
mu.hat2, sigma = sigma.hat2), lty = 1, lwd = 3, col = 1)
```




```
mix_ri.LOGNO$prob

## [1] 0.6081528 0.3918472
```

The first group explains about 60.81% of the distribution, the second group explains about 39.18% of the distribution, for this reason, I assume that two groups of distributions can be enough.

```
aic <- AIC(mix_ri.GA,fit_ri.BCT)
aic$BIC <- c(mix_ri.GA$sbc,fit_ri.BCT$sbc)
aic$LogLik <- c(logLik(mix_ri.GA), logLik(fit_ri.BCT))
aic

##            df       AIC       BIC   LogLik
## mix_ri.GA   5 -1962.715 -1945.885 986.3577
## fit_ri.BCT  4 -1951.670 -1938.206 979.8351
```

The best distribution model which fits the "RI" variable is the mixture of Gamma distributions with the minimum value of AIC and BIC and the maximum value of Log-likelihood.

Goodness of fit

```
gamlss:::LR.test(fit_ri.BCT,mix_ri.GA)

##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
```

```
##        Null model: deviance= -1959.67 with  4 deg. of freedom
##  Altenative model: deviance= -1972.75 with  5 deg. of freedom
##
##  LRT = 13.07964 with 1 deg. of freedom and p-value= 0.0002985225
```

According to the likelihood-ratio test, as the p-value is less than 0.05, the Box-Cox t distribution has been rejected and the mixture of Gamma distribution has been accepted.

Na

It's a continuous quantitative variable that can assume value within range [10.73,17.38].

```
summary(Na)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.73   12.91   13.30   13.41   13.82   17.38
```

According to the position indices, the observations have the smallest value of "Na" of 10.73 and the greatest value of 17.38, at least 25% has a value of 12.92, at least 50% has the value of 13.30 and at least 75% has a value of 13.82. The whole observations have a value above 13.41 and the interquartile distance is 0.91.

In detail, there are 7 outliers out of 214 and they are:

```
boxplot.stats(Na)$out

## [1] 11.45 10.73 11.23 11.02 11.03 17.38 15.79
```

Thanks to the variability and shape indices I can predict the shape of the curve:

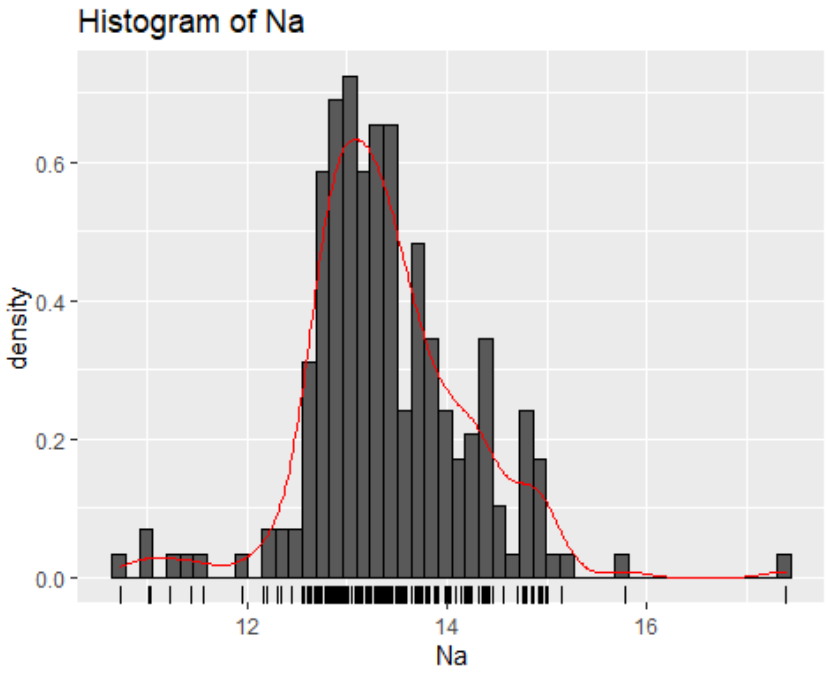
```
labstatR:::skew(Na)

## [1] 0.4509917

labstatR:::kurt(Na)

## [1] 5.953477

sd(Na)

## [1] 0.8166036
```

The coefficient of skewness is slightly positive, and the distribution is right-skewed, this means that the graph is asymmetric, and the values are concentrated on the centre-left side of the graph.

The Kurtosis index value is greater than 3, so we can assume that the distribution is leptokurtic with a higher shape concerning the normal distribution.

The standard deviation has a value close to 1, this means that the values are concentrated near the mean. This is confirmed by plotting the histogram of the "Na" and highlighting the density function:

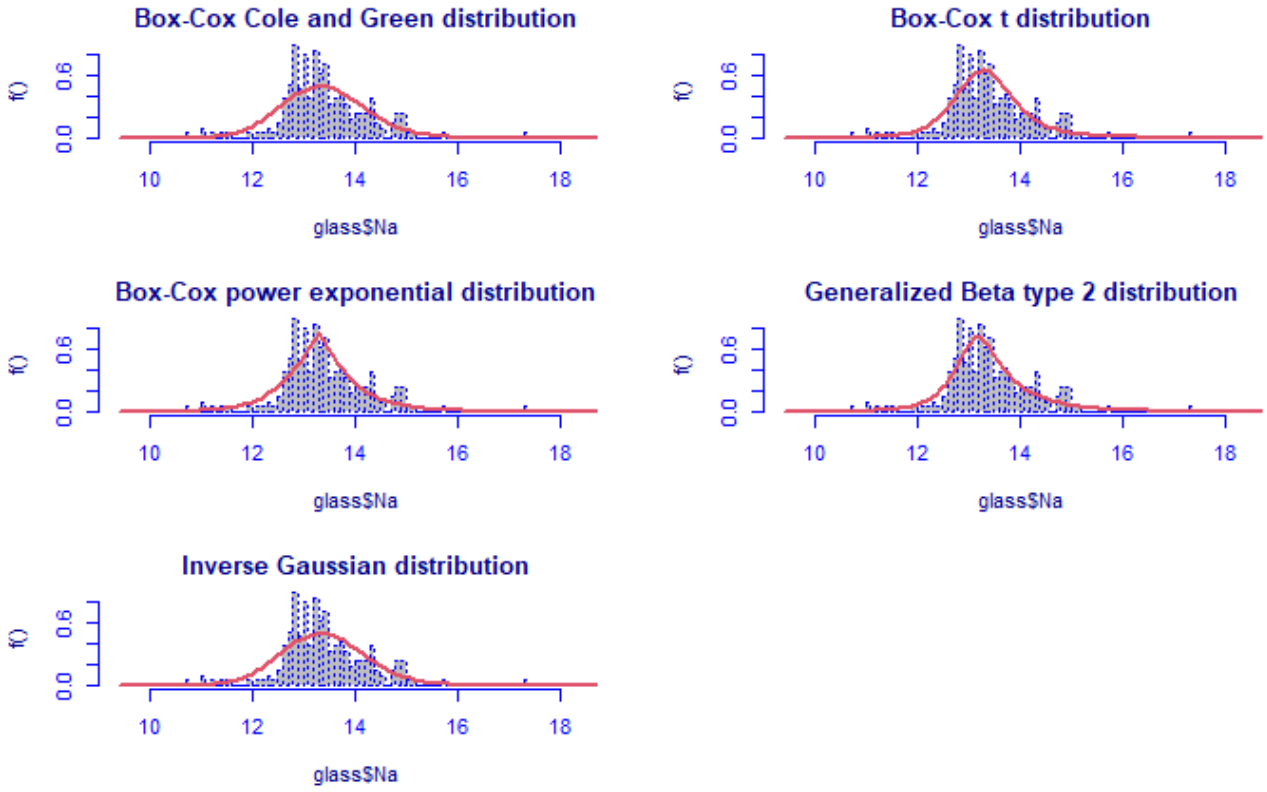
```
ggplot2:::ggplot(glass, aes(x=Na))+
  geom_histogram(aes(y=..density..), color='black',bins=50)+
  geom_density(color='red')+
  geom_rug()+
  ggtitle('Histogram of Na')
```



Single parametric distributions

Throw the following code I will try to establish what is the best distribution which fits the variable "Na". I first introduce five possible distributions which handle the value of the positive real line, and, for each, I will compute the value of the parameters which, in this case, are called mu, sigma, nu, and tau:

```
par(mfrow=c(3,2))

fit_na.BCCG <- gamlss:::histDist(glass$Na, family=BCCG, nbins = 50, main="Box-Cox Cole and
Green distribution")
fit_na.BCT <- gamlss:::histDist(glass$Na, family=BCT, nbins = 50, main="Box-Cox t
distribution")
fit_na.BCPE <- gamlss:::histDist(glass$Na, family=BCPE, nbins = 50, main="Box-Cox power
exponential distribution")
fit_na.GB2 <- gamlss:::histDist(glass$Na, family=GB2, nbins = 50, main="Generalized Beta
type 2 distribution")
fit_na.IG <- gamlss:::histDist(glass$Na, family=IG, nbins = 50, main="Inverse Gaussian
distribution")
```




```
fit_na.BCCG$df.fit
## [1] 3
fitted(fit_na.BCCG, "mu")[1]
## [1] 13.38101
fitted(fit_na.BCCG, "sigma")[1]
## [1] 0.06047485
fitted(fit_na.BCCG, "nu")[1]
## [1] -0.09433756

fit_na.BCT$df.fit
## [1] 4
fitted(fit_na.BCT, "mu")[1]
## [1] 13.34124
fitted(fit_na.BCT, "sigma")[1]
## [1] 0.04375201
fitted(fit_na.BCT, "nu")[1]
## [1] -1.756197
fitted(fit_na.BCT, "tau")[1]
## [1] 3.953332

fit_na.BCPE$df.fit
## [1] 4
fitted(fit_na.BCPE, "mu")[1]
## [1] 13.31378
fitted(fit_na.BCPE, "sigma")[1]
## [1] 0.05997521
fitted(fit_na.BCPE, "nu")[1]
```

```
## [1] -0.5455229
fitted(fit_na.BCPE, "tau")[1]
## [1] 1.161532

fit_na.GB2$df.fit
## [1] 4
fitted(fit_na.GB2, "mu")[1]
## [1] 13.13523
fitted(fit_na.GB2, "sigma")[1]
## [1] 107.959
fitted(fit_na.GB2, "nu")[1]
## [1] 0.2863685
fitted(fit_na.GB2, "tau")[1]
## [1] 0.185553

fit_na.IG$df.fit
## [1] 2
fitted(fit_na.IG, "mu")[1]
## [1] 13.40785
fitted(fit_na.IG, "sigma")[1]
## [1] 0.01653825
```

In the following code I will create a dataframe that shows the ranking of AIC, BIC and Log-likelihood values and the associated degree of freedom:

```
aic <- AIC(fit_na.BCCG,fit_na.BCT,fit_na.BCPE,fit_na.IG,fit_na.GB2)
aic$BIC <- c(fit_na.GB2$sbc,fit_na.BCT$sbc,fit_na.BCPE$sbc,fit_na.IG$sbc,fit_na.BCCG$sbc)
aic$LogLik <- c(logLik(fit_na.GB2), logLik(fit_na.BCT), logLik(fit_na.BCPE), logLik(fit_na
.IG), logLik(fit_na.BCCG))
aic

##             df      AIC      BIC    LogLik
## fit_na.GB2   4 498.2559 511.7198 -245.1279
## fit_na.BCT   4 501.9660 515.4299 -246.9830
## fit_na.BCPE  4 504.9681 518.4320 -248.4841
## fit_na.IG    2 520.9699 527.7018 -258.4849
## fit_na.BCCG  3 522.7828 532.8807 -258.3914
```

The best distribution model that fits the "Na" variable is the Generalized Beta type 2 distribution with 4 parameters. The minimum value of AIC and BIC and the maximum value of Log-likelihood.

Goodness of fit

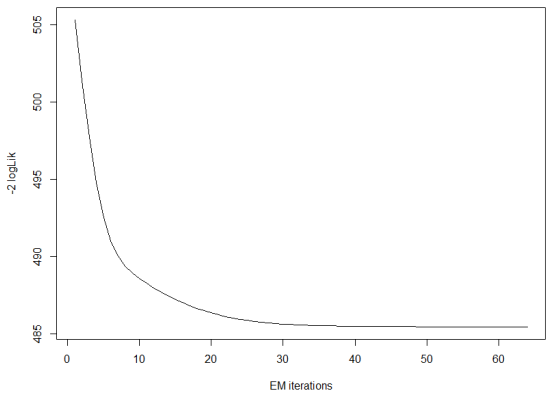
```
gamlss:::LR.test(fit_na.IG,fit_na.GB2)

##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
##        Null model: deviance= 516.9699 with  2 deg. of freedom
##  Alternative model: deviance= 490.2559 with  4 deg. of freedom
##
##  LRT = 26.714 with 2 deg. of freedom and p-value= 1.581716e-06
```

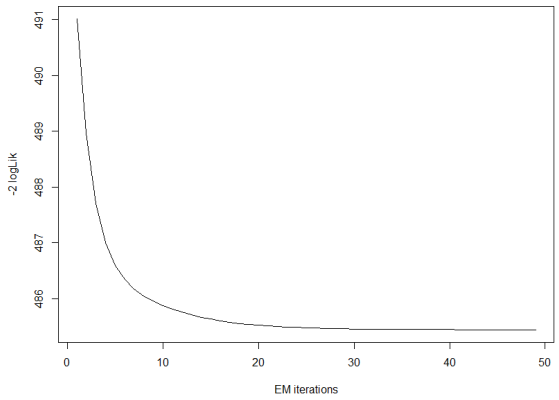
Throw the Likelihood-ratio test, as the p-value is smaller than 0.05, I reject the full model Inverse Gaussian distribution and accept the Generalized Beta type 2 distribution.

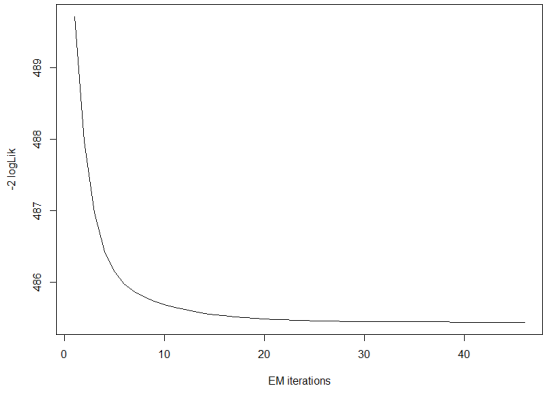
Mixture of distributions

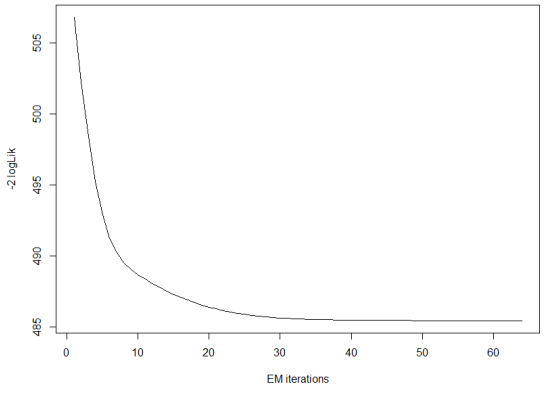
In the following code I will compute a mixture of two Gamma distributions and, to find the best distribution, the algorithm is repeated 5 times:

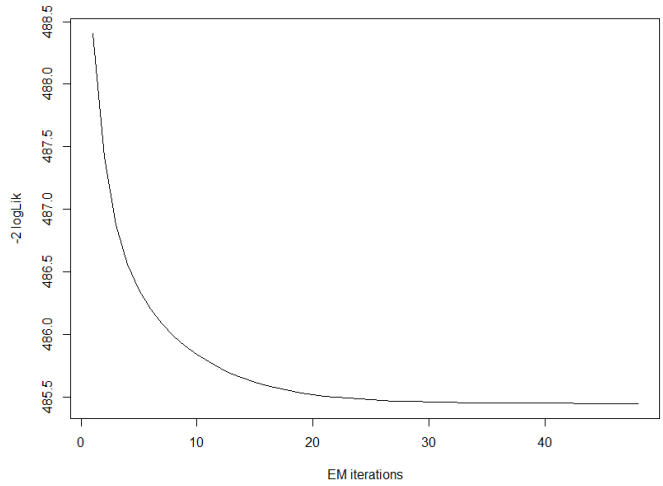
```
mix_na.GA <- gamlss.mx:::gamlssMXfits(n = 5, glass$Na~1, family = GA, K = 2, data = NULL)
```



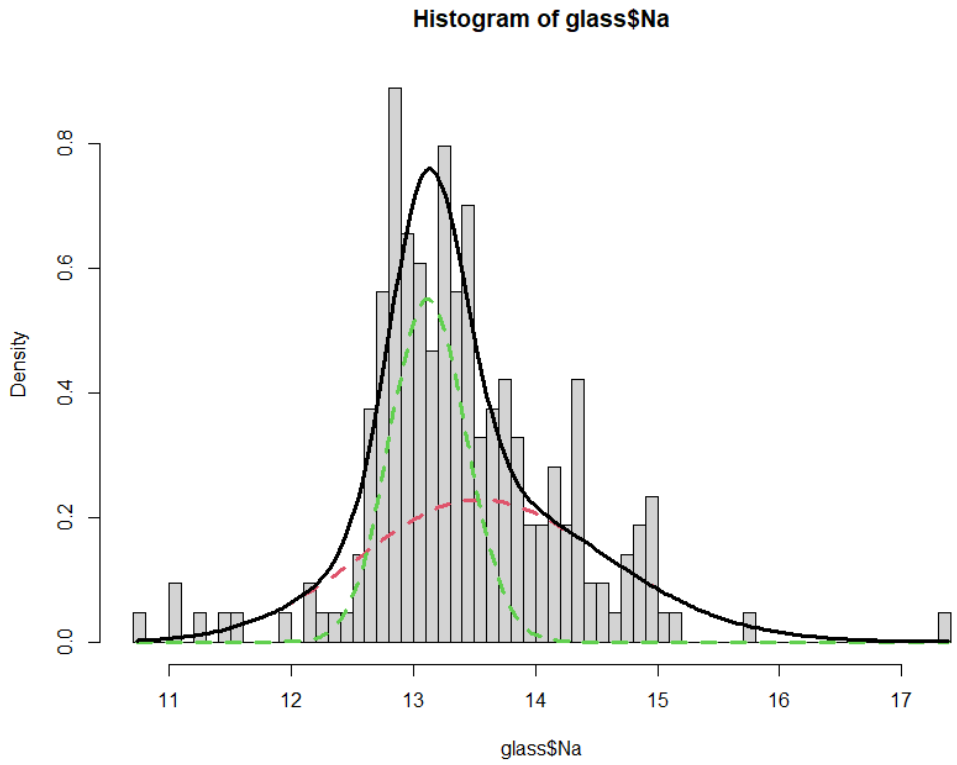









```
mu.hat1 <- exp(mix_na.GA[["models"]][[1]][["mu.coefficients"]])
sigma.hat1 <- exp(mix_na.GA[["models"]][[1]][["sigma.coefficients"]])
mu.hat2 <- exp(mix_na.GA[["models"]][[2]][["mu.coefficients"]])
sigma.hat2 <- exp(mix_na.GA[["models"]][[2]][["sigma.coefficients"]])

hist(glass$Na, breaks = 50,freq = FALSE)
lines(seq(min(glass$Na),max(glass$Na),length=length(glass$Na)),mix_na.GA[["prob"]][1]*dGA(
seq(min(glass$Na),max(glass$Na),length=length(glass$Na)), mu = mu.hat1, sigma =
sigma.hat1),lty=2,lwd=3,col=2)
lines(seq(min(glass$Na),max(glass$Na),length=length(glass$Na)),mix_na.GA[["prob"]][2]*dGA(
seq(min(glass$Na),max(glass$Na),length=length(glass$Na)), mu = mu.hat2, sigma =
sigma.hat2),lty=2,lwd=3,col=3)
lines(seq(min(glass$Na),max(glass$Na),length=length(glass$Na)),
mix_na.GA[["prob"]][1]*dGA(seq(min(glass$Na),max(glass$Na),length=length(glass$Na)), mu =
mu.hat1, sigma = sigma.hat1) +
mix_na.GA[["prob"]][2]*dGA(seq(min(glass$Na),max(glass$Na),length=length(glass$Na)), mu =
mu.hat2, sigma = sigma.hat2), lty = 1, lwd = 3, col = 1)
```




```
mix_na.GA$prob

## [1] 0.4261241 0.5738759
```

The first group explains 42.6% of the distribution and the second group explains 57.3% of the distribution, for this reason, I assume that a mixture of two groups can be enough.

```
aic <- AIC(mix_na.GA,fit_na.GB2)
aic$BIC <- c(mix_na.GA$sbc,fit_na.GB2$sbc)
aic$LogLik <- c(logLik(mix_na.GA), logLik(fit_na.GB2))
aic

##              df      AIC      BIC    LogLik
## mix_na.GA     5 495.4473 512.2771 -242.7236
## fit_na.GB2    4 498.2559 511.7198 -245.1279
```

The best distribution model which fits the "Na" variable is the mixture of Gamma distributions with the minimum value of AIC and BIC and the maximum value of Log-likelihood.

Goodness of fit

```
gamlss:::LR.test(fit_na.GB2,mix_na.GA)

##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
##        Null model: deviance= 490.2559 with  4 deg. of freedom
```

```
##  Alternative model: deviance= 486.6073 with  5 deg. of freedom
##
##  LRT = 3.64859 with 1 deg. of freedom and p-value= 0.0561167
```

According to the Likelihood-ratio test, as the p-value is slightly greater than 0.05, the Generalized Beta type 2 distribution has been accepted instead of the mixture of Gamma distribution.

Mg

It's a continuous quantitative variable within the range [0,4.49].

```
summary(Mg)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.115   3.480   2.685   3.600   4.490
```

According to the position indices, the observations have the smallest value of 0 and the greatest value of 4.49, at least 25% has the value of 2.11, at least 50% has the value of 3.48 and at least 75% has the value of 3.6. The whole observations' value is above 2.68, the interquartile distance is 1.48 and there are no outliers.

Thanks to the variability and shape indices I can predict the shape of the curve:

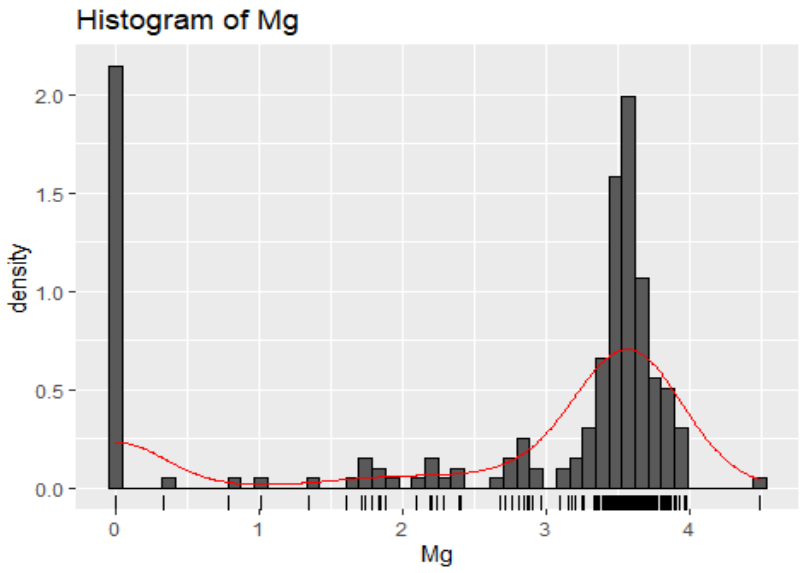
```
labstatR:::skew(Mg)

## [1] -1.144465

labstatR:::kurt(Mg)

## [1] 2.571298

sd(Mg)

## [1] 1.442408
```

The coefficient of skewness is negative, and the distribution is left-skewed, this means that the graph is asymmetric, and the values are concentrated on the right side of the graph.

The Kurtosis index value is smaller than 3, so the platykurtic distribution has a lower central peak concerning the normal distribution.

The standard deviation has a value slightly greater than 1, this means that the values are not so concentrated. This is confirmed by plotting the histogram of "Mg" and highlighting the density function:

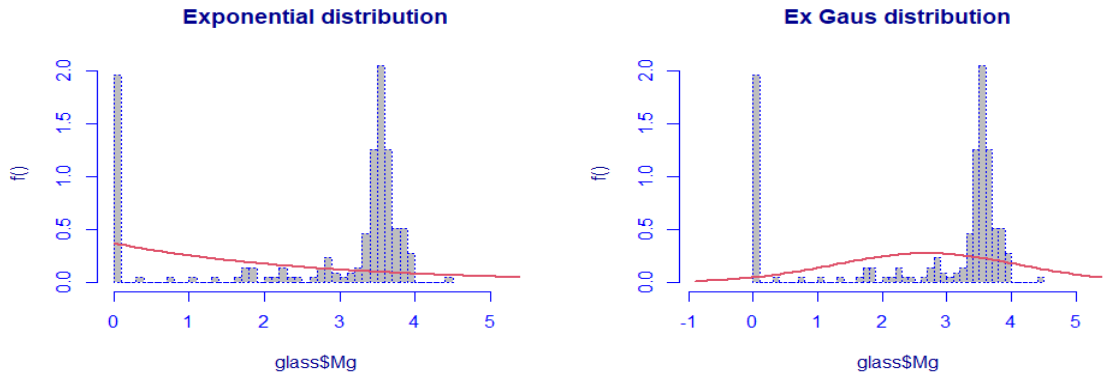
```
ggplot2:::ggplot(glass, aes(x=Mg))+
  geom_histogram(aes(y=..density..), color='black',bins=50)+
  geom_density(color='red')+
  geom_rug()+
  ggtitle('Histogram of Mg')
```



Single parametric distributions

Throw the following code I will try to establish what is the best distribution which fits the variable "Mg". I first introduce two possible distributions which handle the value of the positive real line, and, for each, I will compute the value of the parameters which are called mu, sigma, nu, and tau:

```
par(mfrow=c(1,2))

fit_mg.EXP <- gamlss:::histDist(glass$Mg, family=EXP, nbins = 50, main="Exponential distri
bution")
fit_mg.exGAUS <- gamlss:::histDist(glass$Mg, family=exGAUS, nbins = 50, main="Ex Gaus dist
ribution")
```




```
fit_mg.EXP$df.fit
## [1] 1
fitted(fit_mg.EXP, "mu")[1]
## [1] 2.684533

fit_mg.exGAUS$df.fit
## [1] 3
fitted(fit_mg.exGAUS, "mu")[1]
```

```
## [1] 2.612651
fitted(fit_mg.exGAUS, "sigma")[1]
## [1] 1.437416
fitted(fit_mg.exGAUS, "nu")[1]
## [1] 0.07187082
```

In the next code I will create a dataframe that shows the ranking of AIC, BIC, and Log-likelihood:

```
aic <- AIC(fit_mg.EXP,fit_mg.exGAUS)
aic$BIC <- c(fit_mg.exGAUS$sbc,fit_mg.EXP$sbc)
aic$LogLik <- c(logLik(fit_mg.exGAUS), logLik(fit_mg.EXP))
aic

##                 df      AIC      BIC     LogLik
## fit_mg.exGAUS    3 769.1061 779.2040  -381.5531
## fit_mg.EXP       1 852.6529 856.0188  -425.3264
```

The best distribution model that fits the "Mg" variable is the ex Gaussian with 3 parameters. The minimum value of AIC and BIC and the maximum value of Log-likelihood.

Goodness of fit

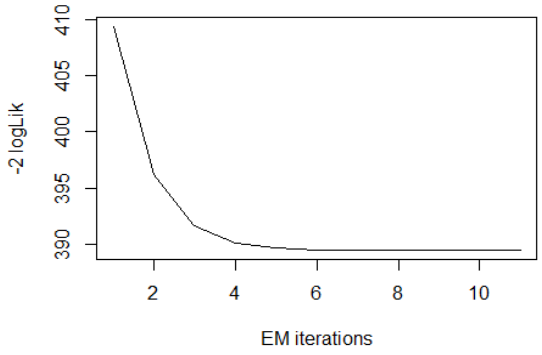
```
gamlss::: LR.test(fit_mg.EXP,fit_mg.exGAUS)

##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
##        Null model: deviance= 850.6529 with  1 deg. of freedom
##  Alternative model: deviance= 763.106 with  3 deg. of freedom
##
##  LRT = 87.54675 with 2 deg. of freedom and p-value= 0
```

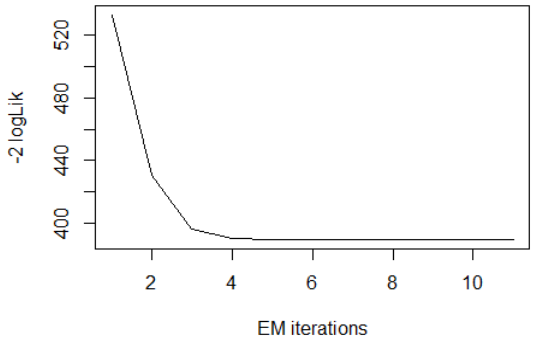
Throw the Likelihood-ratio test, as the p-value is smaller than 0.05, I reject the Exponential distribution and accept the ex Gaussian distribution.

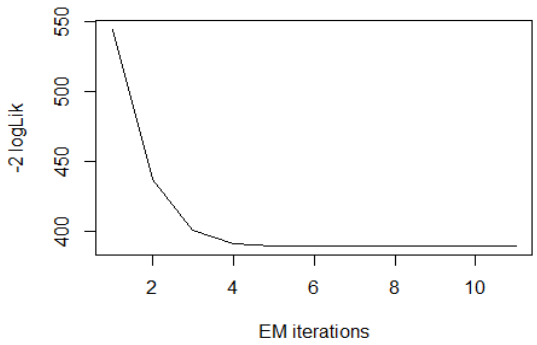
Mixture of distributions

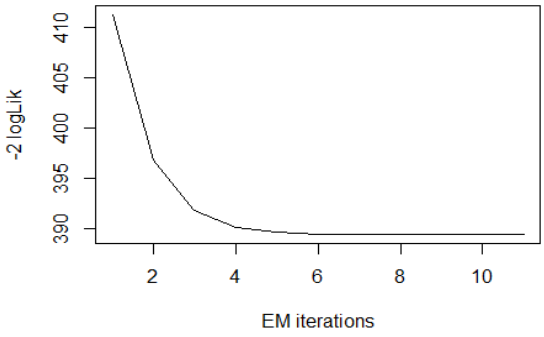
In the following code I will compute a mixture of two Normal distributions and, to find the best distribution, the algorithm is repeated 5 times:

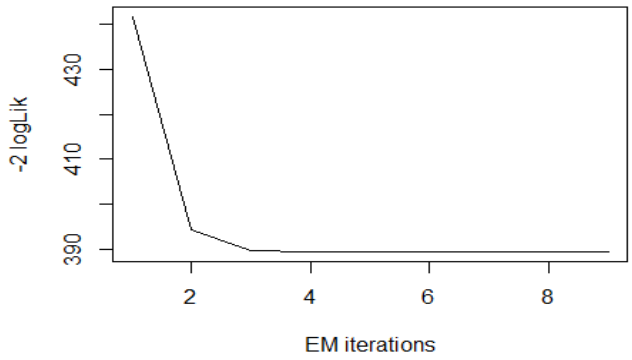
```
mix_mg.NO <- gamlss.mx:::gamlssMXfits(n = 5, glass$Mg~1, family=NO, K = 2, data = NULL)
```





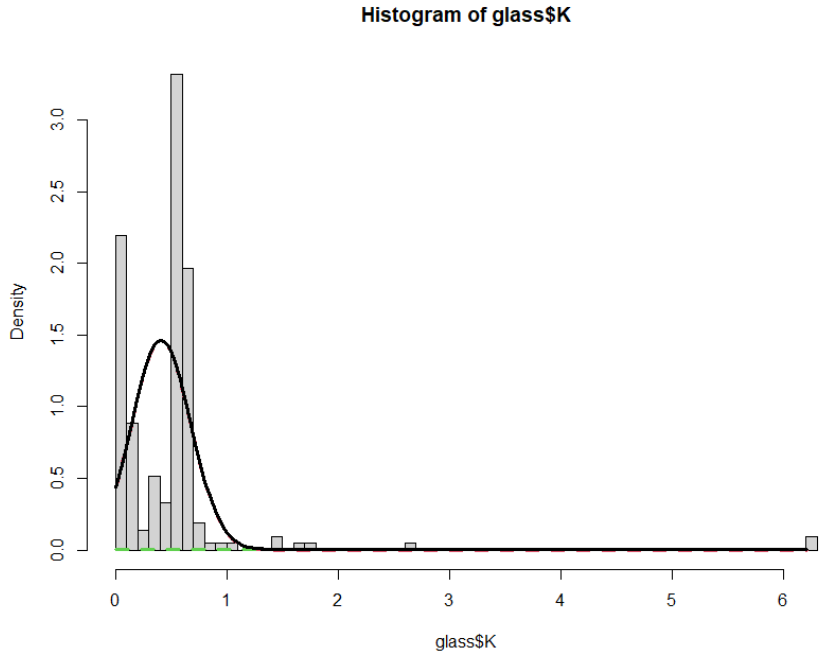







```
mu.hat1 <- mix_mg.NO[["models"]][[1]][["mu.coefficients"]]
sigma.hat1 <- exp(mix_mg.NO[["models"]][[1]][["sigma.coefficients"]])
mu.hat2 <- mix_mg.NO[["models"]][[2]][["mu.coefficients"]]
sigma.hat2 <- exp(mix_mg.NO[["models"]][[2]][["sigma.coefficients"]])

hist(glass$Mg, breaks = 50,freq = FALSE)
```

```
lines(seq(min(glass$Mg),max(glass$Mg),length=length(glass$Mg)),mix_mg.NO[["prob"]][1]*dNO(
seq(min(glass$Mg),max(glass$Mg),length=length(glass$Mg)), mu = mu.hat1, sigma = sigma.hat1
),lty=2,lwd=3,col=2)
lines(seq(min(glass$Mg),max(glass$Mg),length=length(glass$Mg)),mix_mg.NO[["prob"]][2]*dNO(
seq(min(glass$Mg),max(glass$Mg),length=length(glass$Mg)), mu = mu.hat2, sigma = sigma.hat2
),lty=2,lwd=3,col=3)
lines(seq(min(glass$Mg),max(glass$Mg),length=length(glass$Mg)), mix_mg.NO[["prob"]][1]*dNO
(seq(min(glass$Mg),max(glass$Mg),length=length(glass$Mg)), mu = mu.hat1, sigma = sigma.hat
1) + mix_mg.NO[["prob"]][2]*dNO(seq(min(glass$Mg),max(glass$Mg),length=length(glass$Mg)),
mu = mu.hat2, sigma = sigma.hat2), lty = 1, lwd = 3, col = 1)
```




```
mix_mg.NO$prob

## [1] 0.6417187 0.3582813
```

The first group explains 64.17% of the distribution, the second group explains 35.82% of the distribution, so I assume that two groups of distributions can be enough.

```
aic <- AIC(mix_mg.NO,fit_mg.exGAUS)
aic$BIC <- c(fit_mg.NO$sbc,mix_mg.exGAUS$sbc)
aic$LogLik <- c(logLik(fit_mg.NO), logLik(mix_mg.exGAUS))
aic

##                df      AIC      BIC     LogLik
## mix_mg.NO       5 399.4682 779.2040  -381.5531
## fit_mg.exGAUS   2 769.1061 416.2981  -194.73410
```

The best distribution model which fits the “Mg” variable is the mixture of two Normal distribution with 5 parameters, with the minimum value of AIC and BIC and the maximum value of Log-Likelihood.

Al

It's a continuous quantitative variable that can assume value within the range [0.29,3.5].

```
summary(Al)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.290   1.190   1.360   1.445   1.630   3.500
```

According to the position indices, the observations have the smallest value of 0.29 and the greatest value of 3.5, at least 25% has the value of 1.19, at least 50% has a value of 1.36 and at least 75% has a value of 1.63. The whole observations' value is above 1.445.

In detail, there are 18 outliers out of 214:

```
boxplot.stats(Al)$out

##  [1] 0.29 0.47 0.47 0.51 3.50 3.04 3.02 0.34 2.38 2.79 2.68 2.54 2.34 2.66 2.51
## [16] 2.42 2.74 2.88
```

Thanks to the variability and shape indices I can predict the shape of the curve:

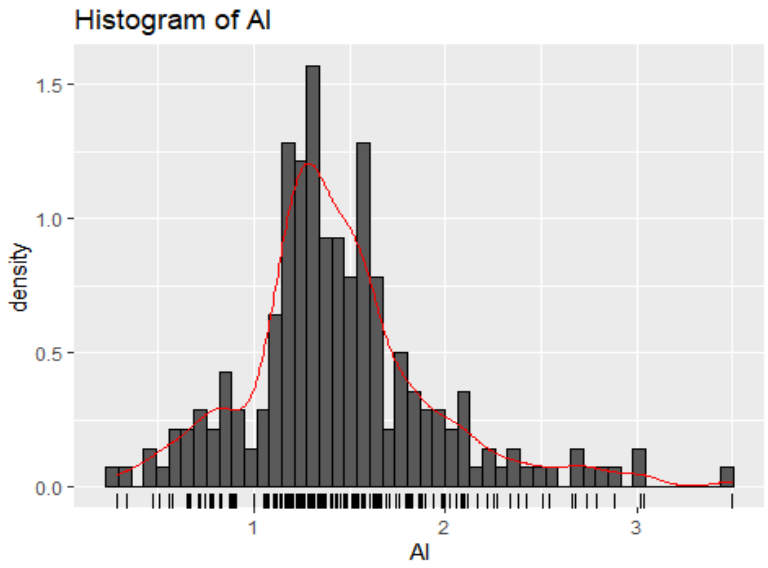
```
labstatR:::skew(Al)

## [1] 0.9009179

labstatR:::kurt(Al)

## [1] 4.984832

sd(Al)

## [1] 0.4992696
```

The coefficient of skewness is positive, and the distribution is right-skewed, this means that the graph is asymmetric, and the values are concentrated on the left side of the graph.

The Kurtosis index value is greater than 3, so the distribution is leptokurtic with a higher shape concerning the normal distribution.

The standard deviation has a small value, this means that the values are concentrated near to the mean. This is confirmed by plotting the histogram and highlighting the density function:

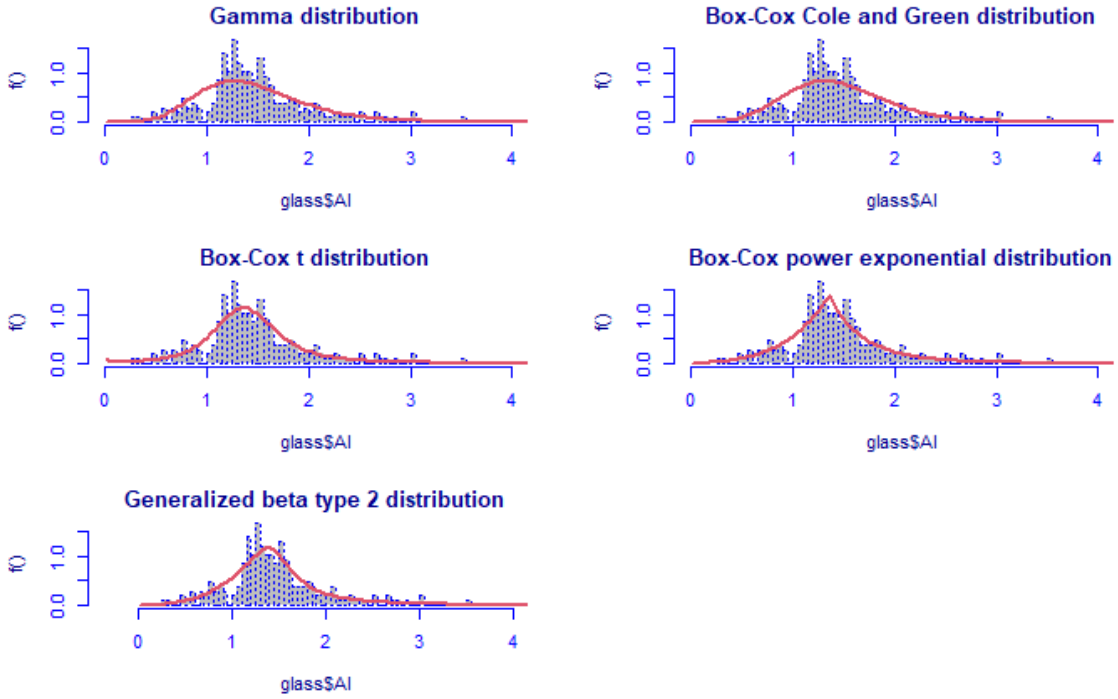
```
ggplot2:::ggplot(glass, aes(x=Al))+
  geom_histogram(aes(y=..density..), color='black',bins=50)+
  geom_density(color='red')+
  geom_rug()+
  ggtitle('Histogram of Al')
```



Single parametric distributions

Throw the following code I will try to establish what is the best distribution which fits the variable "Al". I first introduce five possible distributions which handle the value of the positive real line, and, for each, I will compute the value of the parameters which are called mu, sigma, nu, and tau:

```
par(mfrow=c(3,2))
fit_al.GA <- gamlss:::histDist(glass$Al, family=GA, nbins = 50, main="Gamma distribution")
fit_al.BCCG <- gamlss:::histDist(glass$Al, family=BCCG, nbins = 50, main="Box-Cox Cole and
Green distribution")
fit_al.BCT <- gamlss:::histDist(glass$Al, family=BCT, nbins = 50, main="Box-Cox t distribu
tion")
fit_al.BCPE <- gamlss:::histDist(glass$Al, family=BCPE, nbins = 50, main="Box-Cox power ex
ponential distribution")
fit_al.GB2 <- gamlss:::histDist(glass$Al, family=GB2, nbins = 50, main="Generalized beta t
ype 2 distribution")
```




```
fit_al.GA$df.fit
## [1] 2
fitted(fit_al.GA, "mu")[1]
## [1] 1.444907
fitted(fit_al.GA, "sigma")[1]
## [1] 0.348708

fit_al.BCCG$df.fit
## [1] 3
fitted(fit_al.BCCG, "mu")[1]
## [1] 1.40129
fitted(fit_al.BCCG, "sigma")[1]
## [1] 0.3485044
fitted(fit_al.BCCG, "nu")[1]
## [1] 0.4872865

fit_al.BCT$df.fit
## [1] 4
fitted(fit_al.BCT, "mu")[1]
## [1] 1.391814
fitted(fit_al.BCT, "sigma")[1]
## [1] 0.2337305
fitted(fit_al.BCT, "nu")[1]
## [1] 0.4903103
fitted(fit_al.BCT, "tau")[1]
## [1] 2.821435

fit_al.BCPE$df.fit
## [1] 4
fitted(fit_al.BCPE, "mu")[1]
```

```
## [1] 1.361375
fitted(fit_al.BCPE, "sigma")[1]
## [1] 0.3576653
fitted(fit_al.BCPE, "nu")[1]
## [1] 0.4035379
fitted(fit_al.BCPE, "tau")[1]
## [1] 1.03845

fit_al.GB2$df.fit
## [1] 4
fitted(fit_al.GB2, "mu")[1]
## [1] 1.42961
fitted(fit_al.GB2, "sigma")[1]
## [1] 22.68692
fitted(fit_al.GB2, "nu")[1]
## [1] 0.1616516
fitted(fit_al.GB2, "tau")[1]
## [1] 0.1973659
```

In the next code I will create a dataframe which shows the ranking of AIC, BIC, and Log-likelihood:

```
aic <- AIC(fit_al.GA,fit_al.BCCG,fit_al.BCT,fit_al.BCPE,fit_al.GB2)
aic$BIC <- c(fit_al.BCPE$sbc,fit_al.GB2$sbc,fit_al.BCT$sbc,fit_al.BCCG$sbc,fit_al.GA$sbc)
aic$LogLik <- c(logLik(fit_al.BCPE), logLik(fit_al.GB2), logLik(fit_al.BCT), logLik(fit_al
.BCCG), logLik(fit_al.GA))
aic

##             df      AIC      BIC    LogLik
## fit_al.BCPE  4 279.4617 292.9256 -135.7309
## fit_al.GB2   4 281.1326 294.5965 -136.5663
## fit_al.BCT   4 284.2780 297.7419 -138.1390
## fit_al.BCCG  3 299.6677 309.7657 -146.8339
## fit_al.GA    2 300.0391 306.7711 -148.0196
```

According to the dataframe, the best distribution model which fits the "Al" variable is the Box-Cox power exponential distribution with 4 parameters, with the minimum value of AIC and BIC and the maximum value of Log-likelihood.

Goodness of fit

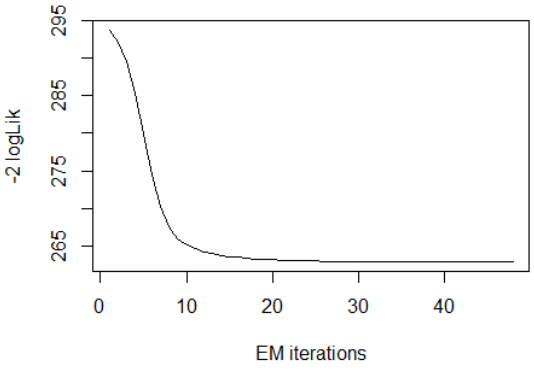
```
gamlss:::LR.test(fit_al.GA,fit_al.BCPE)

##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
##        Null model: deviance= 296.0391 with  2 deg. of freedom
##  Alternative model: deviance= 271.4617 with  4 deg. of freedom
##
##  LRT = 24.57741 with 2 deg. of freedom and p-value= 4.603441e-06
```

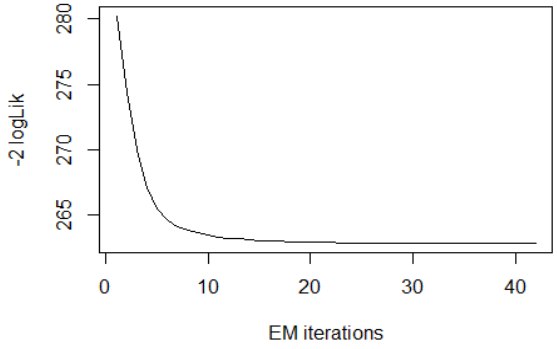
Throw the Likelihood-ratio test, as the p-value is smaller than 0.05, I reject the full model Gaussian distribution and accept the Box-Cox Power Exponential distribution.

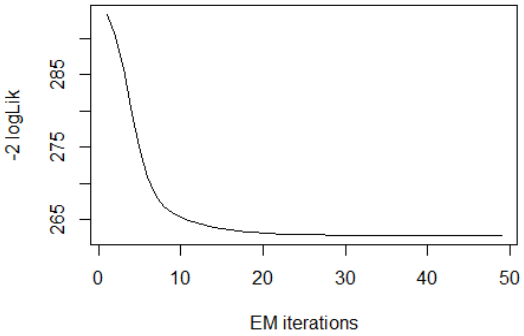
Mixture of distributions

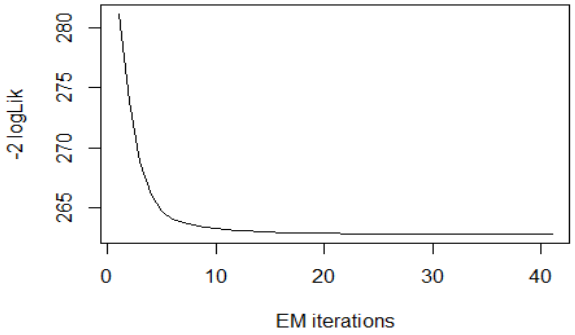
In the following code I will compute a mixture of two Gamma distributions and, to find the best distribution, the algorithm is repeated 5 times

```
mix_al.GA <- gamlss.mx:::gamlssMXfits(n = 5, glass$Al~1, family=GA, K = 2, data = NULL)
```

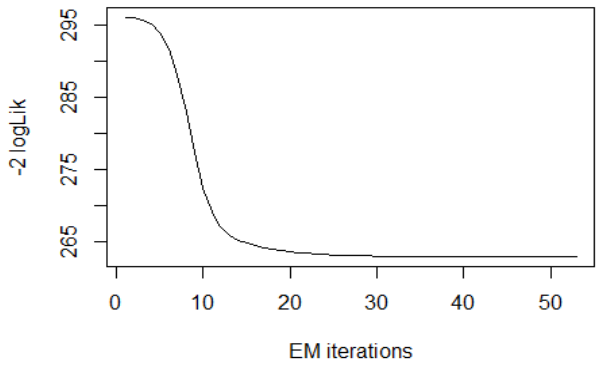




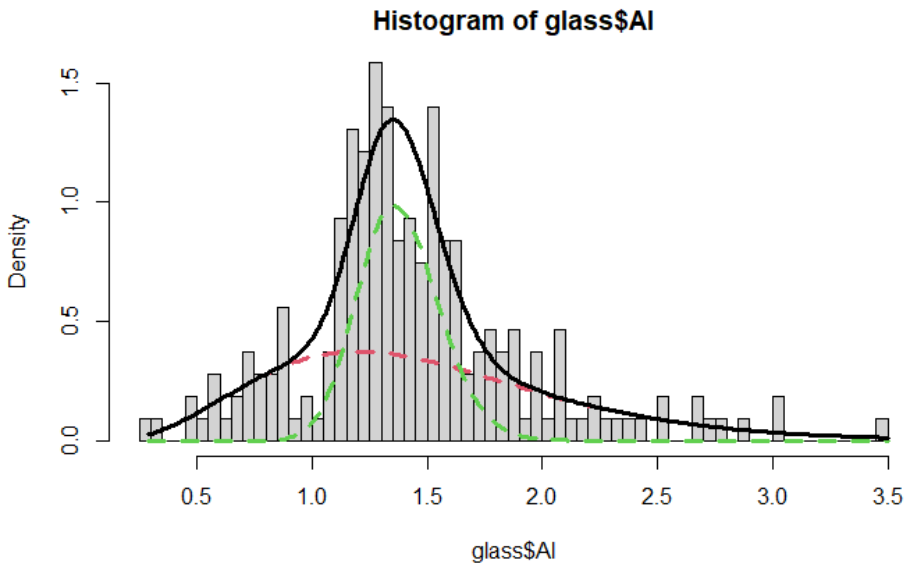




Univariate Analysis - Al




```
mu.hat1 <- exp(mix_al.GA[["models"]][[1]][["mu.coefficients"]])
sigma.hat1 <- exp(mix_al.GA[["models"]][[1]][["sigma.coefficients"]])
mu.hat2 <- exp(mix_al.GA[["models"]][[2]][["mu.coefficients"]])
sigma.hat2 <- exp(mix_al.GA[["models"]][[2]][["sigma.coefficients"]])

hist(glass$Al, breaks = 50,freq = FALSE)
lines(seq(min(glass$Al),max(glass$Al),length=length(glass$Al)),mix_al.GA[["prob"]][1]*dGA(
seq(min(glass$Al),max(glass$Al),length=length(glass$Al)), mu = mu.hat1, sigma =
sigma.hat1),lty=2,lwd=3,col=2)
lines(seq(min(glass$Al),max(glass$Al),length=length(glass$Al)),mix_al.GA[["prob"]][2]*dGA(
seq(min(glass$Al),max(glass$Al),length=length(glass$Al)), mu = mu.hat2, sigma =
sigma.hat2),lty=2,lwd=3,col=3)
lines(seq(min(glass$Al),max(glass$Al),length=length(glass$Al)),
mix_al.GA[["prob"]][1]*dGA(seq(min(glass$Al),max(glass$Al),length=length(glass$Al)), mu =
mu.hat1, sigma = sigma.hat1) +
mix_al.GA[["prob"]][2]*dGA(seq(min(glass$Al),max(glass$Al),length=length(glass$Al)), mu =
mu.hat2, sigma = sigma.hat2), lty = 1, lwd = 3, col = 1)
```




```
mix_al.GA$prob
```

```
## [1] 0.5679206 0.4320794
```

The first group explains 56.79% of our distribution and the second group explains 43.20% of our distribution, so I assume that two distributions are enough.

```
aic <- AIC(mix_al.GA,fit_al.BCPE)
aic$BIC <- c(mix_al.GA$sbc,fit_al.BCPE$sbc)
aic$LogLik <- c(logLik(mix_al.GA), logLik(fit_al.BCPE))
aic
##             df      AIC      BIC    LogLik
## mix_al.GA    5 272.8697 289.6996 -131.4349
## fit_al.BCPE  4 279.4617 292.9256 -135.7309
```

The best distribution model which fits the "Al" variable is the mixture of Gamma distributions with the minimum value of AIC and BIC and the maximum value of Log-likelihood.

Goodness of fit

```
gamlss:::LR.test(fit_al.BCPE, mix_al.GA)

##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
##        Null model: deviance= 271.4617 with  4 deg. of freedom
##  Alternative model: deviance= 262.8697 with  5 deg. of freedom
##
##  LRT = 8.591967 with 1 deg. of freedom and p-value= 0.00337649
```

According to the likelihood-ratio test, as the p-value is less than 0.05, the Box-Cox Power Exponential distribution has been rejected and the mixture of Gamma distribution has been accepted.

Si

It's a continuous quantitative variable that can assume value within range [69.81,75.41].

```
summary(Si)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   69.81   72.28   72.79   72.65   73.09   75.41
```

According to the position indices, the observations have the smallest value of 69.81 and the greatest value of 75.41, at least 25% has a value of 72.28, at least 50% has a value of 72.79 and at least 75% has a value of 73.09. The whole observations' value is above 72.65 and the interquartile distance is 0.81.

In detail, there are 12 outliers out of 214:

```
boxplot.stats(Si)$out

##  [1] 70.57 69.81 70.16 74.45 69.89 70.48 70.70 74.55 75.41 70.26 70.43 75.18
```

Thanks to the variability and shape indices, I can predict the shape of the curve:

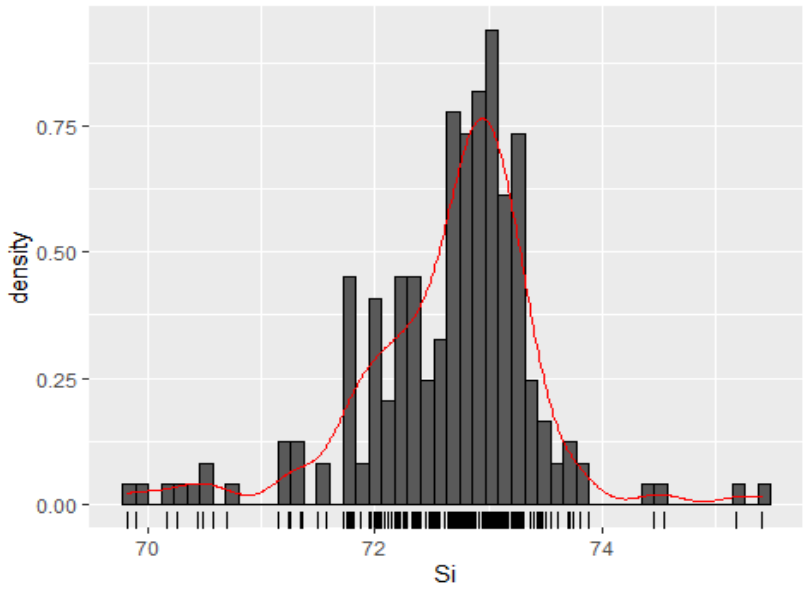
```
labstatR:::skew(Si)

## [1] -0.7253173

labstatR:::kurt(Si)

## [1] 5.871105

sd(Si)

## [1] 0.7745458
```

The coefficient of skewness is negative, and the distribution is left-skewed, this means that the graph is asymmetric, and the values are concentrated on the centre-right side of the graph.

The Kurtosis index value is greater than 3, so I assume that the distribution is leptokurtic with a higher shape concerning the normal distribution.

The standard deviation has a value close to 1, this means that the values are concentrated near the mean. This is confirmed by plotting the histogram of "Si" and highlighting the density function:

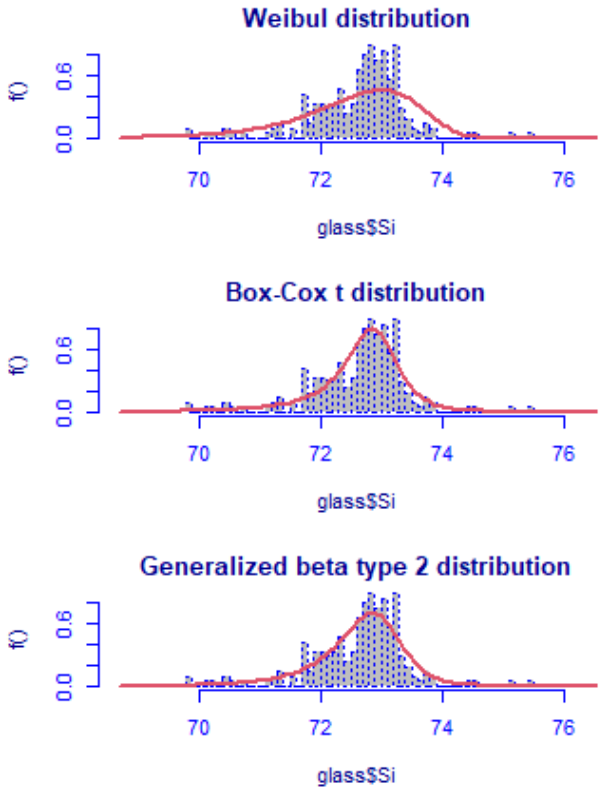
```
ggplot2:::ggplot(glass, aes(x=Si))+
  geom_histogram(aes(y=..density..), color='black',bins=50)+
  geom_rug()+
  geom_density(color='red')
```

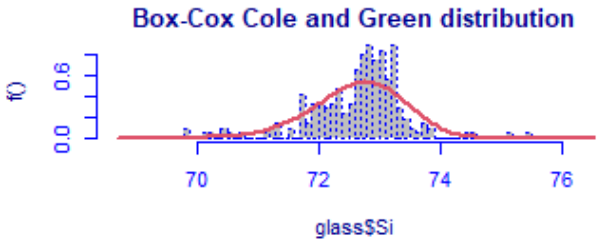


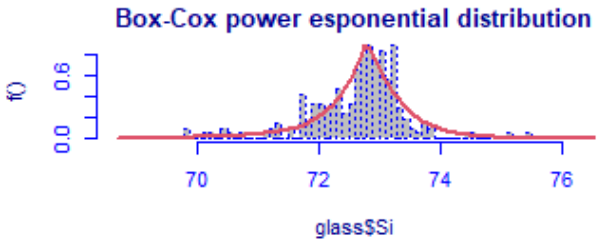
Single parametric distributions

Throw the following code I will try to establish what is the best distribution which fits the variable Si. I first introduce five possible distributions which handle the value of positive real line and for each, I will compute the value of the parameters which are called mu, sigma, nu, and tau:

```
par(mfrow=c(3,2))

fit_si.WEI <- gamlss:::histDist(glass$Si, family=WEI, nbins = 50, main="Weibul distributio
n")
fit_si.BCCG <- gamlss:::histDist(glass$Si, family=BCCG, nbins = 50, main="Box-Cox Cole and
Green distribution")
fit_si.BCT <- gamlss:::histDist(glass$Si, family=BCT, nbins = 50, main="Box-Cox t distribu
tion")
fit_si.BCPE <- gamlss:::histDist(glass$Si, family=BCPE, nbins = 50, main="Box-Cox power es
ponential distribution")
fit_si.GB2 <- gamlss:::histDist(glass$Si, family=GB2, nbins = 50, main="Generalized beta t
ype 2 distribution")
```








```
fit_si.WEI$df.fit
## [1] 2
fitted(fit_si.WEI, "mu")[1]
## [1] 73.01267
fitted(fit_si.WEI, "sigma")[1]
## [1] 91.25856

fit_si.BCCG$df.fit
```

```
## [1] 3
fitted(fit_si.BCCG, "mu")[1]
## [1] 72.69688
fitted(fit_si.BCCG, "sigma")[1]
## [1] 0.01030971
fitted(fit_si.BCCG, "nu")[1]
## [1] 12.46839

fit_si.BCT$df.fit
## [1] 4
fitted(fit_si.BCT, "mu")[1]
## [1] 72.68003
fitted(fit_si.BCT, "sigma")[1]
## [1] 0.007720447
fitted(fit_si.BCT, "nu")[1]
## [1] 80.36508
fitted(fit_si.BCT, "tau")[1]
## [1] 2.023283

fit_si.BCPE$df.fit
## [1] 4
fitted(fit_si.BCPE, "mu")[1]
## [1] 72.79
fitted(fit_si.BCPE, "sigma")[1]
## [1] 0.009990413
fitted(fit_si.BCPE, "nu")[1]
## [1] 16.36079
fitted(fit_si.BCPE, "tau")[1]
## [1] 1.033885

fit_si.GB2$df.fit
## [1] 4
fitted(fit_si.GB2, "mu")[1]
## [1] 73.03016
fitted(fit_si.GB2, "sigma")[1]
## [1] 306.2433
fitted(fit_si.GB2, "nu")[1]
## [1] 0.4016518
fitted(fit_si.GB2, "tau")[1]
## [1] 0.8025492
```

In the next code I will create a dataframe which shows the ranking of AIC, BIC, and Log-likelihood:

```
aic <- AIC(fit_si.WEI,fit_si.BCCG,fit_si.BCT,fit_si.BCPE,fit_si.GB2)
aic$BIC <- c(fit_si.BCT$sbc,fit_si.GB2$sbc,fit_si.BCPE$sbc,fit_si.BCCG$sbc,fit_si.WEI$sbc)
aic$LogLik <- c(logLik(fit_si.BCT), logLik(fit_si.GB2), logLik(fit_si.BCPE), logLik(fit_si
.BCCG), logLik(fit_si.WEI))
aic

##             df      AIC      BIC    LogLik
## fit_si.BCT   4 452.0420 465.5059 -222.0210
```

```
## fit_si.GB2   4 457.6638 471.1277 -224.8319
## fit_si.BCPE  4 463.7746 477.2385 -227.8873
## fit_si.BCCG  3 493.2657 503.3636 -243.6328
## fit_si.WEI   2 530.5924 537.3243 -263.2962
```

The best distribution model that fits the "Si" variable is the Box-Cox-t distribution with 4 parameters. The minimum value of AIC and BIC and the maximum value of Log-likelihood.

Goodness of fit

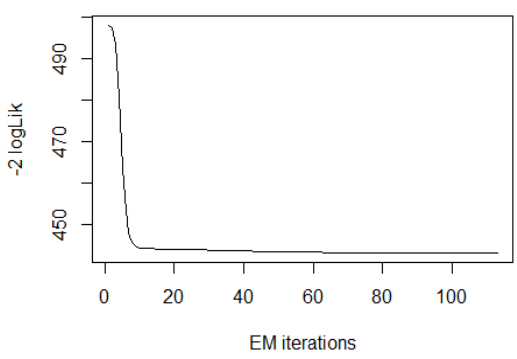
```
gamlss:::LR.test(fit_si.BCPE, fit_si.BCT)

##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
##        Null model: deviance= 455.7746 with  4 deg. of freedom
##  Alternative model: deviance= 444.042 with  4 deg. of freedom
##
##  LRT = 11.73256 with 0 deg. of freedom and p-value= 0
```

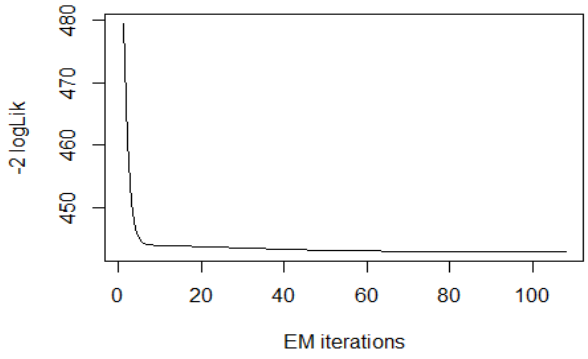
Throw the likelihood-ratio test, as the p-value is less than 0.05, the null hypothesis, which corresponds to the Box-Cox Power Exponential distribution, has been rejected and the Box-Cox t distribution has been accepted.

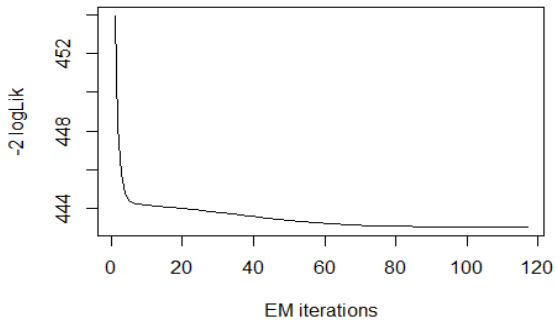
Mixture of distributions

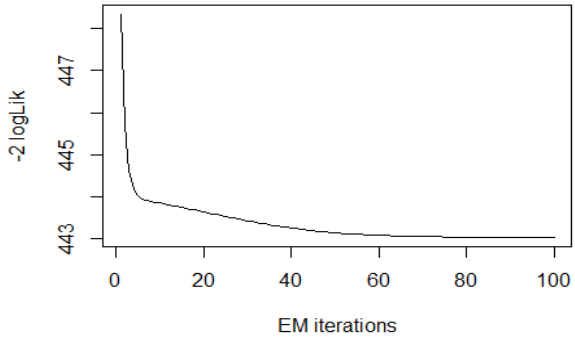
In the following code I will compute a mixture of two Gamma distributions and, to find the best distribution, the algorithm is repeated 5 times:

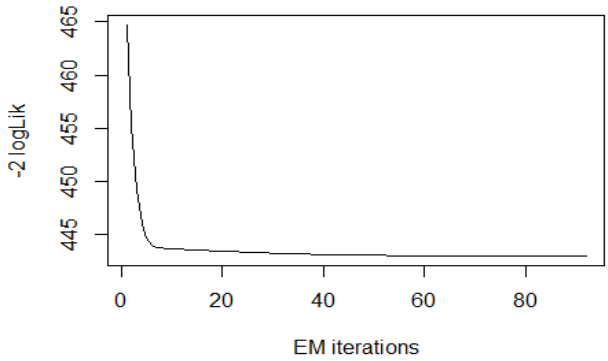
```
mix_si.GA <- gamlss.mx:::gamlssMXfits(n = 5, glass$Si~1, family=GA, K = 2, data = NULL)
```



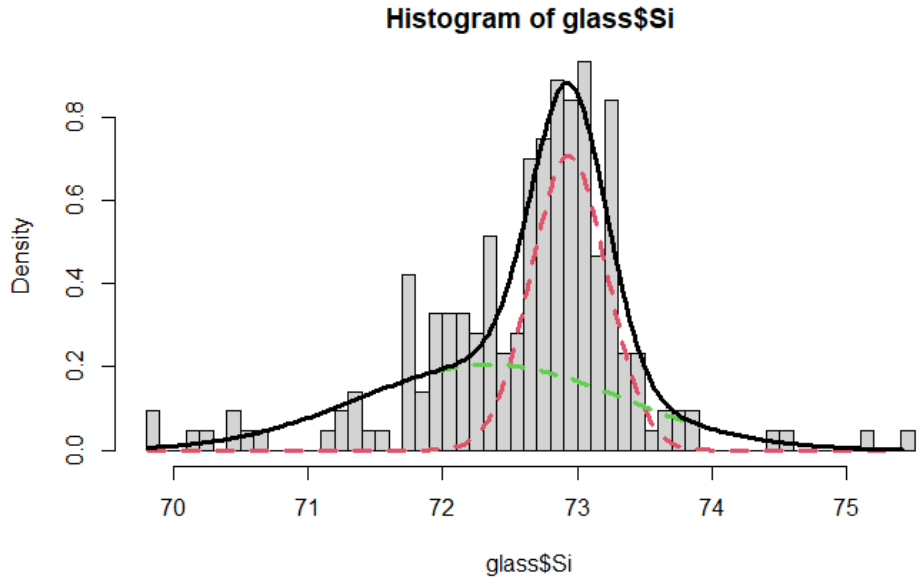









```
mu.hat1 <- exp(mix_si.GA[["models"]][[1]][["mu.coefficients"]])
sigma.hat1 <- exp(mix_si.GA[["models"]][[1]][["sigma.coefficients"]])
mu.hat2 <- exp(mix_si.GA[["models"]][[2]][["mu.coefficients"]])
sigma.hat2 <- exp(mix_si.GA[["models"]][[2]][["sigma.coefficients"]])

hist(glass$Si, breaks = 50,freq = FALSE)
lines(seq(min(glass$Si),max(glass$Si),length=length(glass$Si)),mix_si.GA[["prob"]][1]*dGA(
seq(min(glass$Si),max(glass$Si),length=length(glass$Si)), mu = mu.hat1, sigma =
sigma.hat1),lty=2,lwd=3,col=2)
lines(seq(min(glass$Si),max(glass$Si),length=length(glass$Si)),mix_si.GA[["prob"]][2]*dGA(
seq(min(glass$Si),max(glass$Si),length=length(glass$Si)), mu = mu.hat2, sigma =
sigma.hat2),lty=2,lwd=3,col=3)
lines(seq(min(glass$Si),max(glass$Si),length=length(glass$Si)),
      mix_si.GA[["prob"]][1]*dGA(seq(min(glass$Si),max(glass$Si),length=length(glass$Si)),
mu = mu.hat1, sigma = sigma.hat1) +
mix_si.GA[["prob"]][2]*dGA(seq(min(glass$Si),max(glass$Si),length=length(glass$Si)), mu =
mu.hat2, sigma = sigma.hat2), lty = 1, lwd = 3, col = 1)
```




```
mix_si.GA$prob
```

```
## [1] 0.4970444 0.5029556
```

The first group explains the 49.70% of our distribution and the second group explains the 50.29% of the distribution, so I assume that two distributions are enough.

```
aic <- AIC(mix_si.GA,fit_si.BCT)
aic$BIC <- c(fit_si.BCT$sbc,mix_si.GA$sbc)
aic$LogLik <- c(logLik(fit_si.BCT),logLik(mix_si.GA))
aic
```

```
##            df      AIC      BIC    LogLik
## fit_si.BCT  4 452.0420 465.5059 -222.0210
## mix_si.GA   5 453.0217 469.8515 -221.5108
```

The best distribution model which fits the "Si" variable is the Box-Cox t distribution with the minimum value of AIC and BIC and the maximum value of Log-likelihood.

Goodness of fit

```
gamlss:::LR.test(fit_si.BCT,mix_si.GA)
```

```
##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
##        Null model: deviance= 444.042 with  4 deg. of freedom
##  Alternative model: deviance= 443.0217 with  5 deg. of freedom
##
##  LRT = 1.020389 with 1 deg. of freedom and p-value= 0.3124268
```

According to the likelihood-ratio test, as the p-value is greater than 0.05, the Box-Cox t distribution has been accepted and the mixture of Gamma distribution has been rejected.

K

It's a continuous quantitative variable that can assume value within range [0,6.21].

```
summary(K)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1225  0.5550  0.4971  0.6100  6.2100
```

According to the position indices, the observations have the smallest value of 0 and the greatest value of 6.21, at least 25% has a value of 0.1225, at least 50% has a value of 0.555 and at least 75% has a value of 0.61. The whole observations' value is above 0.4971 and the interquartile distance is 0.49.

In detail, there are 7 outliers out of 214:

```
boxplot.stats(K)$out

## [1] 1.68 6.21 6.21 1.76 1.46 2.70 1.41
```

Thanks to the shape and variability indices I can predict the shape of the curve:

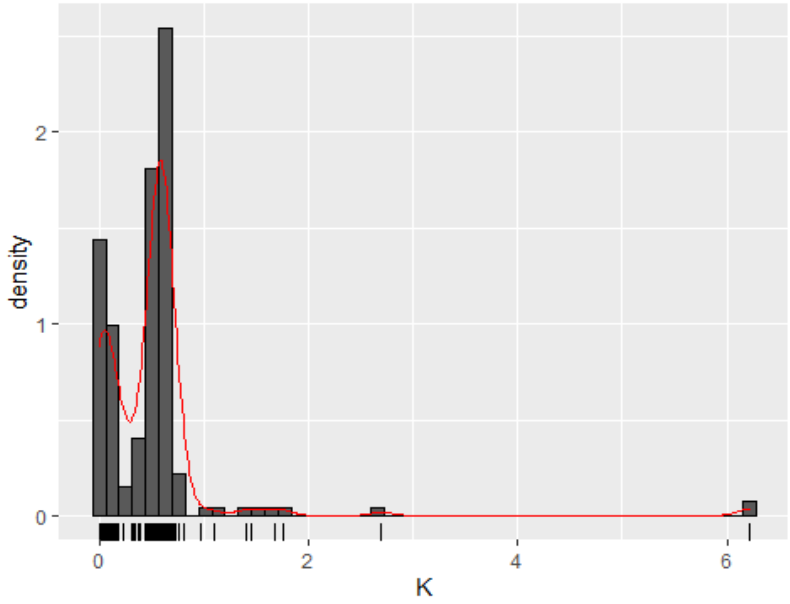
```
labstatR:::skew(K)

## [1] 6.505636

labstatR:::kurt(K)

## [1] 56.39233

sd(K)

## [1] 0.6521918
```

The coefficient of skewness is positive, and the distribution is right-skewed, this means that the graph is asymmetric, and the values are concentrated on the left side of the graph.

The Kurtosis index value is greater than 3, so I assume that the distribution is leptokurtic with a higher shape concerning the normal distribution.

The standard deviation has a small value, this means that a great of the value are concentrated near the mean. This is confirmed by plotting the histogram and highlighting the density function:

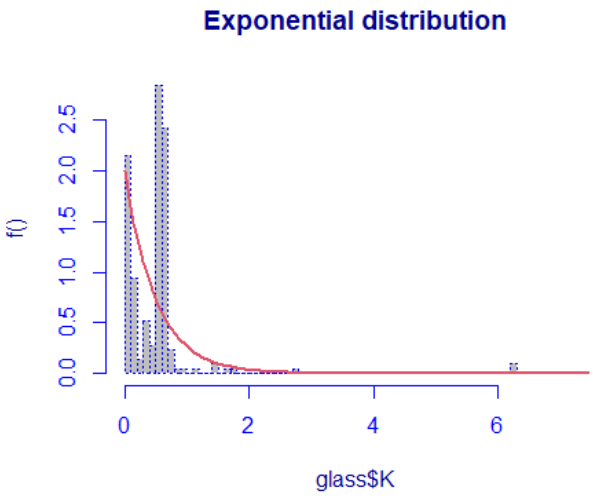
```
ggplot2:::ggplot(glass, aes(x=K))+
  geom_histogram(aes(y=..density..), color='black',bins=50)+
  geom_rug()+
  geom_density(color='red')
```

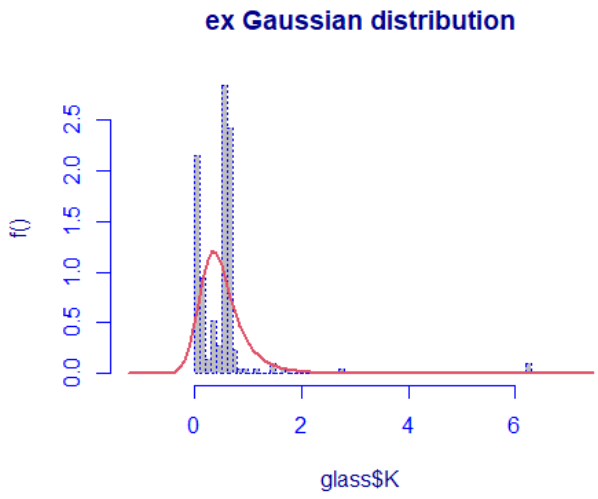


Single parametric distributions

Throw the following code I will try to establish what is the best distribution which fit the variable "K". I first introduce five possible distributions which handle the value of the positive real line and for each, I will compute the value of the parameters which are called mu, sigma, nu, and tau:

```
par(mfrow=c(1,2))

fit_k.EXP <- gamlss:::histDist(glass$K, family=EXP, nbins = 50, main="Exponential distribu
tion")
fit_k.exGAUS <- gamlss:::histDist(glass$K, family=exGAUS, nbins = 50, main="ex Gaussian di
stribution")
```






```
fit_k.EXP$df.fit
```

```
## [1] 1
fitted(fit_k.EXP, "mu")[1]
## [1] 0.4970561

fit_k.exGAUS$df.fit
## [1] 3
fitted(fit_k.exGAUS, "mu")[1]
## [1] 0.1493772
fitted(fit_k.exGAUS, "sigma")[1]
## [1] 0.2176919
fitted(fit_k.exGAUS, "nu")[1]
## [1] 0.3476789
```

In the next code I will create a dataframe that shows the ranking of AIC, BIC, and Log-likelihood:

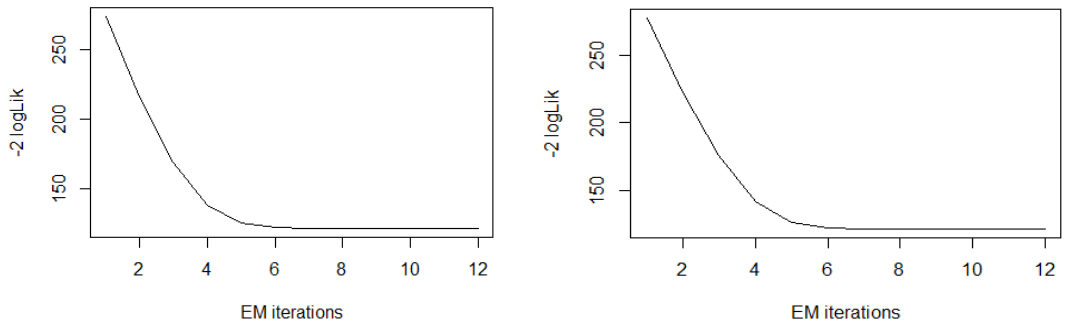
```
aic <- AIC(fit_k.EXP,fit_k.exGAUS)
aic$BIC <- c(fit_k.EXP$sbc,fit_k.exGAUS$sbc)
aic$LogLik <- c(logLik(fit_k.EXP), logLik(fit_k.exGAUS))
aic

##              df       AIC       BIC     LogLik
## fit_k.EXP     1 130.80556 134.17153 -64.402779
## fit_k.exGAUS 3 193.2857  203.3836  -93.64284
```

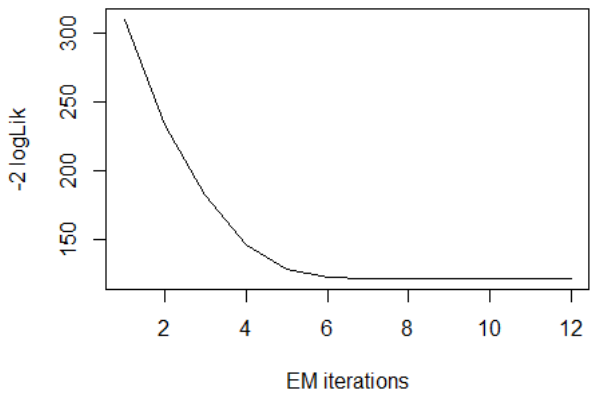
According to the dataframe, the best distribution model which fits the "K" variable is the Exponential with 1 parameter.

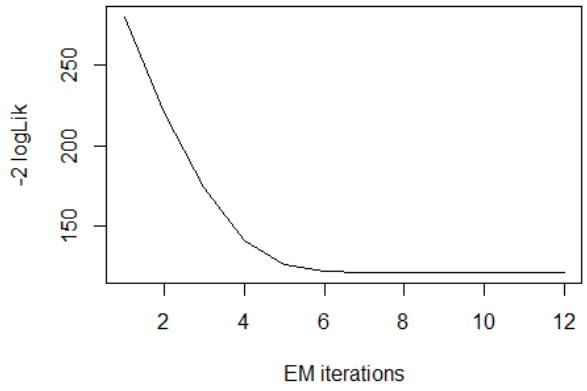
Mixture of distributions

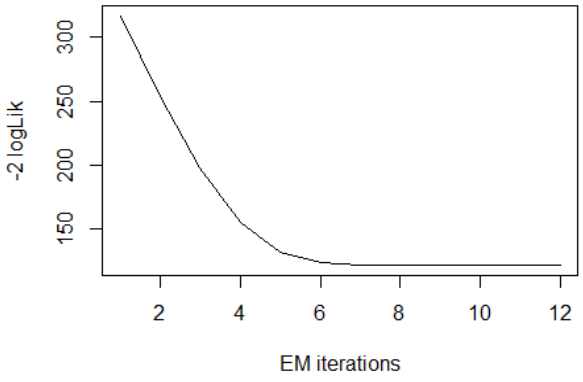
In the following code I will compute a mixture of two Normal distributions and, to find the best distribution, the algorithm is repeated 5 times:

```
mix_k.NO <- gamlss.mx:::gamlssMXfits(n = 5, glass$K~1, family=NO, K = 2, data = NULL)
```

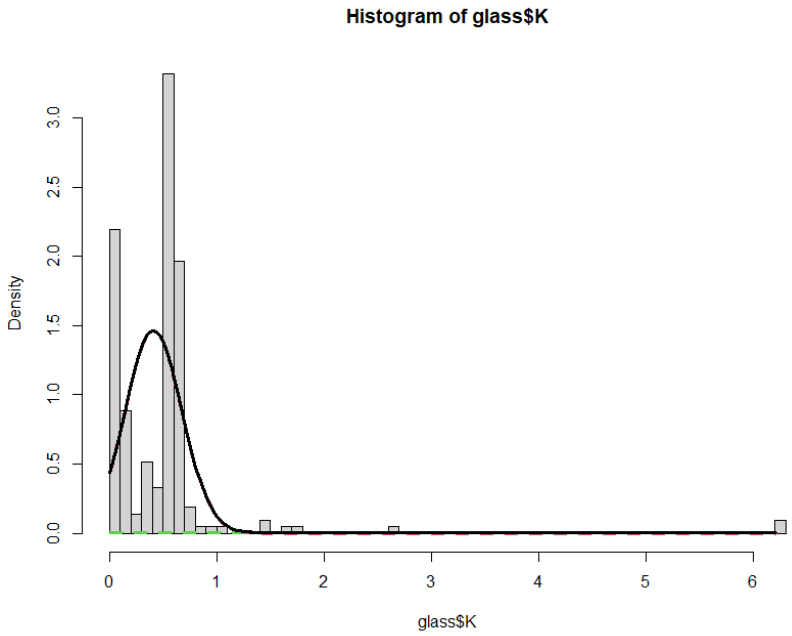









```
mu.hat1 <- mix_k.NO[["models"]][[1]][["mu.coefficients"]]
sigma.hat1 <- exp(mix_k.NO[["models"]][[1]][["sigma.coefficients"]])
mu.hat2 <- mix_k.NO[["models"]][[2]][["mu.coefficients"]]
sigma.hat2 <- exp(mix_k.NO[["models"]][[2]][["sigma.coefficients"]])

hist(glass$K, breaks = 50,freq = FALSE)
lines(seq(min(glass$K),max(glass$K),length=length(glass$K)),mix_k.NO[["prob"]][1]*dNO(seq(
min(glass$K),max(glass$K),length=length(glass$K)), mu = mu.hat1, sigma = sigma.hat1),lty=2
,lwd=3,col=2)
lines(seq(min(glass$K),max(glass$K),length=length(glass$K)),mix_k.NO[["prob"]][2]*dNO(seq(
min(glass$K),max(glass$K),length=length(glass$K)), mu = mu.hat2, sigma = sigma.hat2),lty=2
,lwd=3,col=3)
lines(seq(min(glass$K),max(glass$K),length=length(glass$K)), mix_k.NO[["prob"]][1]*dNO(seq
(min(glass$K),max(glass$K),length=length(glass$K)), mu = mu.hat1, sigma = sigma.hat1) + mi
x_k.NO[["prob"]][2]*dNO(seq(min(glass$K),max(glass$K),length=length(glass$K)), mu = mu.hat
2, sigma = sigma.hat2), lty = 1, lwd = 3, col = 1)
```




```
mix_k.NO$prob

## [1] 0.03600804 0.96399196
```

The first group explains the 0.04% of the distribution and the second group explains the 96.40% of the distribution, so I can predict that the mixture of the models doesn't fit our distribution.

```
aic <- AIC(mix_k.NO,fit_k.EXP)
aic$BIC <- c(fit_k.EXP$sbc,mix_k.NO$sbc)
aic$LogLik <- c(logLik(fit_k.EXP), logLik(mix_k.NO))
aic

##           df       AIC       BIC     LogLik
## fit_k.EXP  1 130.80556 134.17153 -64.402779
## mix_k.NO   5 131.37881 148.20869 -60.689404
```

The best distribution model which fits the "K" variable is the Exponential with 1 parameter, with the minimum value of AIC and BIC and the maximum value of Log-likelihood.

Ca

It's a continuous quantitative variable that can assume values within range [5.430,16.19].

```
summary(Ca)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   5.430   8.240   8.600   8.957   9.172  16.190
```

According to the position indices, the observations have the smallest value of 5.430 and the greatest value of 16.190, at least 25% has a value of 8.240, at least 50% has a value of 8.60 and at

least 75% has a value of 9.172. The whole observations' value is above 8.957 and the interquartile distance is 0.932.

In detail, there are 26 outliers out of 214:

```
boxplot.stats(Ca)$out
```

```
##  [1] 11.64 10.79 13.24 13.30 16.19 11.52 10.99 14.68 14.96 14.40 11.14 13.44
## [13]  5.87 11.41 11.62 11.53 11.32 12.24 12.50 11.27 10.88 11.22  6.65  5.43
## [25]  5.79  6.47
```

Thanks to the variability and shape indices I can predict the shape of the curve:

```
labstatR:::skew(Ca)
```

```
## [1] 2.032677
```

```
labstatR:::kurt(Ca)
```

```
## [1] 9.498968
```

```
sd(Ca)
```

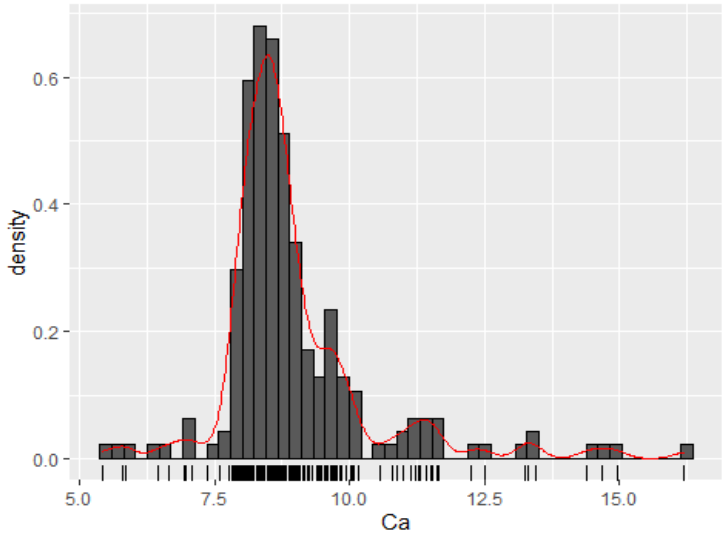
```
## [1] 1.423153
```

The coefficient of skewness is positive, and the distribution is right-skewed, this means that the graph is asymmetric, and the values are concentrated on the left side of the graph.

The Kurtosis index value is greater than 3, so we can assume that the distribution is leptokurtic with a higher shape concerning the normal distribution.

The standard deviation has a value greater than 1, this means that the values are moderately concentrated near the mean. This is confirmed by plotting the histogram and highlighting the density function:

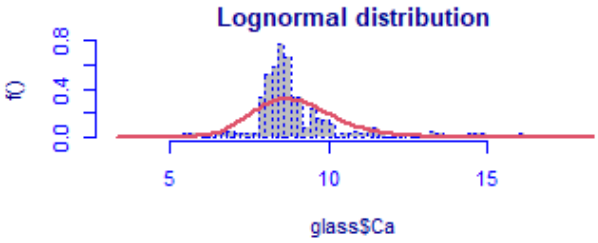
```
ggplot2:::ggplot(glass, aes(x=Ca))+
  geom_histogram(aes(y=..density..), color='black', bins=50)+
  geom_rug()+
  geom_density(color='red')
```

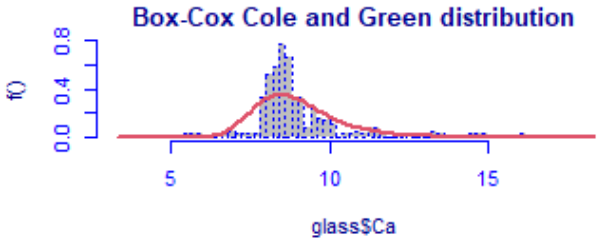


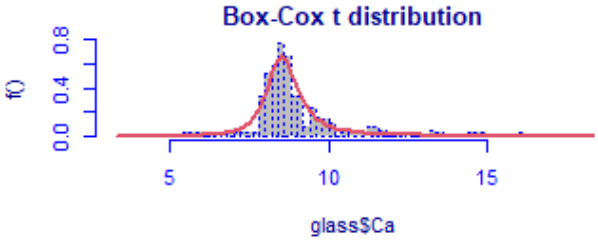
Single parametric distributions

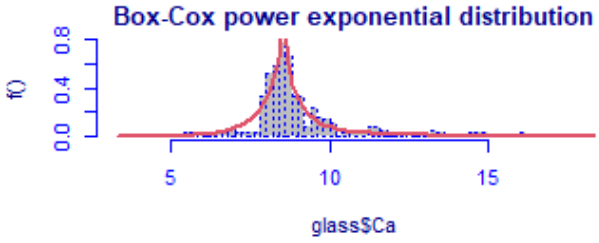
Throw the following code I will try to establish what is the best distribution which fits the variable Ca. I first introduce five possible distributions which handle the value of positive real line and for each, I will compute the value of the parameters which are called mu, sigma, nu, and tau:

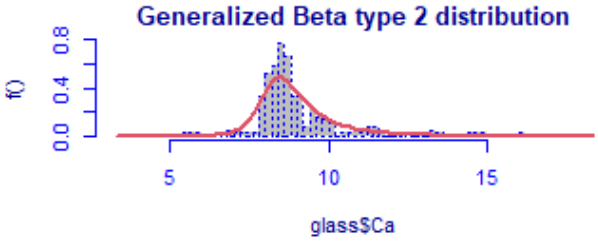
```
par(mfrow=c(3,2))
fit_ca.LOGNO <- gamlss:::histDist(glass$Ca, family=LOGNO, nbins = 50, main="Lognormal dist
ribution")
fit_ca.BCCG <- gamlss:::histDist(glass$Ca, family=BCCG, nbins = 50, main="Box-Cox Cole and
Green distribution")
fit_ca.BCT <- gamlss:::histDist(glass$Ca, family=BCT, nbins = 50, main="Box-Cox t distribu
tion")
fit_ca.BCPE <- gamlss:::histDist(glass$Ca, family=BCPE, nbins = 50, main="Box-Cox power ex
ponential distribution")
fit_ca.GB2 <- gamlss:::histDist(glass$Ca, family=GB2, nbins = 50, main="Generalized Beta t
ype 2 distribution")
```












```
fit_ca.LOGNO$df.fit
## [1] 2
fitted(fit_ca.LOGNO, "mu")[1]
## [1] 2.181504
fitted(fit_ca.LOGNO, "sigma")[1]
## [1] 0.1437452

fit_ca.BCCG$df.fit
## [1] 3
fitted(fit_ca.BCCG, "mu")[1]
## [1] 8.764468
fitted(fit_si.BCCG, "sigma")[1]
## [1] 0.01030971
fitted(fit_si.BCCG, "nu")[1]
## [1] 12.46839

fit_ca.BCT$df.fit
## [1] 4
fitted(fit_ca.BCT, "mu")[1]
## [1] 8.662328
fitted(fit_ca.BCT, "sigma")[1]
## [1] 0.06681865
fitted(fit_ca.BCT, "nu")[1]
## [1] -4.812332
fitted(fit_ca.BCT, "tau")[1]
## [1] 1.627906

fit_ca.BCPE$df.fit
```

```
## [1] 4
fitted(fit_ca.BCPE, "mu")[1]
## [1] 8.55
fitted(fit_ca.BCPE, "sigma")[1]
## [1] 0.1356743
fitted(fit_ca.BCPE, "nu")[1]
## [1] -1.412286
fitted(fit_ca.BCPE, "tau")[1]
## [1] 0.7004667

fit_ca.GB2$df.fit
## [1] 4
fitted(fit_ca.GB2, "mu")[1]
## [1] 8.27091
fitted(fit_ca.GB2, "sigma")[1]
## [1] 27.17604
fitted(fit_ca.GB2, "nu")[1]
## [1] 0.7418024
fitted(fit_ca.GB2, "tau")[1]
## [1] 0.3472989
```

In the next code I will create a dataframe which shows the ranking of AIC, BIC, and Log-likelihood:

```
aic <- AIC(fit_ca.LOGNO,fit_ca.BCCG,fit_ca.BCT,fit_ca.BCPE,fit_ca.GB2)
aic$BIC <- c(fit_ca.BCT$sbc,fit_ca.BCPE$sbc,fit_ca.GB2$sbc,fit_ca.BCCG$sbc,fit_ca.LOGNO$sb
c)
aic$LogLik <- c(logLik(fit_ca.BCT), logLik(fit_ca.BCPE), logLik(fit_ca.GB2), logLik(fit_ca
.BCCG), logLik(fit_ca.LOGNO))
aic

##              df      AIC      BIC    LogLik
## fit_ca.BCT    4 617.6081 631.0720 -304.8040
## fit_ca.BCPE   4 624.2682 637.7321 -308.1341
## fit_ca.GB2    4 640.4233 653.8872 -316.2117
## fit_ca.BCCG   3 698.7394 708.8373 -346.3697
## fit_ca.LOGNO  2 714.7931 721.5251 -355.3966
```

Goodness of fit

According to the dataframe, the best distribution model which fits the "Ca" variable is the Box-Cox t with 4 parameters.

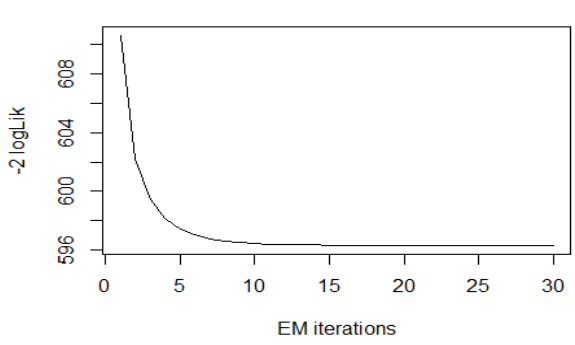
```
gamlss:::LR.test(fit_ca.BCPE,fit_ca.BCT)

##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
##        Null model: deviance= 616.2682 with  4 deg. of freedom
##  Alternative model: deviance= 609.6081 with  4 deg. of freedom
##
##  LRT = 6.660121 with 0 deg. of freedom and p-value= 0
```

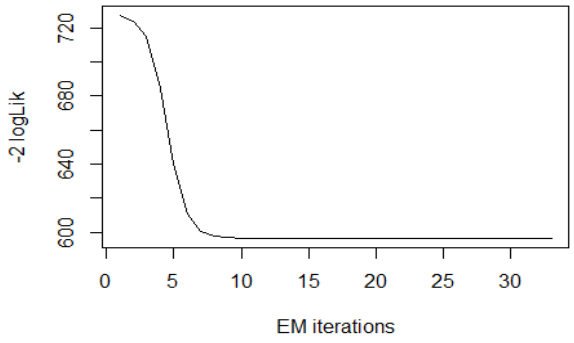
Throw the Likelihood-ratio test, as the p-value is less than 0.05, the null hypothesis, which corresponds to the Box-Cox Power Exponential distribution, has been rejected and the Box-Cox t distribution has been accepted.

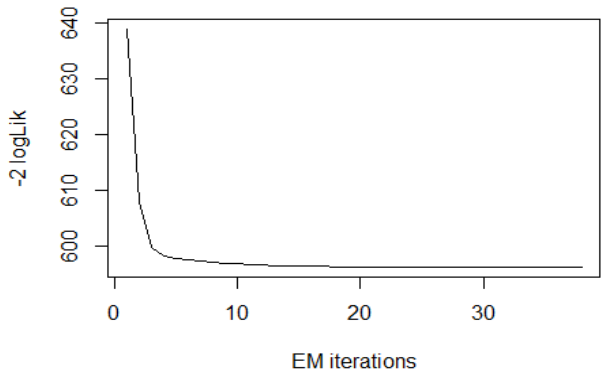
Mixture of distributions

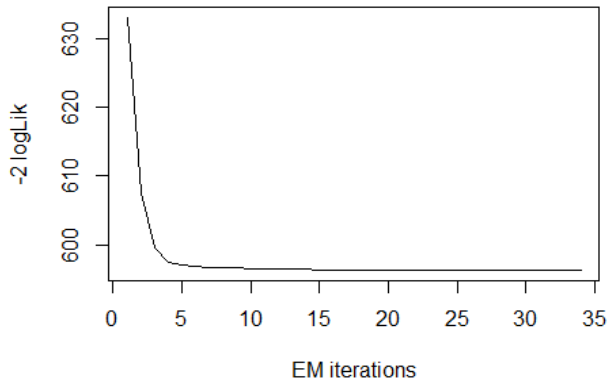
In the following code I will compute a mixture of two Gamma distributions and, to find the best distribution, the algorithm is repeated 5 times:

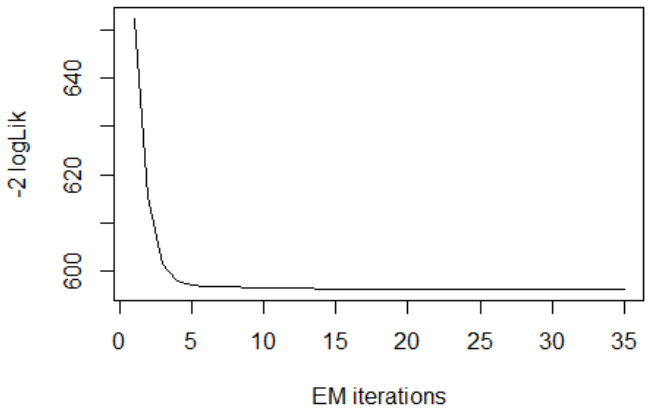
```
mix_ca.GA <- gamlss.mx:::gamlssMXfits(n = 5, glass$Ca~1, family=GA, K = 2, data = NULL)
```



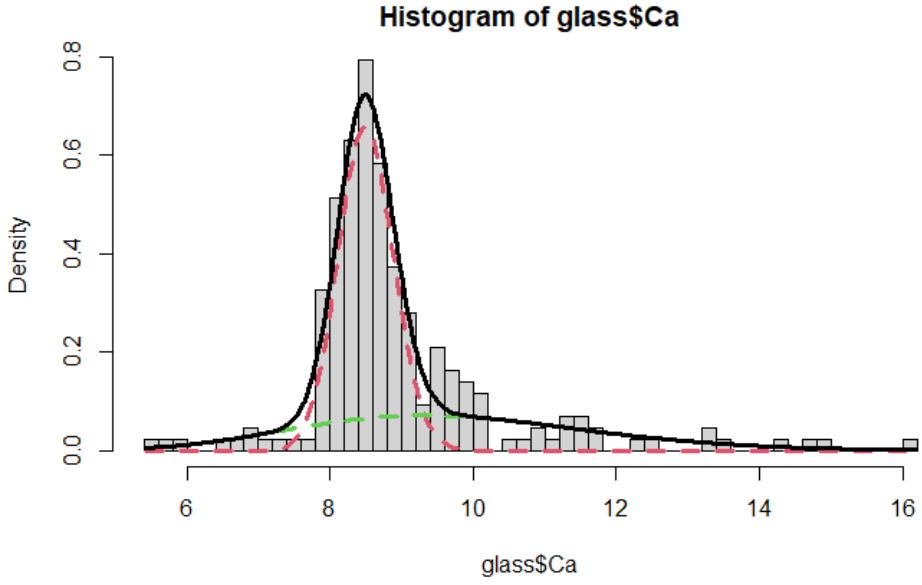









```
mu.hat1 <- exp(mix_ca.GA[["models"]][[1]][["mu.coefficients"]])
sigma.hat1 <- exp(mix_ca.GA[["models"]][[1]][["sigma.coefficients"]])
mu.hat2 <- exp(mix_ca.GA[["models"]][[2]][["mu.coefficients"]])
sigma.hat2 <- exp(mix_ca.GA[["models"]][[2]][["sigma.coefficients"]])

hist(glass$Ca, breaks = 50,freq = FALSE)
lines(seq(min(glass$Ca),max(glass$Ca),length=length(glass$Ca)),mix_ca.GA[["prob"]][1]*dGA(
seq(min(glass$Ca),max(glass$Ca),length=length(glass$Ca)), mu = mu.hat1, sigma =
sigma.hat1),lty=2,lwd=3,col=2)
lines(seq(min(glass$Ca),max(glass$Ca),length=length(glass$Ca)),mix_ca.GA[['prob']][2]*dGA(
seq(min(glass$Ca),max(glass$Ca),length=length(glass$Ca)), mu = mu.hat2, sigma =
sigma.hat2),lty=2,lwd=3,col=3)
lines(seq(min(glass$Ca),max(glass$Ca),length=length(glass$Ca)),
mix_ca.GA[["prob"]][1]*dGA(seq(min(glass$Ca),max(glass$Ca),length=length(glass$Ca)), mu =
mu.hat1, sigma = sigma.hat1) +
mix_ca.GA[["prob"]][2]*dGA(seq(min(glass$Ca),max(glass$Ca),length=length(glass$Ca)), mu =
mu.hat2, sigma = sigma.hat2), lty = 1, lwd = 3, col = 1)
```



```
mix_ca.GA$prob

## [1] 0.3619695 0.6380305
```

The first group explains the 36.20% of the distribution and the second group explains the 63.80% of the distribution, so I can assume that two distributions are enough.

```
aic <- AIC(mix_ca.GA,fit_ca.BCT)
aic$BIC <- c(mix_ca.GA$sbc,fit_ca.BCT$sbc)
aic$LogLik <- c(logLik(mix_ca.GA), logLik(fit_ca.BCT))
aic

##             df      AIC      BIC    LogLik
## mix_ca.GA   5 606.3232 623.1531 -298.1616
## fit_ca.BCT  4 617.6081 631.0720 -304.8040
```

The best distribution model which fits the "Ca" variable is the mixture of Gamma distributions with the minimum value of AIC and BIC and the maximum value of Log-likelihood.

Goodness of fit

```
gamlss:::LR.test(fit_ca.BCT, mix_ca.GA)

##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
##        Null model: deviance= 609.6081 with  4 deg. of freedom
##  Alternative model: deviance= 596.3232 with  5 deg. of freedom
##
##  LRT = 13.28487 with 1 deg. of freedom and p-value= 0.0002675566
```

According to the Likelihood-ratio test, as the p-value is less than 0.05, the Box-Cox t distribution has been rejected and the mixture of Gamma distribution has been accepted.

Ba

It's a continuous quantitative variable that can assume value within range [0,3.15].

```
summary(Ba)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   0.175   0.000   3.150
```

According to the position indices, the observations have the smallest value of 0 and the greatest value of 3.15, at least 25% has a value of 0, at least 50% has a value of 0 and at least 75% has a value of 0. The whole observations' value is above 0.175 and the interquartile distance is 0.

In detail, there are 38 outliers out of 214:

```
boxplot.stats(Ba)$out
```

```
##  [1] 0.09 0.11 0.69 0.14 0.11 3.15 0.27 0.09 0.06 0.15 2.20 0.24 1.19 1.63 1.68
## [16] 0.76 0.64 0.40 1.59 1.57 0.61 0.81 0.66 0.64 0.53 0.63 0.56 1.71 0.67 1.55
## [31] 1.38 2.88 0.54 1.06 1.59 1.64 1.57 1.67
```

Thanks to the variability and shape indices I can predict the shape of the curve:

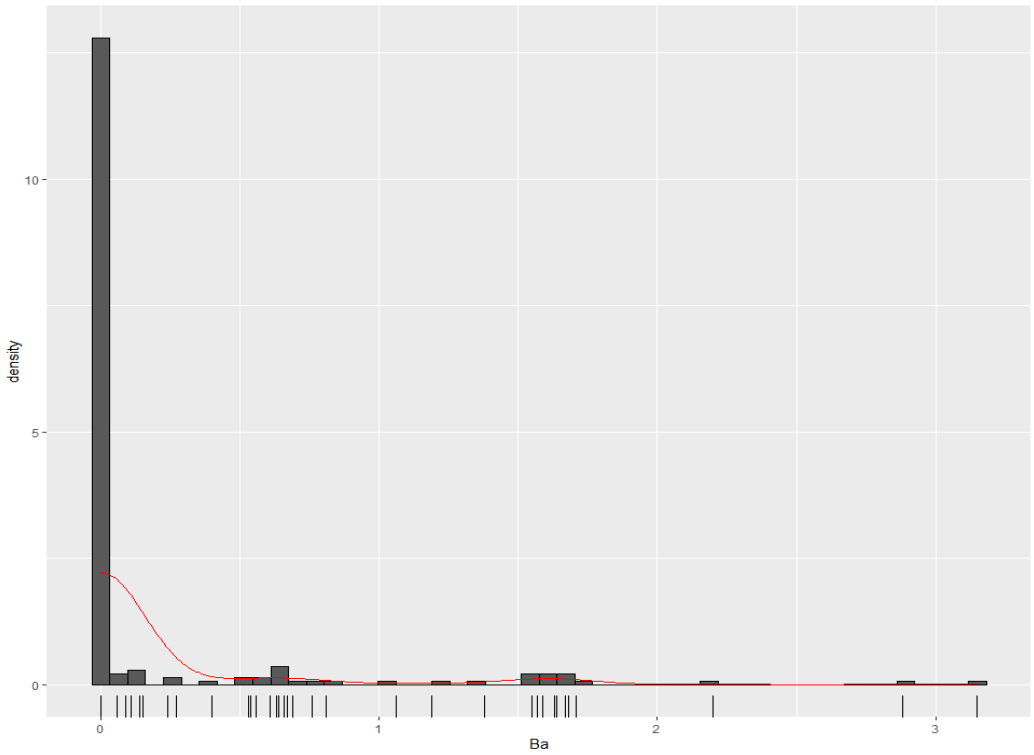
```
labstatR:::skew(Ba)

## [1] 3.392431

labstatR:::kurt(Ba)

## [1] 15.22207

sd(Ba)

## [1] 0.4972193
```

The coefficient of skewness is positive, and the distribution is right-skewed, this means that the graph is asymmetric, and the values are concentrated on the left side of the graph.

The Kurtosis index value is greater than 3, so we can assume that the distribution is leptokurtic with a higher shape concerning the normal distribution.

The standard deviation has a small value, this means that the values are concentrated near the mean. This is confirmed by plotting the histogram of the Si and highlighting the density function:

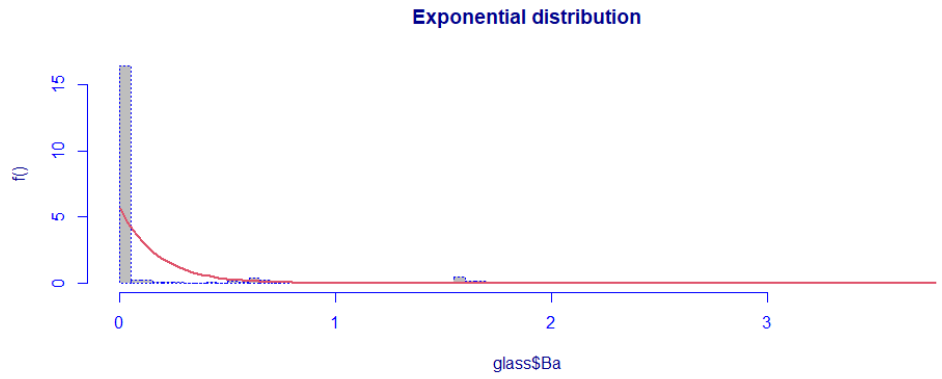
```
ggplot2:::ggplot(glass, aes(x=Ba))+
  geom_histogram(aes(y=..density..), color='black')+
  geom_rug()+
  geom_density(color='red')
```



Single parametric distributions

Throw the following code I introduce the Exponential distribution for the variable "Ba", which fit values of the positive real line, and I will compute the value of the parameters which are called mu, sigma, nu, and tau:

```
fit_ba.EXP <- gamlss:::histDist(glass$Ba, family=EXP, nbins = 50, main="Exponential distri
bution")
```




```
fit_ba.EXP$df.fit
## [1] 1
fitted(fit_ba.EXP, "mu")[1]
## [1] 0.1750467
```

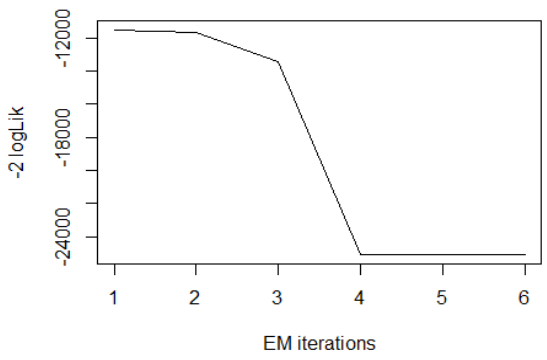
In the next code I will create a dataframe that shows the value of AIC, BIC, and Log-likelihood:

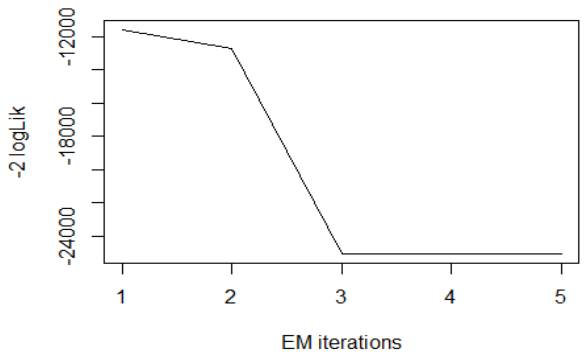
```
aic <- AIC(fit_ba.EXP)
aic$BIC <- fit_ba.EXP$sbc
aic$LogLik <- logLik(fit_ba.EXP)
aic
## [[1]]
## [1] -315.8766
##
## $BIC
## [1] -312.5106
##
## $LogLik
## 'log Lik.' 158.9383 (df=1)
```

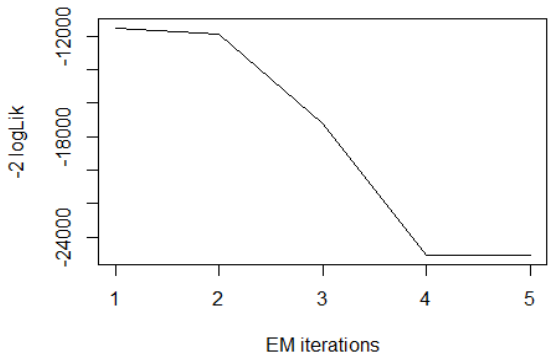
Mixture of distributions

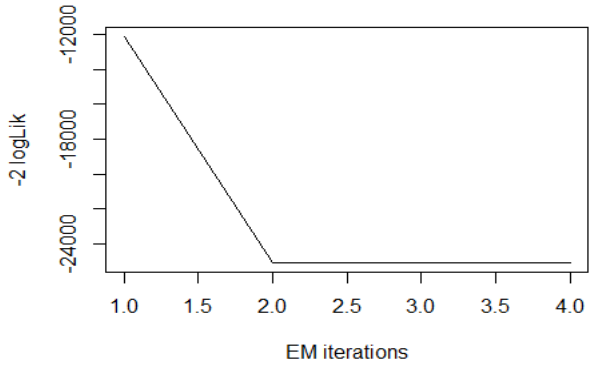
In the following code I will compute a mixture of two Pareto distributions and, to find the best distribution, the algorithm is repeated 5 times:

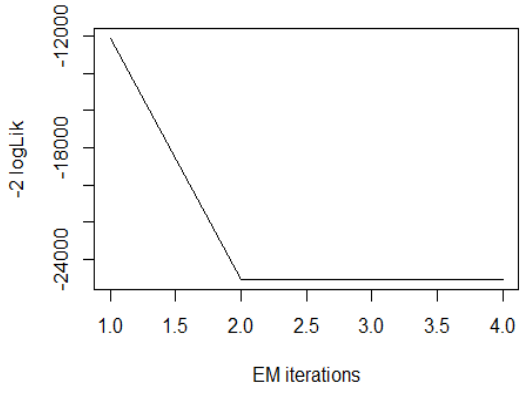
```
mix_ba.PARETO <- gamlss.mx:::gamlssMXfits(n = 5, glass$Ba~1, family=PARETO2, K = 2, data =
NULL)
```





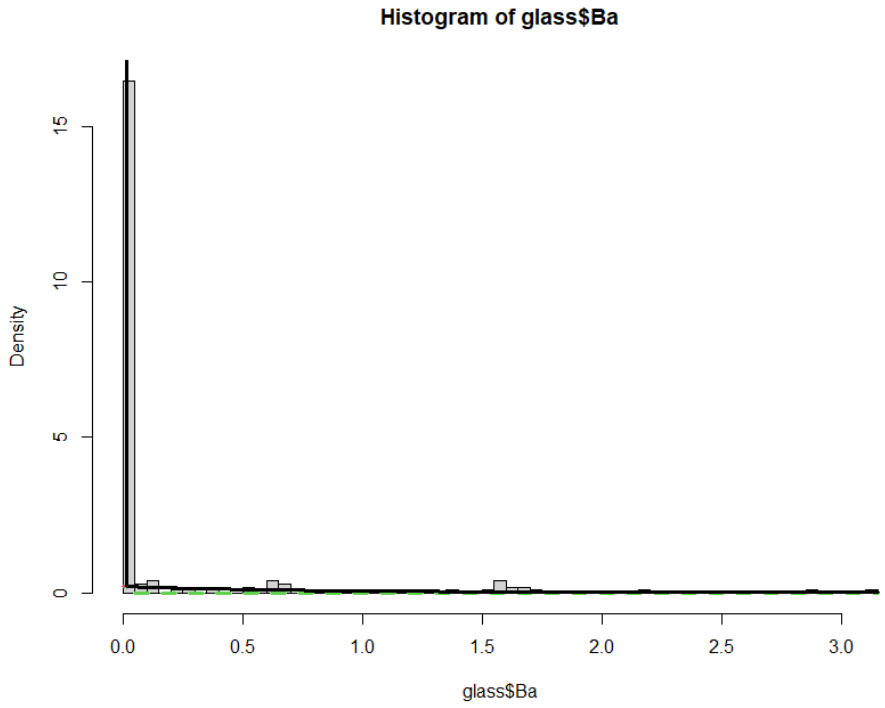







```
mu.hat1 <- exp(mix_ba.PARETO[["models"]][[1]][["mu.coefficients"]])
sigma.hat1 <- exp(mix_ba.PARETO[["models"]][[1]][["sigma.coefficients"]])
mu.hat2 <- exp(mix_ba.PARETO[["models"]][[2]][["mu.coefficients"]])
sigma.hat2 <- exp(mix_ba.PARETO[["models"]][[2]][["sigma.coefficients"]])

hist(glass$Ba, breaks = 50,freq = FALSE)
lines(seq(min(glass$Ba),max(glass$Ba),length=length(glass$Ba)),mix_ba.PARETO[["prob"]][1]*
```

```
dPARETO2(seq(min(glass$Ba),max(glass$Ba),length=length(glass$Ba)), mu = mu.hat1, sigma = s
igma.hat1),lty=2,lwd=3,col=2)
lines(seq(min(glass$Ba),max(glass$Ba),length=length(glass$Ba)),mix_ba.PARETO[["prob"]][2]*
dPARETO2(seq(min(glass$Ba),max(glass$Ba),length=length(glass$Ba)), mu = mu.hat2, sigma = s
igma.hat2),lty=2,lwd=3,col=3)
lines(seq(min(glass$Ba),max(glass$Ba),length=length(glass$Ba)), mix_ba.PARETO[["prob"]][1]
*dPARETO2(seq(min(glass$Ba),max(glass$Ba),length=length(glass$Ba)), mu = mu.hat1, sigma =
sigma.hat1) + mix_ba.PARETO[["prob"]][2]*dPARETO2(seq(min(glass$Ba),max(glass$Ba),length=l
ength(glass$Ba)), mu = mu.hat2, sigma = sigma.hat2), lty = 1, lwd = 3, col = 1)
```




```
mix_ba.PARETO$prob

## [1] 0.8224299 0.1775701
```

The first group explains 82.24% of our distribution and the second group explains 17.76% of our distribution, so I can assume that two distributions are enough.

```
aic <- AIC(mix_ba.PARETO,fit_ba.EXP)
aic$BIC <- c(mix_ba.PARETO$sbc,fit_ba.EXP$sbc)
aic$LogLik <- c(logLik(mix_ba.PARETO), logLik(fit_ba.EXP))
aic

##                df         AIC         BIC     LogLik
## mix_ba.PARETO  5 -25056.1150 -25039.2852 12533.0575
## fit_ba.EXP     1   -315.8766   -312.5106   158.9383
```

The best distribution model which fits the “Ba” variable is the mixture of Pareto distributions with the minimum value of AIC and BIC and the maximum value of Log-likelihood.

Goodness of fit

```
gamlss:::LR.test(fit_ba.EXP, mix_ba.PARETO)

##  Likelihood Ratio Test for nested GAMLSS models.
##  (No check whether the models are nested is performed).
##
##        Null model: deviance= -317.8766 with  1 deg. of freedom
##  Altenative model: deviance= -25066.12 with  5 deg. of freedom
##
##  LRT = 24748.24 with 4 deg. of freedom and p-value= 0
```

According to the likelihood-ratio test, as the p-value is less than 0.05, the Exponential distribution has been rejected and the mixture of Pareto distribution has been accepted.

Fe

It's a continuous quantitative variable that can assume value within range [0,0.51].

```
summary(Fe)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.05701 0.10000 0.51000
```

According to the position indices, the observations have the smallest value of 0 and the greatest value of 0.51, at least 25% has a value of 0, at least 50% has a value of 0 and at least 75% has a value of 0.1. The whole observations' value is above 0.57011 and the interquartile distance is 0.1.

In detail, there are 7 outliers out of 214:

```
boxplot.stats(Fe)$out

##  [1] 0.26 0.30 0.31 0.32 0.34 0.28 0.29 0.28 0.35 0.37 0.51 0.28
```

Thanks to the variability and shape indices I can predict the shape of the curve:

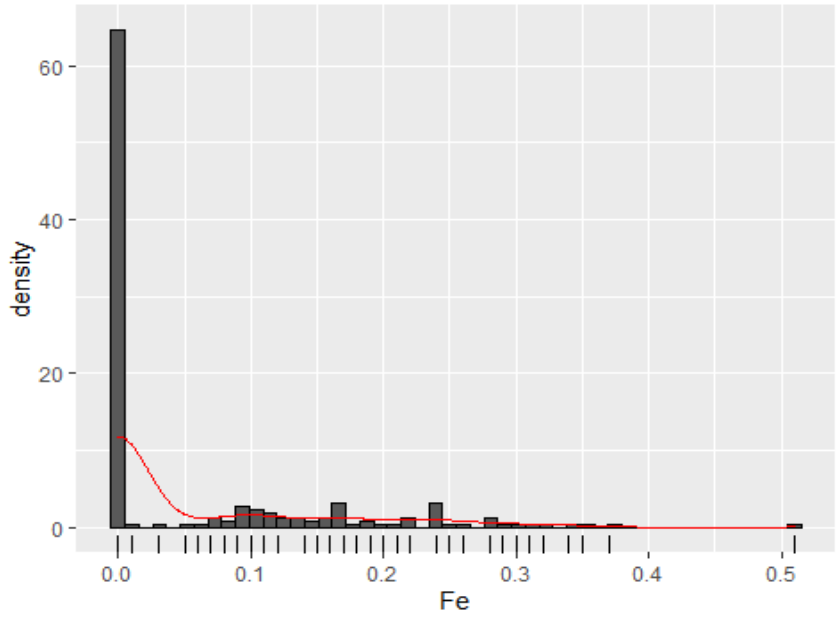
```
labstatR:::skew(Fe)

## [1] 1.742007

labstatR:::kurt(Fe)

## [1] 5.572318

sd(Fe)

## [1] 0.0974387
```

The coefficient of skewness is positive, and the distribution is right-skewed, this means that the graph is asymmetric, and the values are concentrated on the left side of the graph.

The Kurtosis index value is greater than 3, so we can assume that the distribution is leptokurtic with a higher shape concerning the normal distribution.

The standard deviation has a small value, this means that a great of the value are concentrated. This is confirmed by plotting the histogram of the Fe and highlighting the density function:

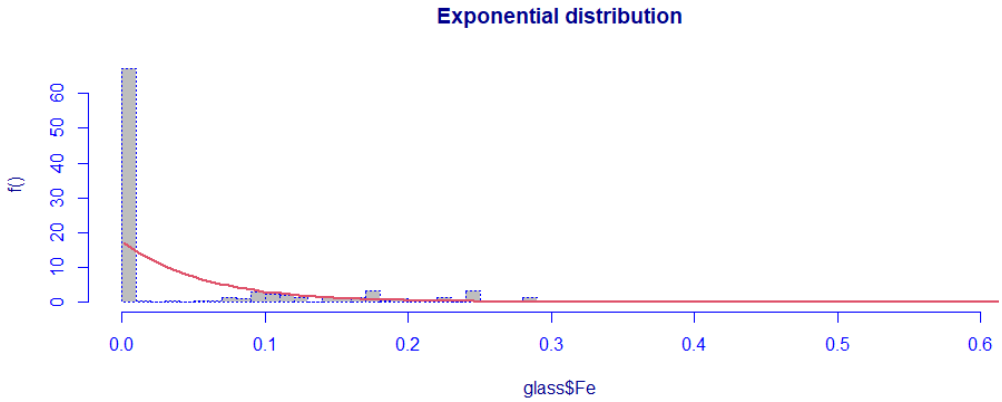
```
ggplot2:::ggplot(glass, aes(x=Fe))+
  geom_histogram(aes(y=..density..), color='black', bins=50)+
  geom_rug()+
  geom_density(color='red')
```



Single parametric distributions

Throw the following code I introduce the Exponential distribution which fit value on the positive real line and I will compute the value of the parameters which are called mu, sigma, nu, and tau:

```
fit_fe.EXP <- gamlss:::histDist(glass$Fe, family=EXP, nbins = 50, main="Exponential
distribution")
```




```
fit_fe.EXP$df.fit
```

```
## [1] 1
fitted(fit_fe.EXP, "mu")[1]
## [1] 0.05700935
```

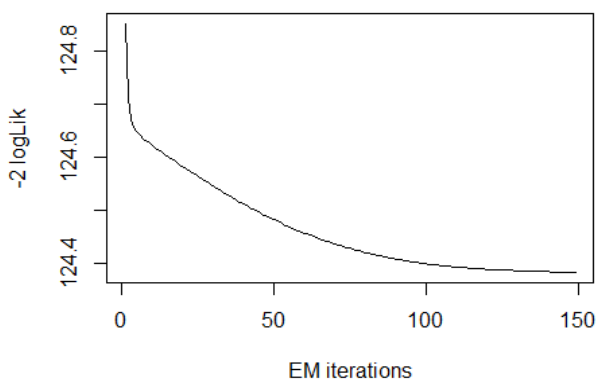
I create a dataframe that show the value of AIC, BIC, and Log-likelihood:

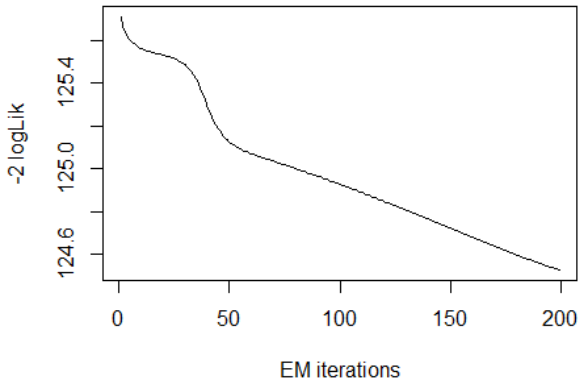
```
aic <- AIC(fit_fe.EXP)
aic$BIC <- fit_fe.EXP$sbc

aic$LogLik <- logLik(fit_fe.EXP)
aic

## [[1]]
## [1] -796.0231
##
## $BIC
## [1] -792.6572
##
## $LogLik
## 'log Lik.' 399.0116 (df=1)
```

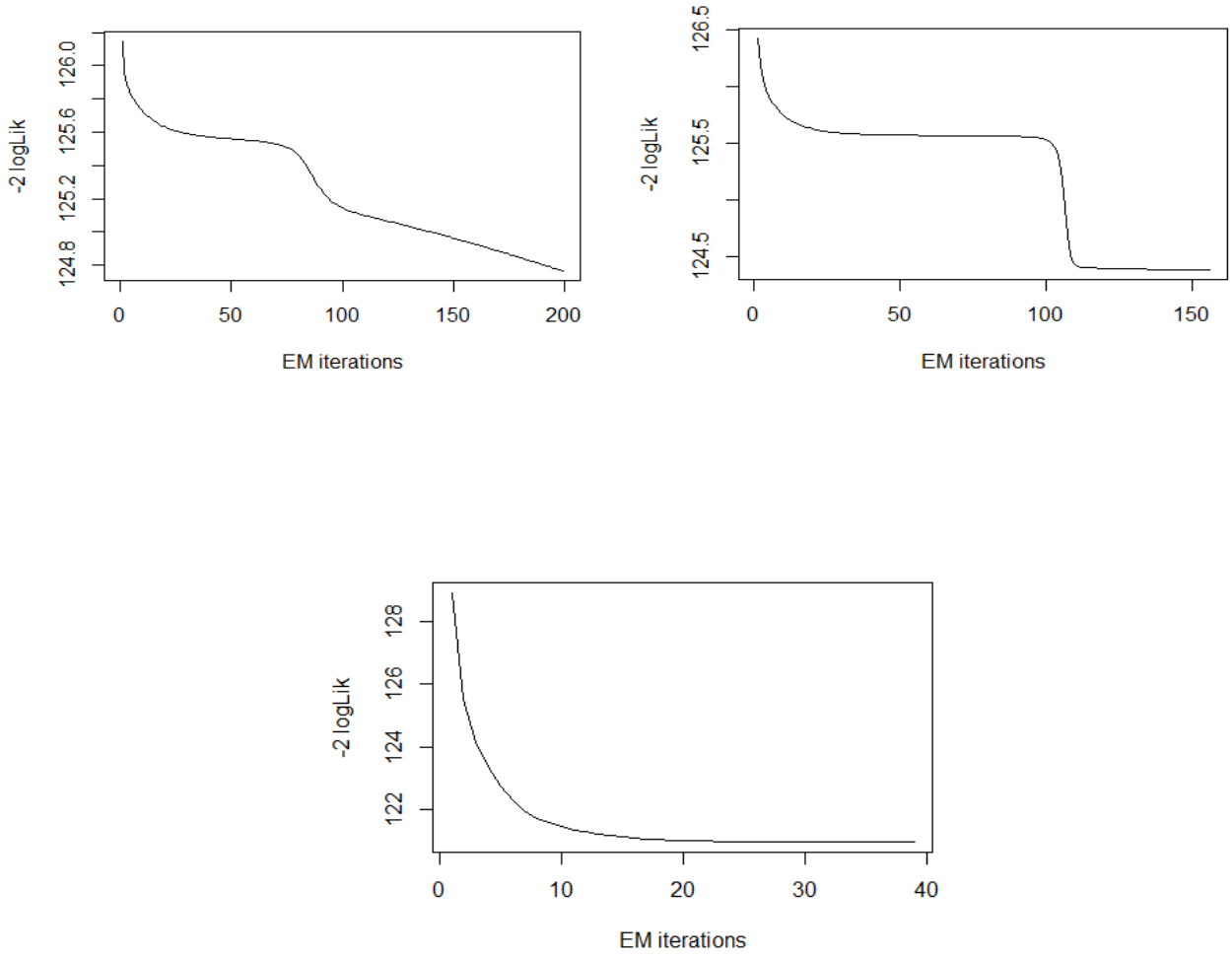
Mixture of distributions

In the following code, I will compute a mixture of two Beta-Inflated distributions and, to find the best distribution, the algorithm is repeated 5 times:

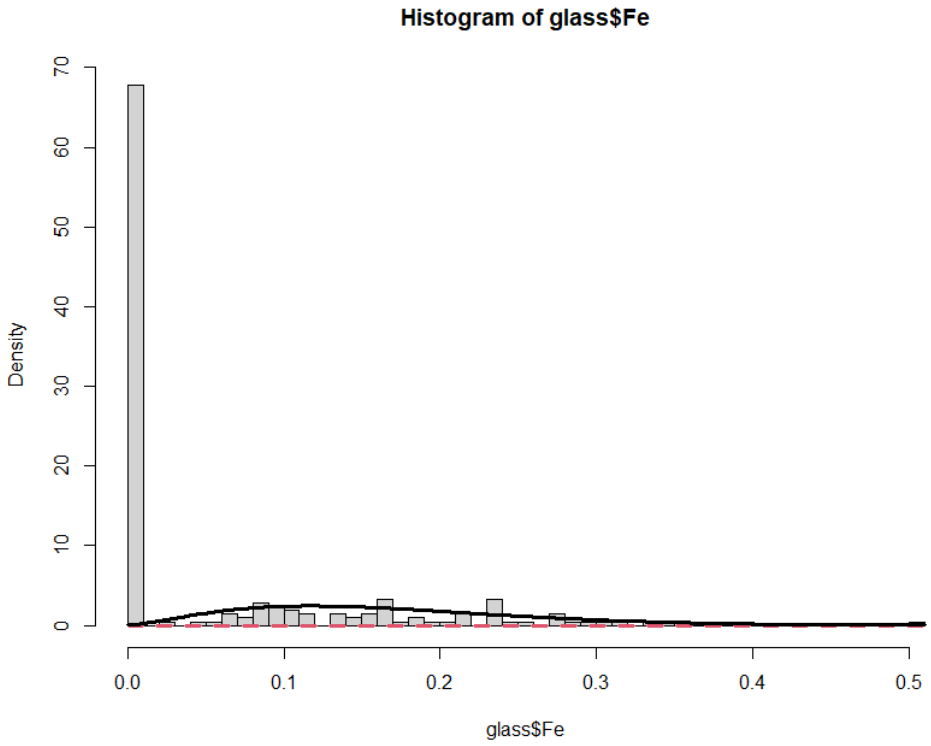
```
mix_fe.BEZI <- gamlss.mx:::gamlssMXfits(n = 5, glass$Fe~1, family=BEZI, K = 2, data = NULL
)
```








```
mu.hat1 <- plogis(mix_fe.BEZI[["models"]][[1]][["mu.coefficients"]])
sigma.hat1 <- exp(mix_fe.BEZI[["models"]][[1]][["sigma.coefficients"]])

mu.hat2 <- plogis(mix_fe.BEZI[["models"]][[2]][["mu.coefficients"]])
sigma.hat2 <- exp(mix_fe.BEZI[["models"]][[2]][["sigma.coefficients"]])

hist(glass$Fe, breaks = 50,freq = FALSE)
lines(seq(min(glass$Fe),max(glass$Fe),length=length(glass$Fe)),mix_fe.BEZI[["prob"]][1]*dB
EZI(seq(min(glass$Fe),max(glass$Fe),length=length(glass$Fe)), mu = mu.hat1, sigma =
sigma.hat1),lty=2,lwd=3,col=2)
lines(seq(min(glass$Fe),max(glass$Fe),length=length(glass$Fe)),mix_fe.BEZI[["prob"]][2]*dB
EZI(seq(min(glass$Fe),max(glass$Fe),length=length(glass$Fe)), mu = mu.hat2, sigma =
sigma.hat2),lty=2,lwd=3,col=3)
lines(seq(min(glass$Fe),max(glass$Fe),length=length(glass$Fe)),
mix_fe.BEZI[["prob"]][1]*dBEZI(seq(min(glass$Fe),max(glass$Fe),length=length(glass$Fe)),
mu = mu.hat1, sigma = sigma.hat1) +
mix_fe.BEZI[["prob"]][2]*dBEZI(seq(min(glass$Fe),max(glass$Fe),length=length(glass$Fe)),
mu = mu.hat2, sigma = sigma.hat2), lty = 1, lwd = 3, col = 1)
```




```
mix_fe.BEZI$prob

## [1] 0.6833097 0.3166903
```

The first group explains the 68.33% of the distribution and the second group explains the 31.67% of the distribution, so I can assume that two distributions are enough.

```
aic <- AIC(mix_fe.BEZI,fit_fe.EXP)
aic$BIC <- c(mix_fe.EXP$sbc,fit_fe.BEZI$sbc)
aic$LogLik <- c(logLik(mix_fe.EXP), logLik(fit_fe.BEZI))
aic

##             df       AIC        BIC    LogLik
## fit_fe.EXP   1 -796.0231   86.46474  399.01157
## mix_fe.BEZI  7   62.9029   86.46474 -24.45145
```

As we can see from the dataframe, the best distribution model which fits the "Fe" variable is the Exponential distribution with the minimum value of AIC and BIC and the maximum value of Log-likelihood.

Type

It's a categorical qualitative variable that has been transformed as a factor to perform a better analysis. The whole type of glass analyzed are:

```
glass$Type <- factor(Type, labels = c("Building Windows FP", "Building Windows NFP", "Vehi
cle Window FP", "Containers", "Tableware", "Headlamp"))
```

```
levels(glass$Type)

## [1] "Building Windows FP"  "Building Windows NFP" "Vehicle Window FP"    "Containers"
## "Tableware"

## [6] "Headlamp"
```

In the following code I show how many values correspond to each type of glass in the absolute frequency table:

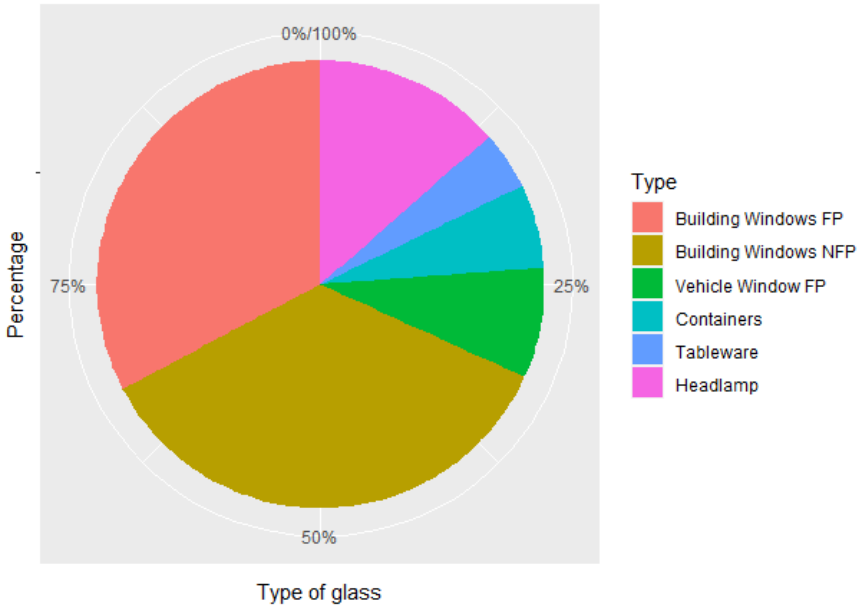
```
table(Type)

## Type
## Building Windows FP Building Windows NFP Vehicle Window FP Containers Tableware
##                  70                   76                17         13         9
## Headlamp
##       29
```

Considering the previous table, I decided to compute the relative frequency table to get the percentage:

```
round(table(Type)/length(Type)*100, digit=2)

## Type
## Building Windows FP Building Windows NFP Vehicle Window FP Containers Tableware
##               32.71                35.51              7.94       6.07      4.21
## Headlamp
##    13.55
```

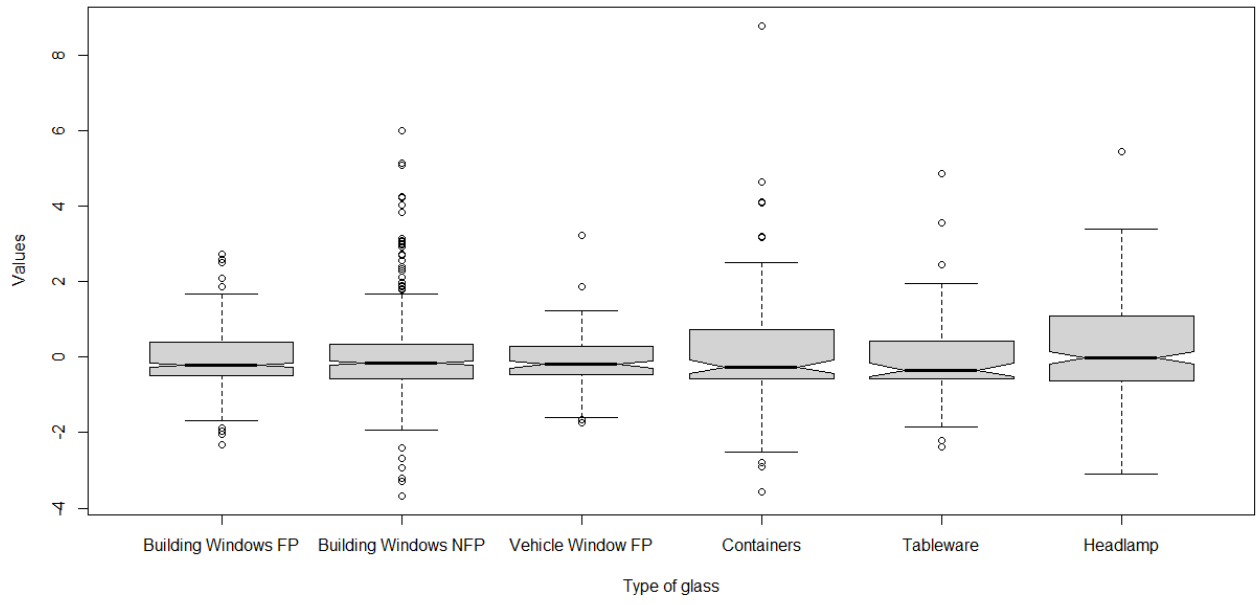
According to the frequency table, I can say that the dataset is not balanced 32.71% of the analyzed glass are building windows float-processed, 35.51% are building windows non-float-processed, 7.94% are vehicle windows float-processed, 6.07% are containers, 4.21% are tableware and 13.55% are headlamps. I plot the frequency table in a pie chart to translate the percentage in a graphical point of view:

```
ggplot2:::ggplot(glass, aes(y="", fill=Type)) +
  geom_bar(aes(x=..count../sum(..count..))) +
  scale_x_continuous(labels=scales::percent) +
  ylab('Percentage') +
  xlab('Type of glass') +
  coord_polar()
```



Throw the following code I want to analyze the position indices of the variable "Type":

```
boxplot <- boxplot(scaled_glass ~ glass$Type, notch=TRUE, xlab='Type of glass',
ylab='Values')
boxplot
table(boxplot$group)
## 1  2   3  4  5  6
## 12 38  4 10  5  1
```

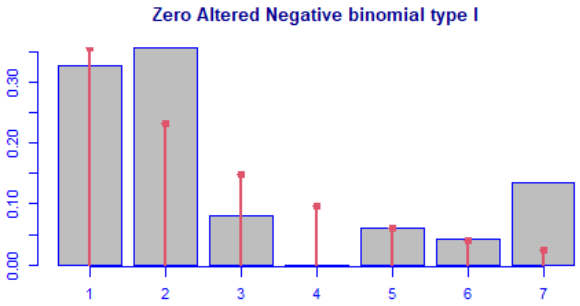


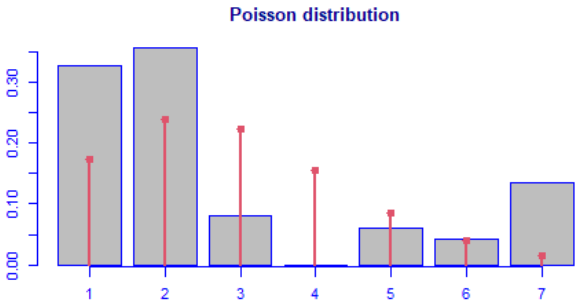
The headlamp has the highest value of interquartile distance and, as printed in the table above, the smallest number of outliers. The type of glass with the highest number of outliers is Building Windows Float-processed, with 38 outliers.

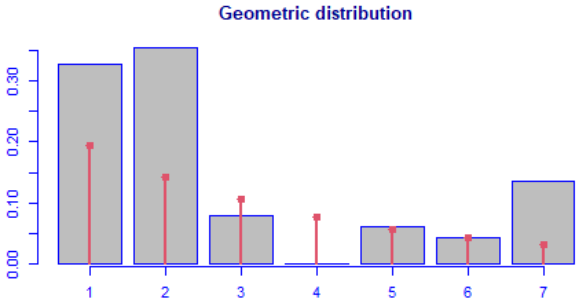
Single parametric distributions

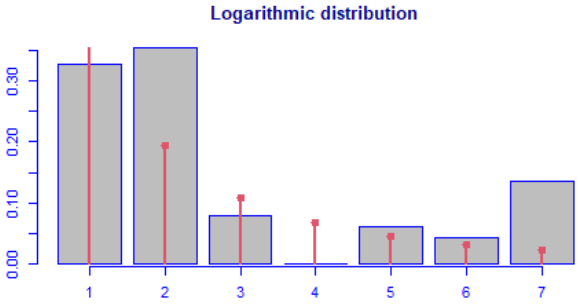
Throw the following code I will try to establish what is the best distribution which fits the variable "Type". I first introduce five discrete distributions. For each, I will show the parameters:

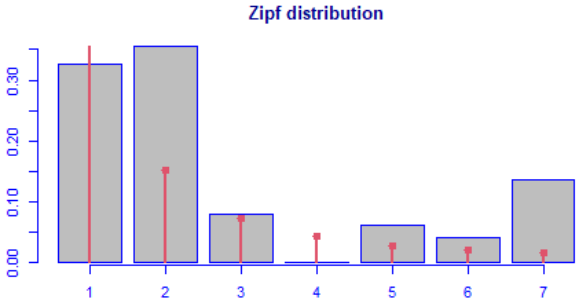
```
par(mfrow=c(3,2))

fit_type.ZANBI <- gamlss:::histDist(glass$Type, family=ZANBI, nbins = 50, main='Zero
Altered Negative binomial type I')
fit_type.PO <- gamlss:::histDist(glass$Type, family=PO, nbins = 50, main='Poisson
distribution')
fit_type.GEOM <- gamlss:::histDist(glass$Type, family=GEOM, nbins = 50, main='Geometric
distribution')
fit_type.LG <- gamlss:::histDist(glass$Type, family=LG, nbins = 50, main='Logarithmic
distribution')
fit_type.ZIPF <- gamlss:::histDist(glass$Type, family=ZIPF, nbins = 50, main='Zipf
distribution')
```












```
fit_type.ZANBI$df.fit
## [1] 3
fitted(fit_type.ZANBI, "mu")[1]
## [1] 1.897525
fitted(fit_type.ZANBI, "sigma")[1]
## [1] 0.4494074
fitted(fit_type.ZANBI, "nu")[1]
## [1] 2.080381e-10

fit_type.PO$df.fit
## [1] 1
fitted(fit_type.PO, "mu")[1]
## [1] 2.542056

fit_type.GEOM$df.fit
```

```
## [1] 1
fitted(fit_type.GEOM, "mu")[1]
## [1] 2.542056

fit_type.LG$df.fit
## [1] 1
fitted(fit_type.LG, "mu")[1]
## [1] 0.8070366

fit_type.ZIPF$df.fit
## [1] 1
fitted(fit_type.ZIPF, "mu")[1]
## [1] 0.8503
```

I create a dataframe that show a ranking of AIC, BIC, and Log-likelihood:

```
aic <- AIC(fit_type.ZANBI,fit_type.ZIPF, fit_type.PO, fit_type.GEOM, fit_type.LG)
aic$BIC <- c(fit_type.ZANBI$sbc,fit_type.LG$sbc, fit_type.ZIPF$sbc, fit_type.PO$sbc, fit_t
ype.GEOM$sbc)
aic$LogLik <- c(logLik(fit_type.ZANBI),logLik(fit_type.LG), logLik(fit_type.ZIPF), logLik(
fit_type.PO), logLik(fit_type.GEOM))
aic

##                df      AIC      BIC    LogLik
## fit_type.ZANBI  3 731.1809 741.2788 -362.5904
## fit_type.LG     1 759.9982 763.3642 -378.9991
## fit_type.PO     1 791.6041 836.3093 -415.4717
## fit_type.ZIPF   1 832.9433 794.9701 -394.8021
## fit_type.GEOM   1 904.2215 907.5875 -451.1107
```

The best distribution model which fits the "Type" variable is the Zero Altered negative binomial type I distribution with minimum AIC and BIC and maximum Log-likelihood.

Goodness of fit

```
gf <- vcd:::goodfit(glass$Type,method='MinChisq')
summary(gf)

##
##   Goodness-of-fit test for poisson distribution
##
##              X^2 df     P(> X^2)
## Pearson 107.7717  5 1.211357e-21
```

Throw the Pearson's chi-square test I reject the null hypothesis as the area on the right of X^2 is smaller than 0.05, so the variable 'Type' distribution is not a Poisson distribution.

Multivariate Analysis

Principal Component Analysis

The Principal Component Analysis (PCA) allows for a summary of the information contained in the dataset and finds the best representation on a lower-dimensional space: as this dataset contains 9 numerical variables, performing a multivariate analysis would imply the analysis of 36 scatter plots. For this reason, I will use the PCA, which will help me visualize the variables in a lower-dimensional space that explains the most variability of the original space.

As the PCA can be applied only to numerical continuous variables, I will compute it only in the first 9 variables of the dataset, excluding the "Type" variable.

The first requirement is that there must be a correlation between variables before grouping them into PC:

```
as.dist(round(cor(glass[-10]),2))
```

```
##       RI    Na    Mg    Al    Si     K    Ca    Ba
## Na -0.19
## Mg -0.12 -0.27
## Al -0.41  0.16 -0.48
## Si -0.54 -0.07 -0.17 -0.01
## K  -0.29 -0.27  0.01  0.33 -0.19
## Ca  0.81 -0.28 -0.44 -0.26 -0.21 -0.32
## Ba  0.00  0.33 -0.49  0.48 -0.10 -0.04 -0.11
## Fe  0.14 -0.24  0.08 -0.07 -0.09 -0.01  0.12 -0.06
```

I can see from the correlation matrix that there is a correlation between variables: the most positively correlated variables are "Ca" and "RI", "Al" and "Ba" and this means that an increment of the value of "Ca" will imply the increment of the value of "RI".

There is a negative correlation between variables "Si" and "RI", "Al" and "RI", "Mg" and "Al" and this means that an increase of the value of "Si" would imply a decrement of the value of "RI" and so for the other variables.

For all this reason I assume that the Principal Component Analysis is well justified.

To evaluate the difference between variables, I apply the mean and the variance for each variable:

```
apply(glass[-10],2,mean)
```

```
##          RI          Na          Mg          Al          Si           K
##  1.51836542 13.40785047  2.68453271  1.44490654 72.65093458  0.49705607
##          Ca          Ba          Fe
##  8.95696262  0.17504673  0.05700935
```

```
apply(glass[-10],2,var)
```

```
##           RI           Na           Mg           Al           Si           K
## 9.222541e-06 6.668414e-01 2.080540e+00 2.492702e-01 5.999212e-01 4.253542e-01
##           Ca           Ba           Fe
## 2.025366e+00 2.472270e-01 9.494300e-03
```

As the values are significantly different and there are different scales of measure, I need to standardize the dataset to have mean equal zero and standard deviation equal one, to make the data comparable:

```
scaled_glass <- apply(glass[-10],2,scale)
head(scaled_glass)
```

```
##              RI         Na        Mg         Al          Si           K
## [1,]  0.8708258  0.2842867 1.2517037 -0.6908222 -1.12444556 -0.67013422
## [2,] -0.2487502  0.5904328 0.6346799 -0.1700615  0.10207972 -0.02615193
## [3,] -0.7196308  0.1495824 0.6000157  0.1904651  0.43776033 -0.16414813
## [4,] -0.2322859 -0.2422846 0.6970756 -0.3102663 -0.05284979  0.11184428
## [5,] -0.3113148 -0.1688095 0.6485456 -0.4104126  0.55395746  0.08117845
## [6,] -0.7920739 -0.7566101 0.6416128  0.3506992  0.41193874  0.21917466
##              Ca         Ba         Fe
## [1,] -0.1454254 -0.3520514 -0.5850791
## [2,] -0.7918771 -0.3520514 -0.5850791
## [3,] -0.8270103 -0.3520514 -0.5850791
## [4,] -0.5178378 -0.3520514 -0.5850791
## [5,] -0.6232375 -0.3520514 -0.5850791
## [6,] -0.6232375 -0.3520514  2.0832652
```

The variance and the loadings of the Principal Component are computed by applying the eigendecomposition to the covariance matrix of the scaled dataset:

```
glass.cov <- cov(scaled_glass)
glass.eigen <- eigen(glass.cov)
str(glass.eigen)
```

```
## List of 2
##  $ values : num [1:9] 2.511 2.05 1.405 1.158 0.914 ...
##  $ vectors: num [1:9, 1:9] 0.545 -0.258 0.111 -0.429 -0.229 ...
##  - attr(*, "class")= chr "eigen"
```

The eigenvalues indicate the variance for each of the 9 principal components and the eigenvectors are the loadings of each Principal Component. In the next step, I will extract the eigenvectors and I will create a matrix called phi, which has been flip-signed to obtain the original from the mirror image:

```
phi <- glass.eigen$vectors
phi <- -phi
row.names(phi) <- c("RI", "Na", "Mg", "Al", "Si","K","Ca","Ba","Fe")
colnames(phi) <- c("PC1", "PC2", 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9')
phi
```

```
##            PC1         PC2           PC3         PC4          PC5         PC6
## RI -0.5451766  0.28568318  0.0869108293 -0.14738099 -0.073542700 -0.11528772
## Na  0.2581256  0.27035007 -0.3849196197 -0.49124204  0.153683304  0.55811757
## Mg -0.1108810 -0.59355826  0.0084179590 -0.37878577  0.123509124 -0.30818598
## Al  0.4287086  0.29521154  0.3292371183  0.13750592  0.014108879  0.01885731
## Si  0.2288364 -0.15509891 -0.4587088382  0.65253771  0.008500117 -0.08609797
## K   0.2193440 -0.15397013  0.6625741197  0.03853544 -0.307039842  0.24363237
## Ca -0.4923061  0.34537980 -0.0009847321  0.27644322 -0.188187742  0.14866937
## Ba  0.2503751  0.48470218  0.0740547309 -0.13317545  0.251334261 -0.65721884
## Fe -0.1858415 -0.06203879  0.2844505524  0.23049202  0.873264047  0.24304431
```

```
##            PC7         PC8         PC9
## RI -0.08186724  0.75221590  0.02573194
## Na -0.14858006  0.12769315 -0.31193718
## Mg  0.20604537  0.07689061 -0.57727335
## Al  0.69923557  0.27444105 -0.19222686
## Si -0.21606658  0.37992298 -0.29807321
## K  -0.50412141  0.10981168 -0.26050863
## Ca  0.09913463 -0.39870468 -0.57932321
## Ba -0.35178255 -0.14493235 -0.19822820
## Fe -0.07372136  0.01627141 -0.01466944
```

Throw the output information I can predict that, in terms of weight, variables "RI", "Ca" and "Al" contribute to the first Principal Component, and "Mg", "Ba", and "Ca" contribute to the second Principal Component.

In the next code I will compute the Principal Component Scores of each value:

```
PC1 <- scaled_glass %*% phi[,1]
PC2 <- scaled_glass %*% phi[,2]
PC3 <- scaled_glass %*% phi[,3]
PC4 <- scaled_glass %*% phi[,4]
PC5 <- scaled_glass %*% phi[,5]
PC6 <- scaled_glass %*% phi[,6]
PC7 <- scaled_glass %*% phi[,7]
PC8 <- scaled_glass %*% phi[,8]
PC9 <- scaled_glass %*% phi[,9]

PC <- data.frame(PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, PC9)
head(PC)
```

```
##          PC1        PC2        PC3         PC4        PC5        PC6
## 1 -1.1484468 -0.5282491 -0.3712253 -1.72485681 -0.2513465 -0.3394143
## 2  0.5727942 -0.7580105 -0.5554059 -0.75845396 -0.2564694  0.1156889
## 3  0.9379605 -0.9276609 -0.5536094 -0.20577184 -0.2369503 -0.1263341
## 4  0.1417509 -0.9594279 -0.1168507 -0.41475157 -0.4751847 -0.2851361
## 5  0.3502710 -1.0886966 -0.4839440 -0.06894065 -0.4310792 -0.2973353
## 6  0.2895876 -1.3209105  0.8666466  0.92562711  1.8104158  0.1408914
##           PC7          PC8          PC9
## 1  0.39467467  0.196698883  0.016308248
## 2  0.02410142  0.283548730 -0.010764556
## 3  0.36665788  0.095716252  0.021589567
## 4  0.05237392  0.002952415  0.091820202
## 5 -0.15819858  0.190868054  0.002309129
## 6  0.26366789 -0.032862334 -0.001769212
```

Computing the optimal number of Principal Component

Once I computed the Principal Component Score, I try to establish the optimal number of Principal Components which best represent the dataset throw the proportion of variance explained, the Kaiser's rule, and the Scree plot.

To carry as much information as possible, about 80% of variance explained is retained a good value:

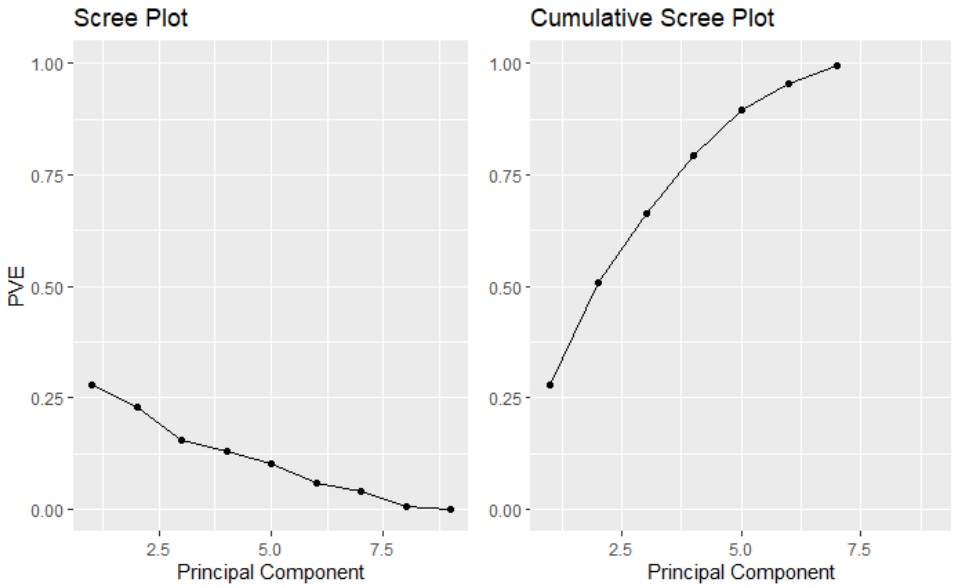
```
PVE <- glass.eigen$values/sum(glass.eigen$values)
PVE <-round(PVE, 3)
cumsum(PVE)

## [1] 0.279 0.507 0.663 0.792 0.894 0.953 0.994 1.001 1.001
```

To compute the proportion of variance explained by each Principal Component, I simply divided the variance explained by each Principal Component by the total variance explained by all Principal Components. According to the cumulative proportion of variance explained, the first four Principal Components together explain the 79.2% of our distribution.

```
pr.out <- prcomp(scaled_glass, scale=FALSE)
pr.var <- pr.out$sdev^2
which(pr.var>1)

## [1] 1 2 3 4
```

The function prcomp() is another way to perform the Principal Component Analysis which prints out a lot of useful information. One of these is the standard deviation of each Principal Component, which has been squared to obtain the variance explained. According to the Kaiser's rule, values greater than 1 should be considered, for this reason, I should consider the first four Principal Components.

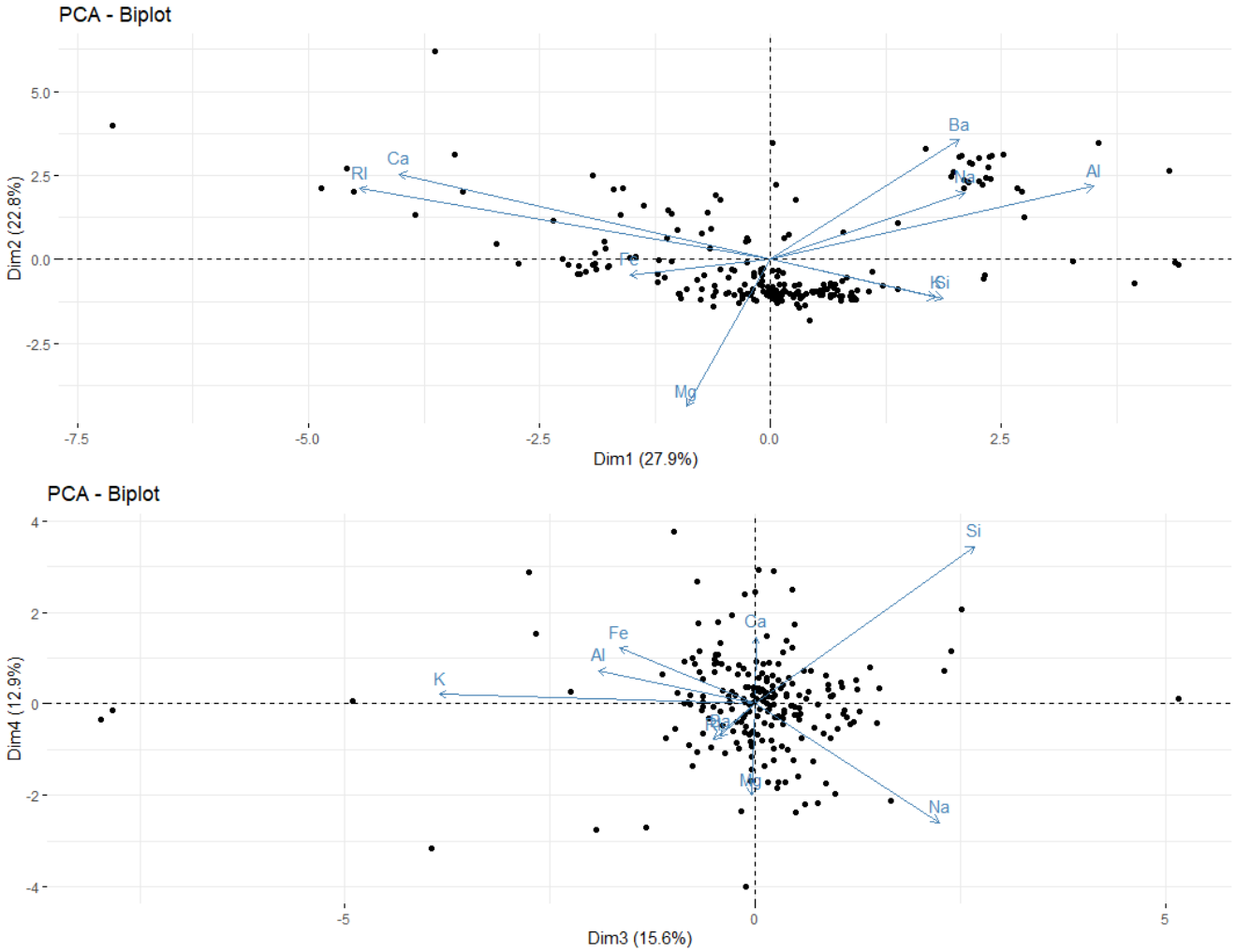
```
PVEplot <- ggplot2:::qplot(c(1:9), PVE) +
  geom_line() +
  xlab("Principal Component") +
  ylab("PVE") +
  ggtitle("Scree Plot") +
  ylim(0, 1)
cumPVE <- ggplot2:::qplot(c(1:9), cumsum(PVE)) +
  geom_line() +
  xlab("Principal Component") +
  ylab(NULL) +
  ggtitle("Cumulative Scree Plot") +
  ylim(0,1)
gridExtra:::grid.arrange(PVEplot, cumPVE, ncol = 2)
```



Looking at the scree plot it's difficult to capture the elbow, so I computed the cumulative scree plot, also, in this case, it would be reasonable to consider the first four Principal Components.

With the following code I can visualize the biplot which will allow me to analyze the Principal Components and the original variables represented by arrows:

```
PC1.2 <- factoextra:::fviz_pca_biplot(pr.out, axes=c(1,2), geom='point')
PC3.4 <- factoextra:::fviz_pca_biplot(pr.out, axes=c(3,4), geom='point')
grid.arrange(PC1.2,PC3.4, nrow=2)
```



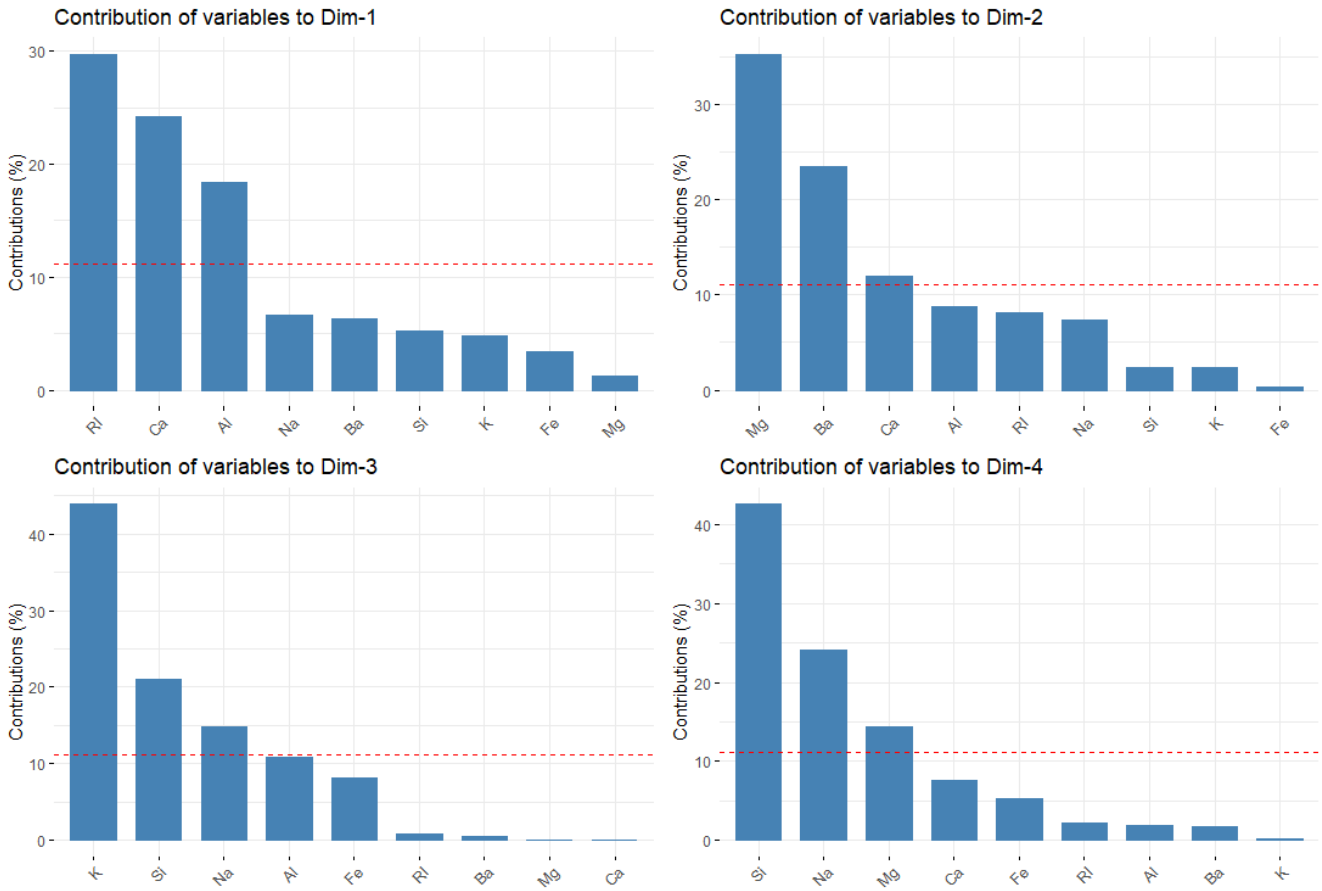
The first Principal Component is the direction along which there is the greatest variability in the data. The other principal components are linear combinations of the variables that are uncorrelated to the previous principal components and have the largest variance.

If the arrows follow the same direction of the Principal Component, observations that are far from the origin in direction of the PC axis will have values above the average, observations that are near to the origin will have values around the average, and observations which are far from the origin in the opposite direction of the PC axis will have values below the average.

In addition, arrows that are far from the origin are well represented on the PC space and the angle of the arrow shows the correlation between the original variable and the Principal Component. The more the arrows are near between them, the more are correlated.

Thanks to the following code, I plot the contribution of each variable for each chosen dimension:

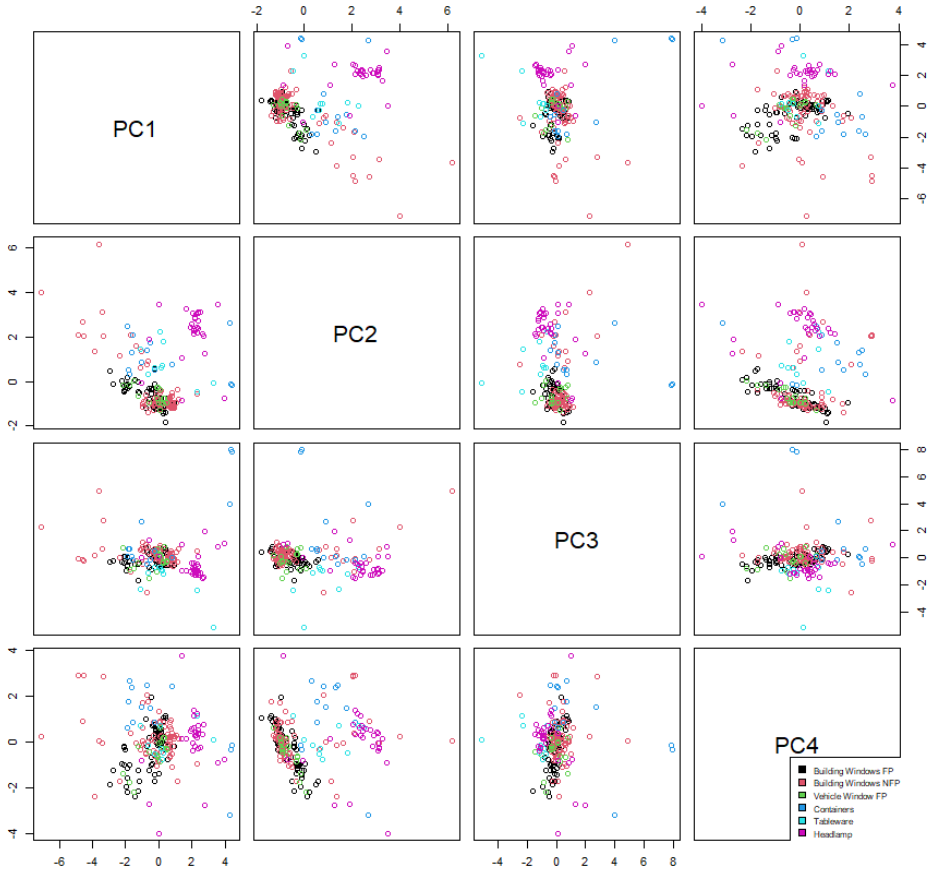
```
pc1 <- factoextra:::fviz_contrib(pr.out, 'var', axes=1)
pc2 <- factoextra:::fviz_contrib(pr.out, 'var', axes=2)
pc3 <- factoextra:::fviz_contrib(pr.out, 'var', axes=3)
pc4 <- factoextra:::fviz_contrib(pr.out, 'var', axes=4)
gridExtra:::grid.arrange(pc1,pc2,pc3,pc4)
```



Variables "RI", "Ca" are negatively correlated to PC1, variable "Al" is positively correlated to PC1 and together contribute to the first Principal Component.
Variables "Ba" and "Ca" are positively correlated to PC2 and variable "Mg" is negatively correlated to PC2 and together contribute to the second Principal Component.
Variables "Si" and "Na" are positively correlated to PC3 and variable "K" is negatively correlated to PC3 and together contribute to the third Principal Component.
Variables "Na" and "Mg" are negatively correlated to PC4 and variable "Si" is positively correlated to PC4 and together contribute to the fourth Principal Component.

In the following code I plot the first four Principal Components by highlighting the types of glass:

```
glass$Type <- factor(Type, labels = c("Building Windows FP", "Building Windows NFP", "Vehi
cle Window FP", "Containers", "Tableware", "Headlamp"))
pairs(PC[1:4], col=glass$Type)
legend("bottomright", inset=0.03, cex=0.5, fill = unique(glass$Type), legend = c( levels(g
lass$Type)))
```



By looking at the scatter plot on the Principal Component space, I can see at least two groups: the first group coincides with headlamp glasses and the second group coincide with all the other type of glass.

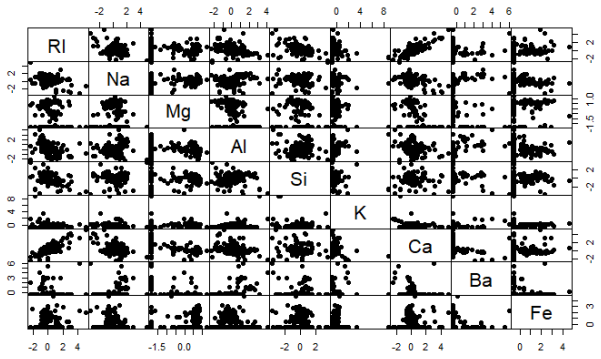
Cluster Analysis

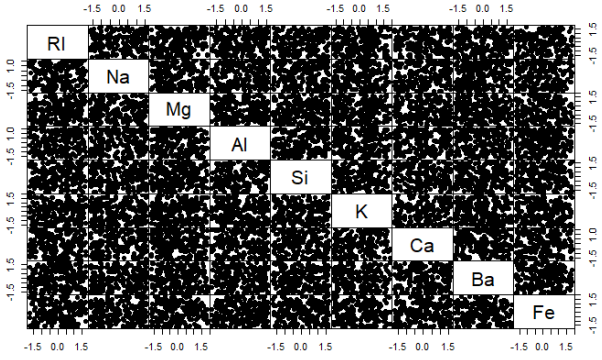
This statistical method will allow me to identify groups of units within the dataset that are similar and have the same behaviour. While the observations within the same group are similar, observations between each group are as different as possible from each other. A previous standardization has been applied to the dataset to make the values comparable between each other.

Accessing cluster tendency

Before clustering I need to verify if the cluster analysis is feasible or not, that is, whether applying to the cluster is suitable or not. To verify if clustering is a reasonable method, I compared the dataset with a randomly generated one as well as verified if they contain meaningful clusters.

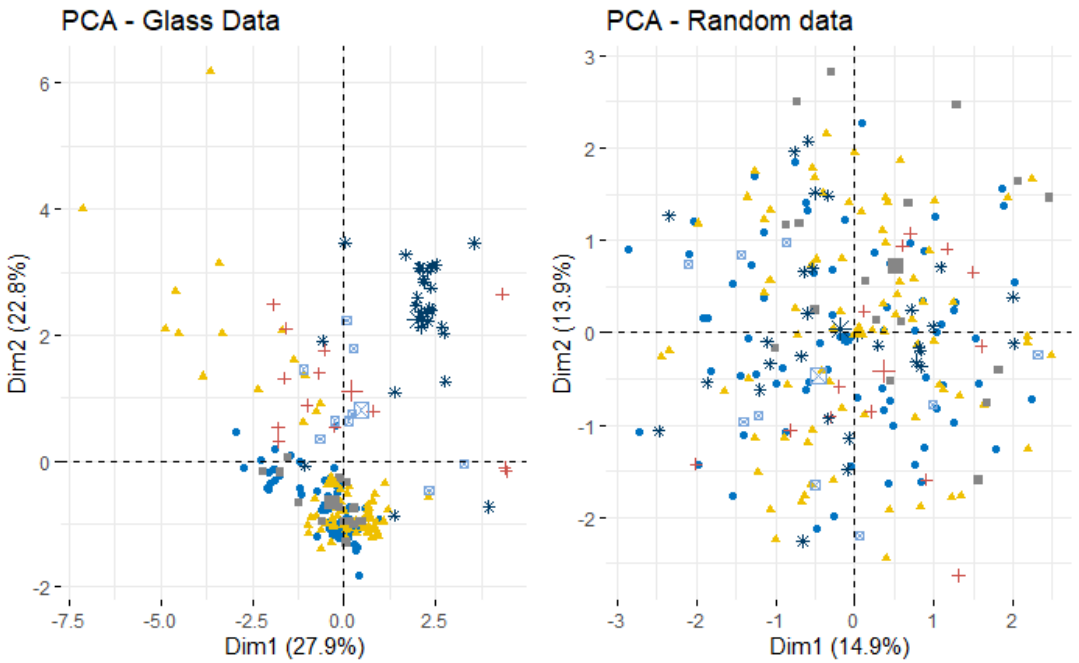
```
random_glass<- apply(glass[-10], 2, function(x){runif(length(x), min(x), max(x))})
random_glass <- as.data.frame(random_glass)
random_glass <- scale(random_glass)
pairs(scaled_glass, gap=0, pch=16)
pairs(random_glass, gap=0, pch=16)
```





The first plot seems to have a clustering tendency instead of the second one. I summarize the same example on the first two PC spaces by highlighting the categorical variable:

```
random <- factoextra:::fviz_pca_ind(prcomp(scaled_glass), title = "PCA - Glass Data",
             habillage = glass$Type,
             palette = "jco",
             geom = "point")+guides(col=F, size=F, shape=F)

original <- fviz_pca_ind(prcomp(random_glass), title = "PCA - Random data",
                         habillage=glass$Type,
                         palette='jco',
                         geom = "point")+guides(col=F,size=F, shape=F)

gridExtra:::grid.arrange(random,original, ncol=2)
```



There is a high difference between the two datasets: throw the analysis of the first plot I can assume that there are at least two clusters, instead, also in this case there isn't any cluster tendency in the random-generated data.

In the code below I examine the clustering tendency of the datasets by using two statistical methods, the first method is Hopkin's statistic:

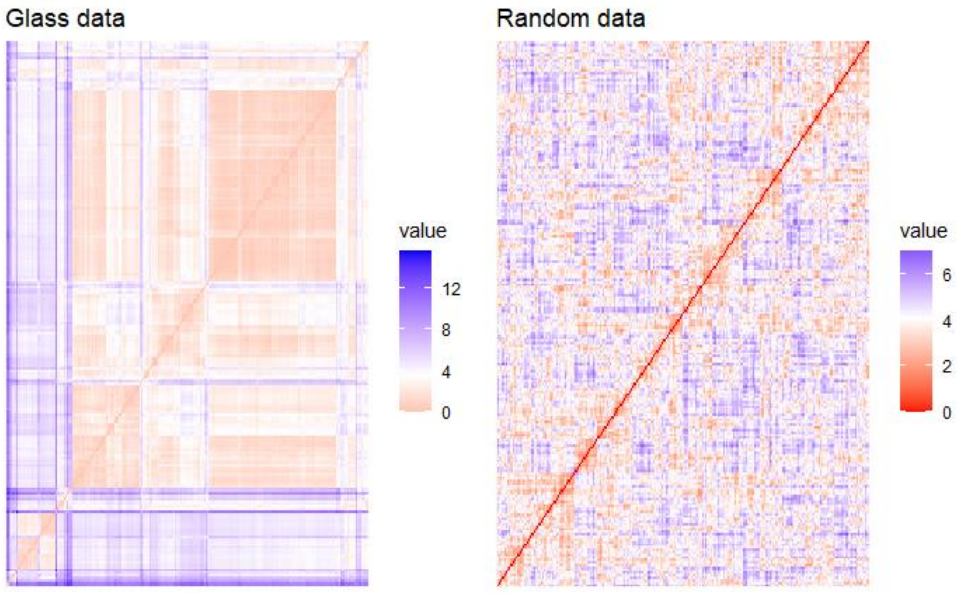
```
clustertend:::hopkins(scaled_glass, n = nrow(scaled_glass)-1)

## $H
## [1] 0.1252253

clustertend:::hopkins(random_glass, n = nrow(random_glass)-1)

## $H
## [1] 0.4940128
```

Hopkin's statistic is an index with a range of [0,1], a lower value corresponds to the presence of cluster tendency. The first value, which refers to the original scaled dataset, is close to 0 and this means that there is cluster tendency, so the application of clustering methods is justified.

The H index which refers to the scaled random-generated dataset is close to 0.5 and this means that the data are uniformly distributed.

The second method is the VAT algorithm which consists of a visual method of accessing the cluster tendency:

```
vat_glass <- factoextra:::fviz_dist(dist(scaled_glass), show_labels = FALSE)+
  labs(title = "Glass data")
vat_random <- factoextra:::fviz_dist(dist(random_glass), show_labels = FALSE)+
  labs(title = "Random data")
gridExtra:::grid.arrange(vat_glass,vat_random, ncol=2)
```



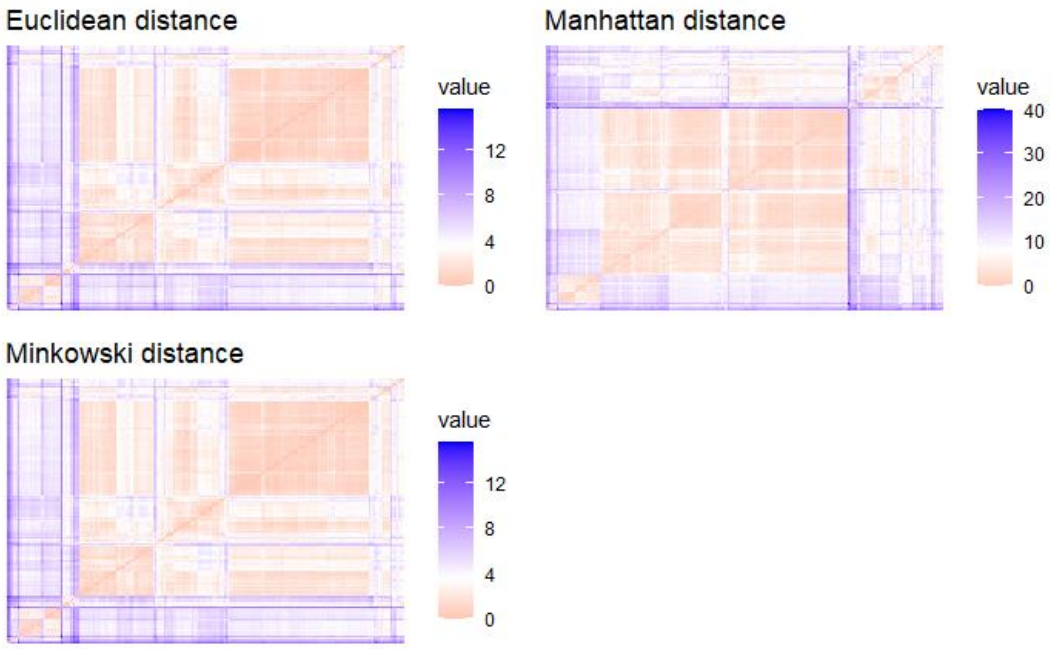
The red zone highlights a high level of similarity between values. There are two relevant red zones in the first plot instead of the second plot.

Distance matrix

The classification of the observations requires three different methods which allow computing the distance between observations: Euclidean distance, Manhattan distance, and Minkowski distance.

From this computation I obtained a distance matrix for each method, which will be reported in three different plots:

```
euclid <- stats:::dist(scaled_glass, method = "euclidean")
manhattan <- stats:::dist(scaled_glass, method = "manhattan")
minkowski <- stats:::dist(scaled_glass, method = "minkowski")

gridExtra:::grid.arrange(factoextra:::fviz_dist(euclid, show_labels=FALSE)+labs(title='Euc
lidean distance'), factoextra:::fviz_dist(manhattan, show_labels=FALSE)+labs(title='Manhat
tan distance'), factoextra:::fviz_dist(minkowski, show_labels=FALSE)+labs(title='Minkowski
distance'),ncol=2,nrow=2)
```



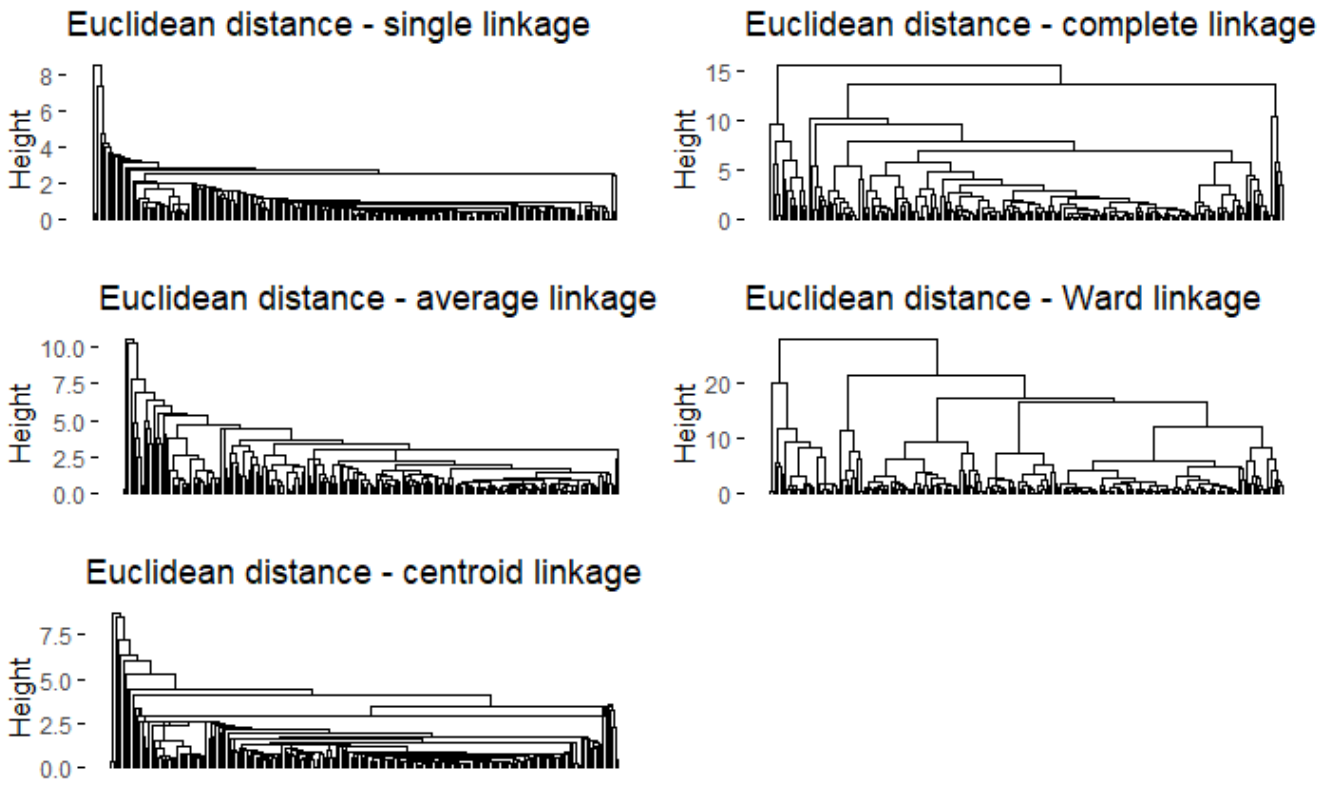
The red zone in the plot measures a high similarity (low dissimilarity) while the blue zone on the plot measures a low similarity (high dissimilarity). By using different methods to compute dissimilarity between groups of the dataset, different zones have been highlighted.

Agglomerative clustering: hierarchical

Hierarchical clustering is a method that builds a hierarchy of clusters. As a bottom-up approach, each observation starts on its cluster, and pairs of clusters are merged by moving up on the hierarchy and going on until it reaches a unique cluster. This type of clustering method is visualized thanks to the dendrogram. In addition, the linkage criterion specifies the dissimilarity of sets as a function of the pairwise distances of observations in the sets.

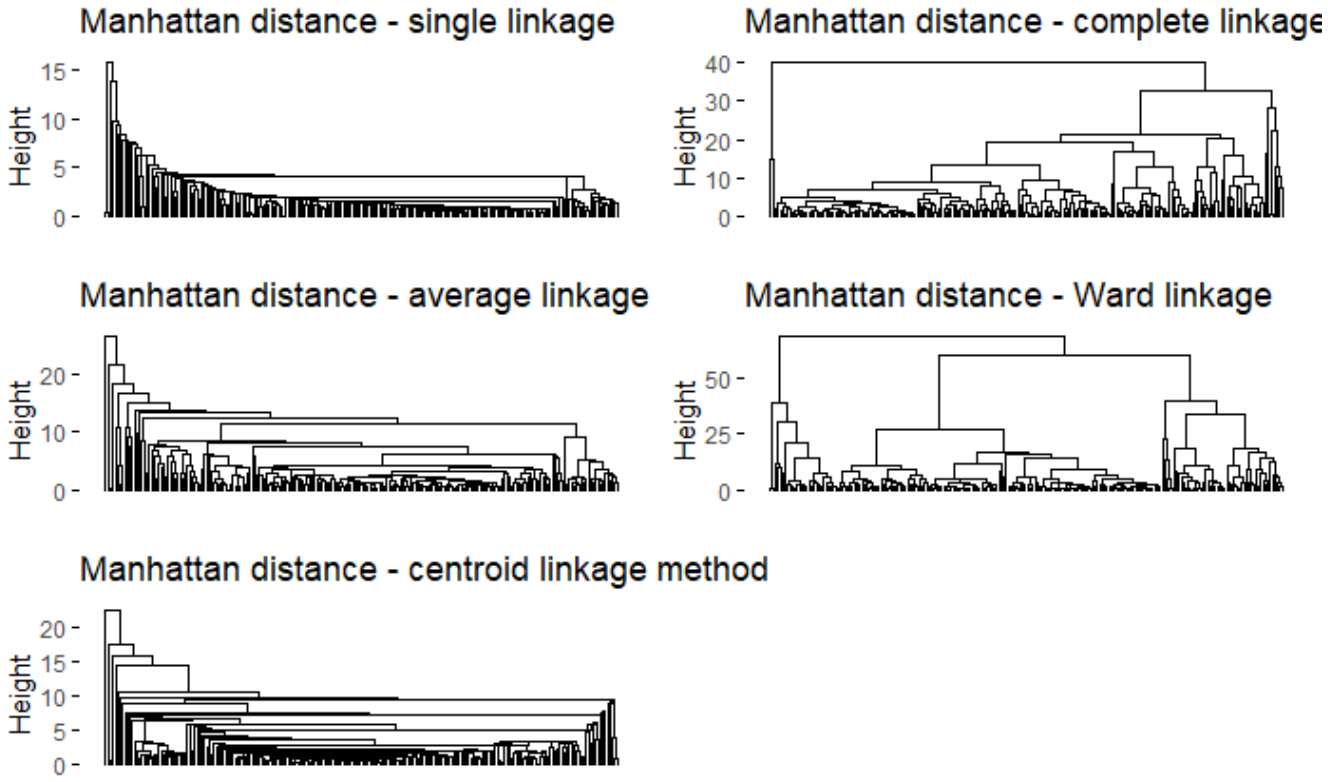
In this report single linkage method, complete linkage method, average linkage method, Ward's linkage method, and centroid linkage method are visualized by using each type of distance. For the Euclidean distance I obtained the following dendrograms:

```
single_euclid <- stats:::hclust(d = euclid, method = "single")
complete_euclid <- stats:::hclust(d = euclid, method = "complete")
average_euclid <- stats:::hclust(d = euclid, method = "average")
ward_euclid <- stats:::hclust(d = euclid, method = "ward.D2")
centroid_euclid <- stats:::hclust(d = euclid, method = "centroid")

gridExtra:::grid.arrange(factoextra:::fviz_dend(single_euclid, cex = 0.5, main='Euclidean
distance - single linkage', show_labels=FALSE), factoextra:::fviz_dend(complete_euclid, ce
x = 0.5, main='Euclidean distance - complete linkage', show_labels=FALSE), factoextra:::fv
iz_dend(average_euclid, cex = 0.5, main='Euclidean distance - average linkage', show_label
s=FALSE), factoextra:::fviz_dend(ward_euclid, cex = 0.5, main='Euclidean distance - Ward l
inkage', show_labels=FALSE), factoextra:::fviz_dend(centroid_euclid, cex = 0.5, main='Eucl
idean distance - centroid linkage', show_labels=FALSE), ncol=2,nrow=3)
```



For the Manhattan distance I obtained the following dendrograms:

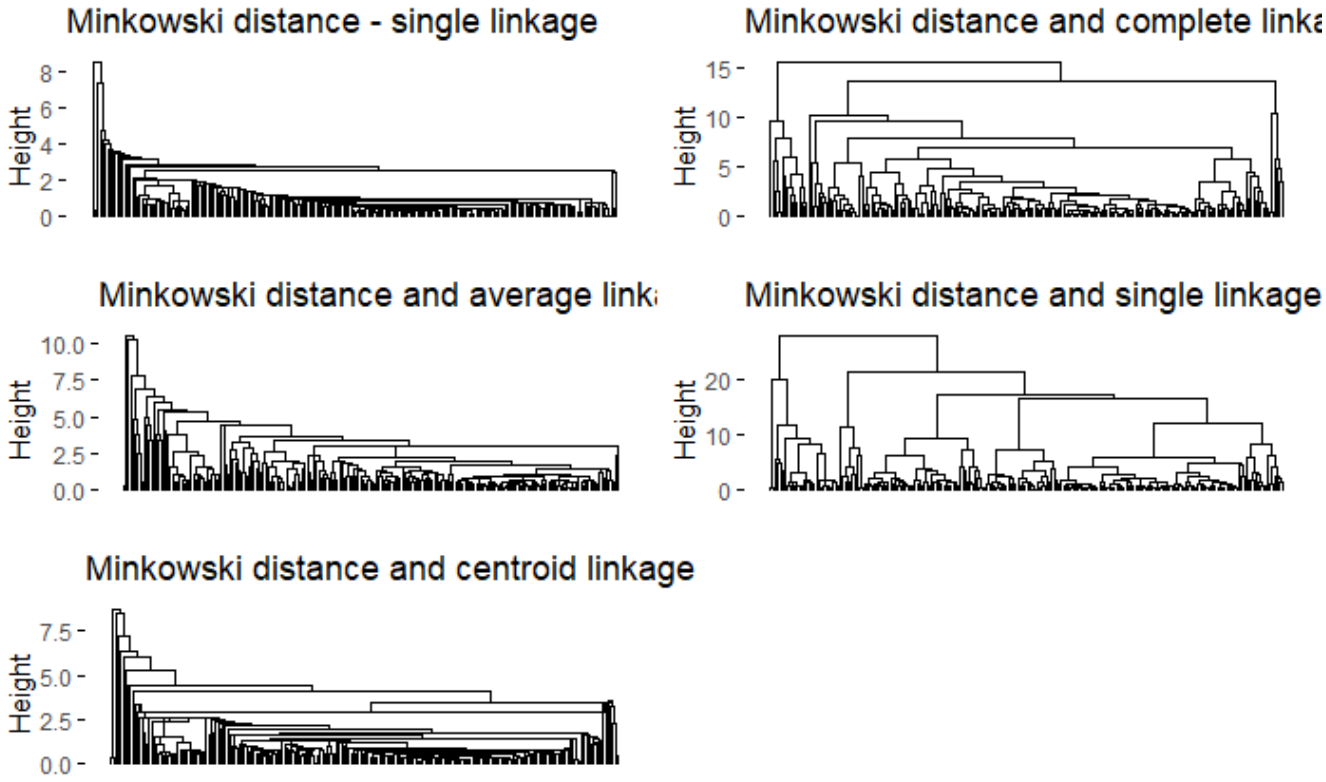
```
single_manhattan <- stats:::hclust(d = manhattan, method = "single")
complete_manhattan <- stats:::hclust(d = manhattan, method = "complete")
average_manhattan <- stats:::hclust(d = manhattan, method = "average")
ward_manhattan <- stats:::hclust(d = manhattan, method = "ward.D2")
centroid_manhattan <- stats:::hclust(d = manhattan, method = "centroid")

gridExtra:::grid.arrange(factoextra:::fviz_dend(single_manhattan, cex = 0.5, main='Manhatt
an distance - single linkage',show_labels=FALSE), factoextra:::fviz_dend(complete_manhatta
n, cex = 0.5, main='Manhattan distance - complete linkage',show_labels=FALSE), factoextra:
::fviz_dend(average_manhattan, cex = 0.5, main='Manhattan distance - average linkage',show
_labels=FALSE), factoextra:::fviz_dend(ward_manhattan, cex = 0.5, main='Manhattan distance
- Ward linkage',show_labels=FALSE), factoextra:::fviz_dend(centroid_manhattan, cex = 0.5,
main='Manhattan distance - centroid linkage method',show_labels=FALSE))
```



For the Minkowski distance I obtained the following dendrograms:

```
single_minkowski <- stats:::hclust(d = minkowski, method = "single")
complete_minkowski <- stats:::hclust(d = minkowski, method = "complete")
average_minkowski <- stats:::hclust(d = minkowski, method = "average")
ward_minkowski <- stats:::hclust(d = minkowski, method = "ward.D2")
centroid_minkowski <- stats:::hclust(d = minkowski, method = "centroid")

gridExtra:::grid.arrange(factoextra:::fviz_dend(single_minkowski, cex = 0.5, main='Minkows
ki distance - single linkage', show_labels=FALSE),
             factoextra:::fviz_dend(complete_minkowski, cex = 0.5, main='Minkowski distanc
e and complete linkage', show_labels=FALSE),
             factoextra:::fviz_dend(average_minkowski, cex = 0.5, main='Minkowski distance
and average linkage', show_labels=FALSE),
             factoextra:::fviz_dend(ward_minkowski, cex = 0.5, main='Minkowski distance an
```

```
d single linkage', show_labels=FALSE),
             factoextra:::fviz_dend(centroid_minkowski, cex = 0.5, main='Minkowski distanc
e and centroid linkage', show_labels=FALSE))
```



The correlation between the cophenetic distances and the original distances measure how well the cluster tree represents the data. In the following code I computed a dataframe with the correlation values obtained by each distance and each linkage method:

```
Euclidean <- c(cor(euclid, stats:::cophenetic(single_euclid)), cor(euclid, stats:::cophene
tic(complete_euclid)), cor(euclid, stats:::cophenetic(average_euclid)), cor(euclid, stats:
::cophenetic(ward_euclid)), cor(euclid, stats:::cophenetic(centroid_euclid)))
Manhattan <- c(cor(manhattan, stats:::cophenetic(single_manhattan)), cor(manhattan, stats:
::cophenetic(complete_manhattan)), cor(manhattan, stats:::cophenetic(average_manhattan)),
cor(manhattan, stats:::cophenetic(ward_manhattan)), cor(manhattan, stats:::cophenetic(cent
roid_manhattan)))
Minkowski <- c(cor(minkowski, stats:::cophenetic(single_minkowski)), cor(minkowski, stats:
::cophenetic(complete_minkowski)), cor(minkowski, stats:::cophenetic(average_minkowski)),
cor(minkowski, stats:::cophenetic(ward_minkowski)), cor(minkowski, stats:::cophenetic(cent
roid_minkowski)))


cophenetic.distance_matrix <- data.frame(Euclidean, Manhattan, Minkowski)
rownames(cophenetic.distance_matrix) <- c('Single', 'Complete', 'Average', 'Ward', 'Centro
id')
cophenetic.distance_matrix

##          Euclidean Manhattan Minkowski
## Single   0.8764943 0.8681716 0.8764943
## Complete 0.7221467 0.8100737 0.7221467
```

```
## Average  0.9297960 0.9183818 0.9297960
## Ward     0.6698684 0.6906438 0.6698684
## Centroid 0.9146228 0.8892455 0.9146228
```

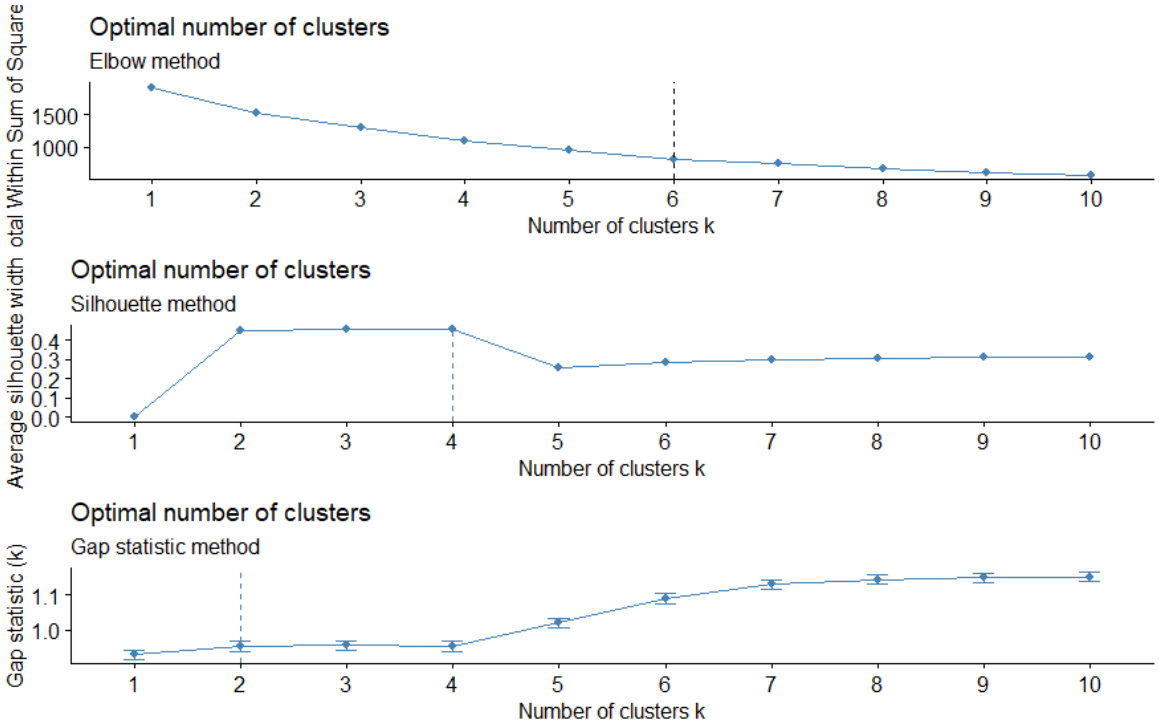
The closer the value of the correlation coefficient is to 1, the more accurately the clustering solution reflects our data. According to the dataframe above, the best linkage method is the average linkage method which has the same value for Euclidean and Minkowski distance.

For this clustering approach, I choose the average linkage method on the Euclidean distance.

Computing the optimal number of clusters

Before cutting the dendrogram I computed the optimal number of clusters by analyzing direct methods and statistical testing methods. Direct methods involve the elbow method and the average silhouette width method, statistical testing methods involve the gap statistic.

```
wss <- factoextra:::fviz_nbclust(scaled_glass, hcut, method = "wss") + geom_vline(xinterce
pt = 6, linetype = 2) + labs(subtitle = "Elbow method")
silhouette <- factoextra:::fviz_nbclust(scaled_glass, hcut, method = "silhouette") + labs(
subtitle = "Silhouette method")
gapstat <- factoextra:::fviz_nbclust(scaled_glass, hcut, method = "gap_stat", nboot = 200)
+labs(subtitle = "Gap statistic method")
gridExtra:::grid.arrange(wss, silhouette, gapstat, nrow=3, ncol=1)
```



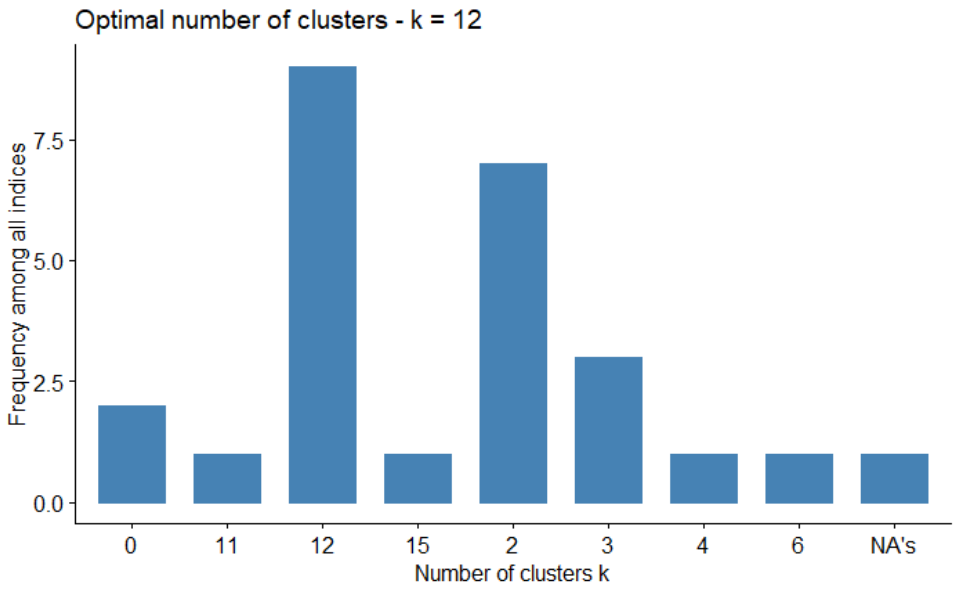
According to the elbow method, the optimal number of clusters is 6, according to the average Silhouette width method the number of clusters is 4 and according to the gap statistic method, the optimal number of clusters is 2.

In addition, I computed the optimal number of clusters throw other 30 indices:

```
factoextra:::fviz_nbclust(NbClust:::NbClust(data = scaled_glass, diss = NULL, distance = "
euclidean", min.nc = 2, max.nc = 15, method = 'average'))
```

```
## [1] "Frey index : No clustering structure in this data set"
```

```
## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 7 proposed  2 as the best number of clusters
## * 3 proposed  3 as the best number of clusters
## * 1 proposed  4 as the best number of clusters
## * 1 proposed  6 as the best number of clusters
## * 1 proposed  11 as the best number of clusters
## * 9 proposed  12 as the best number of clusters
## * 1 proposed  15 as the best number of clusters
## * 1 proposed  NA's as the best number of clusters
##
## Conclusion
## ========================
## * According to the majority rule, the best number of clusters is  12.
```



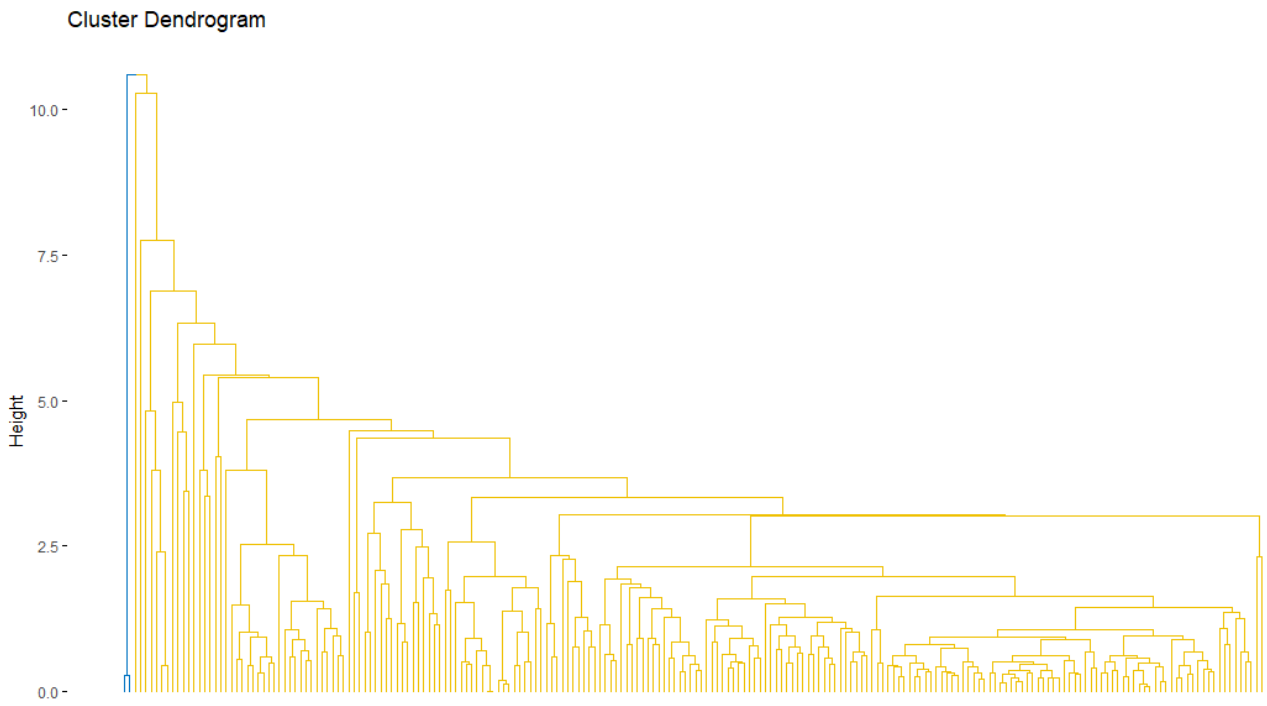

```
grp <- stats:::cutree(average_euclid, k = 2)
table(grp)
```

```
## grp
##   1   2
## 212   2
```

I choose to cut my dendrogram into 2 groups. The clusters are not balanced the first group contains 212 observations; the second group contains 2 observations.

In the following plot I displayed the dendrogram obtained with the average linkage method and Euclidean distance cut into 2 groups:

```
factoextra:::fviz_dend(average_euclid, k = 2, cex = 0.5, k_colors = 'jco', show_labels=FAL
SE)
```

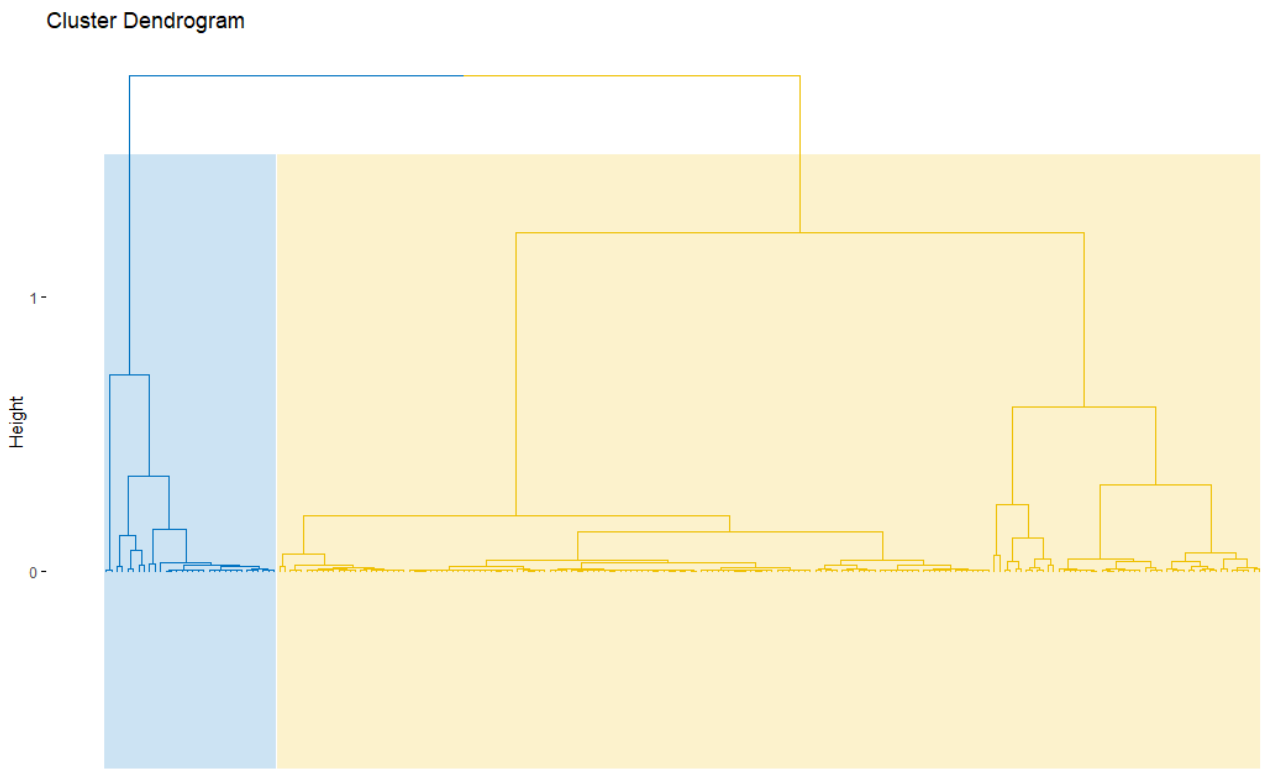



```
tail(average_euclid$merge)

##        [,1] [,2]
## [208,] -202  207
## [209,]  205  208
## [210,]  204  209
## [211,] -185  210
## [212,] -107  211
## [213,]   19  212

tail(average_euclid$height)

## [1]  5.970826  6.331326  6.895060  7.757997 10.285270 10.599864
```

By looking at the iteration steps, the dendrogram will merge two groups, highlighted by different colours in the dendrogram, at 10.285.

Thanks to the dimensionality reduction gained throw the PCA I computed the hierarchical clustering on the PC space by using Ward's criterion on the selected Principal Component with the function HCPC() in the "FactoMineR" package:

```
res.pca <- FactoMineR:::PCA(glass[-10], ncp=4, graph=FALSE)
res.hcpc <- FactoMineR:::HCPC(res.pca, nb.clust=-1, min=2, graph=FALSE)
factoextra:::fviz_dend(res.hcpc, cex=0.7, palette='jco', rect=TRUE, rect_fill=TRUE, rect_b
order='jco', show_labels=FALSE)
```



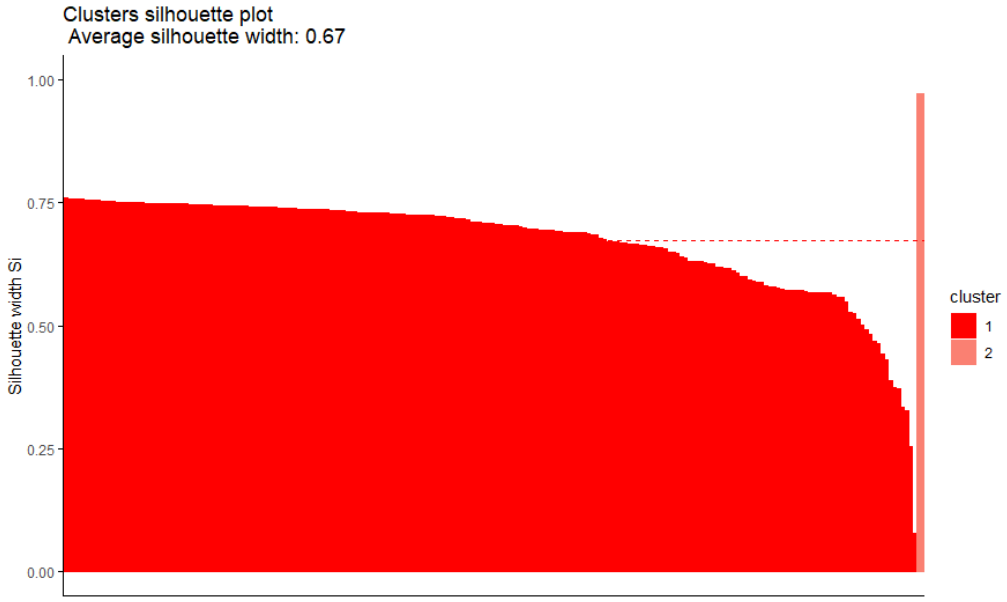
In this case, thanks to the reduction from 9 to 4 dimensions, the dendrogram will display two different groups at about 1.5 distance.

Cluster validation

To evaluate the goodness of clustering algorithm results I used the internal validation statistics which uses the internal information of the clustering process to evaluate the goodness of a clustering structure and external validation statistics which consists in comparing the results of cluster analysis to an externally known result.

Regarding the internal validation statistics, I first computed the Silhouette width:

```
hclust_validation <- factoextra:::eclust(scaled_glass, "hclust", k = 2, nstart = 25, graph
= FALSE, hc_metric='euclidean', hc_method='average')

factoextra:::fviz_silhouette(hclust_validation, palette = c('red','salmon'),ggtheme = them
e_classic())

##    cluster size ave.sil.width
## 1        1  212          0.67
## 2        2    2          0.97
```



This index can reach values in the range [-1,1], a high value indicates that the object is well matched to its cluster and poorly matched to neighbouring clusters, so the configuration is appropriate. A low or negative value means that the clustering configuration may have too many or too few clusters.

In this case, according to the average Silhouette width, the cluster configuration is appropriate, each cluster fits the observations and there aren't observations with negative Silhouette width.

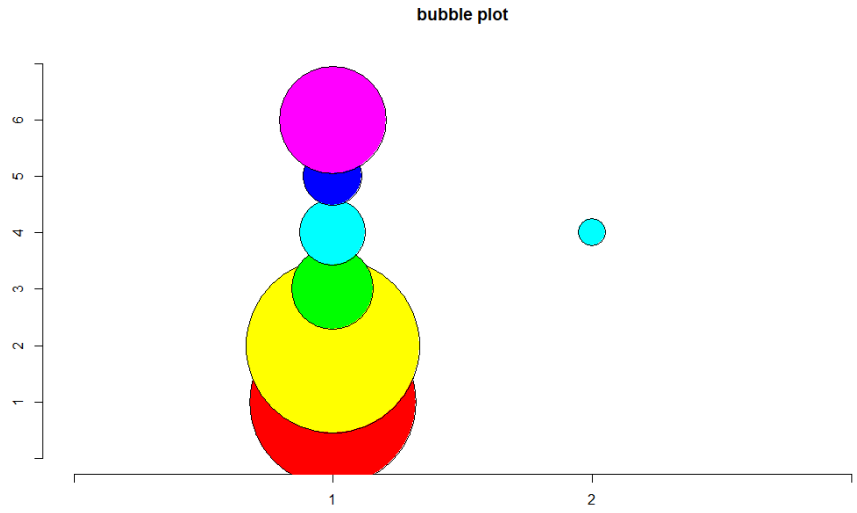
I also computed the Dunn index as internal cluster validation:

```
hclust_stats <- fpc:::cluster.stats(stats:::dist(scaled_glass), hclust_validation$cluster)
hclust_stats$dunn

## [1] 0.5438607
```

As this index must be maximized, we can say that according to the Dunn index the clusters aren't compact and well separated.

Regarding the external validation, I first plot the confusion matrix between the external information (the variable 'Type'), and the clusters:

```
labstatR:::bubbleplot(table(glass$Type, hclust_validation$cluster))
```

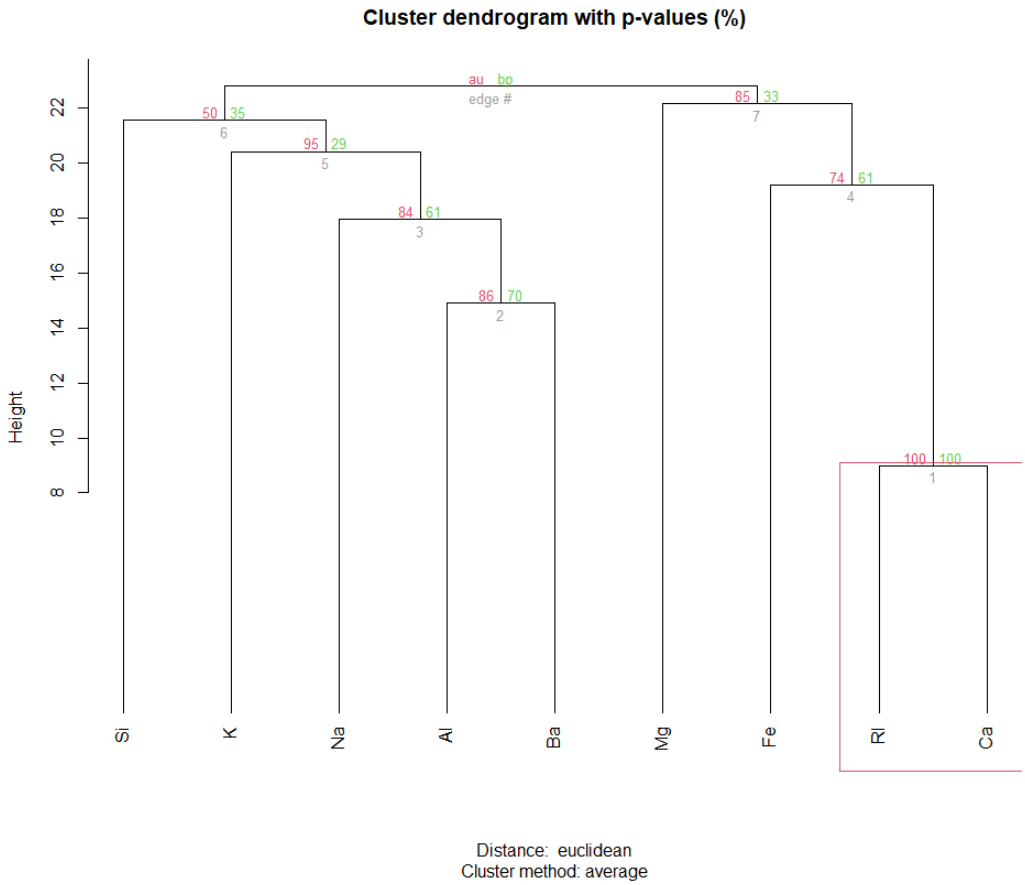


According to the plot, I can assume that the external information doesn't influence the clustering method because cluster 1 assumes a high value of almost all types of glass. To quantify the agreement between clusters and external information I used the corrected Rand and Meila's VI indexes:

```
clust_stats <- fpc:::cluster.stats(d = dist(scaled_glass),
                                   Type, hclust_validation$cluster)

clust_stats$corrected.rand

## [1] 0.01057863

clust_stats$vi

## [1] 1.507846
```

The corrected Rand index goes from [-1,1] and should be maximized instead the Meila's VI index should be minimized. In this case, an agreement between the external information and clusters is 0.10, and disagreement between the external information and clusters is 1.50 so the clusters are not influenced by external information. An extra validation has been computing throw the p-value for the Hierarchical clustering:

```
pv <- pvclust:::parPvclust(data=scaled_glass, method.hclust = "average", method.di
st = "euclidean", nboot = 10)

plot(pv, hang=-1)
pvclust:::pvrect(pv)
```



Clusters with AU values greater or equal to 95% are strongly supported by the data. The clusters have been indicated with a red rectangle and are “RI” and “Ca” which merge at first about 9.

Partitioning clustering

In this sub-chapter, I perform the partitioning clustering method which involves the k-means and k-medoids method.

In this type of clustering, the number of clusters is specified a priori and the data are split in such a way that the cohesion is maximized, and the within-cluster dissimilarity is maximized.

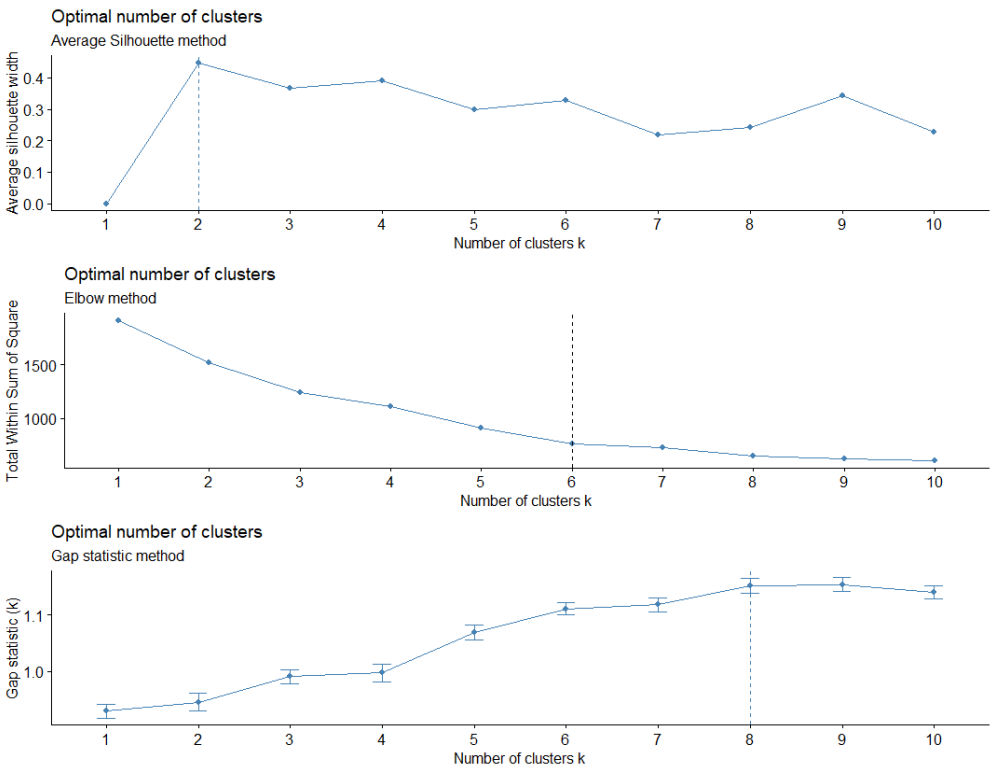
K-means

In the K-means clustering method, each cluster is represented by the mean of the observations which it contains. The algorithm starts with k random centroids and the observations are assigned to the nearest centroid using Euclidean distance to form clusters. At this point, the mean of each cluster is recomputed, and observations are reassigned according to the updated mean. The algorithm will continue until convergence is achieved.

Computing the optimal number of clusters

As the algorithm needs to know the number of clusters a priori, I computed the optimal number of k by using the elbow, gap-statistic, and average silhouette width method:

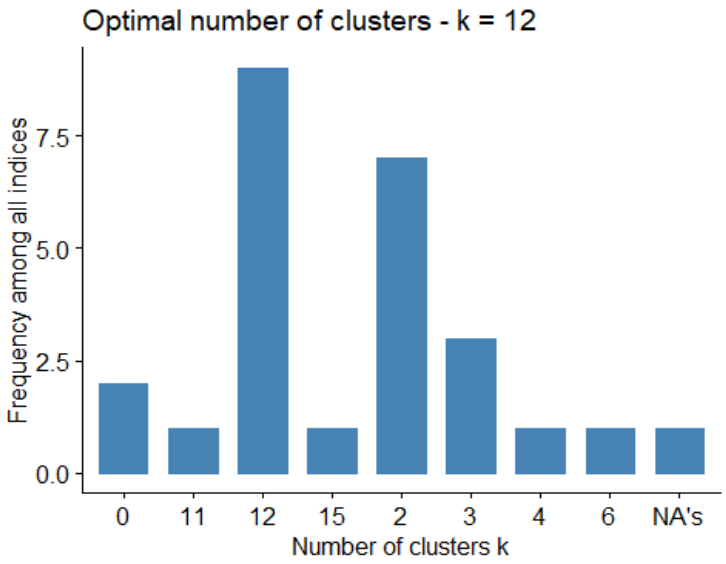
```
km1 <- factoextra:::fviz_nbclust(scaled_glass, kmeans, method = "silhouette") + labs(subti
tle='Average Silhouette method')

km2 <-  factoextra:::fviz_nbclust(scaled_glass, kmeans, method = "wss" ) +geom_vline(xinte
rcept = 6, linetype = 2)+labs(subtitle='Elbow method')

km3 <- factoextra:::fviz_nbclust(scaled_glass, kmeans, method = "gap_stat") +labs(subtitle
='Gap statistic method')
gridExtra:::grid.arrange(km1,km2,km3,nrow=3)
```



I also computed the optimal number of clusters by analyzing 30 additional indices:

```
factoextra:::fviz_nbclust(NbClust:::NbClust(data = scaled_glass, diss = NULL, distance = "
euclidean", min.nc = 2, max.nc = 15, method = 'average'))
```

```
## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 7 proposed  2 as the best number of clusters
## * 3 proposed  3 as the best number of clusters
## * 1 proposed  4 as the best number of clusters
## * 1 proposed  6 as the best number of clusters
## * 1 proposed  11 as the best number of clusters
## * 9 proposed  12 as the best number of clusters
## * 1 proposed  15 as the best number of clusters
## * 1 proposed  NA's as the best number of clusters
##
## Conclusion
## ==========================
## * According to the majority rule, the best number of clusters is  12.
```



There are different results for each method. In this report, k-means partitioning clustering will be computed by using k=2:

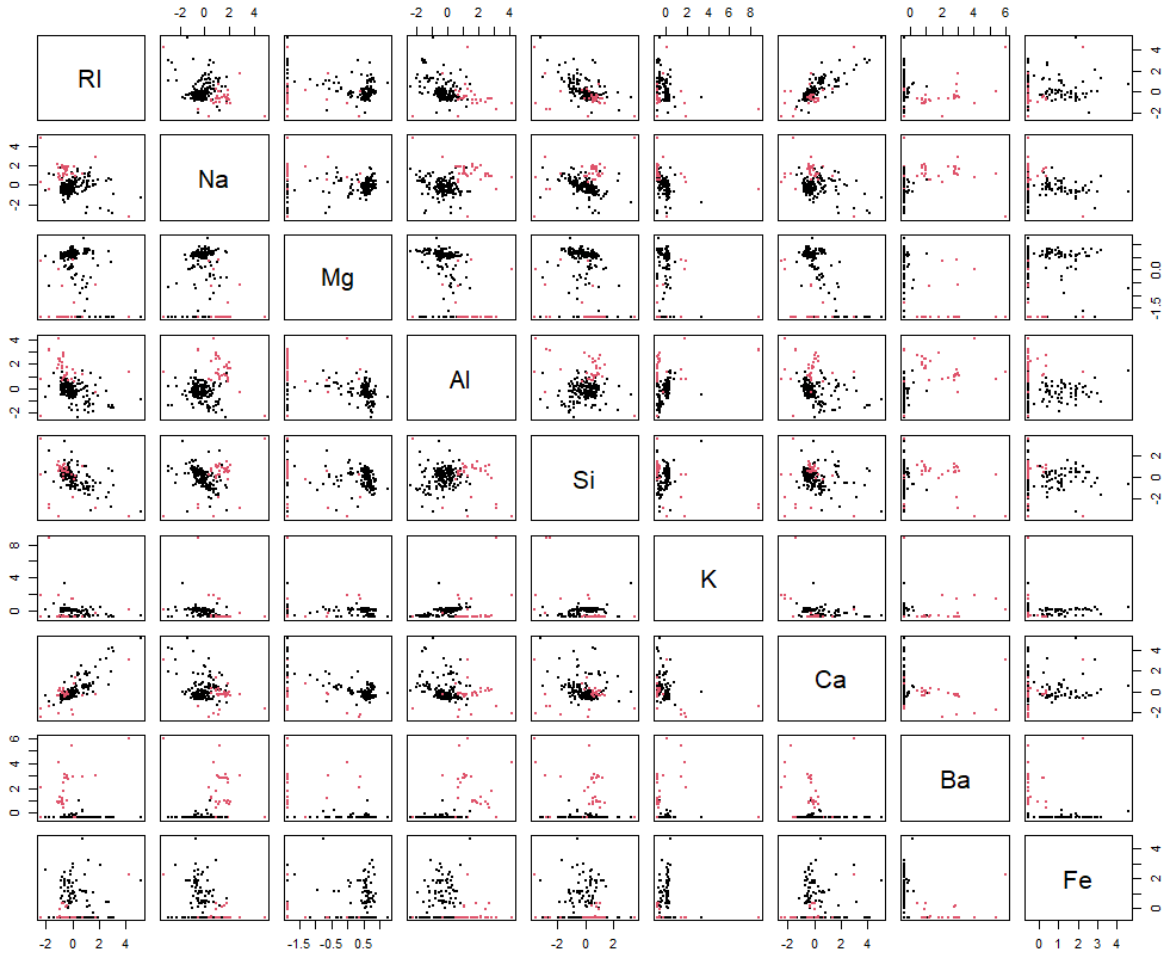
```
km.res <- kmeans(scaled_glass, 2, nstart = 25)
km.res

## K-means clustering with 2 clusters of sizes 181, 33
##
## Cluster means:
##            RI         Na         Mg         Al          Si           K
## 1  0.09076112 -0.2259117  0.2875011 -0.2791711 -0.01597200 -0.03089581
## 2 -0.49781101  1.2390915 -1.5768998  1.5312114  0.08760399  0.16945885
##            Ca         Ba          Fe
## 1  0.06331658 -0.3303839  0.06868228
## 2 -0.34728183  1.8121057 -0.37671187
##
## Clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 2 2 1 2
## [186] 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 988.3260 530.8212
##  (between_SS / total_SS =  20.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

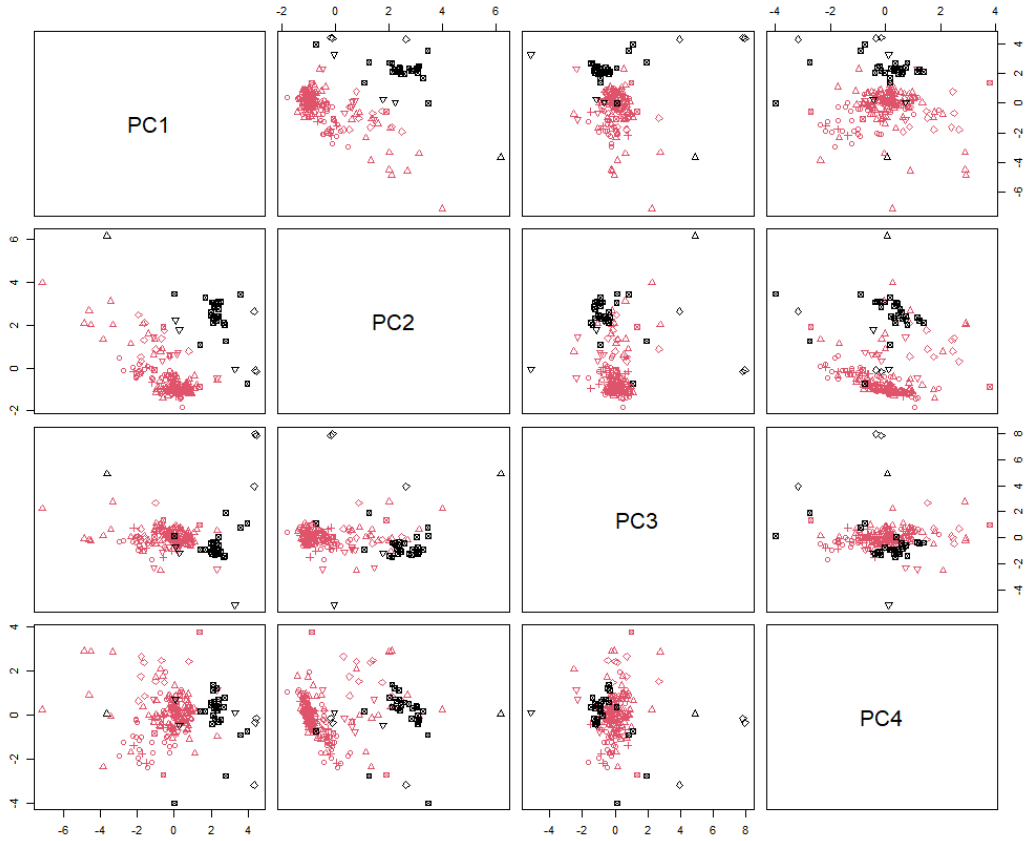
The information above shows the size of each cluster, the mean of each cluster according to each variable, the vector of the cluster to which each point belongs, and the within-cluster sum of the square. Within a cluster sum of squares should be as small as possible and between clusters sum

of squares should be as high as possible, in this case, the distance between groups is not high, so I can imagine that the clusters are not well separated.

To visualize clusters in the original and pc space:

```
cl.km <- km.res$cluster
pairs(scaled_glass, pch=16, cex=0.5, col=cl.km)
pairs(PC[1:4], col=cl.km, pch=glass$Type)
```



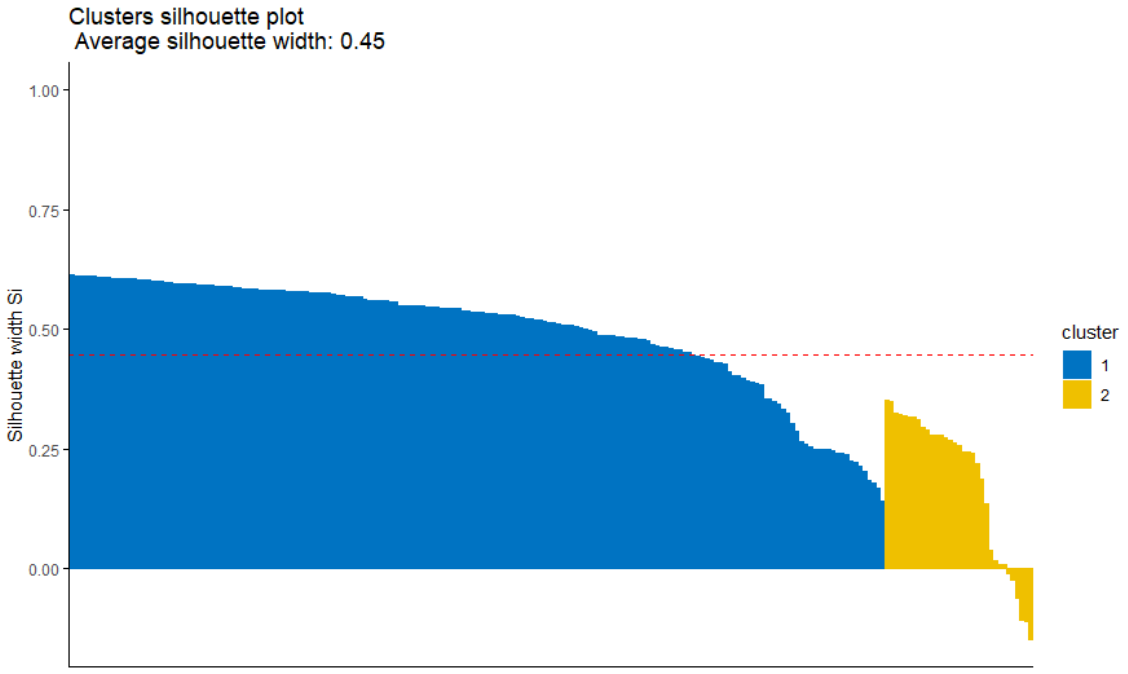


The red cluster contains observations with high values of aluminium and barium and the black cluster contains observations with a high value of magnesium. As we can see, observations with higher values of aluminium and barium coincide with headlamps, observations with higher values of magnesium coincide with building windows and vehicle windows both float-processed and non-float processed.

K-means validation

The internal validation for k-means partitioning cluster method has produced the following results:

```
km.res <- factoextra:::eclust(scaled_glass, "kmeans", k = 2, nstart = 25, graph = FALSE)
factoextra:::fviz_silhouette(km.res, palette = "jco",
                ggtheme = theme_classic())

##   cluster size ave.sil.width
## 1       1   181          0.50
## 2       2    33          0.18
```



According to the average Silhouette width which has a value of 0.45, the cluster configuration is not so appropriate, and some observations in the second cluster are not well clustered as they assume a negative value of Silhouette width:

```
silinfo <- km.res$silinfo
sil <- km.res$silinfo$widths[, 1:3]
neg_sil_index <- which(sil[, "sil_width"] < 0)
sil[neg_sil_index, drop = FALSE]

##     cluster neighbor   sil_width
## 185       2        1 -0.008551532
## 107       2        1 -0.022429991
## 186       2        1 -0.062019236
## 183       2        1 -0.105741685
## 191       2        1 -0.110592107
## 182       2        1 -0.146611131
```

From the dataframe above I can assume that each of the six observations which assume the negative value of Silhouette width would fit better in the first cluster.

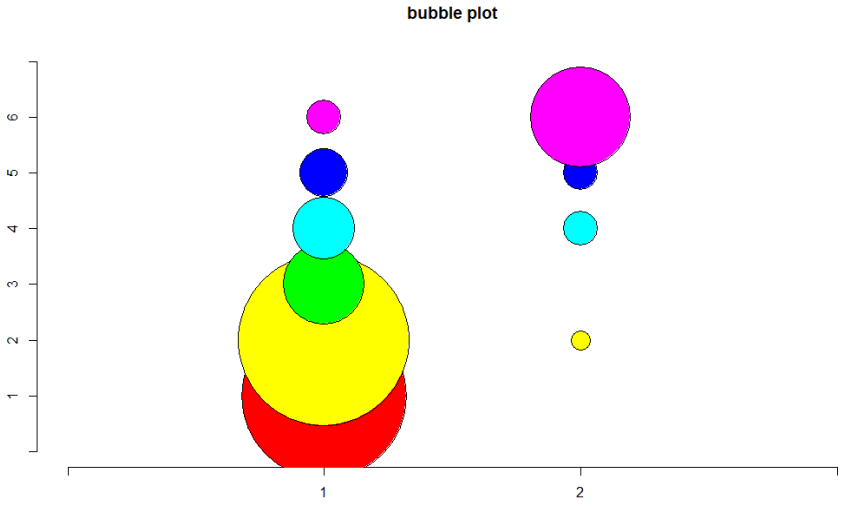
I also computed the Dunn index as internal cluster validation:

```
km_stats <- fpc:::cluster.stats(dist(scaled_glass), km.res$cluster)
km_stats$dunn

## [1] 0.08557619
```

Since the Dunn index must be maximized, we can say that according to this index the observations are not well clustered.

Regarding the external validation, corrected Rand and Meila's VI indexes have been computed in addition to the plot of the confusion matrix of the clusters and the external information (variable 'Type'):

```
labstatR:::bubbleplot(table(glass$Type, km.res$cluster))

clust_stats <- fpc:::cluster.stats(d = dist(scaled_glass), Type, km.res$cluster)

clust_stats$corrected.rand

## [1] 0.1929584

clust_stats$vi

## [1] 1.337794
```



Thanks to the plot I can affirm that cluster two represents headlamps and cluster 1 represent all the other types of glass. Agreement between the external information and clusters is 0.19, disagreement between external information and clusters is 1.34 so we can assume that the external information doesn't influence the clustering algorithm.

K-medoids

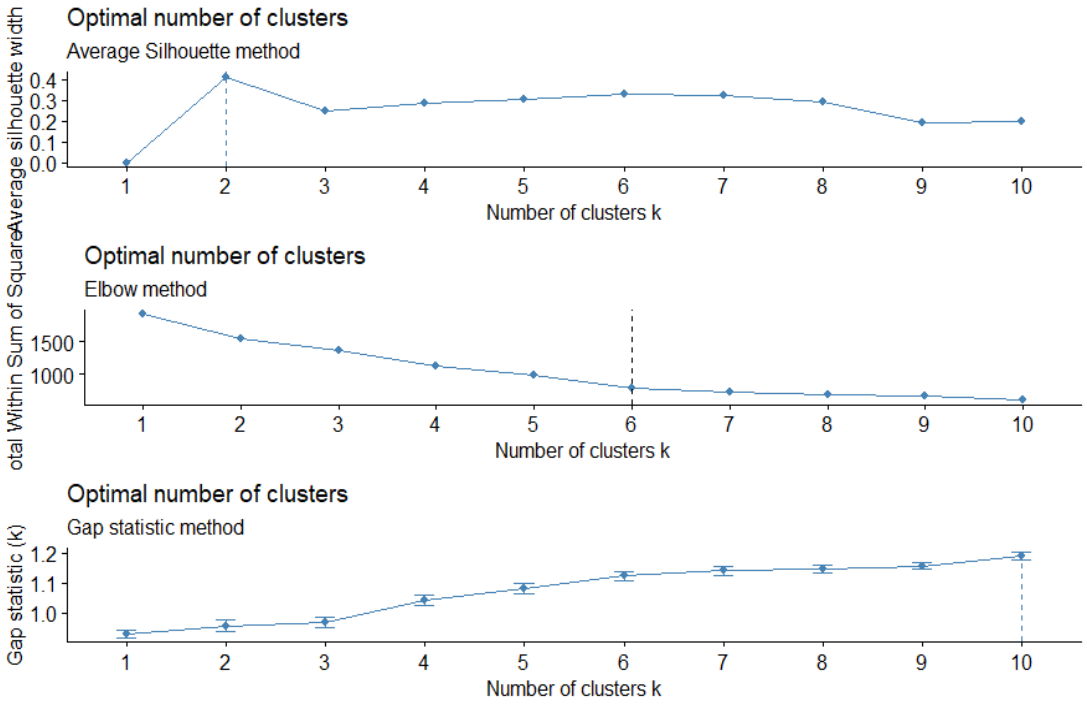
In this clustering method, each cluster is represented by one of the observations in the cluster, which is called medoid and is usually the most centrally located point in the cluster. The medoids must be interpreted as a representative example of the members of that cluster. This second method is supposed to be more robust and less sensitive to outliers because the cluster centre isn't represented by the mean.

As the k-medoids clustering method works with the distance matrix, I will compute the PAM clustering method with Euclidean and Manhattan distance.

Computing the optimal number of clusters - Euclidean distance

I first applied the PAM clustering method with the Euclidean distance:

```
km1 <- factoextra:::fviz_nbclust(scaled_glass, pam, method = "silhouette") +
labs(subtitle='Average Silhouette method')

km2 <-  factoextra:::fviz_nbclust(scaled_glass, pam, method = "wss" )
+geom_vline(xintercept = 6, linetype = 2)+labs(subtitle='Elbow method')

km3 <- factoextra:::fviz_nbclust(scaled_glass, pam, method = "gap_stat")
+labs(subtitle='Gap statistic method')
gridExtra:::grid.arrange(km1,km2,km3,nrow=3)
```



In this case I computed pam with k=2:

```
pam.euclid <- cluster:::pam(scaled_glass,2,metric='euclidean')
pam.euclid
```

```
## Medoids:
##       ID          RI          Na         Mg          Al         Si           K
## [1,]  43 -0.1894786 -0.2422846  0.489090 -0.2301493 0.1408121  0.1425101
## [2,] 198 -0.3607078  1.5823462 -1.861147  1.7928057 0.8121733 -0.7621317
##                Ca         Ba         Fe
## [1,] -0.257851750 -0.3520514 -0.5850791
## [2,] -0.004892386  0.9753308 -0.5850791
## Clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 2 1 2
## [186] 1 1 1 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2
## Objective function:
##   build    swap
## 2.17918 2.17918
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

The printed output represents the medoids for each cluster and a vector of clusters which each observation belongs to.

```
pam.euclid$clusinfo

##      size max_diss  av_diss diameter separation
## [1,]  183 10.19137 2.110446 14.13302   1.247174
## [2,]   31 10.42106 2.584937 15.67933   1.247174
```

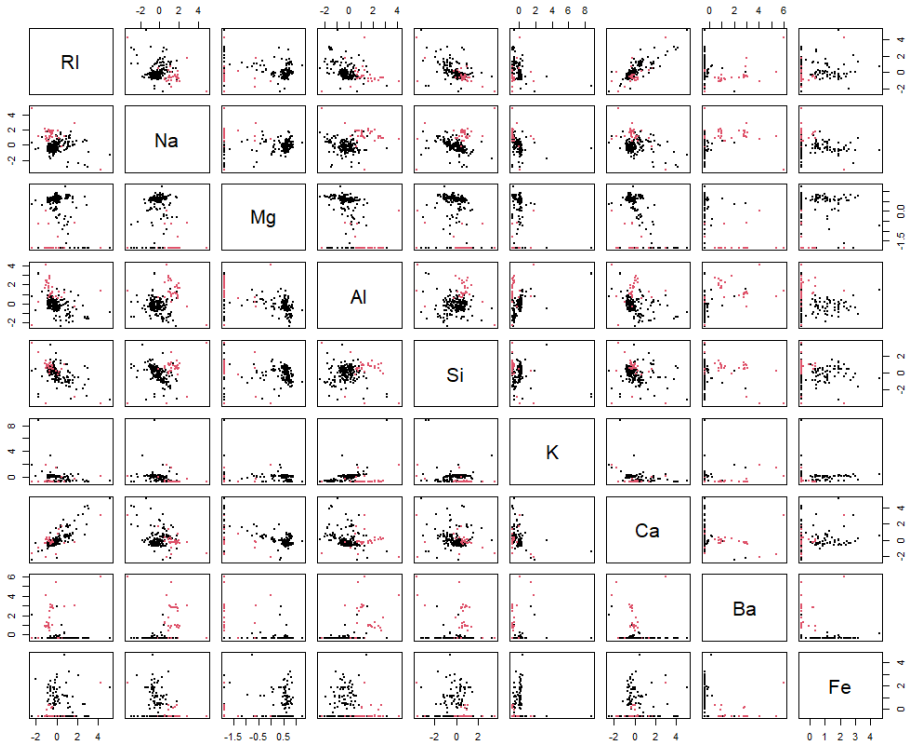
In this data frame, each row corresponds to a cluster, the first two columns represent the maximal and average dissimilarity between observations in the cluster and the medoids, the diameter of the cluster, such as the maximal dissimilarity between two observations of the cluster, and the separation such as the minimal dissimilarity between two observations of two different clusters.

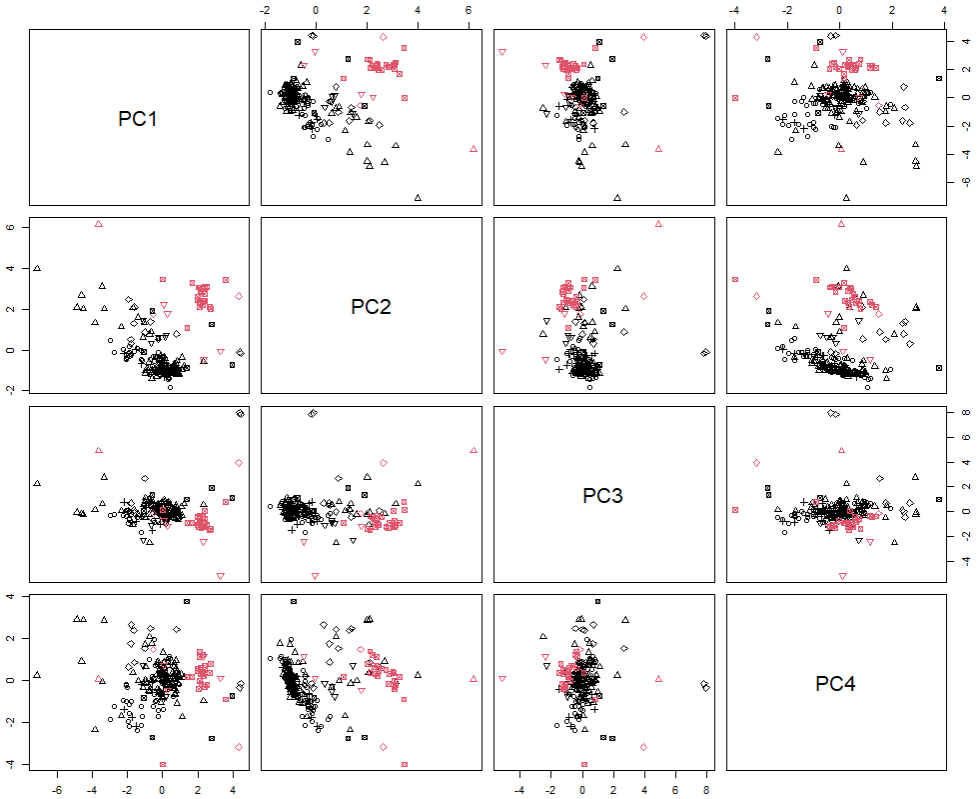
```
pam.euclid$isolation

##  1  2
## no no
## Levels: no L L*
```

The clusters are not isolated so maybe they will appear overlapped.

I can now visualize the clusters according to the PAM clustering method and the Euclidean distance in both original and PC space:

```
cl.pam1 <- pam.euclid$clustering
pairs(scaled_glass, pch=16, cex=0.5, col=cl.pam1)

pairs(PC[1:4], col=cl.pam1, pch=glass$Type)
```


RI
Na
Mg
Al
Si
K
Ca
Ba
Fe

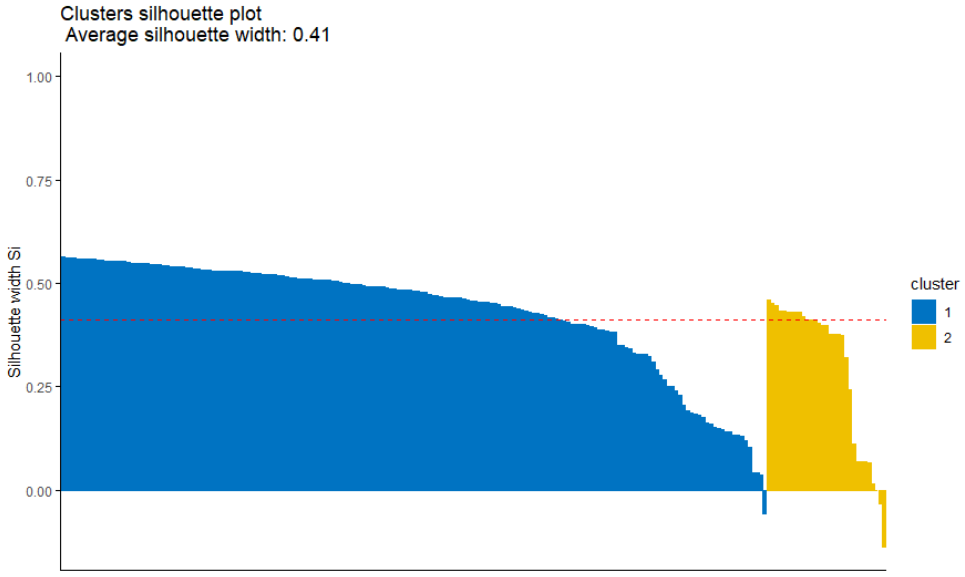

PC1
PC2
PC3
PC4

The red cluster contains observations with higher values of aluminium and barium, the black cluster contains observations with a higher value of magnesium. Also in this case it's possible to compare the clustering with the type of glass: the red cluster represents headlamps and the black cluster represents building windows and vehicle windows, both float and non-float processed.

Cluster validation - Euclidean distance

According to the internal validation, I first computed the average silhouette width:

```
pam.euclid <- factoextra:::eclust(scaled_glass, "pam", k = 2, nstart = 25, graph = FALSE)
factoextra:::fviz_silhouette(pam.euclid, palette = "jco",
                ggtheme = theme_classic())

##   cluster size ave.sil.width
## 1       1  183          0.43
## 2       2   31          0.29
```



The dataframe below shows the size and the average silhouette width of each cluster, while the total value is 0.41, for this reason, I assume that the dataset according to this clustering method is not well clustered.

In addition, there are some observations which assume negative value:

```
sil <- pam.euclid$silinfo$widths[, 1:3]
neg_sil_index <- which(sil[, "sil_width"] < 0)
sil[neg_sil_index, drop = FALSE]

##     cluster neighbor    sil_width
## 187       1        2 -0.057533452
## 107       2        1 -0.001887105
## 181       2        1 -0.032526131
## 170       2        1 -0.136858873
```

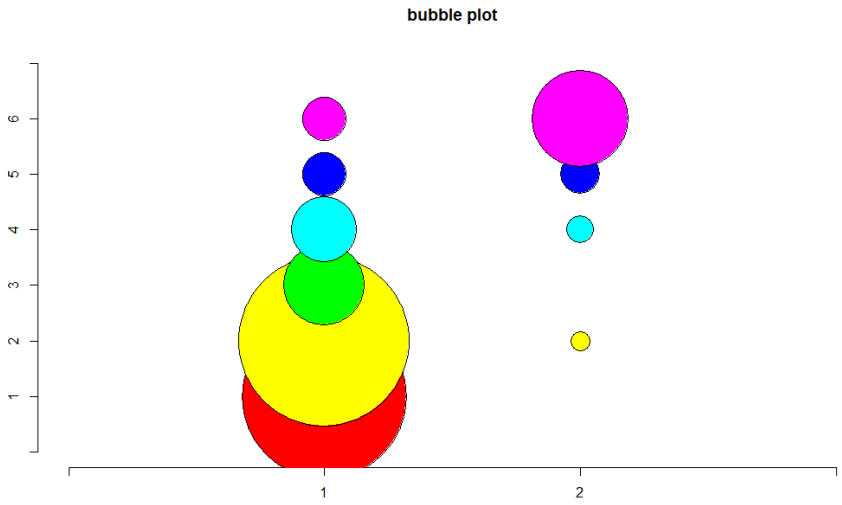
There are 4 observations out of 214 which assume the negative value of Silhouette width and that means that those observations would fit better in the neighbour cluster.

```
kme_stats <- fpc:::cluster.stats(dist(scaled_glass), pam.euclid$cluster)
kme_stats$dunn

## [1] 0.07954256
```

The Dunn index is 0.08 so we can assume that according to this index the observations are not well-clustered.

Regarding external validation, I first computed the plot of the confusion matrix:

```
labstatR:::bubbleplot(table(glass$Type, pam.euclid$cluster))
```



According to the plot, it seems that cluster 2 represents headlamps and cluster 1 represents almost all the other types of glass.

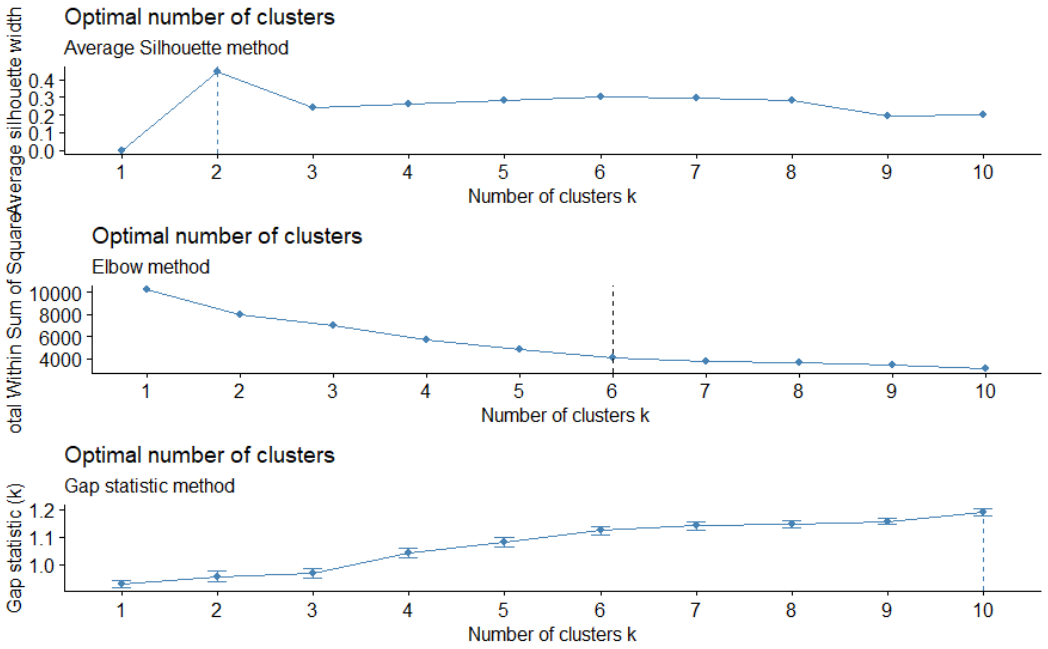
```
fpc:::cluster.stats(d = dist(scaled_glass),Type, pam.euclid$cluster)$vi

## [1] 1.379261

fpc:::cluster.stats(d = dist(scaled_glass),Type, pam.euclid$cluster)$corrected.rand

## [1] 0.1748818
```

Disagreement between cluster solution and external information is 1.38 and agreement between cluster solution and external information is 0.17 so we can confirm that the external information doesn't influence the clustering algorithm.

Computing optimal number of clusters - Manhattan distance

Secondly, I computed the PAM algorithm according to the Manhattan distance:

```
pm1 <- factoextra:::fviz_nbclust(scaled_glass, pam, method = "silhouette", diss=manhattan)
+ labs(subtitle='Average Silhouette method')

pm2 <-  factoextra:::fviz_nbclust(scaled_glass, pam, method = "wss", diss=manhattan )
+geom_vline(xintercept = 6, linetype = 2)+labs(subtitle='Elbow method')

pm3 <- factoextra:::fviz_nbclust(scaled_glass, pam, method = "gap_stat", diss=manhattan)
+labs(subtitle='Gap statistic method')
gridExtra:::grid.arrange(pm1,pm2,pm3,nrow=3)
```



According to these methods, I computed the PAM algorithm with Manhattan distance and k=2:

```
pam.man <- cluster:::pam(scaled_glass,2,metric='manhattan')
pam.man

## Medoids:
##       ID         RI         Na         Mg         Al         Si          K
## [1,]  43 -0.1894786 -0.2422846  0.489090 -0.2301493 0.1408121  0.1425101
## [2,] 205 -0.7229236  1.8884923 -1.861147  1.6526009 0.8379949 -0.7621317
##              Ca         Ba         Fe
## [1,] -0.2578517 -0.3520514 -0.5850791
## [2,] -0.1735320  0.9954427 -0.5850791
## Clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
## [186] 1 1 1 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2
## Objective function:
##    build     swap
## 4.648352 4.648352
##
```

```
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

The printed output represents the medoids for each cluster and a vector of clusters to which each observation belongs to.

```
pam.man$clusinfo

##      size max_diss  av_diss diameter separation
## [1,]  183 21.32106 4.475231 31.34456   2.735625
## [2,]   31 26.95564 5.670320 40.28822   2.735625
```

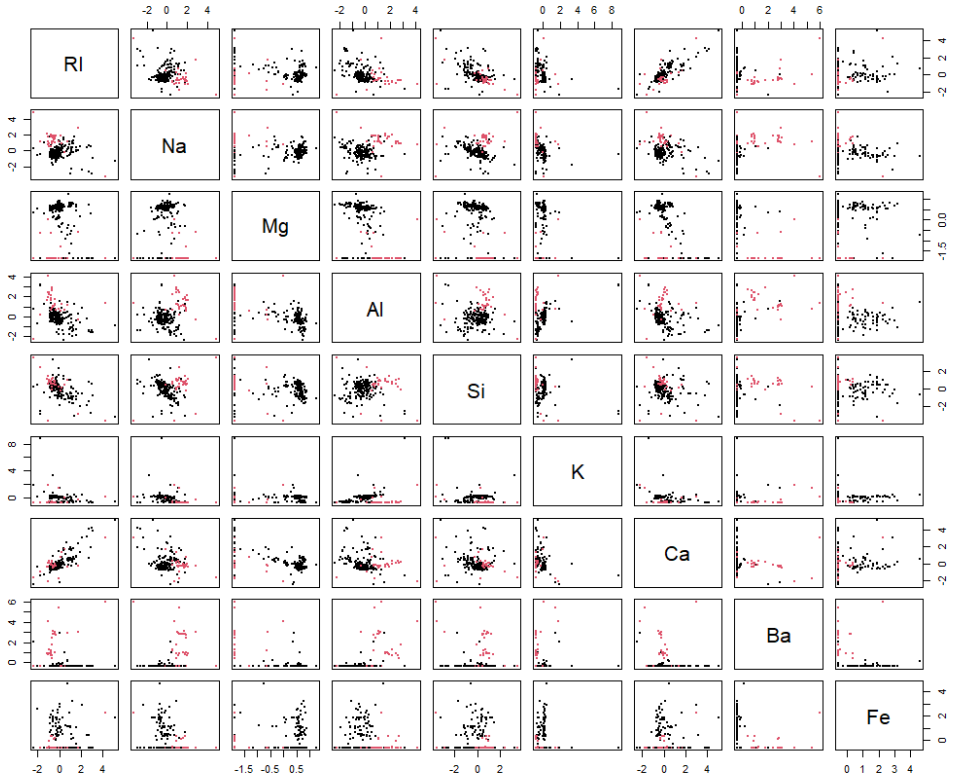
In the dataframe above, each row corresponds to a cluster, the first two columns represent the maximal and average dissimilarity between observations in the cluster and the medoids, the diameter of the cluster, such as the maximal dissimilarity between two observations of the cluster, and the separation such as the minimal dissimilarity between two observations of two different clusters.

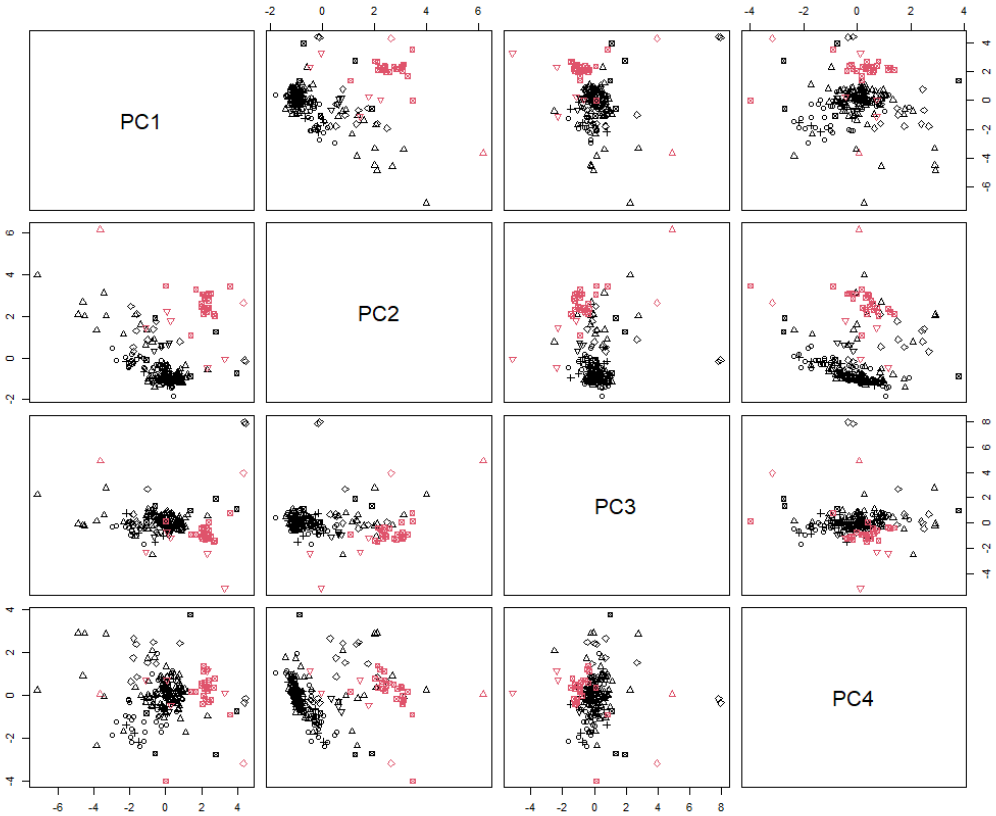
```
pam.man$isolation

##  1  2
## no no
## Levels: no L L*
```

All the clusters are not isolated so I can predict that maybe they will be overlapped.

I can now visualize the clusters in the original space and the PC space:

```
cl.pam2 <- pam.man$clustering
pairs(scaled_glass,  pch=16, cex=0.5, col=cl.pam2)

pairs(PC[1:4], col=cl.pam2, pch=glass$Type)
```


RI
Na
Mg
Al
Si
K
Ca
Ba
Fe

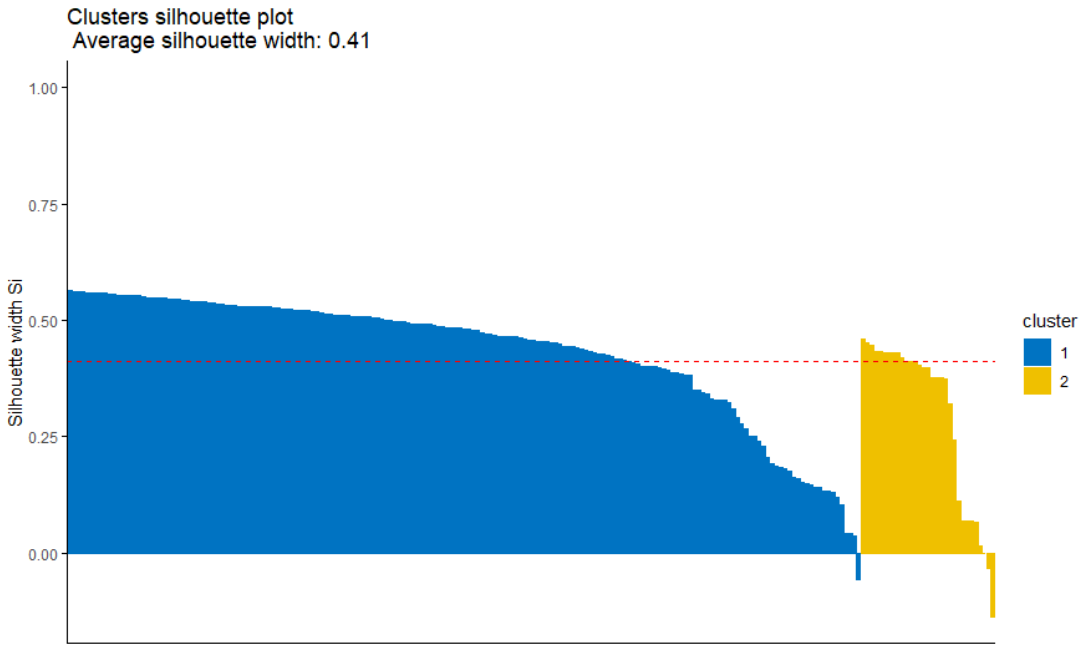

PC1
PC2
PC3
PC4

The red cluster contains observations with higher values of aluminium and barium and the black cluster contains observations with value of magnesium above the mean. By highlighting the type of glass in the same plot, it's possible to see that the red cluster represents headlamps and the black cluster represents building windows and vehicle windows, both float and non-float processed.

Cluster validation - Manhattan distance

I first analyze the internal validation of the clusters:

```
pam.man <- factoextra:::eclust(scaled_glass, "pam", k =2, nstart = 25, graph = FALSE)
factoextra:::fviz_silhouette(pam.man, palette = "jco",
                ggtheme = theme_classic())

##   cluster size ave.sil.width
## 1       1  183          0.43
## 2       2   31          0.29
```



The dataframe above shows the size and the average silhouette width of each cluster, while the total value is 0.41. According to this, I can affirm that the dataset is not well-clustered thanks to the PAM clustering method and Manhattan distance.

There are some observations which assume negative value:

```
sil <- pam.man$silinfo$widths[, 1:3]
neg_sil_index <- which(sil[, "sil_width"] < 0)
sil[neg_sil_index, , drop = FALSE]

##     cluster neighbor    sil_width
## 187       1        2 -0.057533452
```

```
## 107       2        1 -0.001887105
## 181       2        1 -0.032526131
## 170       2        1 -0.136858873
```

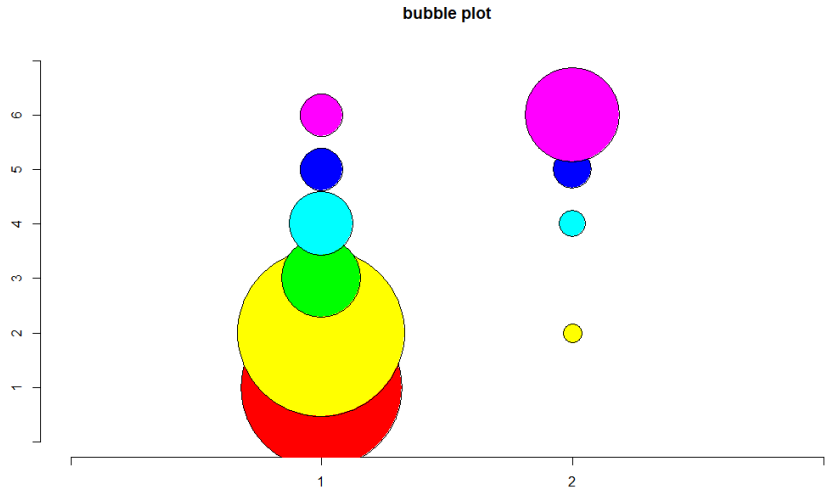
There are 4 observations out of 214 which assumes a negative Silhouette width value and the data frame shows the cluster to which they belong. According to this, each of the neighbour clusters would fit better in each observation.

```
kme_stats <- fpc:::cluster.stats(dist(scaled_glass), pam.man$cluster)
kme_stats$dunn

## [1] 0.07954256
```

As the Dunn index should be maximized, according to this index the cluster validation has produced bad results.

Regarding the external validation statistics, I first computed the plot of the confusion matrix:

```
labstatR:::bubbleplot(table(glass$Type, pam.man$cluster))
```



The plot shows that cluster 2 represent the headlamps and cluster 1 represents all the other types of glass.

```
fpc:::cluster.stats(d = dist(scaled_glass), Type, pam.man$cluster)$vi

## [1] 1.379261

fpc:::cluster.stats(d = dist(scaled_glass),Type, pam.man$cluster)$corrected.rand

## [1] 0.1748818
```

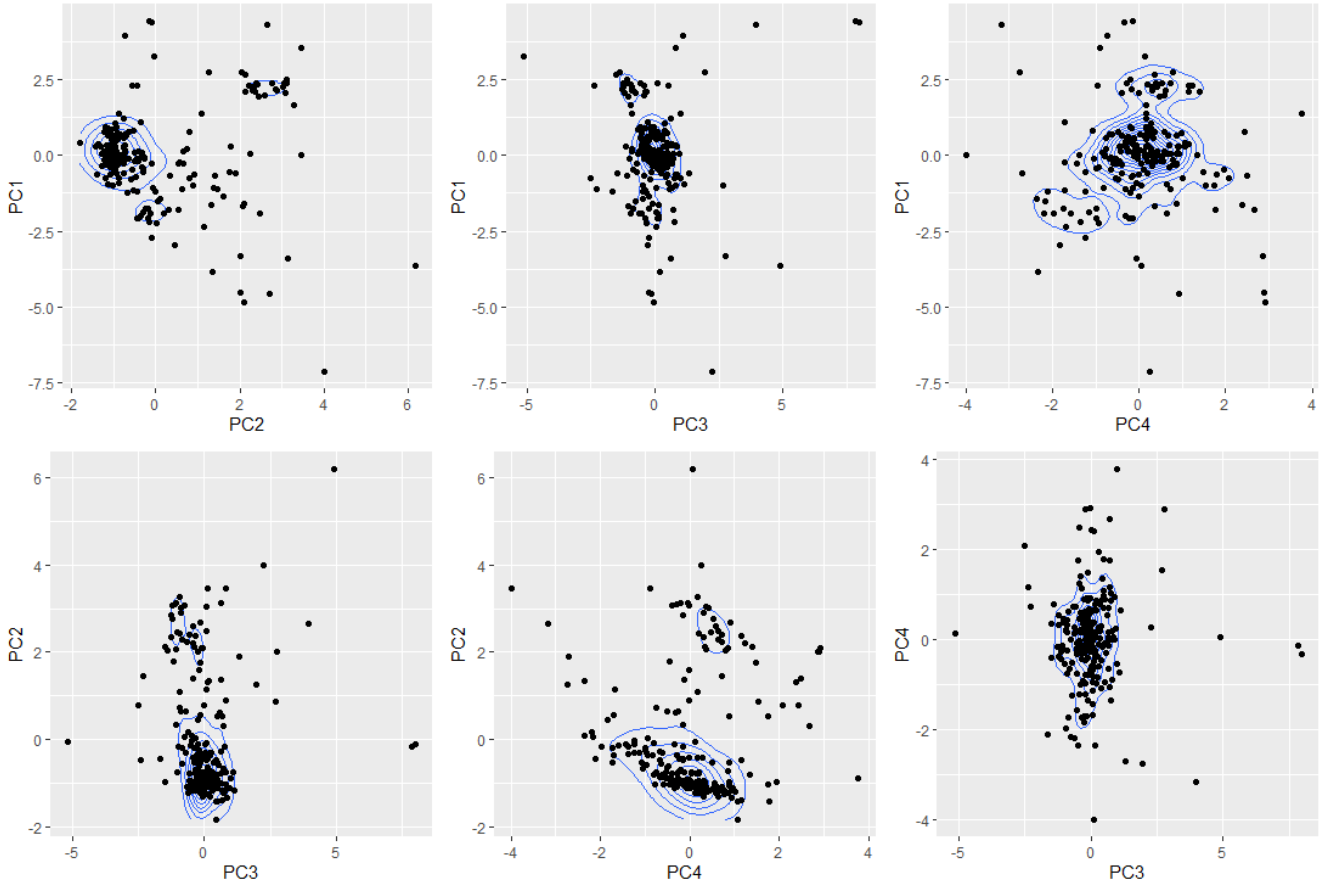
Disagreement between clusters and external information is 1.38, an agreement between clusters and external information is 0.17 so I can confirm that external information doesn't influence the clusters.

Model-based clustering

This type of clustering method considers the dataset as a mixture of two or more distributions that correspond to clusters, so each distribution will have its probability mass or density function. Each observation can belong to different distribution with a different probability.

I made a graphical representation of the data throw the Kernel smooth density to understand if the values have a cluster behavior:

```
pc1.2 <- ggplot2:::ggplot(PC, aes(x=PC2, y=PC1)) + geom_density_2d() + geom_point()
pc1.3 <- ggplot2:::ggplot(PC, aes(x=PC3, y=PC1)) + geom_density_2d() + geom_point()
pc1.4 <- ggplot2:::ggplot(PC, aes(x=PC4, y=PC1)) + geom_density_2d() + geom_point()
pc2.3 <- ggplot2:::ggplot(PC, aes(x=PC3, y=PC2)) + geom_density_2d() + geom_point()
pc2.4 <- ggplot2:::ggplot(PC, aes(x=PC4, y=PC2)) + geom_density_2d() + geom_point()
pc3.4 <- ggplot2:::ggplot(PC, aes(x=PC3, y=PC4)) + geom_density_2d() + geom_point()
grid.arrange(pc1.2,pc1.3,pc1.4,pc2.3,pc2.4,pc3.4, ncol=3)
```

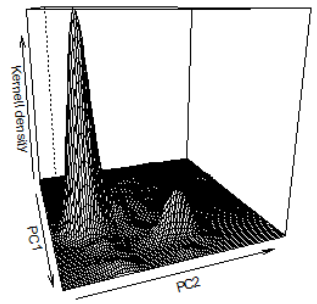


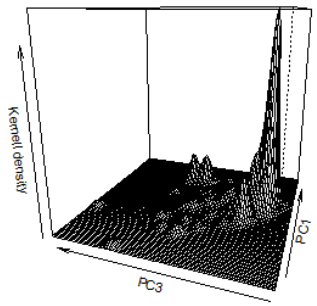
By representing the Kernel smooth density in the Principal Component space, I can affirm that the distribution has at least four clusters. The shape of each cluster appears ellipsoidal with different volumes and orientations.

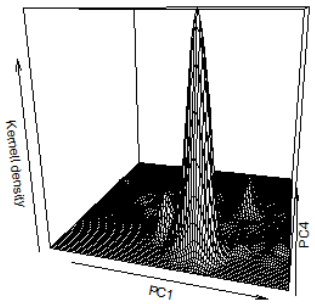
In the plot below I represented 3D Kernel smooth density:

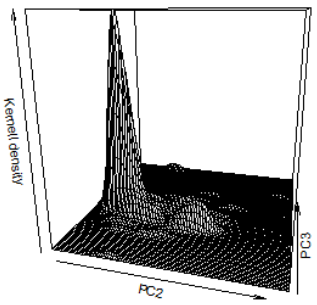
```
par(mfrow=c(2,3))
griglia1.2 <- KernSmooth:::bkde2D(PC[1:2], gridsize = c(101,101), bandwidth=c(density(PC[,
```

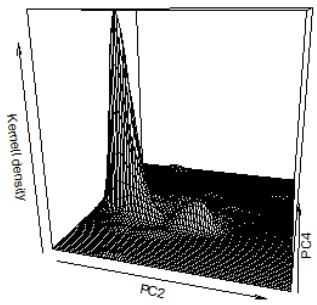
```
1])$bw,density(PC[,2])$bw))
pc1.2 <- persp(x = griglia1.2$x1, y = griglia1.2$x2, z = griglia1.2$fhat, xlab='PC1', ylab
='PC2', zlab='Kernell density', theta = 15, axes = TRUE, box = TRUE)
griglia1.3 <- KernSmooth:::bkde2D(PC[1:3], gridsize = c(101,101), bandwidth=c(density(PC[,
1])$bw,density(PC[,3])$bw))
persp(x = griglia1.3$x1, y = griglia1.3$x2, z = griglia1.3$fhat, xlab='PC1', ylab='PC3', z
lab='Kernell density', theta = 15, axes = TRUE, box = TRUE)
griglia1.4 <- KernSmooth:::bkde2D(PC[1:4], gridsize = c(101,101), bandwidth=c(density(PC[,
1])$bw,density(PC[,4])$bw))
persp(x = griglia1.4$x1, y = griglia1.4$x2, z = griglia1.4$fhat,xlab='PC1', ylab='PC4', zl
ab='Kernell density', theta = 15, axes = TRUE, box = TRUE)
griglia2.3 <- KernSmooth:::bkde2D(PC[2:3], gridsize = c(101,101), bandwidth=c(density(PC[,
2])$bw,density(PC[,3])$bw))
persp(x = griglia2.3$x1, y = griglia2.3$x2, z = griglia2.3$fhat, xlab='PC2', ylab='PC3', z
lab='Kernell density', theta = 15, axes = TRUE, box = TRUE)
griglia2.4 <- KernSmooth:::bkde2D(PC[2:4], gridsize = c(101,101), bandwidth=c(density(PC[,
2])$bw,density(PC[,4])$bw))
persp(x = griglia2.4$x1, y = griglia2.4$x2, z = griglia2.4$fhat, xlab='PC2', ylab='PC4', z
lab='Kernell density', theta = 15, axes = TRUE, box = TRUE)
griglia3.4 <- KernSmooth:::bkde2D(PC[3:4], gridsize = c(101,101), bandwidth=c(density(PC[,
3])$bw,density(PC[,4])$bw))
persp(x = griglia3.4$x1, y = griglia3.4$x2, z = griglia3.4$fhat, xlab='PC3', ylab='PC4', z
lab='Kernell density', theta = 15, axes = TRUE, box = TRUE)
```

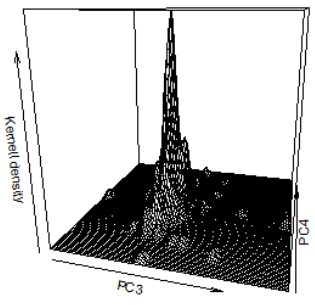












In this report I analyzed the dataset to understand if it comes from a mixture of Gaussian distributions:

```
mod_clust <- mclust:::Mclust(scaled_glass, G = 1:9, modelNames = NULL)
summary(mod_clust)

## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEV (ellipsoidal, equal shape) model with 5 components:
##
##  log-likelihood   n  df       BIC       ICL
##       -12.11575 214 242 -1322.798 -1324.732
##
## Clustering table:
##  1  2  3  4  5
## 36 76 45 37 20
```

According to the BIC value, the optimal model is composed of a mixture of 5 ellipsoidal clusters with variable volume, and variable orientation.

In addition, 36 observations belong to the first component, 76 observations belong to the second component, 45 observations belong to the third component, 37 observations belong to the fourth component, 20 observations belong to the fifth component.

The log-likelihood of the optimal BIC is -12.116, the estimated parameters are 242, the n column corresponds to the number of observations and the ICL value of the selected model is 1324.732. To visualize the whole BIC values:

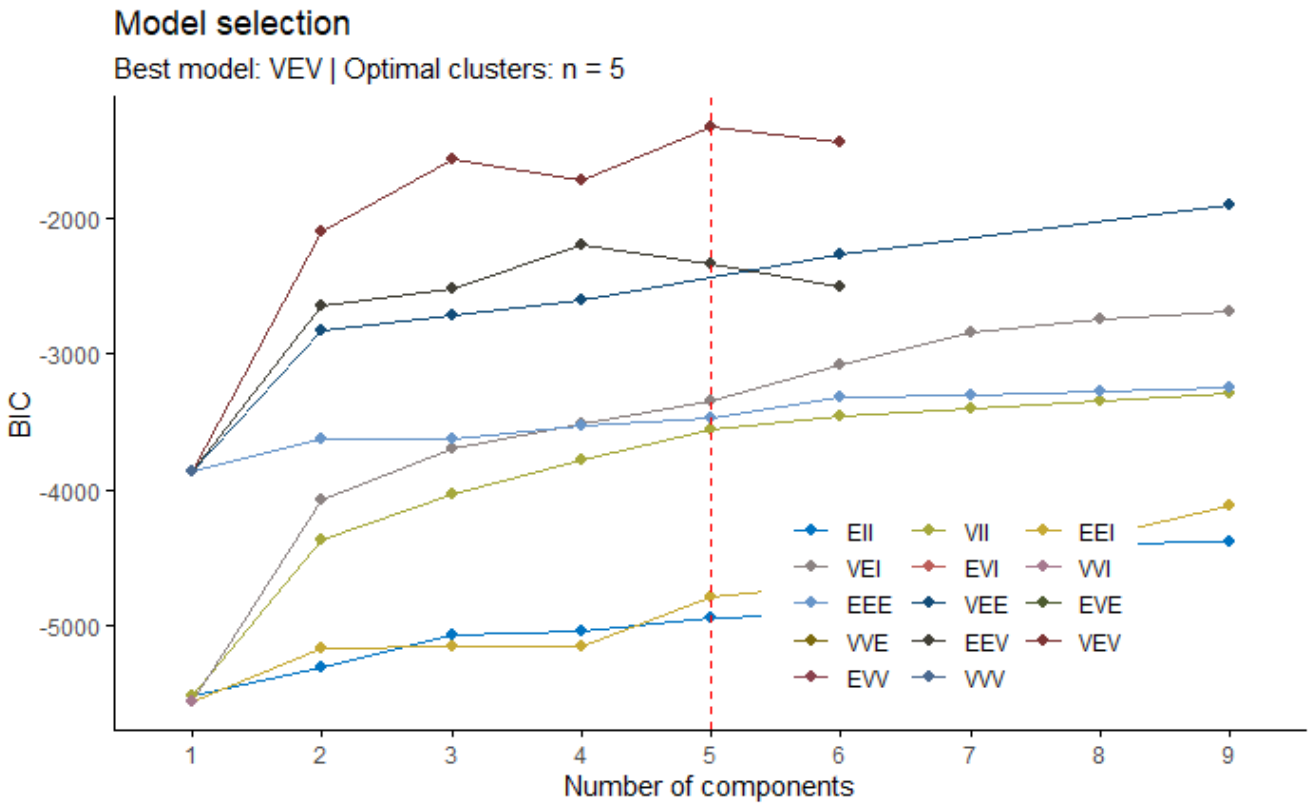
```
mod_clust$BIC

## Bayesian Information Criterion (BIC):
##         EII       VII       EEI       VEI       EVI       VVI       EEE
## 1 -5510.390 -5510.390 -5553.318 -5553.318 -5553.318 -5553.318 -3866.578
## 2 -5297.923 -4363.708 -5163.715 -4075.571        NA        NA -3626.847
## 3 -5070.865 -4034.932 -5148.170 -3700.126        NA        NA -3618.166
## 4 -5044.131 -3776.754 -5143.493 -3514.393        NA        NA -3520.064
## 5 -4933.348 -3558.717 -4790.135 -3340.231        NA        NA -3466.552
## 6 -4909.650 -3456.029 -4689.577 -3080.420        NA        NA -3319.931
## 7 -4522.563 -3406.409 -4349.463 -2834.398        NA        NA -3302.856
## 8 -4390.736 -3341.249 -4337.570 -2740.640        NA        NA -3268.757
## 9 -4373.833 -3290.207 -4119.237 -2680.132        NA        NA -3249.653
##         VEE       EVE       VVE       EEV       VEV       EVV       VVV
## 1 -3866.578 -3866.578 -3866.578 -3866.578 -3866.578 -3866.578 -3866.578
## 2 -2828.795        NA        NA -2648.617 -2097.475        NA        NA
## 3 -2712.819        NA        NA -2523.861 -1572.503        NA        NA
## 4 -2607.856        NA        NA -2197.444 -1720.677        NA        NA
## 5        NA        NA        NA -2333.250 -1322.798        NA        NA
## 6 -2272.062        NA        NA -2509.409 -1436.353        NA        NA
## 7        NA        NA        NA        NA        NA        NA        NA
## 8        NA        NA        NA        NA        NA        NA        NA
## 9 -1901.950        NA        NA        NA        NA        NA        NA
##
## Top 3 models based on the BIC criterion:
```

```
##     VEV,5     VEV,6     VEV,3
## -1322.798 -1436.353 -1572.503
```

The dataframe contains all the BIC values of each variable according to the different models. The top three models are VEV mixture of 5, 6, and 3 components.

For a better interpretation, throw the code below it's possible to visualize the plot of the BIC values:

```
factoextra:::fviz_mclust(mod_clust, "BIC", palette = "jco")
```

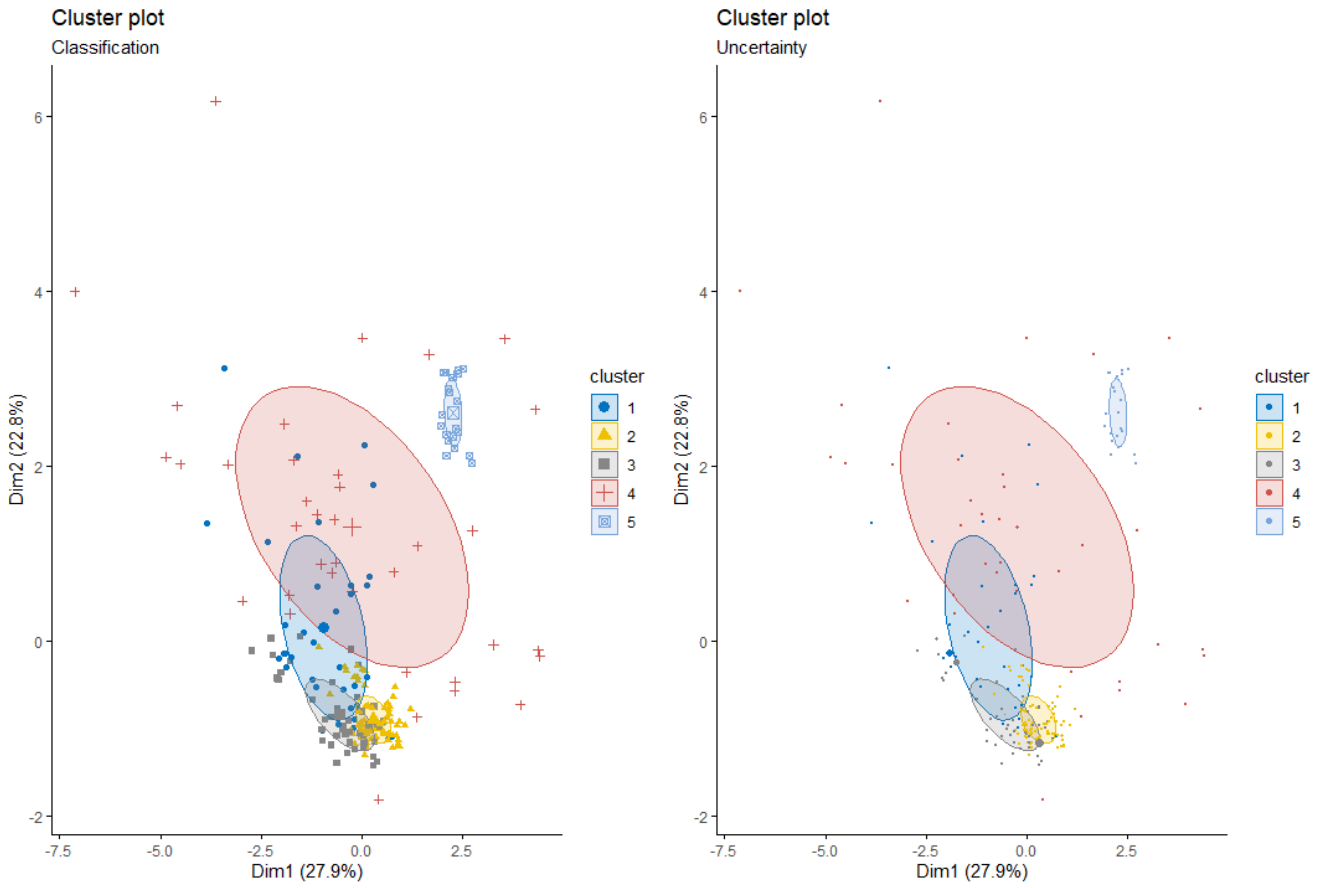



```
head(mod_clust$z)

##              [,1]         [,2]         [,3]         [,4] [,5]
## [1,] 9.999999e-01 2.529074e-23 1.228248e-26 1.402203e-07    0
## [2,] 2.187361e-05 9.999781e-01 6.394014e-12 1.739080e-09    0
## [3,] 5.707745e-04 9.994292e-01 1.416965e-28 6.304533e-08    0
## [4,] 4.414414e-25 1.000000e+00 1.059559e-82 2.610824e-13    0
## [5,] 3.116467e-09 1.000000e+00 2.837382e-21 1.003001e-11    0
## [6,] 2.107727e-04 0.000000e+00 9.997890e-01 1.883353e-07    0
```

The matrix above represents the probability that each observation belongs to each class.

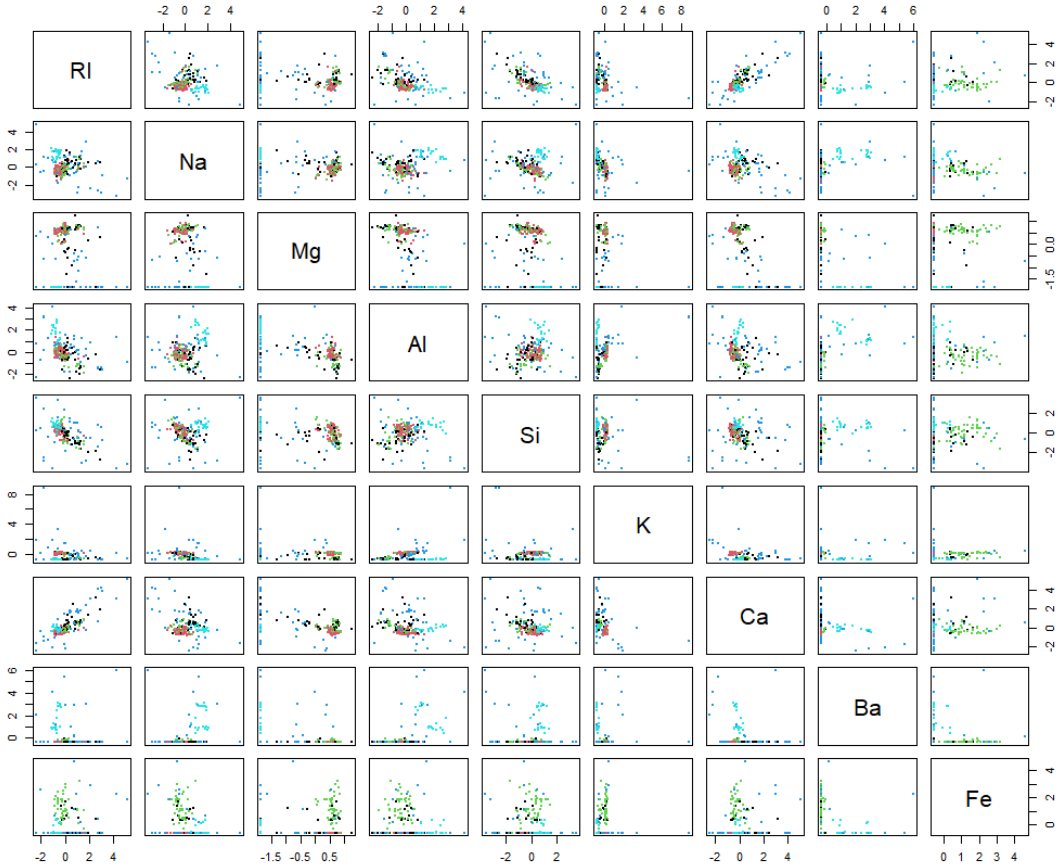
In the following plot we can see the classification of each observation on the first two principal component space and the uncertainty, highlighted as bigger points which has a low probability to belong to clusters:

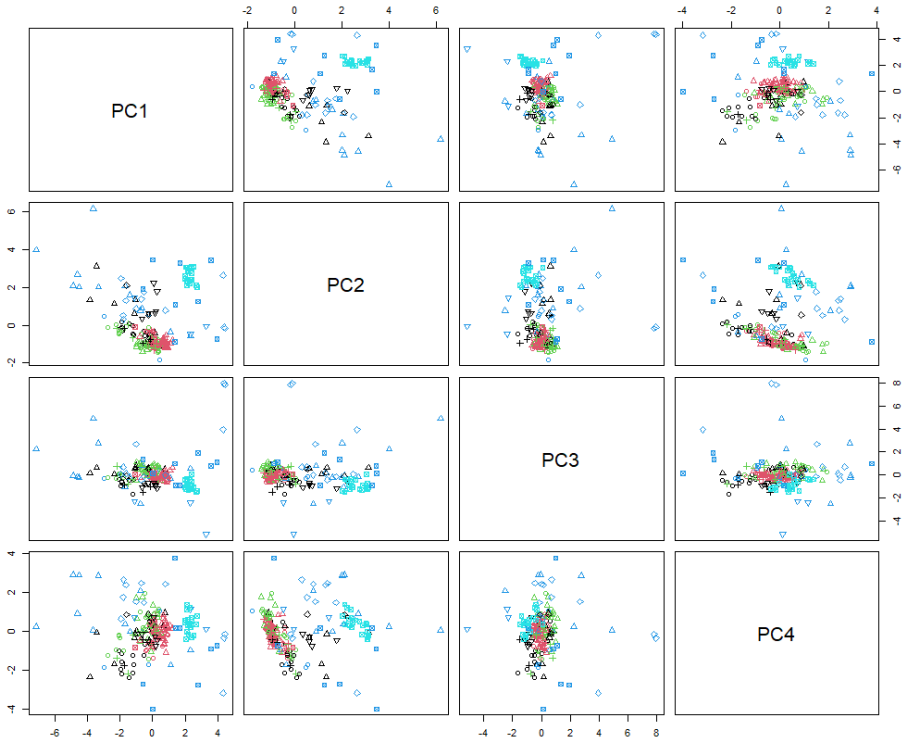
```
class <- fviz_mclust(mod_clust, "classification", geom = "point", pointsize = 1.5, palette
= "jco")
uncert <- fviz_mclust(mod_clust, "uncertainty", palette = "jco")

grid.arrange(class,uncert, ncol=2)
```



Throw the following code I analyzed the clusters both in the original and PC space:

```
pairs(scaled_glass, col = mod_clust$classification, pch=16, cex=0.5)

pairs(PC[1:4], col=mod_clust$classification, pch=Type)
```

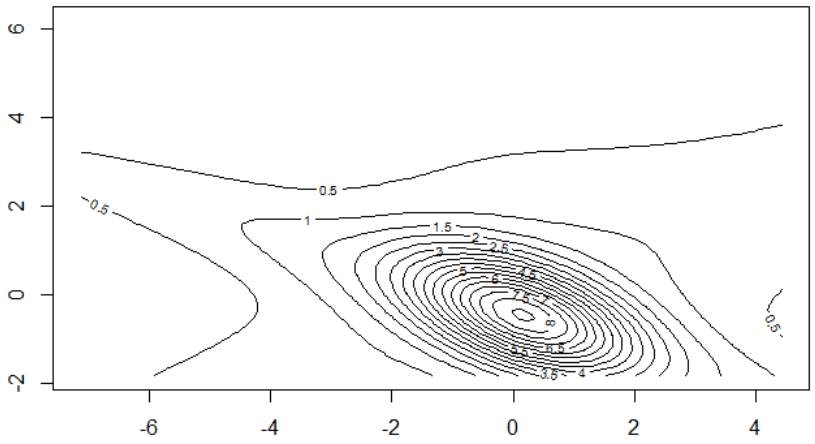

RI
Na
Mg
Al
Si
K
Ca
Ba
Fe


PC1
PC2
PC3
PC4

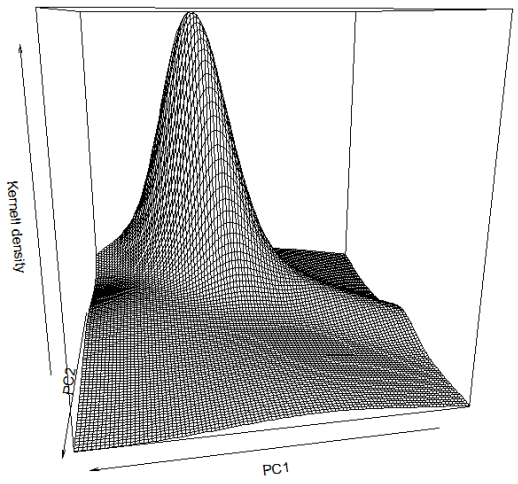
By highlighting the type of glass together with the cluster, I can see that the clusters almost coincide with the types of glasses.

In the following code I computed the Kernel smooth density of the selected mixture model both 2D and 3D in the first two Principal Component spaces:

```
x  <- mclust:::grid1(100, range = range(PC[,1]))
y  <- mclust:::grid1(100, range = range(PC[,2]))
xy <- mclust:::grid2(x,y)
xyDens <- dens(modelName = mod_clust$modelName, data = xy, parameters =
mod_clust$parameters)
xyDens <- matrix(xyDens, nrow = length(x), ncol = length(y))
contour(x = x, y = y, z = xyDens, nlevels = 20)
```




```
persp(x = x,
      y = y,
      z = xyDens,
      xlab = "PC1",
      ylab = "PC2",
      zlab = "Kernell density",
      theta=170)
```



Cluster validation

External validation has been computed to understand if the result from the model-based clustering is good:

```
table(glass$Type, mod_clust$classification)

##                         1  2  3  4  5

##  Building Windows FP   14 32 21  3  0

##  Building Windows NFP 11 35 18 12  0

##  Vehicle Window FP      3  8  6  0  0

##  Containers             2  0  0 11  0

##  Tableware              6  0  0  3  0

##  Headlamp               0  1  0  8 20
```

In the matrix below It seems that that the fifth group represents headlamps, the first three groups represent buildings and vehicles glass, either float and non-float processed and fourth group represents containers.

```
fpc:::cluster.stats(d = dist(scaled_glass),Type, mod_clust$classification)$vi

## [1] 2.125523

mclust:::adjustedRandIndex(glass$Type, mod_clust$classification)
```

```
## [1] 0.1470175
```

Disagreement between cluster solution and external information is 2.13 and agreement between cluster solution and external information is 0.15 so we can confirm that the external information doesn't influence the clustering algorithm.

Computing the best algorithm

Internal measures

We can use information in the data to access to quality of the clustering thanks to the connectivity, silhouette coefficient, and Dunn index:

```
clmethods <- c('hierarchical','kmeans','pam')
intern <- clValid:::clValid(scaled_glass, nClust = 2:5, clMethods = clmethods, validation
= "internal")

summary(intern)

##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                                  2       3       4       5
##
## hierarchical Connectivity   3.8579  6.7869  9.7159 17.6512
##              Dunn           0.5439  0.5302  0.3814  0.1962
##              Silhouette     0.6735  0.6493  0.5497  0.5020
## kmeans       Connectivity   6.6202 33.4040 42.8091 51.0841
##              Dunn           0.2258  0.1308  0.0866  0.1036
##              Silhouette     0.6418  0.4082  0.4441  0.4352
## pam          Connectivity  24.5464 59.6290 71.6718 74.8762
##              Dunn           0.0795  0.0367  0.0456  0.0104
##              Silhouette     0.4112  0.2502  0.2828  0.3019
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 3.8579 hierarchical 2
## Dunn         0.5439 hierarchical 2
## Silhouette   0.6735 hierarchical 2
```

In the dataframe, hierarchical, k-means and PAM clustering methods have been analyzed throw Connectivity, Dunn, and Silhouette indices.

The hierarchical clustering method with 2 clusters is the best clustering method for the dataset.

Stability measures

This type of validation compares the stability of clustering results with the clusters obtained after each variable is removed, one at a time.

```
stab <- clValid:::clValid(scaled_glass, nClust = 2:5, clMethods = clmethods, validation =
"stability")

summary(stab)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                         2      3      4      5
##
## hierarchical APN  0.0010 0.0031 0.0183 0.0144
##              AD   3.4856 3.4171 3.3694 3.2224
##              ADM  0.0271 0.0540 0.1465 0.1558
##              FOM  0.9883 0.9740 0.9631 0.8975
## kmeans       APN  0.0202 0.1484 0.1062 0.1127
##              AD   3.4902 3.3043 2.7786 2.7273
##              ADM  0.1138 0.6543 0.4445 0.5230
##              FOM  0.9858 0.9537 0.8583 0.8448
## pam          APN  0.0826 0.1198 0.1266 0.1079
##              AD   3.2496 2.9720 2.6744 2.5054
##              ADM  0.3794 0.4838 0.4943 0.4982
##              FOM  0.9684 0.9302 0.8759 0.8514
##
## Optimal Scores:
##
##     Score  Method       Clusters
## APN 0.0010 hierarchical 2
## AD  2.5054 pam          5
## ADM 0.0271 hierarchical 2
## FOM 0.8448 kmeans       5
```

Also, in this case, each cluster obtained with hierarchical, k-means, and PAM clustering methods have been analyzed throw APN, AD, ADM, and FOM indices.

According to the APN index, the best algorithm is the hierarchical clustering method with 2 clusters, according to the AD index the best algorithm is the PAM clustering method with 5 clusters, according to the ADM index the best algorithm is hierarchical clustering method with 2 clusters and according to the FOM index, the best algorithm is k-means clustering method with 5 clusters.

Conclusions

According to the univariate analysis of each variable, most of the continuous variables come from a mixture of distributions, instead, the categorical variable comes from a Zero Altered Negative Binomial Type I distribution. Each variable has many outliers and neither of those is symmetric distribution. In addition, as the mean and the variance are significatively different and there are different scales of measure, standardization has been applied to the dataset.

According to the multivariate analysis:

- Thanks to the dimensionality reduction, almost 80% of the dataset can be explained by 4 Principal Components and it allows me to compare only six plots. Even if the variables are correlated with each other, this is not an optimal result as no more than 3 Principal Components would be the optimal solution. The first two Principal Components explain almost 50% of the distribution and the first three Principal Components explain almost 60% of the distribution and this is not a good compromise.
- The H index was very low, and thus justify a clustering tendency of the dataset. Throw the VAT algorithm it's possible to predict the presence of two clusters that are not well balanced.
    - Concerning the agglomerative clustering method, two clusters were highlighted on the dendrogram of the original space and merge at 10.3, in the dimensionally reduced space the clusters merge at about 1.5. Even if the clusters are not well balanced, better results have been computed thanks to the Principal Component space.
    - The partitioning clustering shows two relevant clusters: headlamps belong to the first cluster and building windows and vehicle windows, both float and non-float processed, belong to the second cluster.
    - By looking at the first Kernel smooth density, it seems that the dataset contains at least four distributions with elliptical different shapes and orientations. However, the BIC values suggested five VEV clusters which almost coincide with the types of glass.

According to these considerations, the internal and stability measures suggest that the best clustering algorithm is the agglomerative hierarchical clustering method by highlighting the last two clusters with the highest distance.

In my opinion, the best classification according to the data set is the model-based clustering which highlights different types of glass as they come from different distributions to form a mixture of distribution.