DATA ANALYSIS'S REPORT

Fires in Algerian Forests

M1000049555
Fabio Spampinato

This dataset is about fires in algerian forest, it includes 243 observations from two different Algeria's regions. It is composed by 7 numerical variables and 1 categorical variable that says to us if a Fire occur or not in that specifical day.

```
data = algerian[-c(123,124,125,169),-c(1,2,3,11,12,13)]
data5 = algerian[-c(123,124,125,169),-c(2,3,11,12,13)]
data2 = data[,-8]
str(data)
```

```
library(GGally)

library(cluster)

library(mclust)

library(DataExplorer)

library(factoextra)

library(gamlss)

library(gridExtra)

library(tidyverse)

library(ggcorrplot)

library(clustertend)

library(clValid)

library(NbClust)

library(moments)


library(fpc)

data$Temperature = as.numeric(as.character(data$Temperature))
data$RH = as.numeric(as.character(data$RH))
data$Ws = as.numeric(as.character(data$Ws))
data$Rain = as.numeric(as.character(data$Rain))
data$FFMC = as.numeric(as.character(data$FFMC))
data$DMC = as.numeric(as.character(data$DMC))
data$DC = as.numeric(as.character(data$DC))
data$Classes = as.factor(as.numeric(as.character(data$Classes)))
```

```
str(data)
```

with this function we can look at the structure of our data to get an overview about our future work

```
 'data.frame':    243 obs. of  8 variables:
  $ Temperature: num  29 29 26 25 27 31 33 30 25 28 ...
  $ RH         : num  57 61 82 89 77 67 54 73 88 79 ...
  $ Ws         : num  18 13 22 13 16 14 13 15 13 12 ...
  $ Rain       : num  0 1.3 13.1 2.5 0 0 0 0 0.2 0 ...
  $ FFMC       : num  65.7 64.4 47.1 28.6 64.8 82.6 88.2 86.6 52.9 73.2 ...
  $ DMC        : num  3.4 4.1 2.5 1.3 3 5.8 9.9 12.1 7.9 9.5 ...
  $ DC         : num  7.6 7.6 7.1 6.9 14.2 22.2 30.5 38.3 38.8 46.3 ...
  $ Classes    : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 2 1 1 ...
```

```
head(data,10)
```

```
    Temperature RH Ws Rain FFMC  DMC   DC Classes
 1           29 57 18  0.0 65.7  3.4  7.6       0
 2           29 61 13  1.3 64.4  4.1  7.6       0
 3           26 82 22 13.1 47.1  2.5  7.1       0
 4           25 89 13  2.5 28.6  1.3  6.9       0
 5           27 77 16  0.0 64.8  3.0 14.2       0
 6           31 67 14  0.0 82.6  5.8 22.2       1
 7           33 54 13  0.0 88.2  9.9 30.5       1
 8           30 73 15  0.0 86.6 12.1 38.3       1
 9           25 88 13  0.2 52.9  7.9 38.8       0
 10          28 79 12  0.0 73.2  9.5 46.3       0
```

```
summary(data)
```

```
   Temperature          RH              Ws             Rain        
  Min.   :22.00   Min.   :21.00   Min.   : 6.00   Min.   : 0.000  
  1st Qu.:30.00   1st Qu.:52.50   1st Qu.:14.00   1st Qu.: 0.000  
  Median :32.00   Median :63.00   Median :15.00   Median : 0.000  
  Mean   :32.15   Mean   :62.04   Mean   :15.49   Mean   : 0.763  
  3rd Qu.:35.00   3rd Qu.:73.50   3rd Qu.:17.00   3rd Qu.: 0.500  
  Max.   :42.00   Max.   :90.00   Max.   :29.00   Max.   :16.800  
       FFMC            DMC              DC          Classes  
  Min.   :28.60   Min.   : 0.70   Min.   :  6.90   0:106  
  1st Qu.:71.85   1st Qu.: 5.80   1st Qu.: 12.35   1:137  
  Median :83.30   Median :11.30   Median : 33.10          
  Mean   :77.84   Mean   :14.68   Mean   : 49.43          
  3rd Qu.:88.30   3rd Qu.:20.80   3rd Qu.: 69.10          
  Max.   :96.00   Max.   :65.90   Max.   :220.40
```

The command summary() returns us a first descriptive analysis of the dataset, highlighting for the qualitative values the absolute frequencies and the quantitative variables descriptive indices as Minimum, Maximum, Median, Mean, and Quartiles. From this preliminary analysis, we can see how all the variables have a different range of values and magnitudes, and how for further and more in-depth analysis it will be necessary to standardize and scale the data.

UNIVARIATE ANALYSIS

In this first part of the report, it is going to be analyzed each variable of the dataset, taken individually. In detail, we will implement an analysis of the frequency distributions by computing the absolute and the relative frequencies for each variable.

1)Classes: it is a categorical variable that can assume only values 0 or 1, 0 in case of not fire and 1 in case of fire.

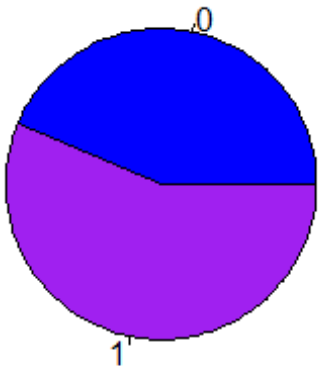
```
table(data$Classes)
```

```
  0   1
106 137
```

By the use of function table we can see how many fires occurs.

We can use also a pieplot to give us a graphical idea of the distribution

```
pie(table(data$Classes), col=c("blue","purple"))
```



2)Temperature: it is a continuous variable, that means that can assume whatever values in an interval, in this case is defined in the interval $0,\infty$. It refers to the total temperature in Celsius Degrees.

```
unique(data$Temperature)
```

```
 [1] 29 26 25 27 31 33 30 28 32 34 35 36 37 22 24 38 39 40 42
```

```
length(unique(data$Temperature))
```

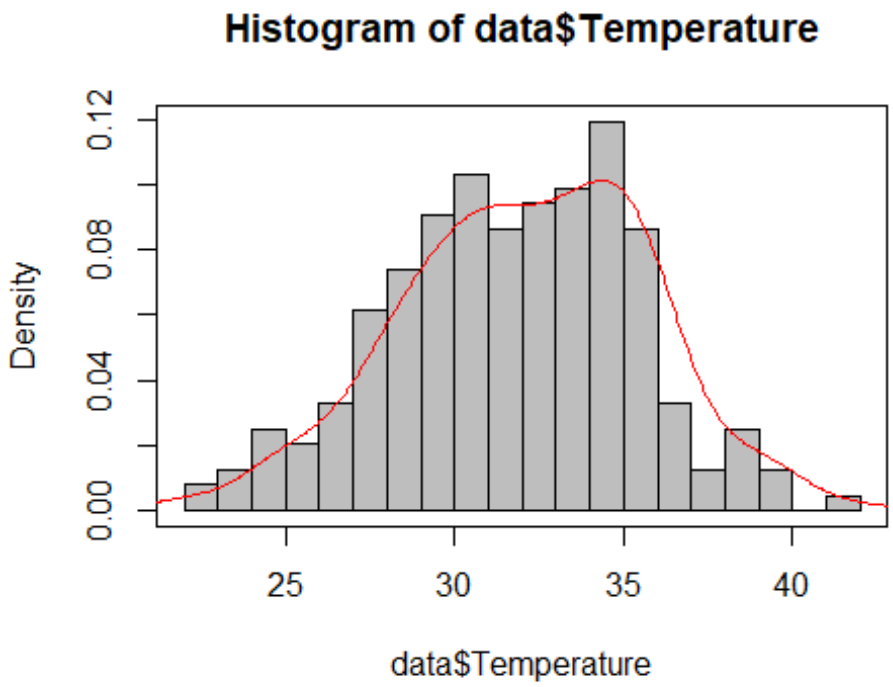
```
 [1] 19
```

The variable takes 19 unique modalities. In the following code are computed the absolute frequencies and represented with an histogram.

```
table(data$Temperature)
```

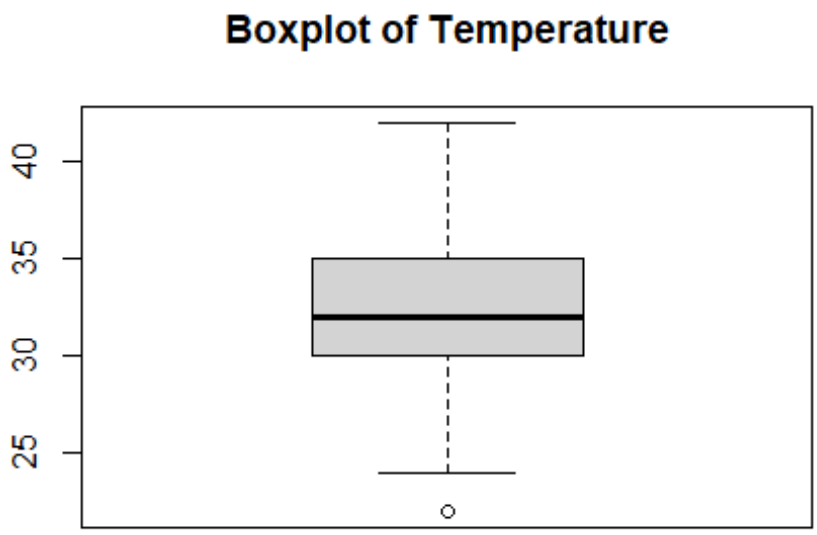
```
 22 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 42
  2  3  6  5  8 15 18 22 25 21 23 24 29 21  8  3  6  3  1
```

```
hist(data$Temperature,col = 'grey', breaks = 19 ,freq = FALSE)
box()
lines(density(data$Temperature),col = 'red')
```



In the histogram the red line represents the density of the distribution of the Temperature variable.

```
boxplot(data$Temperature,main = 'Boxplot of Temperature')
```



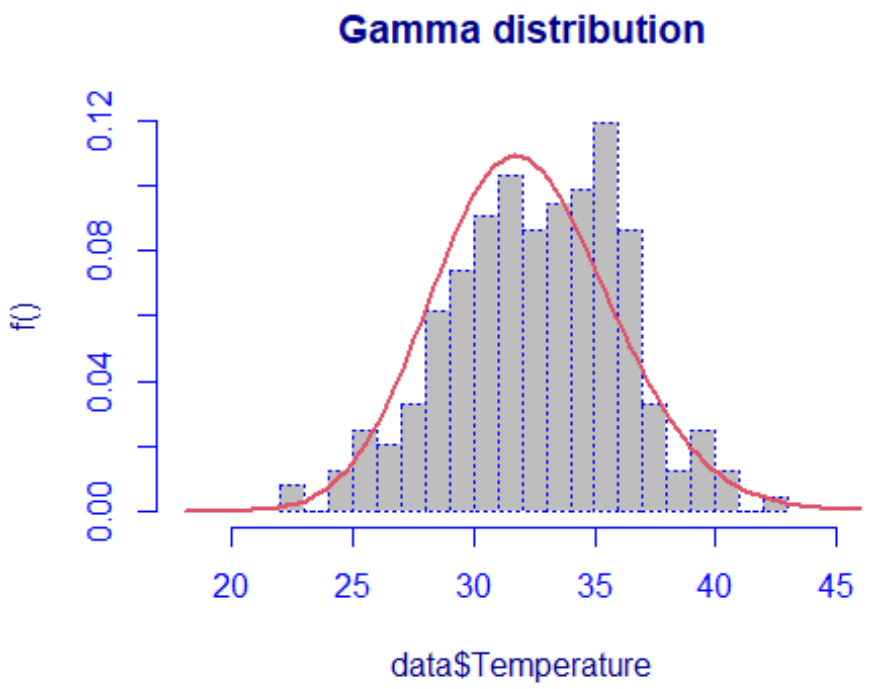
The boxplot gives a clear representation of the descriptive statistics we have obtained through the summary, except for the mean. In particular, the black lines within the box represents the median, the two lines which contours the box represent the first (on the bottom) and the third Quartile (on the top) and finally, the two lines outside the box represent the maximum (on the top) and the minimum values (on the bottom). The distance between the first and the third quartile is defined as the Interquartile range (IQR).

```
kurtosis(data$Temperature)

 [1] 2.836888

skewness(data$Temperature)

 [1] -0.1901443
```

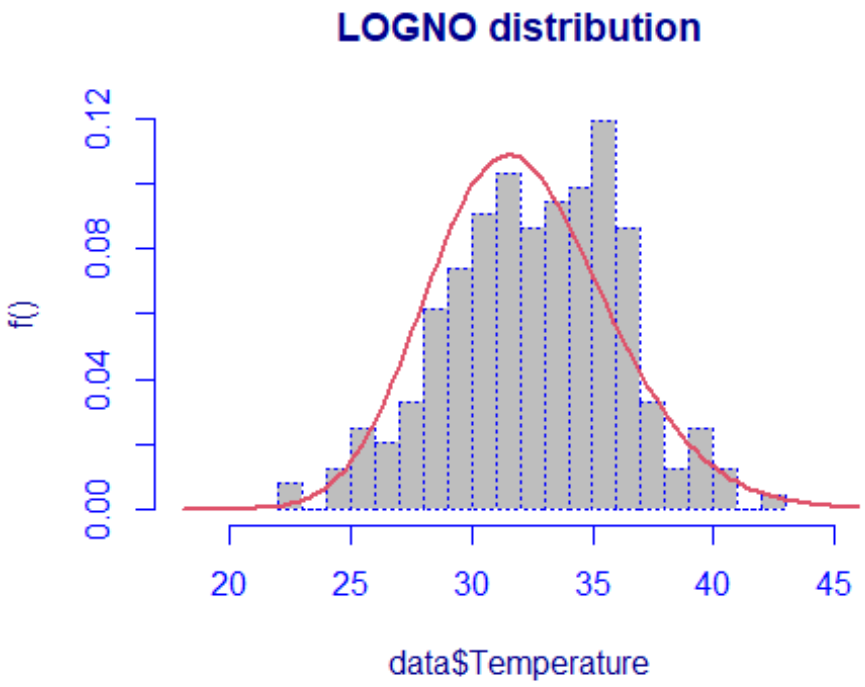
By computing the skewness and the kurtosis, it seems that the Temperature distribution is lowly negative skewed, and the kurtosis' value is greater than one, which means that the distribution is leptokurtic and has heavy tails.

DATA FITTING FOR TEMPERATURE: In the following code, we will try to establish a theoretical model to be fitted with the Temperature variable, by looking at three different criterions: the Log-likelihood value (which must be maximized), the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC) (which must be minimized).

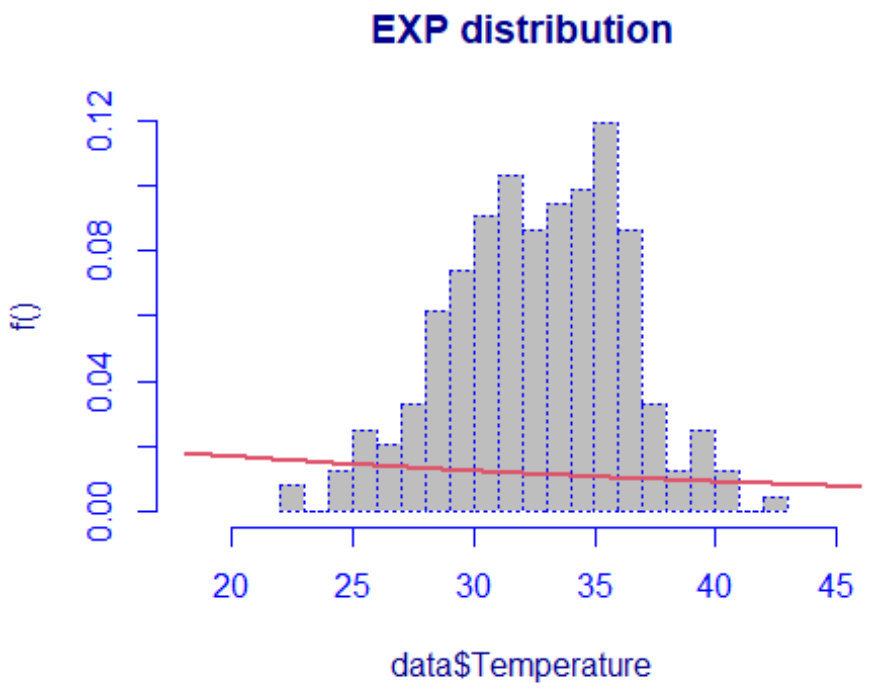
```
fit.GA = histDist(data$Temperature,family = GA, nbins=19, main='Gamma
distribution')
```



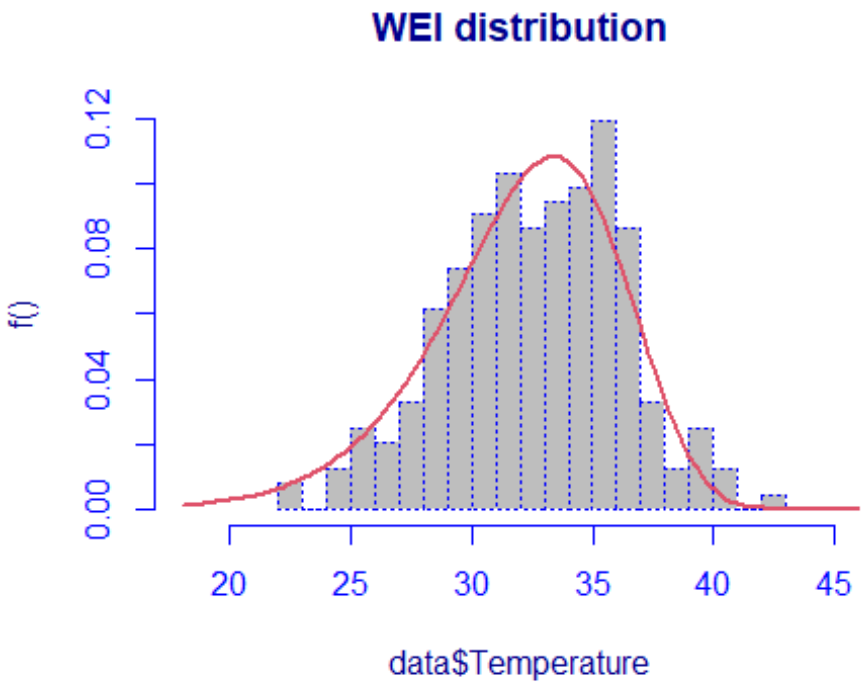

```
logLik(fit.GA)

 'log Lik.' -660.1611 (df=2)

AIC(fit.GA)

 [1] 1324.322

fit.GA$sbc

 [1] 1331.308

fit.LOGNO = histDist(data$Temperature,family = LOGNO, nbins=19, main='LOGNO
distribution')

 Warning in MLE(ll2, start = list(eta.mu = eta.mu, eta.sigma = eta.sigma), :
 possible convergence problem: optim gave code=1 false convergence (8)
```



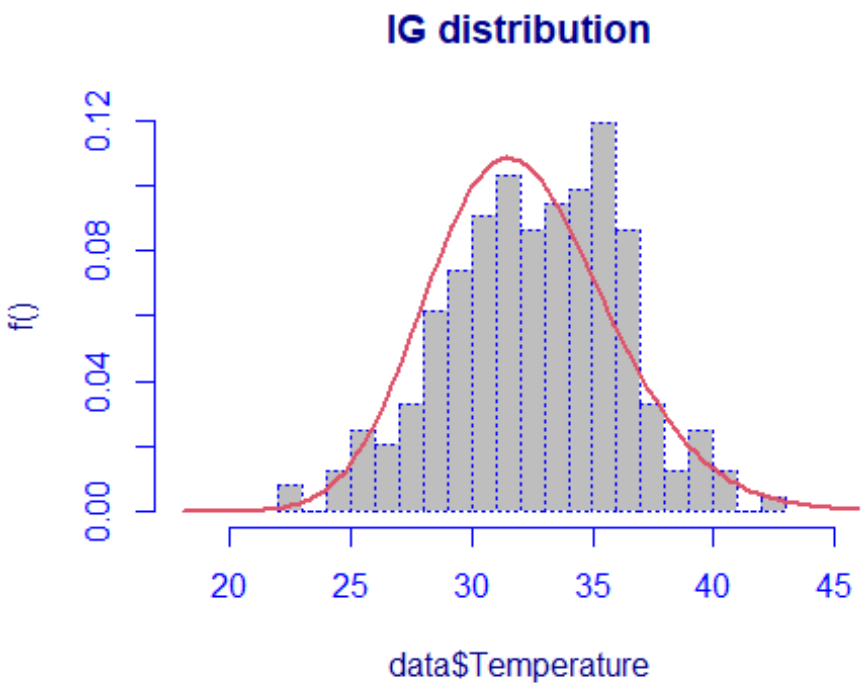

```
logLik(fit.LOGNO)

 'log Lik.' -662.2874 (df=2)

AIC(fit.LOGNO)

 [1] 1328.575

fit.LOGNO$sbc

 [1] 1335.561

fit.EXP = histDist(data$Temperature,family = EXP, nbins=19, main='EXP
distribution')
```




```
logLik(fit.EXP)

 'log Lik.' -1086.327 (df=1)

AIC(fit.EXP)

 [1] 2174.655

fit.EXP$sbc

 [1] 2178.148

fit.WEI = histDist(data$Temperature,family = WEI, nbins=19, main='WEI
distribution')
```



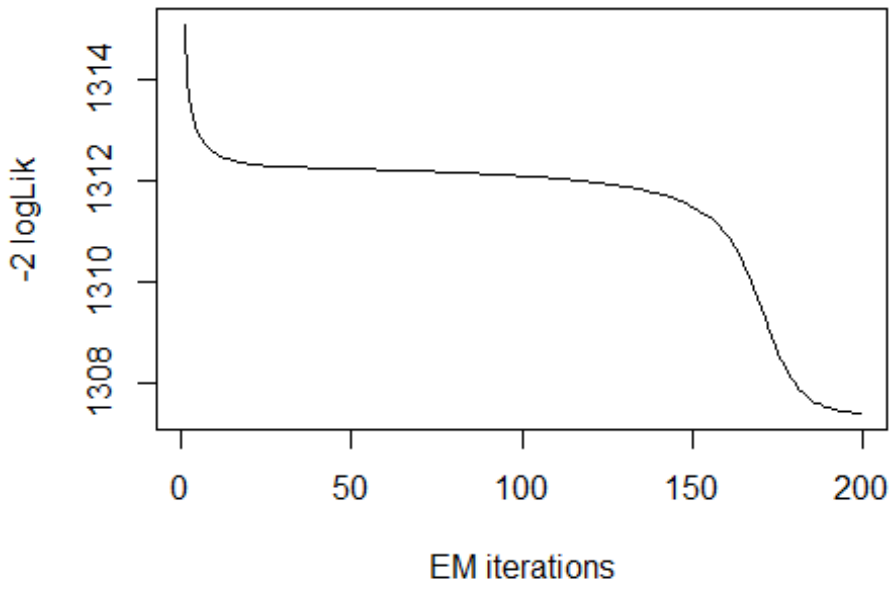

```
logLik(fit.WEI)
 'log Lik.' -660.2984 (df=2)
AIC(fit.WEI)
 [1] 1324.597
fit.WEI$sbc
 [1] 1331.583
fit.IG = histDist(data$Temperature,family = IG, nbins=19, main='IG distribution')
```



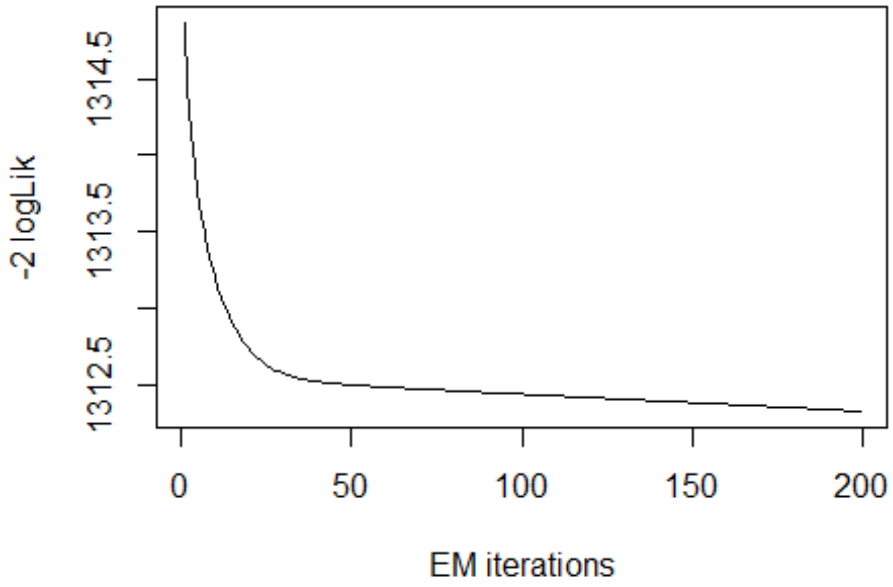

```
logLik(fit.IG)

 'log Lik.' -662.3286 (df=2)

AIC(fit.IG)

 [1] 1328.657

fit.IG$sbc

 [1] 1335.643

Temperature.fitted =
matrix(c(logLik(fit.GA),+AIC(fit.GA),+fit.GA$sbc,+logLik(fit.LOGNO),+AIC(fit.LOGNO
),+fit.LOGNO$sbc,+logLik(fit.EXP),+AIC(fit.EXP),+fit.EXP$sbc,+logLik(fit.WEI),+AIC
(fit.WEI),+fit.WEI$sbc,+logLik(fit.IG),AIC(fit.IG),fit.IG$sbc),nrow=5,ncol=3,byrow
=TRUE)
colnames(Temperature.fitted) = c('Loglikelihood','AIC','BIC')
rownames(Temperature.fitted) = c('GA','LOGNO','EXP','WEI','IG')
Temperature.fitted

       Loglikelihood      AIC      BIC
 GA        -660.1611 1324.322 1331.308
 LOGNO     -662.2874 1328.575 1335.561
 EXP      -1086.3273 2174.655 2178.148
 WEI       -660.2984 1324.597 1331.583
 IG        -662.3286 1328.657 1335.643

By looking at this matrix we can see how we get the best AIC AND BIC values with
Gamma.
```

Mixture of two Gamma distributions : For a further and more complete analysis we are going to compare the result of the previous fitting analysis with the outcome of fitting mixture of two Gamma distributions. The purpose of this analysis is to assess which kind of fitting approach yield the best result. Note that the algorithm will be repeated 5 times in order to have a more stable result.

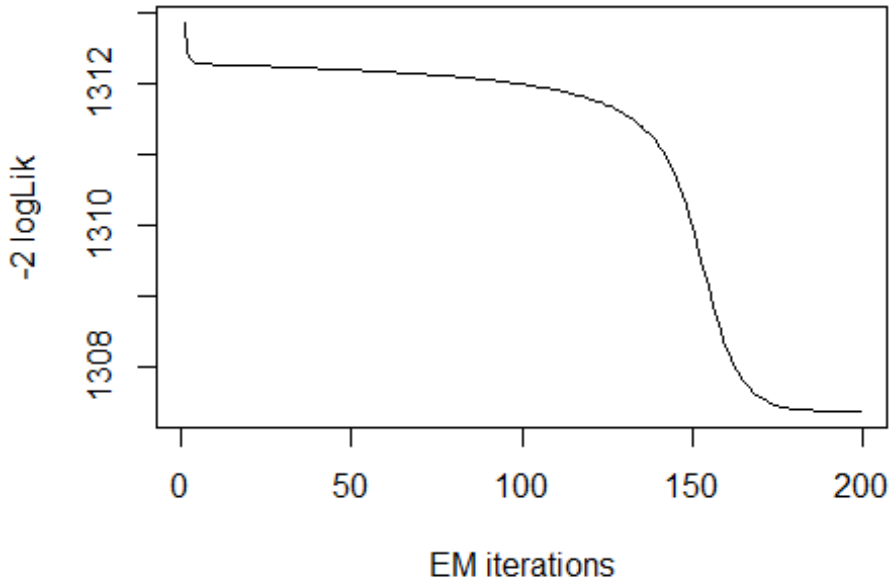
```
fit.GA.2 <- gamlssMXfits(n = 5, data$Temperature~1, family = GA, K = 2, data =
NULL)
```



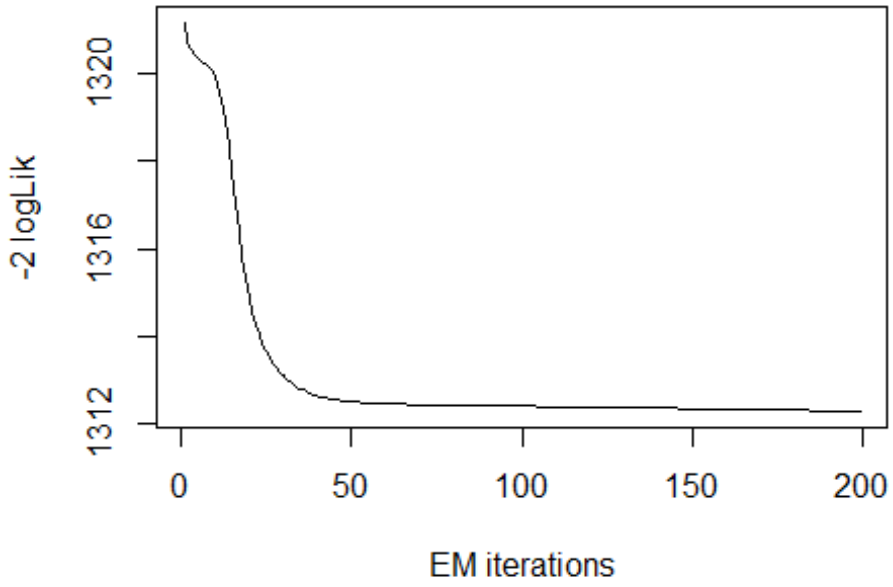

```
model= 1
```



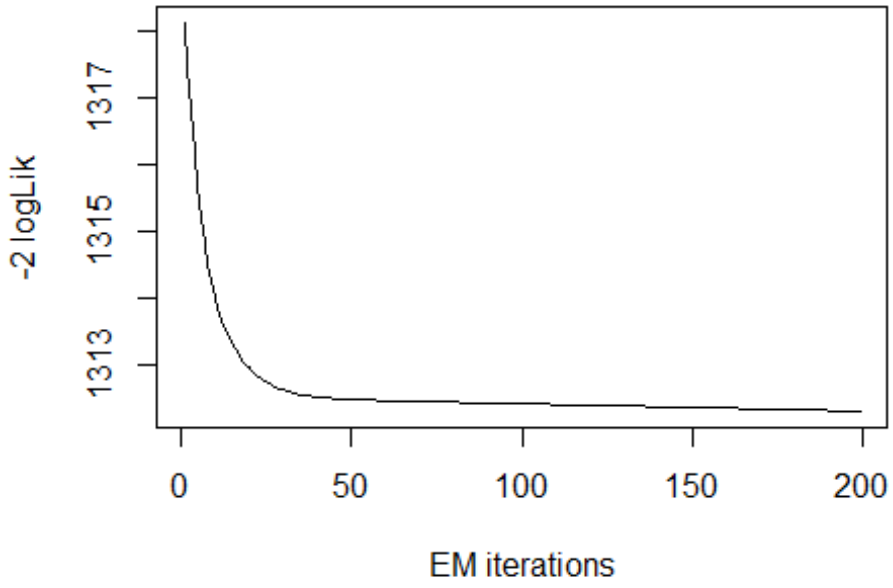

```
model= 2
```




```
model= 3
```

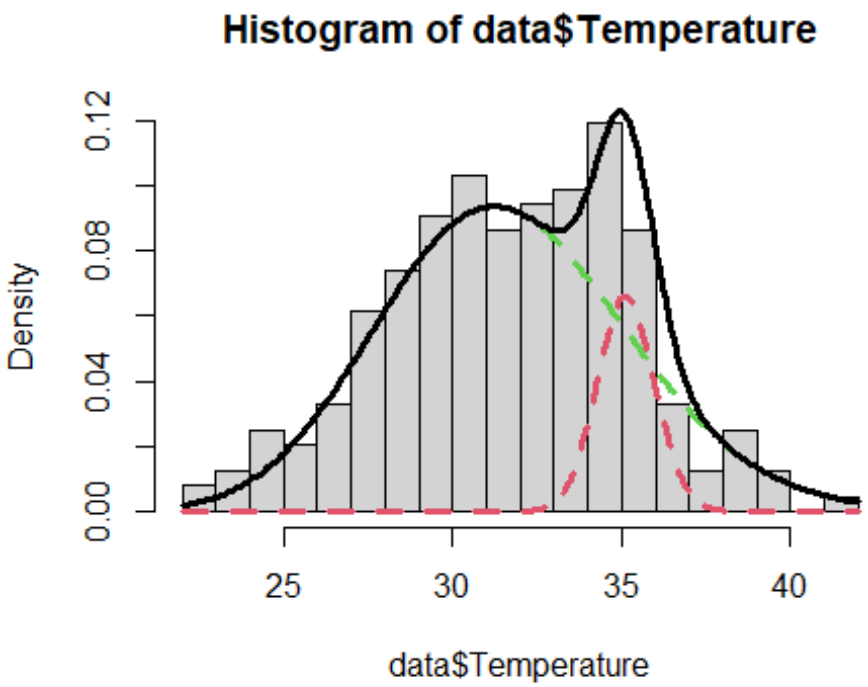



```
model= 4
```




```
model= 5

logLik(fit.GA.2)
```

```
 'log Lik.' -653.6808 (df=5)

fit.GA.2$prob

 [1] 0.1392766 0.8607234

AIC(fit.GA.2)

 [1] 1317.362

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

hist(data$Temperature, breaks = 19,freq = FALSE)
lines(seq(min(data$Temperature),max(data$Temperature),length=length(data$Temperatu
re)),fit.GA.2[["prob"]][1]*dGA(seq(min(data$Temperature),max(data$Temperature),len
gth=length(data$Temperature)), mu = mu.hat1, sigma =
sigma.hat1),lty=2,lwd=3,col=2)
lines(seq(min(data$Temperature),max(data$Temperature),length=length(data$Temperatu
re)),fit.GA.2[["prob"]][2]*dGA(seq(min(data$Temperature),max(data$Temperature),len
gth=length(data$Temperature)), mu = mu.hat2, sigma =
sigma.hat2),lty=2,lwd=3,col=3)
lines(seq(min(data$Temperature),max(data$Temperature),length=length(data$Temperatu
re)),

fit.GA.2[["prob"]][1]*dGA(seq(min(data$Temperature),max(data$Temperature),length=l
ength(data$Temperature)), mu = mu.hat1, sigma = sigma.hat1) +

fit.GA.2[["prob"]][2]*dGA(seq(min(data$Temperature),max(data$Temperature),length=l
ength(data$Temperature)), mu = mu.hat2, sigma = sigma.hat2),
      lty = 1, lwd = 3, col = 1)
```



In the last graph there are three peaks: precisely the red line is the first distribution, the green line is the second distribution, and the black segment is the overall mixture of the two distributions. The first Gamma distribution represents 14% of the mixture while the second one accounts for the 86%.

3) Relative Humidity (RH) : it is also a continuous variable that assume values in [0:100] because it is a %. It measures the level of RH in presence of fire and not fire.

```
unique(data$RH)
```

```
 [1] 57 61 82 89 77 67 54 73 88 79 65 81 84 78 80 55 62 66 64 53 47 50 68 75 76
[26] 63 69 70 59 48 45 60 51 52 58 86 74 71 49 44 41 42 90 87 72 46 37 36 56 43
[51] 83 29 34 33 35 39 31 21 40 24 38 26
```

```
length(unique(data$RH))
```

```
[1] 62
```

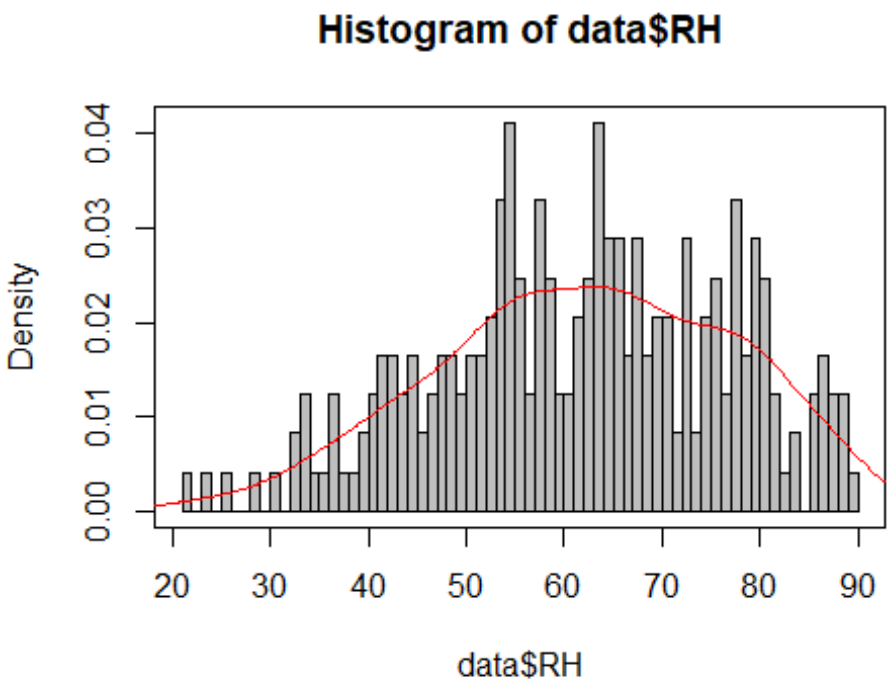
The variable takes 147 different modalities. In the following code are computed the absolute frequencies and represented with a histogram.

```
table(data$RH)
```

```
21 24 26 29 31 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
 1  1  1  1  1  2  3  1  1  3  1  1  2  3  4  4  3  4  2  3  4  4  3  4  4  5
54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
```

```
  8 10  6  3  8  6  3  3  5  6 10  7  7  4  7  4  5  5  2  7  2  5  6  3  8  4
 80 81 82 83 84 86 87 88 89 90
  7  6  3  1  2  3  4  3  3  1
```

```
hist(data$RH, col = 'grey', breaks = 62 ,freq = FALSE)
box()
lines(density(data$RH),col = 'red')
```




```
kurtosis(data$RH)
```

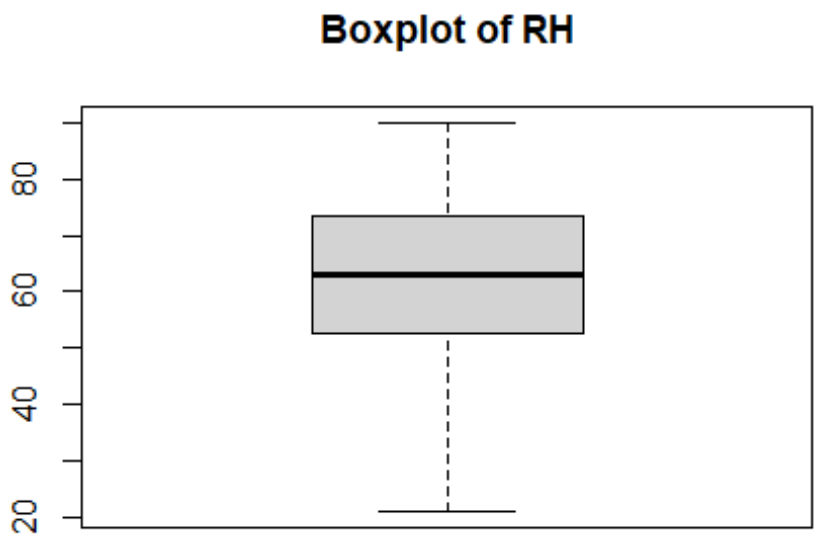
```
 [1] 2.476879
```

```
skewness(data$RH)
```

```
 [1] -0.2412892
```

By computing the skewness and the kurtosis, it seems that the RH distribution is lowly negative skewed, and the kurtosis' value is greater than one, which means that the distribution is leptokurtic and has heavy tails.

```
boxplot(data$RH,main = 'Boxplot of RH')
```

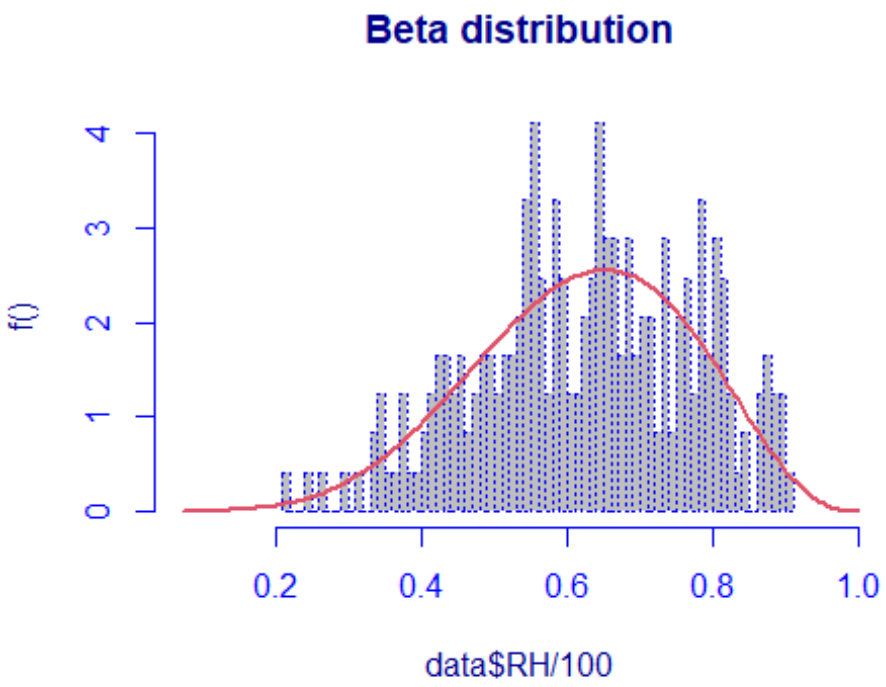


By looking at the median of this boxplot we can see that has a similar distance between the first and the third quartile and we can also see and say that there are no outliers.

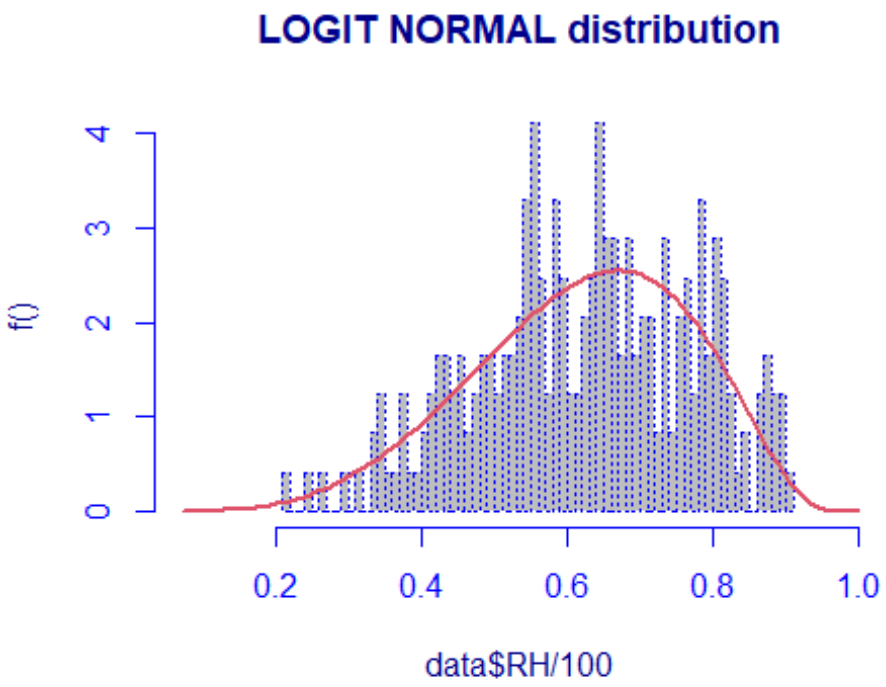
In the following code, we will try to establish a theoretical model to be fitted with the RH variable, by looking at three different criteria: the Log-likelihood value (which must be maximized), the Akaike Information Criterion (AIC), and the Bayesian Information criterion (BIC) (which must be minimized).

DATA FITTING FOR RH: in this case we have a variable with support [0:100] so we need to consider different distributions in order to fit our variable.

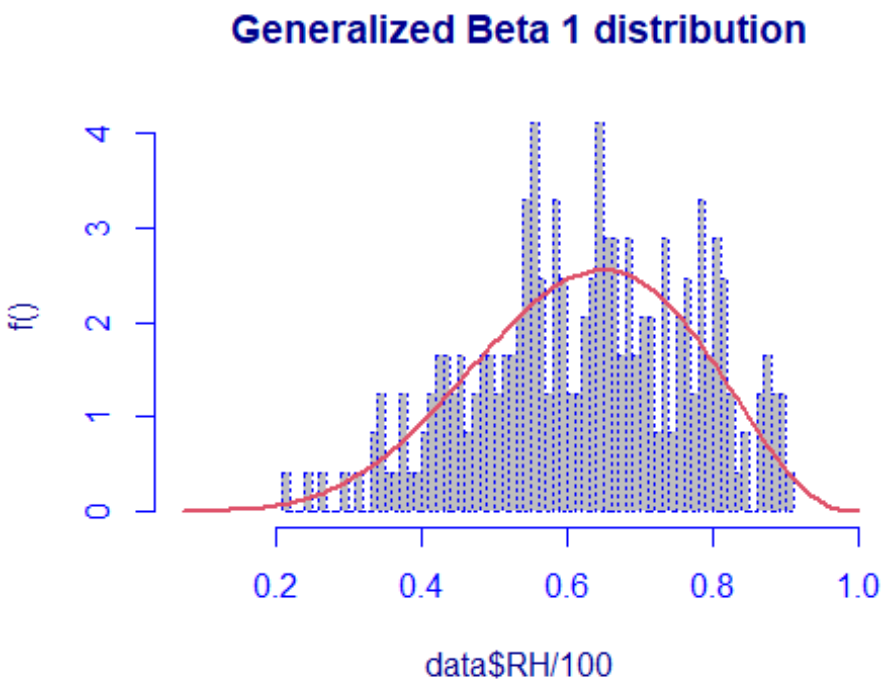
```
fit.BE = histDist(data$RH/100,family = BE, nbins=62, main='Beta distribution')
```




```
logLik(fit.BE)

 'log Lik.' 125.3712 (df=2)

AIC(fit.BE)

 [1] -246.7424

fit.BE$sbc

 [1] -239.7563

fit.LOGITNO = histDist(data$RH/100,family = LOGITNO, nbins=62, main='LOGIT NORMAL
distribution')
```




```
logLik(fit.LOGITNO)

 'log Lik.' 125.7825 (df=2)

AIC(fit.LOGITNO)

 [1] -247.5649

fit.LOGITNO$sbc

 [1] -240.5788

fit.GB1 = histDist(data$RH/100,family = GB1, nbins=62, main='Generalized Beta 1
distribution')
```

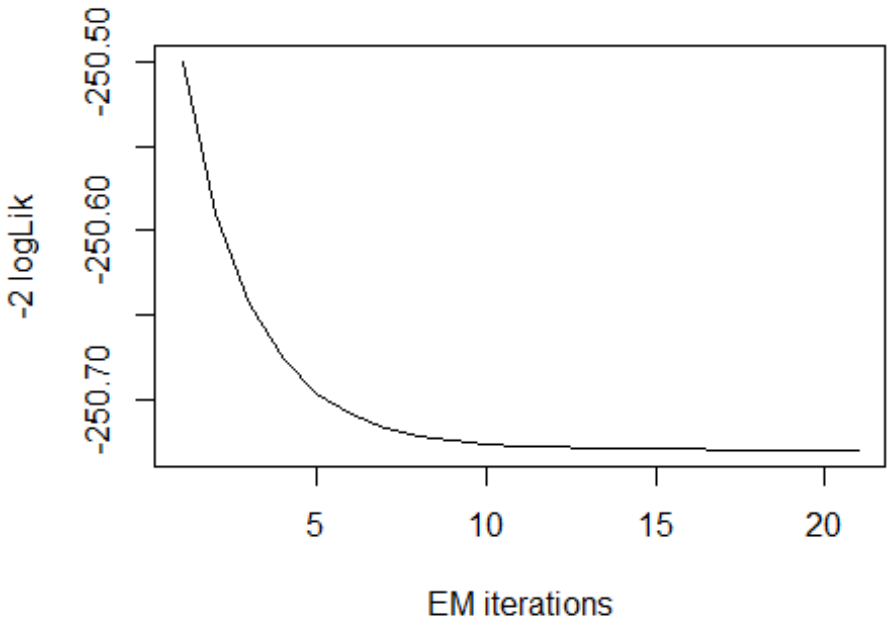



```
logLik(fit.GB1)

 'log Lik.' 125.3783 (df=4)

AIC(fit.GB1)

 [1] -242.7566

fit.GB1$sbc

 [1] -228.7843

RH.fitted =
matrix(c(logLik(fit.BE),+AIC(fit.BE),+fit.BE$sbc,+logLik(fit.LOGITNO),+AIC(fit.LOG
ITNO),+fit.LOGITNO$sbc,+logLik(fit.GB1),+AIC(fit.GB1),+fit.GB1$sbc),nrow=3,ncol=3,
byrow=TRUE)
colnames(RH.fitted) = c('Loglikelihood','AIC','BIC')
rownames(RH.fitted) = c('BE','LOGITNO','GB1')
RH.fitted

          Loglikelihood       AIC       BIC
 BE           125.3712 -246.7424 -239.7563
 LOGITNO      125.7825 -247.5649 -240.5788
 GB1          125.3783 -242.7566 -228.7843

By looking at the matrix we can see how the LOGITNO has the lowest and so bests
values for AIC and BIC.
```

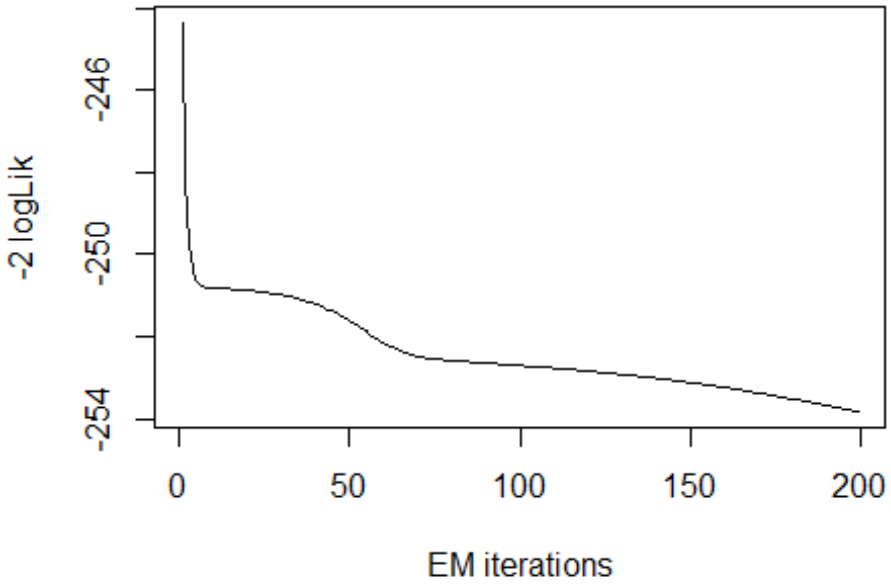
RH Mixture of 2 Beta distributions: For a further and more complete analysis we are going to compare the result of the previous fitting analysis with the outcome of fitting mixture of two

Beta distributions. The purpose of this analysis is to assess which kind of fitting approach yield the best result. Note that the algorithm will be repeated 5 times in order to have a more stable result

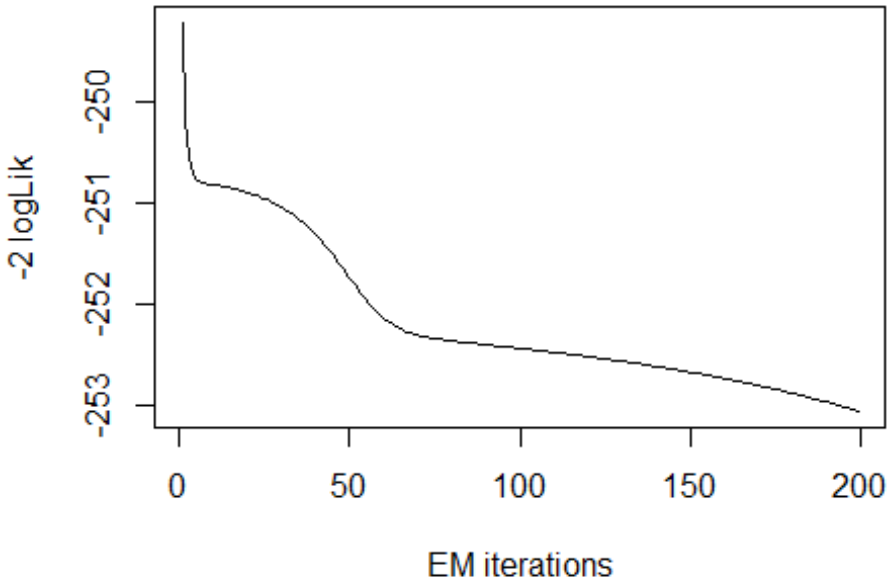
```
fit.BE.2 <- gamlssMXfits(n = 5, data$RH/100~1, family = BE, K = 2, data = NULL)
```



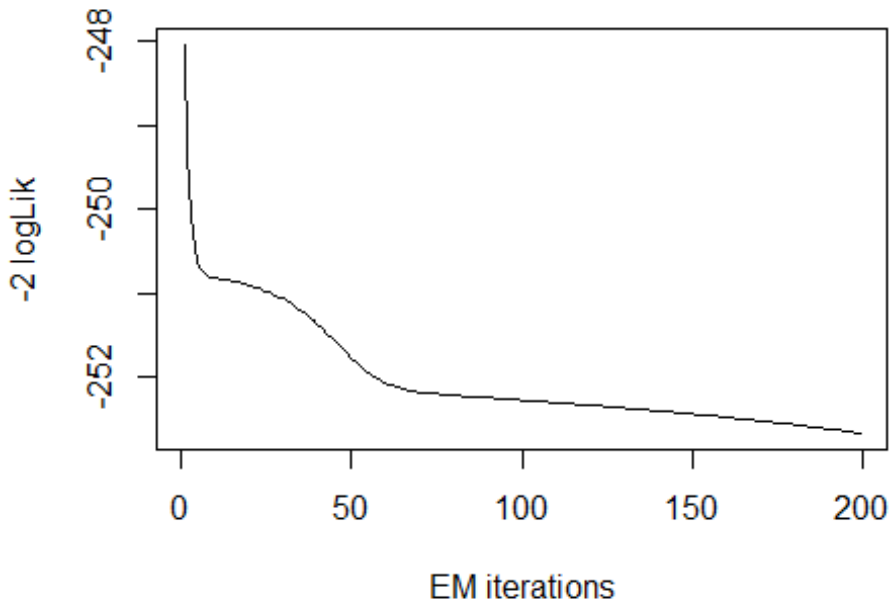

```
model= 1
```



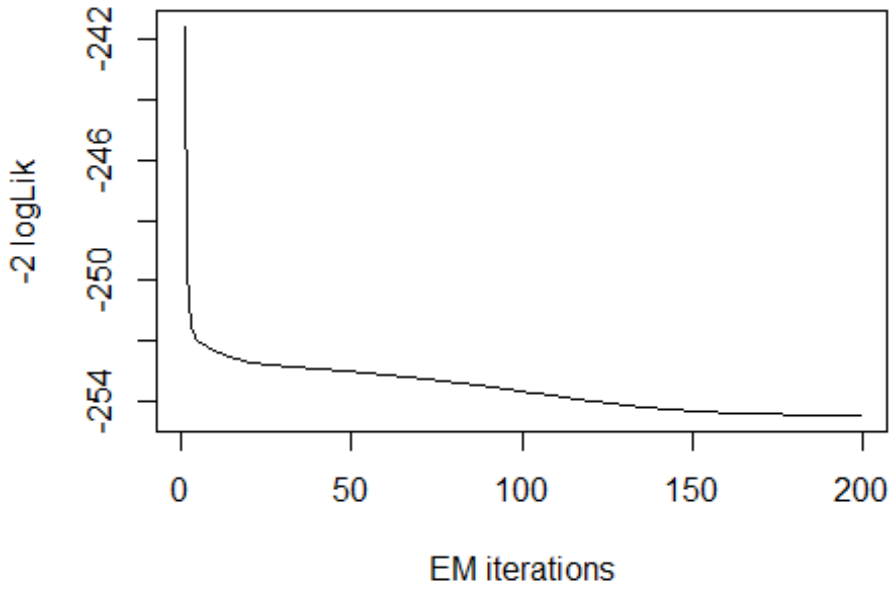

```
model= 2
```




```
model= 3
```

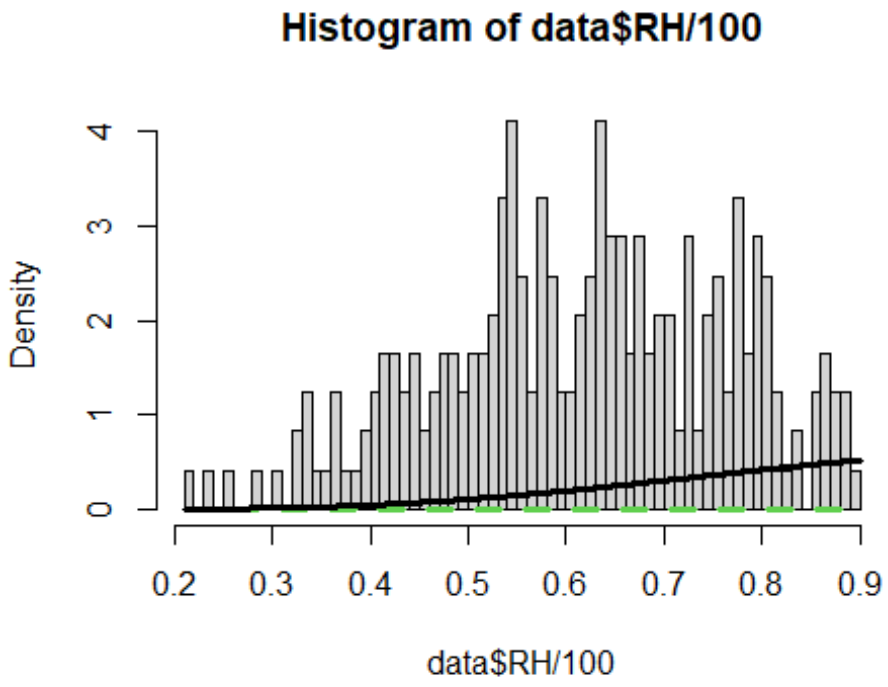



```
model= 4
```




```
 model= 5

logLik(fit.BE.2)
```

```
 'log Lik.' 127.2267 (df=5)

mu.hat1 <- exp(fit.BE.2[["models"]][[1]][["mu.coefficients"]])
sigma.hat1 <- exp(fit.BE.2[["models"]][[1]][["sigma.coefficients"]])
mu.hat2 <- exp(fit.BE.2[["models"]][[2]][["mu.coefficients"]])
sigma.hat2 <- exp(fit.BE.2[["models"]][[2]][["sigma.coefficients"]])

hist(data$RH/100, breaks = 62,freq = FALSE)
lines(seq(min(data$RH/100),max(data$RH/100),length=length(data$RH/100)),fit.BE.2[[
"prob"]][1]*dGA(seq(min(data$RH/100),max(data$RH/100),length=length(data$RH/100)),
mu = mu.hat1, sigma = sigma.hat1),lty=2,lwd=3,col=2)
lines(seq(min(data$RH/100),max(data$RH/100),length=length(data$RH/100)),fit.BE.2[[
"prob"]][2]*dGA(seq(min(data$RH/100),max(data$RH/100),length=length(data$RH/100)),
mu = mu.hat2, sigma = sigma.hat2),lty=2,lwd=3,col=3)
lines(seq(min(data$RH/100),max(data$RH/100),length=length(data$RH/100)),

fit.BE.2[["prob"]][1]*dGA(seq(min(data$RH/100),max(data$RH/100),length=length(data
$RH/100)), mu = mu.hat1, sigma = sigma.hat1) +

fit.BE.2[["prob"]][2]*dGA(seq(min(data$RH/100),max(data$RH/100),length=length(data
$RH/100)), mu = mu.hat2, sigma = sigma.hat2),
      lty = 1, lwd = 3, col = 1)
```



4) Wind Speed: it is a continuous variable that as a support [0: ∞], let's see how it distributed. It refers to the intensity of wind during all the days.

```
unique(data$Ws)

 [1] 18 13 22 16 14 15 12 19 21 20 17 26 11 10  9  8  6 29
```

```
length(unique(data$Ws))
```

```
 [1] 18
```

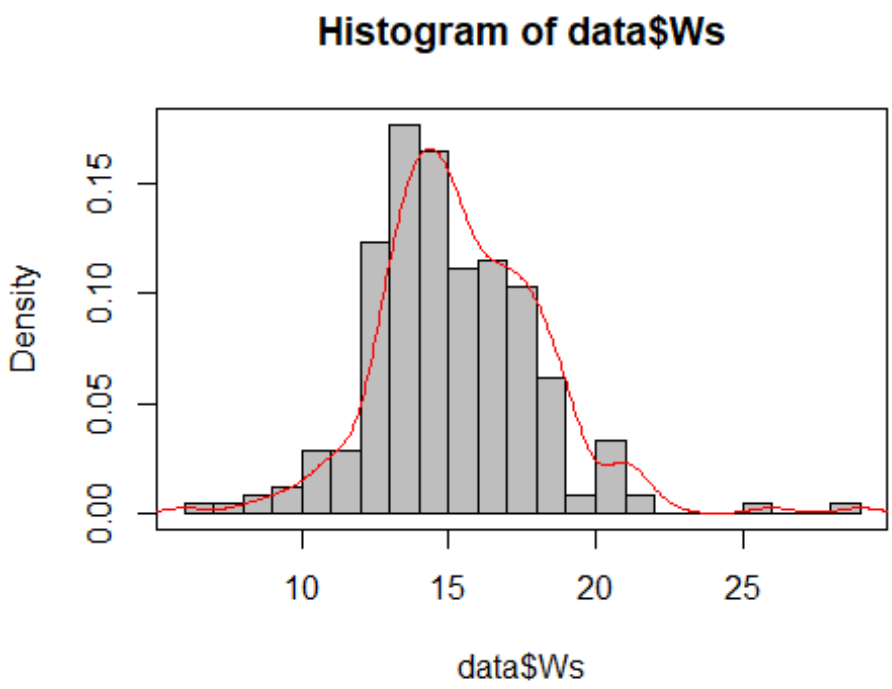
Computing the length of unique we can see the modalities that our variable assume and in this case are 18.

Let’s see the frequency and plot an hist to give us a graphical rappresentation.

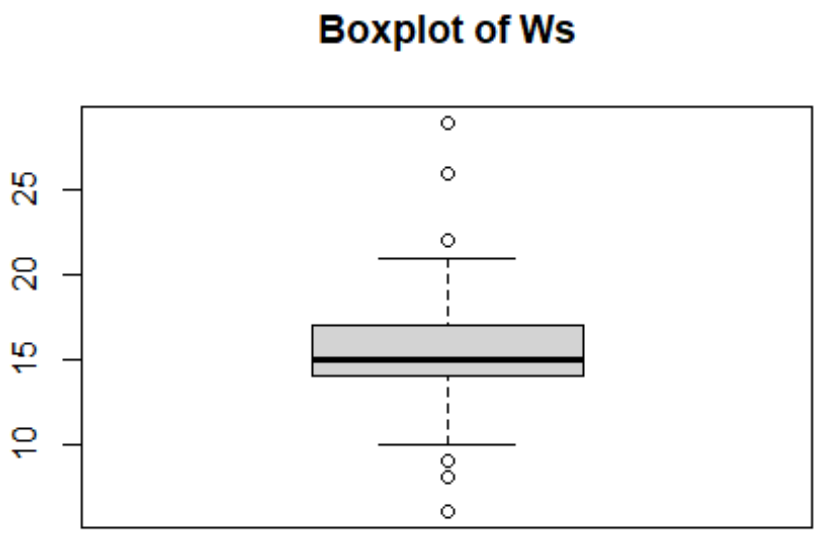
```
table(data$Ws)
```

```
  6   8   9 10 11 12 13 14 15 16 17 18 19 20 21 22 26 29
  1   1   2   3   7   7 30 43 40 27 28 25 15   2   8   2   1   1
```

```
hist(data$Ws,col = 'grey', breaks = 18 ,freq = FALSE)
box()
lines(density(data$Ws),col = 'red')
```




```
boxplot(data$Ws, main='Boxplot of Ws')
```

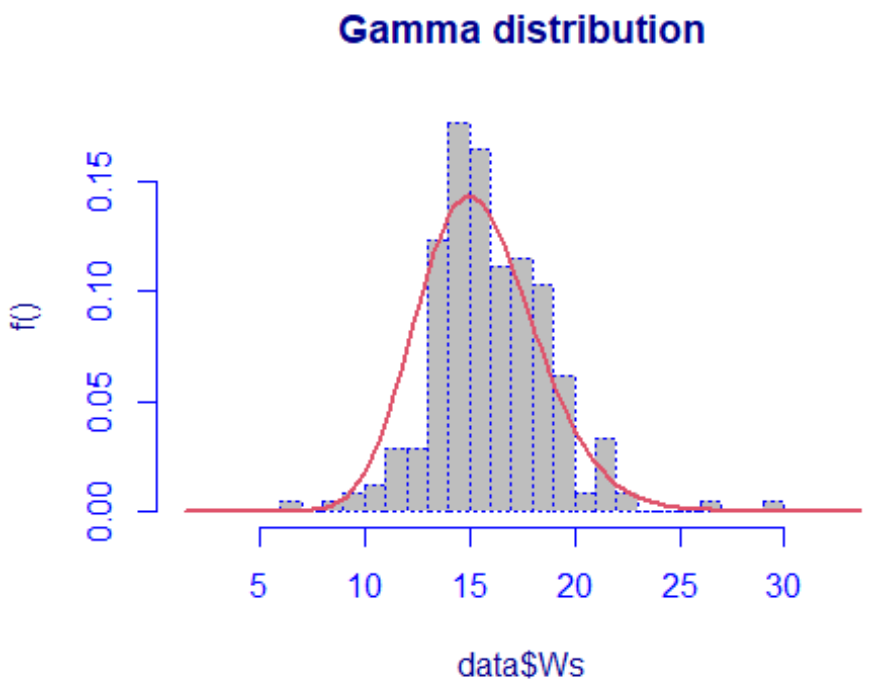



```
By looking at this boxplot we can see how there are some outliers and how first
quartile is closer to the median than the third quartile.

kurtosis(data$Ws)

 [1] 5.543479

skewness(data$Ws)

 [1] 0.5521504
```

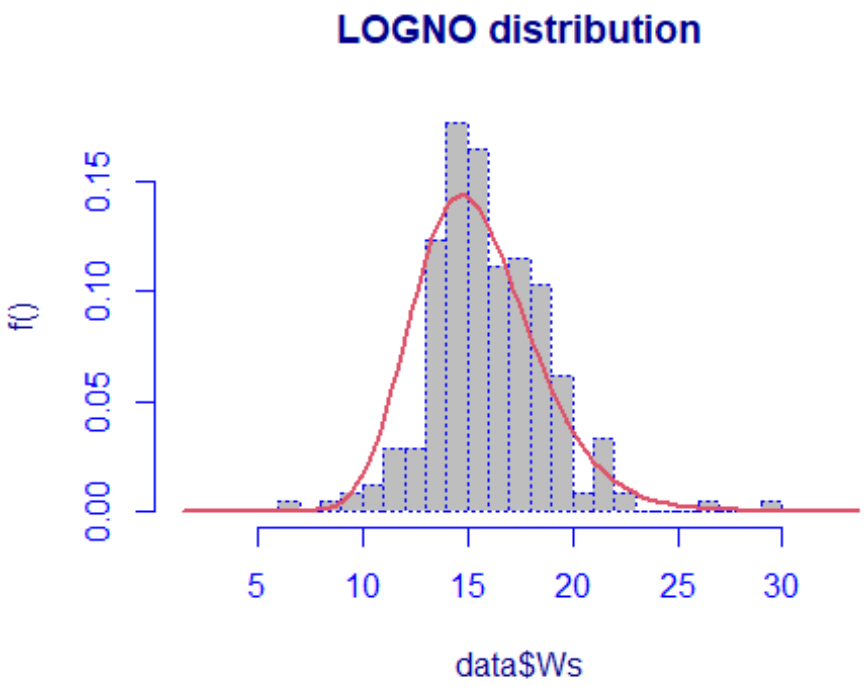
By computing the skewness and the kurtosis, it seems that the Ws distribution is lowly positive skewed, and the kurtosis' value is greater than one, which means that the distribution is leptokurtic and has heavy tails.

DATA FITTING FOR WS: In the following code, we will try to establish a theoretical model to be fitted with the Ws variable, by looking at three different criterions: the Log-likelihood value (which must be maximized), the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC) (which must be minimized).

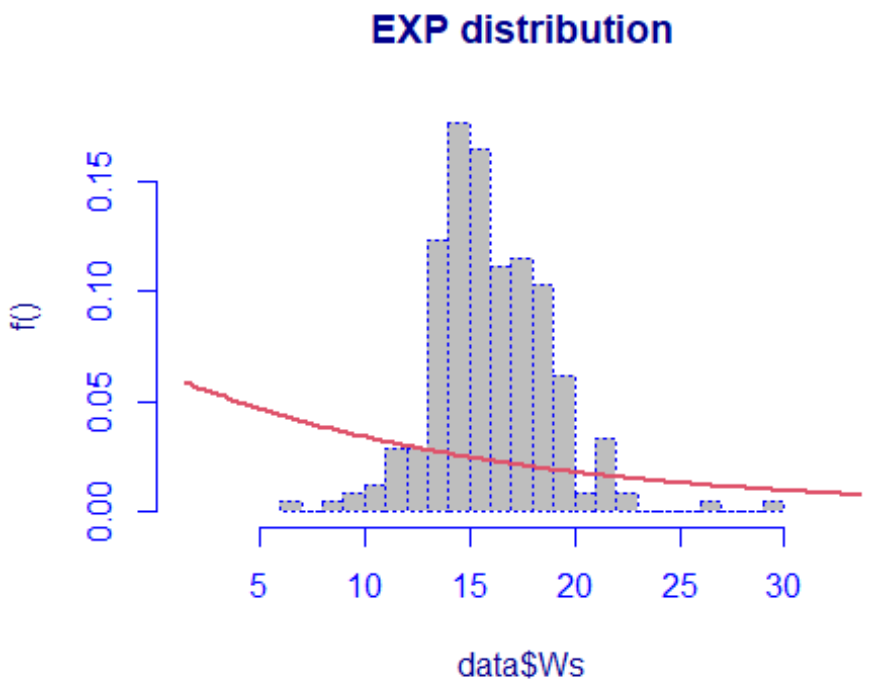
```
fit.GA = histDist(data$Ws,family = GA, nbins=18, main='Gamma distribution')
```



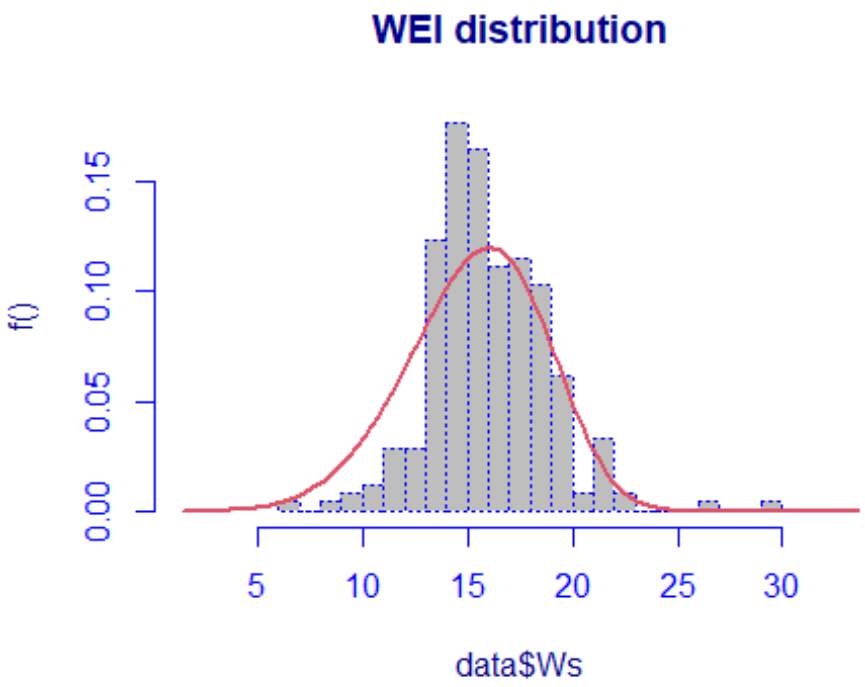

```
logLik(fit.GA)
 'log Lik.' -594.2088 (df=2)
AIC(fit.GA)
 [1] 1192.418
fit.GA$sbc
 [1] 1199.404
fit.LOGNO = histDist(data$Ws,family = LOGNO , nbins=18, main='LOGNO distribution')
```



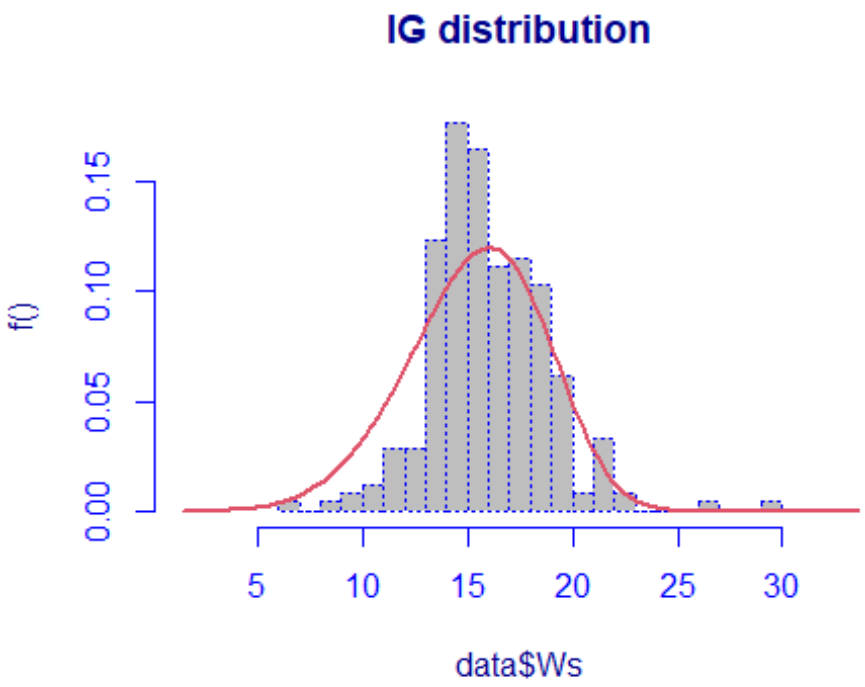

```
logLik(fit.LOGNO)
 'log Lik.' -597.1655 (df=2)
AIC(fit.LOGNO)
 [1] 1198.331
fit.LOGNO$sbc
 [1] 1205.317
fit.EXP = histDist(data$Ws,family = EXP , nbins=18, main='EXP distribution')
```




```
logLik(fit.EXP)
 'log Lik.' -908.9273 (df=1)
AIC(fit.EXP)
 [1] 1819.855
fit.EXP$sbc
 [1] 1823.348
fit.WEI = histDist(data$Ws,family = WEI , nbins=18, main='WEI distribution')
```




```
logLik(fit.WEI)
 'log Lik.' -613.4149 (df=2)
AIC(fit.WEI)
 [1] 1230.83
fit.WEI$sbc
 [1] 1237.816
fit.IG = histDist(data$Ws,family = WEI , nbins=18, main='IG distribution')
```




```
logLik(fit.IG)

 'log Lik.' -613.4149 (df=2)

AIC(fit.IG)

 [1] 1230.83

fit.IG$sbc

 [1] 1237.816

Ws.fitted =
matrix(c(logLik(fit.GA),+AIC(fit.GA),+fit.GA$sbc,+logLik(fit.LOGNO),+AIC(fit.LOGNO
),+fit.LOGNO$sbc,+logLik(fit.EXP),+AIC(fit.EXP),+fit.EXP$sbc,+logLik(fit.WEI),+AIC
(fit.WEI),+fit.WEI$sbc,+logLik(fit.IG),AIC(fit.IG),fit.IG$sbc),nrow=5,ncol=3,byrow
=TRUE)
colnames(Ws.fitted) = c('Loglikelihood','AIC','BIC')
rownames(Ws.fitted) = c('GA','LOGNO','EXP','WEI','IG')
Ws.fitted

       Loglikelihood      AIC      BIC
 GA        -594.2088 1192.418 1199.404
 LOGNO     -597.1655 1198.331 1205.317
 EXP       -908.9273 1819.855 1823.348
 WEI       -613.4149 1230.830 1237.816
 IG        -613.4149 1230.830 1237.816

According to theory the best model in this case is the Gamma, because it has the
lowest level of AIC and BIC.
```

5) Rain: it refers to the mm of rain of everyday,it is also a continuous variable, and has a support [0: ∞]

```
unique(data$Rain)
```

```
  [1]  0.0  1.3 13.1  2.5  0.2  1.2  0.5  3.1  0.7  0.6  0.3  0.1  0.4  1.0  1.4
 [16]  0.8 16.8  7.2 10.1  3.8  0.9  1.8  4.6  8.3  5.8  4.0  2.0  4.7  8.7  4.5
 [31]  1.1  1.7  2.2  6.0  1.9  2.9  4.1  6.5  4.4
```

```
length(unique(data$Rain))
```

```
 [1] 39
```

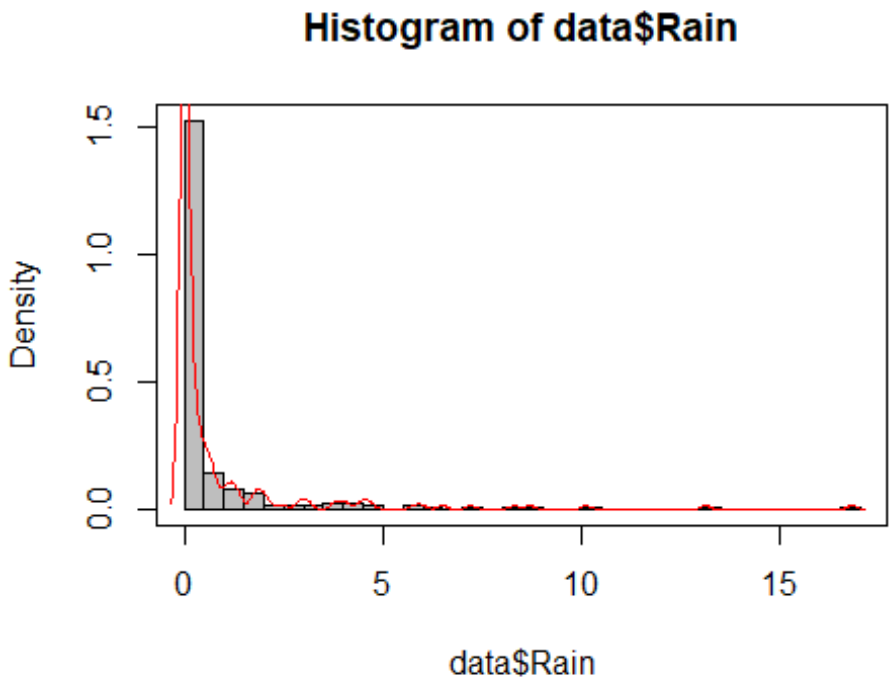
By looking at length function we can see that our variable assume 39 modalities.

Now let's look at frequency and plot them into a graphic with hist() function.

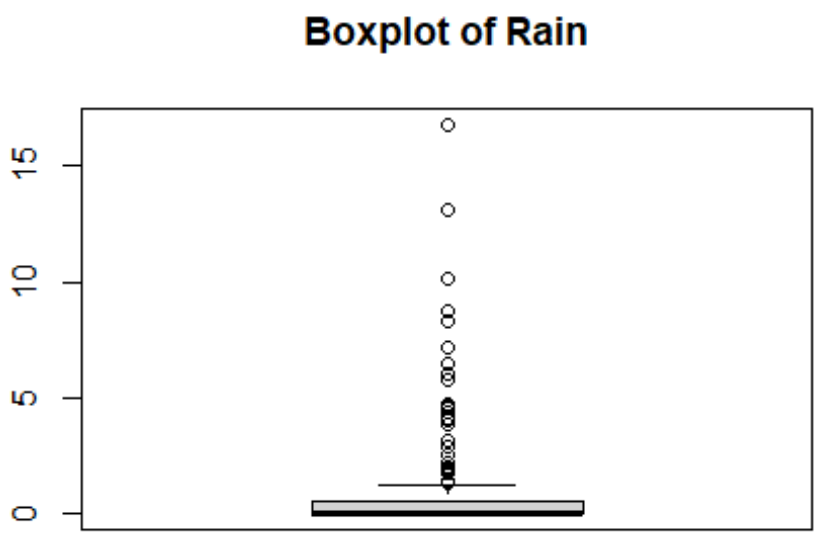
```
table(data$Rain)
```

```
   0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9    1  1.1  1.2  1.3  1.4  1.7
 133   18   11   10    8    5    6    6    2    1    2    3    3    2    2    1
 1.8  1.9    2  2.2  2.5  2.9  3.1  3.8    4  4.1  4.4  4.5  4.6  4.7  5.8    6
   3    1    3    1    1    2    2    2    1    1    1    1    1    1    1    1
 6.5  7.2  8.3  8.7 10.1 13.1 16.8
   1    1    1    1    1    1    1
```

```
hist(data$Rain,col = 'grey', breaks = 39 ,freq = FALSE)
box()
lines(density(data$Rain),col = 'red')
```



```
boxplot(data$Rain, main='Boxplot of Rain')
```



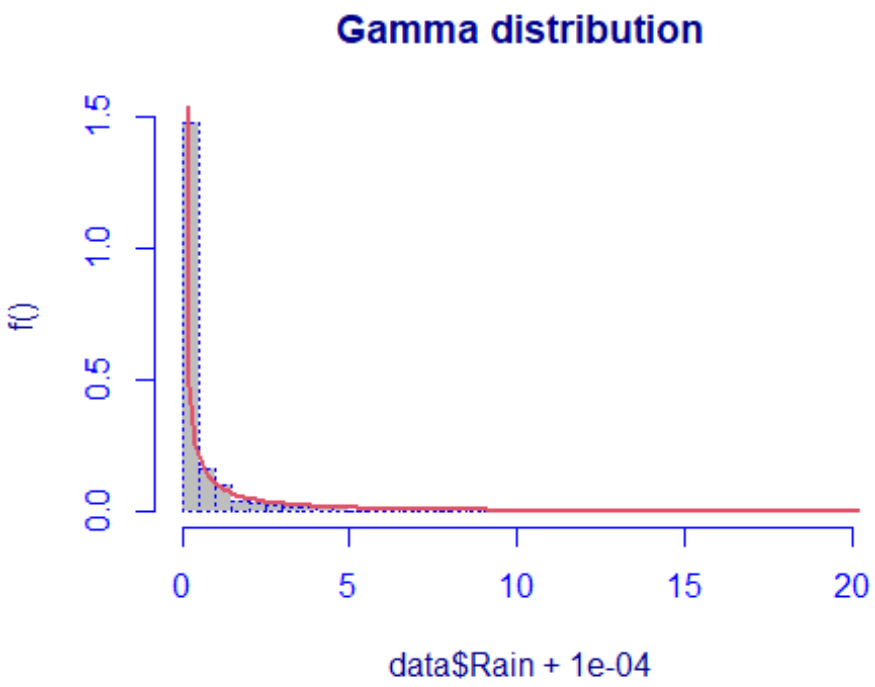

```
By looking at this boxplot we can say that median coincide with the first quartile
and we also have a lot of outliers, so we imagine that mean and median will be not
the same.

kurtosis(data$Rain)

 [1] 28.27011

skewness(data$Rain)

 [1] 4.54038
```

By computing the skewness and the kurtosis, it seems that the Rain distribution is highly positive skewed, and the kurtosis' value is greater than one, which means that the distribution is leptokurtic and has heavy tails.

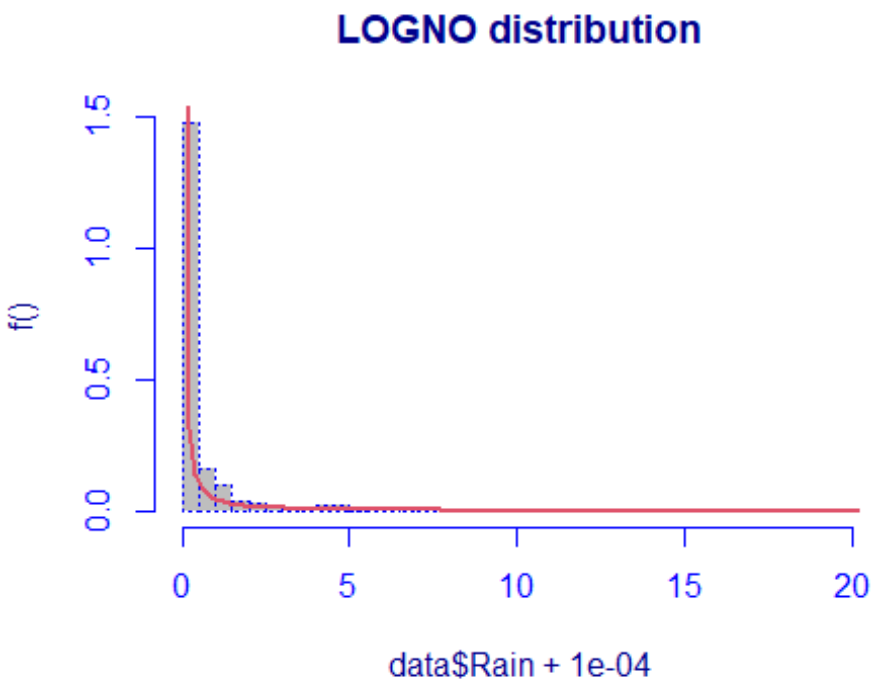
DATA FITTING FOR RAIN:

in the following code, we will try to establish a theoretical model to be fitted with the Rain variable, by looking at three different criterions: the Log-likelihood value (which must be maximized), the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC) (which must be minimized).

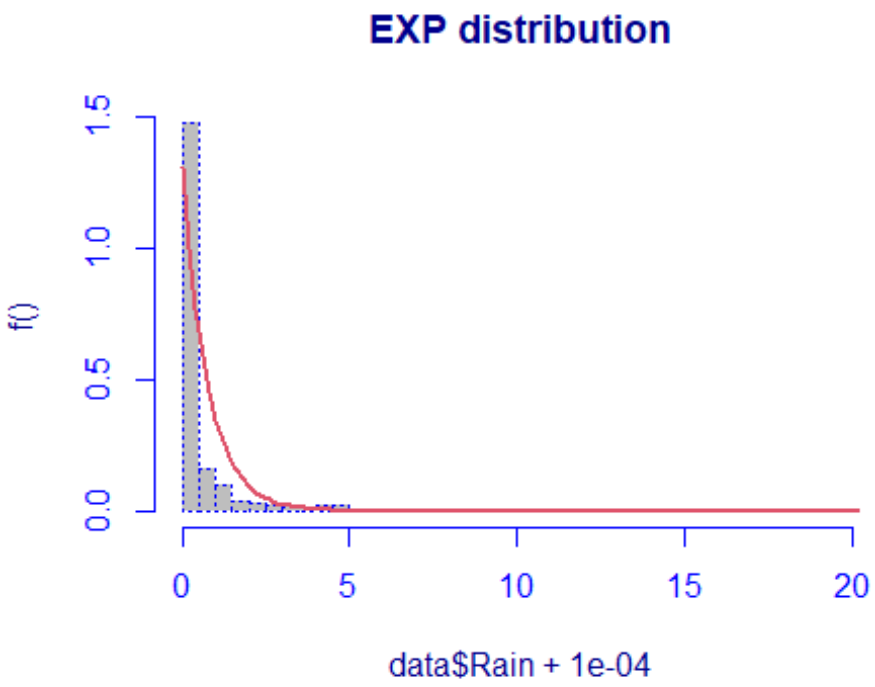
```
fit.GA = histDist(data$Rain+0.0001,family = GA , nbins=39, main='Gamma
distribution')
```



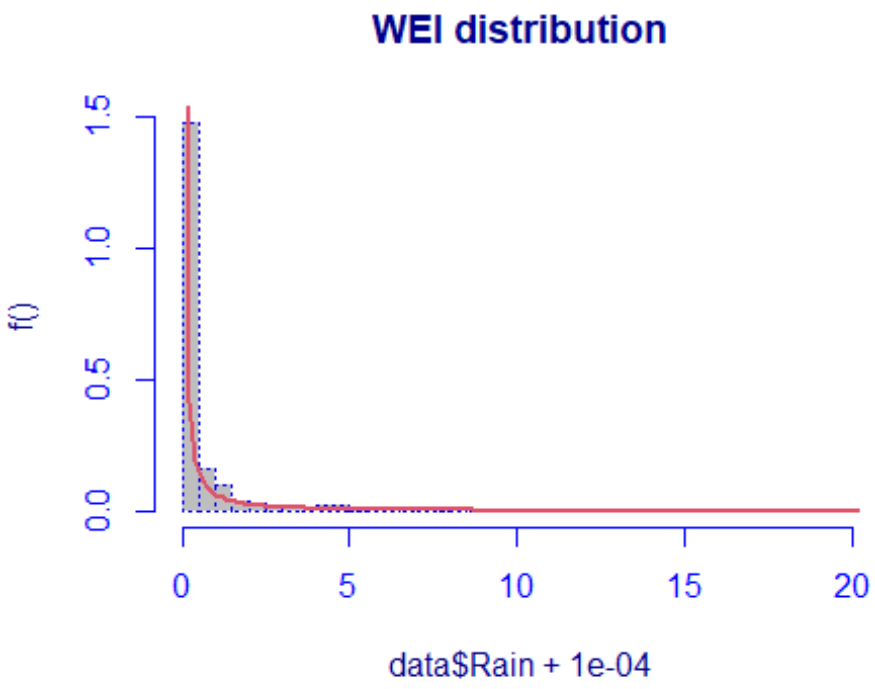

```
logLik(fit.GA)

 'log Lik.' 538.5315 (df=2)

AIC(fit.GA)

 [1] -1073.063

fit.GA$sbc

 [1] -1066.077

fit.LOGNO = histDist(data$Rain+0.0001,family = LOGNO , nbins=39, main='LOGNO
distribution')
```



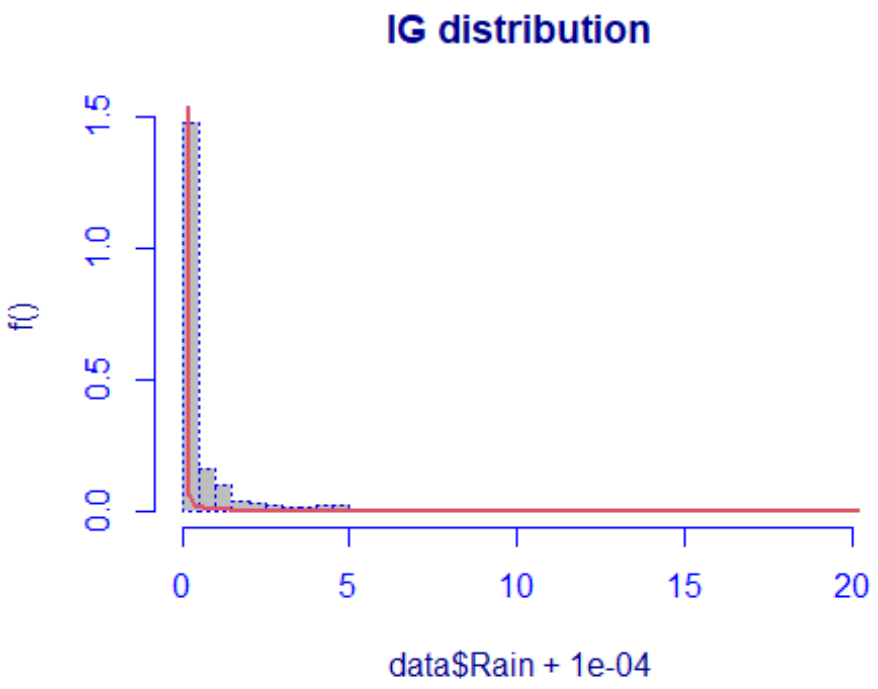

```
logLik(fit.LOGNO)

 'log Lik.' 559.387 (df=2)

AIC(fit.LOGNO)

 [1] -1114.774

fit.LOGNO$sbc

 [1] -1107.788

fit.EXP = histDist(data$Rain+0.0001,family = EXP , nbins=39, main='EXP
distribution')
```




```
logLik(fit.EXP)

 'log Lik.' -177.2892 (df=1)

AIC(fit.EXP)

 [1] 356.5784

fit.EXP$sbc

 [1] 360.0715

fit.WEI = histDist(data$Rain+0.0001,family = WEI , nbins=39, main='WEI
distribution')
```




```
logLik(fit.WEI)
 'log Lik.' 545.4928 (df=2)
AIC(fit.WEI)
 [1] -1086.986
fit.WEI$sbc
 [1] -1080
fit.IG = histDist(data$Rain+0.0001,family = IG , nbins=39, main='IG distribution')
```




```
logLik(fit.IG)

 'log Lik.' 512.3187 (df=2)

AIC(fit.IG)

 [1] -1020.637

fit.IG$sbc

 [1] -1013.651

Rain.fitted =
matrix(c(logLik(fit.GA),+AIC(fit.GA),+fit.GA$sbc,+logLik(fit.LOGNO),+AIC(fit.LOGNO
),+fit.LOGNO$sbc,+logLik(fit.EXP),+AIC(fit.EXP),+fit.EXP$sbc,+logLik(fit.WEI),+AIC
(fit.WEI),+fit.WEI$sbc,+logLik(fit.IG),AIC(fit.IG),fit.IG$sbc),nrow=5,ncol=3,byrow
=TRUE)
colnames(Rain.fitted) = c('Loglikelihood','AIC','BIC')
rownames(Rain.fitted) = c('GA','LOGNO','EXP','WEI','IG')
Rain.fitted

       Loglikelihood        AIC        BIC
 GA         538.5315 -1073.0630 -1066.0768
 LOGNO      559.3870 -1114.7740 -1107.7879
 EXP       -177.2892   356.5784   360.0715
 WEI        545.4928 -1086.9857 -1079.9995
 IG         512.3187 -1020.6374 -1013.6513

By looking at this matrix we can see how the lowest value of AIC and BIC  is
reached by the LOGNO fitting model.
```

5)FFMC : Fine Fuel Moisture Code (FFMC) index from the FWI system, is an indicator of the water of the fine fuels, so called undecomposed, that are found at a depth of 1-2 cm and estimates the degree of flammability. It is also a continuous variable and has support [0: ∞]

```
unique(data$FFMC)
```

```
  [1] 65.7 64.4 47.1 28.6 64.8 82.6 88.2 86.6 52.9 73.2 84.5 84.0 50.0 59.0 49.4
 [16] 36.1 37.3 56.9 79.9 59.8 81.0 79.1 81.4 85.9 86.7 86.8 89.0 89.1 88.7 59.9
 [31] 55.7 63.1 80.1 87.0 80.0 85.6 66.6 81.1 75.1 81.8 73.9 60.7 72.6 82.8 85.4
 [46] 88.1 73.4 68.2 70.0 84.3 89.2 90.3 86.5 87.2 78.8 78.0 76.6 85.0 86.4 77.1
 [61] 87.4 88.9 81.3 82.4 80.2 89.3 89.4 88.3 88.6 89.5 85.8 84.9 90.1 72.7 52.5
 [76] 46.0 30.5 42.6 68.4 80.8 75.8 69.6 62.0 56.1 58.5 71.0 40.9 47.4 44.9 78.1
 [91] 87.7 83.8 87.8 77.8 73.7 68.3 48.6 82.0 85.7 77.5 45.0 57.1 48.7 79.4 83.7
[106] 71.4 90.6 72.3 53.4 66.8 62.2 65.5 64.6 60.2 86.2 78.3 74.2 85.3 86.0 92.5
[121] 79.7 63.7 87.6 84.7 88.0 90.5 82.3 74.8 85.2 84.6 86.1 89.9 93.9 91.5 87.3
[136] 72.8 73.8 87.5 93.3 93.7 93.8 70.5 69.7 91.7 94.2 93.0 91.9 83.9 92.0 96.0
[151] 94.3 82.7 91.2 92.1 92.2 91.0 79.2 37.9 75.4 82.2 73.5 66.1 64.5 83.3 82.5
[166] 83.1 59.5 84.2 79.5 61.3 41.1 45.9 67.3
```

```
length(unique(data$FFMC))
```

```
[1] 173
```

Assumes 173 different modalities.

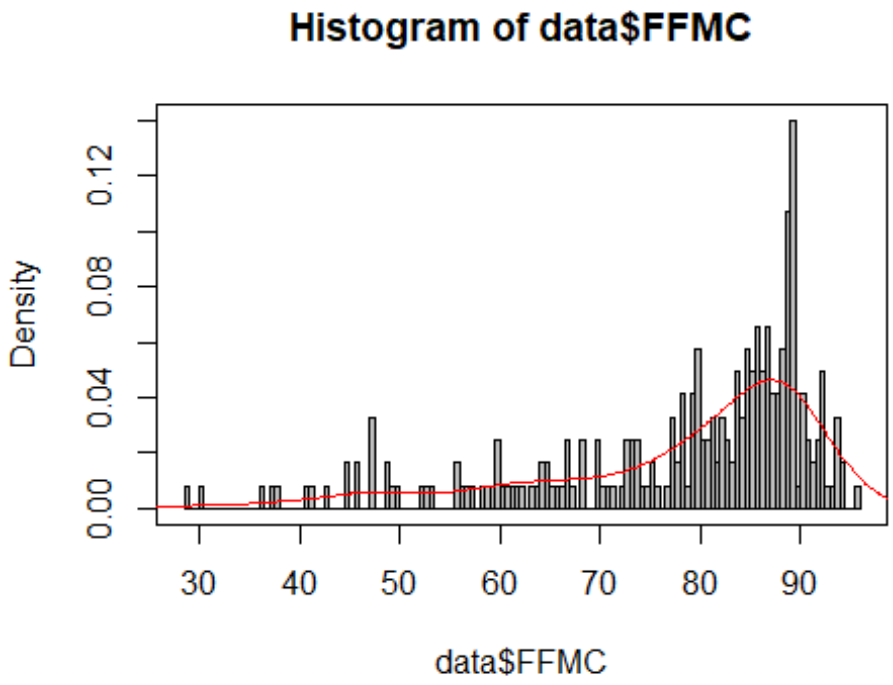
Now look at the distribution of frequency and plot it in a Hist.

```
table(data$FFMC)
```

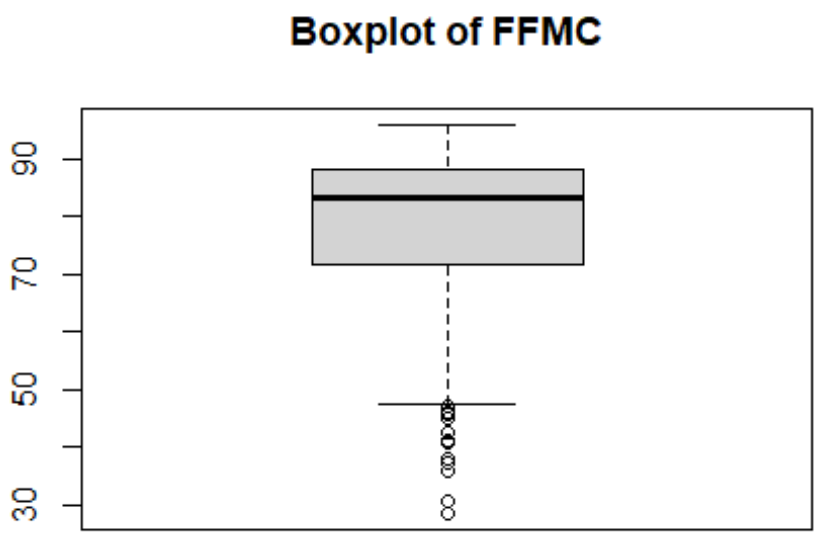
```
 28.6 30.5 36.1 37.3 37.9 40.9 41.1 42.6 44.9   45 45.9   46 47.1 47.4 48.6 48.7
    1    1    1    1    1    1    1    1    1    1    1    1    1    3    1    1
 49.4   50 52.5 52.9 53.4 55.7 56.1 56.9 57.1 58.5   59 59.5 59.8 59.9 60.2 60.7
    1    1    1    1    1    2    1    1    1    1    1    1    1    2    1    1
 61.3   62 62.2 63.1 63.7 64.4 64.5 64.6 64.8 65.5 65.7 66.1 66.6 66.8 67.3 68.2
    1    1    1    1    1    1    1    1    1    1    1    1    2    1    1    1
 68.3 68.4 69.6 69.7   70 70.5   71 71.4 72.3 72.6 72.7 72.8 73.2 73.4 73.5 73.7
    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 73.8 73.9 74.2 74.8 75.1 75.4 75.8 76.6 77.1 77.5 77.8   78 78.1 78.3 78.8 79.1
    1    1    1    1    1    1    1    1    2    2    1    1    2    3    1    1
 79.2 79.4 79.5 79.7 79.9   80 80.1 80.2 80.8   81 81.1 81.3 81.4 81.8   82 82.2
    2    1    1    2    3    2    1    2    2    1    1    1    2    1    1    1
 82.3 82.4 82.5 82.6 82.7 82.8 83.1 83.3 83.7 83.8 83.9   84 84.2 84.3 84.5 84.6
    1    1    1    1    1    1    1    1    2    1    2    1    1    1    2    2
 84.7 84.9   85 85.2 85.3 85.4 85.6 85.7 85.8 85.9   86 86.1 86.2 86.4 86.5 86.6
    2    1    2    1    1    4    2    1    1    2    2    1    2    1    2    2
 86.7 86.8   87 87.2 87.3 87.4 87.5 87.6 87.7 87.8   88 88.1 88.2 88.3 88.6 88.7
    1    2    3    1    1    2    1    1    1    2    1    3    1    3    2    2
 88.9   89 89.1 89.2 89.3 89.4 89.5 89.9 90.1 90.3 90.5 90.6   91 91.2 91.5 91.7
    7    2    4    2    4    5    2    1    2    1    2    2    1    1    1    1
```

```
 91.9   92 92.1 92.2 92.5   93 93.3 93.7 93.8 93.9 94.2 94.3   96
   1    1    2    2    2    1    1    1    1    2    1    1    1
```

```
hist(data$FFMC,col = 'grey', breaks = 173 ,freq = FALSE)
box()
lines(density(data$FFMC),col = 'red')
```




```
boxplot(data$FFMC, main='Boxplot of FFMC')
```



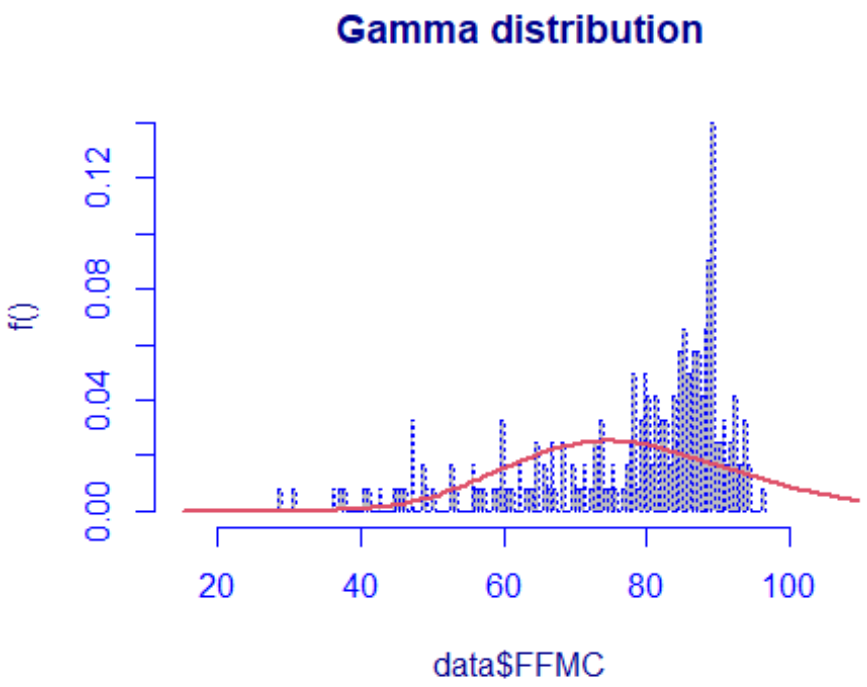
By looking at this boxplot we can see how we have some outliers and we can also see that median is closer to the third quartile than the first.

By computing the skewness and the kurtosis, it seems that the FFMC distribution is highly negative skewed, and the kurtosis' value is greater than one, which means that the distribution is leptokurtic and has heavy tails.

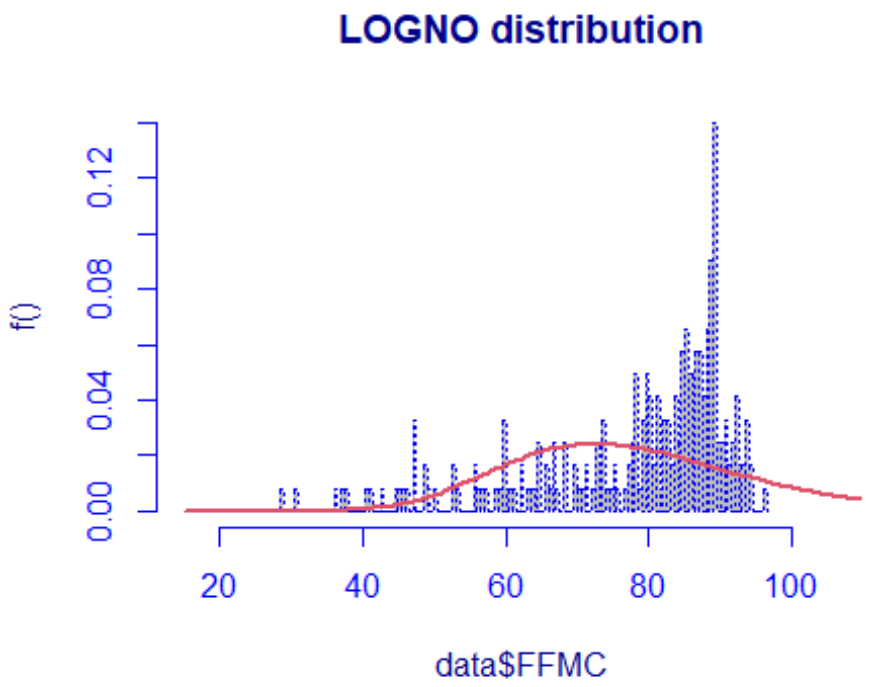
```
kurtosis(data$FFMC)

 [1] 3.994219

skewness(data$FFMC)

 [1] -1.311967
```

DATA FITTING FOR FFMC: In the following code, we will try to establish a theoretical model to be fitted with the FFMC variable, by looking at three different criterions: the Log-likelihood value (which must be maximized), the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC) (which must be minimized).

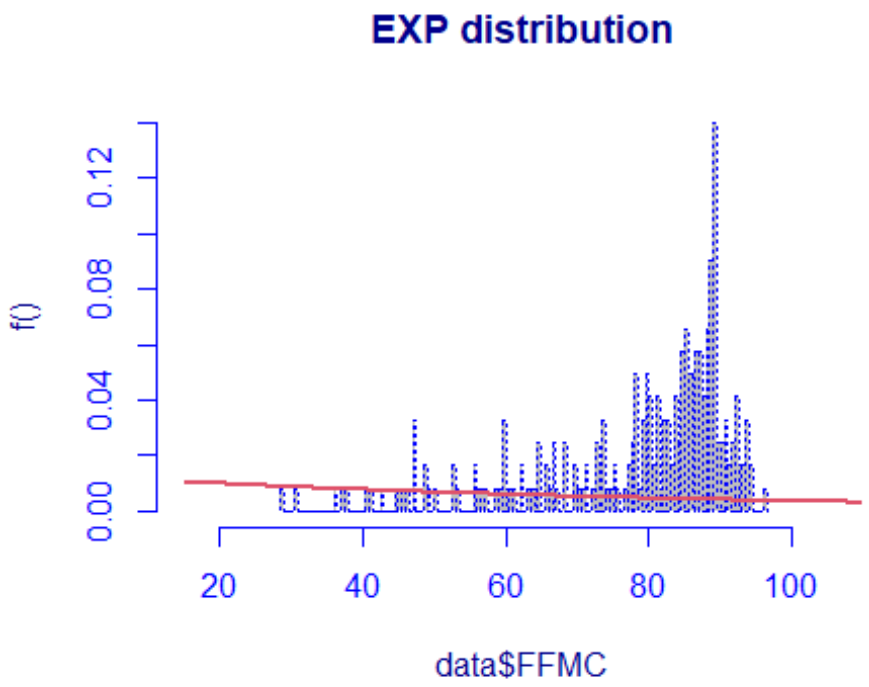
```
fit.GA = histDist(data$FFMC,family = GA , nbins=173, main='Gamma distribution')
```



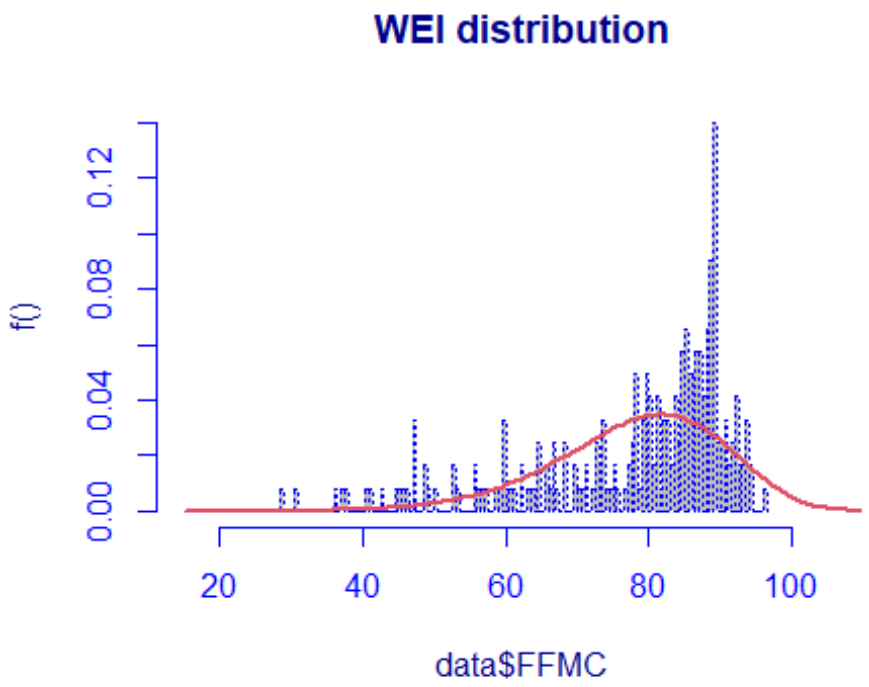

```
logLik(fit.GA)

 'log Lik.' -1016.552 (df=2)

AIC(fit.GA)

 [1] 2037.103

fit.GA$sbc

 [1] 2044.089

fit.LOGNO = histDist(data$FFMC,family = LOGNO , nbins=173, main='LOGNO
distribution')
```



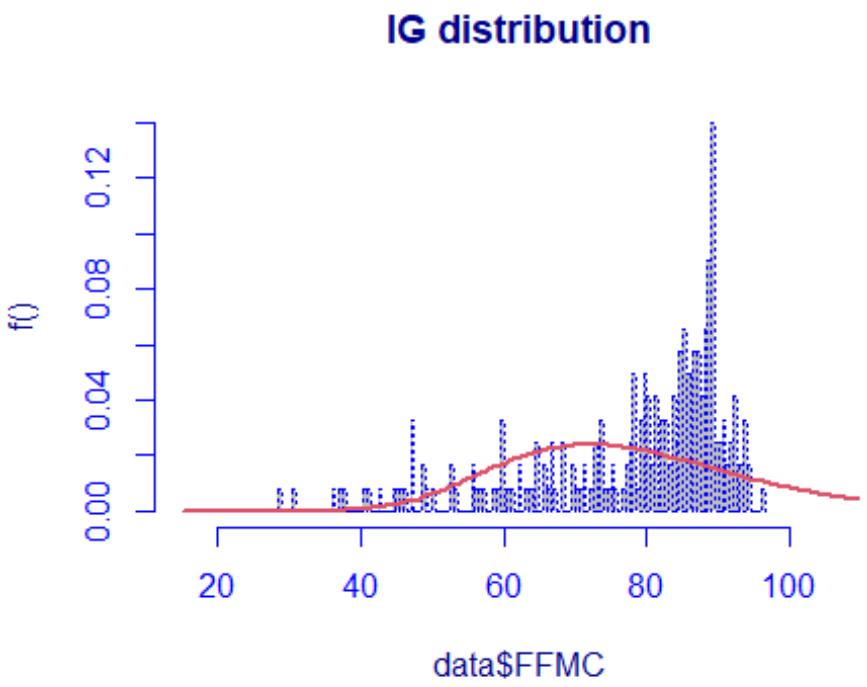

```
logLik(fit.LOGNO)
 'log Lik.' -1031.604 (df=2)
AIC(fit.LOGNO)
 [1] 2067.207
fit.LOGNO$sbc
 [1] 2074.193
fit.EXP = histDist(data$FFMC,family = EXP , nbins=173, main='EXP distribution')
```




```
logLik(fit.EXP)
 'log Lik.' -1301.189 (df=1)
AIC(fit.EXP)
 [1] 2604.377
fit.EXP$sbc
 [1] 2607.871
fit.WEI = histDist(data$FFMC,family = WEI , nbins=173, main='WEI distribution')
```




```
logLik(fit.WEI)
 'log Lik.' -964.3328 (df=2)
AIC(fit.WEI)
 [1] 1932.666
fit.WEI$sbc
 [1] 1939.652
fit.IG = histDist(data$FFMC,family = IG , nbins=173, main='IG distribution')
```




```
logLik(fit.IG)

 'log Lik.' -1033.634 (df=2)

AIC(fit.IG)

 [1] 2071.268

fit.IG$sbc

 [1] 2078.254

FFMC.fitted =
matrix(c(logLik(fit.GA),+AIC(fit.GA),+fit.GA$sbc,+logLik(fit.LOGNO),+AIC(fit.LOGNO
),+fit.LOGNO$sbc,+logLik(fit.EXP),+AIC(fit.EXP),+fit.EXP$sbc,+logLik(fit.WEI),+AIC
(fit.WEI),+fit.WEI$sbc,+logLik(fit.IG),AIC(fit.IG),fit.IG$sbc),nrow=5,ncol=3,byrow
=TRUE)
colnames(FFMC.fitted) = c('Loglikelihood','AIC','BIC')
rownames(FFMC.fitted) = c('GA','LOGNO','EXP','WEI','IG')
FFMC.fitted

       Loglikelihood      AIC      BIC
 GA       -1016.5516 2037.103 2044.089
 LOGNO    -1031.6035 2067.207 2074.193
 EXP      -1301.1887 2604.377 2607.871
 WEI       -964.3328 1932.666 1939.652
 IG       -1033.6338 2071.268 2078.254

By looking at the matrix we can see how Weibull model is the best for fitting,
because it gave us the best values for AIC and BIC.
```

7) DMC : Duff Moisture Code from the FWI system, it is a continuous variable and has a support [0: ∞]

```
unique(data$DMC)
```

```
  [1]  3.4  4.1  2.5  1.3  3.0  5.8  9.9 12.1  7.9  9.5 12.5 13.8  6.7  4.6  1.7
 [16]  1.1  1.9  4.5  6.3  7.0  8.2 11.2 14.2 17.8 21.6 25.5 18.4 22.9  2.4  2.6
 [31]  7.6 10.9  9.7  7.7  6.0  8.1  7.8  5.2  9.4 12.0 12.3 18.5 16.4 10.5  9.6
 [46] 17.1 22.2 24.4 26.7 28.5 31.9  4.8  5.7 11.1 13.0 15.5 11.3 14.8 18.6 21.7
 [61] 15.6 19.0 11.7 16.0 20.0 23.2 25.9 29.6 33.5 37.6 40.5 43.9 45.6 47.0 50.2
 [76] 54.2 25.2  8.7  0.7  1.2  3.6  3.2  2.1  2.2  0.9  6.4  9.8 13.5 16.5 10.6
 [91]  5.5  8.3  7.1  2.9  2.7  8.4  8.5 13.3 18.2 21.3 11.4  7.2  4.2  3.9  4.4
[106]  3.8 10.0 12.8 20.9 27.2 17.9 13.6 18.7  8.0 12.6 18.0 19.4 21.1 23.9 27.8
[121] 32.7 39.6 44.2 46.6 10.8 11.8 15.7 19.5 23.8 28.3 23.0 23.6 11.0 15.8 22.5
[136] 16.9 22.3 22.6 30.3 35.9 34.4 36.9 41.1 46.1 51.3 56.3 61.3 65.9 37.0 20.7
[151] 24.8  4.0  3.3  6.6  4.7  6.5 11.5 21.2 25.8 24.9 26.1 29.4 11.9  3.5  4.3
```

```
length(unique(data$DMC))
```

```
 [1] 165
```

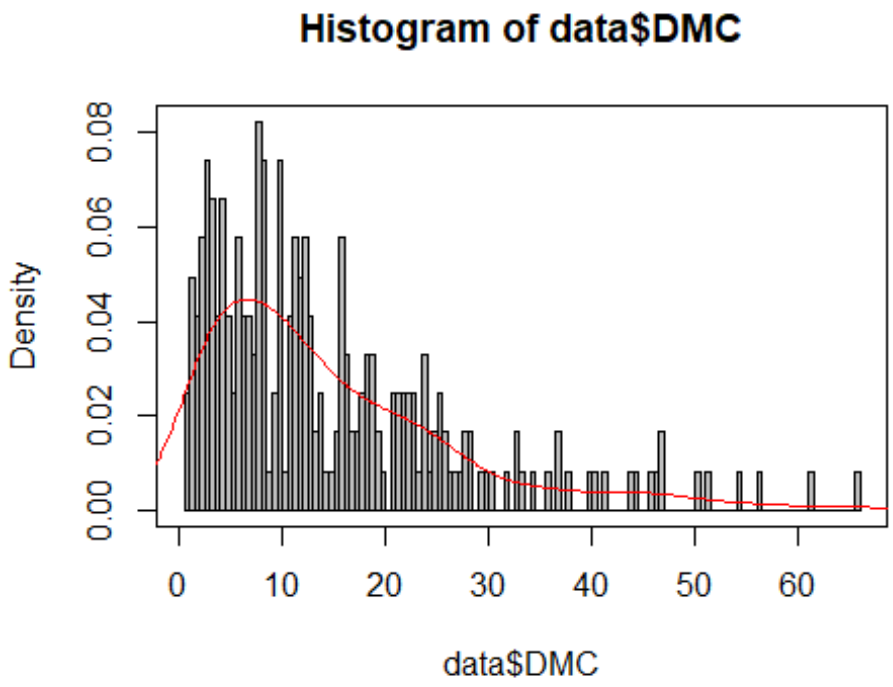
We can see how it assumes 165 different modalities.

Now we can look at frequency's distribution and plot it into an hist to get a better visual rappresentation.

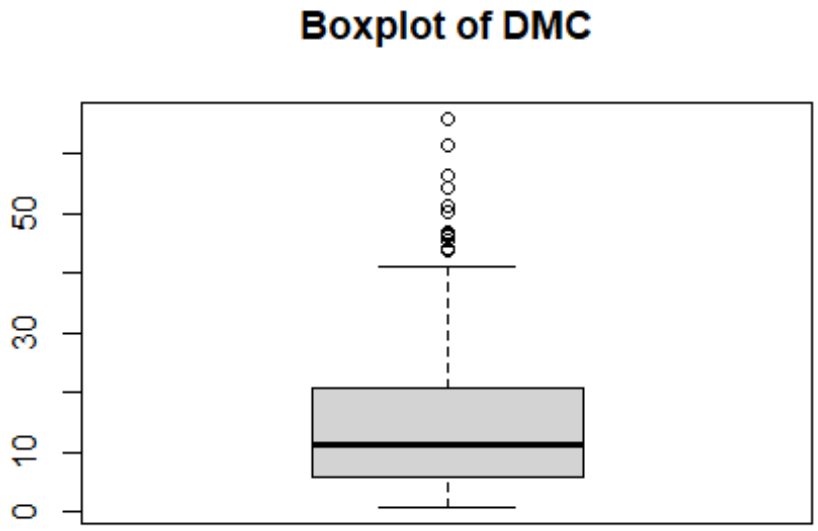
```
table(data$DMC)
```

```
  0.7  0.9  1.1  1.2  1.3  1.7  1.9  2.1  2.2  2.4  2.5  2.6  2.7  2.9    3  3.2
    1    2    2    1    3    1    4    1    2    1    3    3    1    2    3    3
  3.3  3.4  3.5  3.6  3.8  3.9    4  4.1  4.2  4.3  4.4  4.5  4.6  4.7  4.8  5.2
    1    3    1    1    2    1    1    2    1    1    3    1    3    1    1    2
  5.5  5.7  5.8    6  6.3  6.4  6.5  6.6  6.7    7  7.1  7.2  7.6  7.7  7.8  7.9
    1    1    3    3    1    2    2    1    1    3    2    2    1    2    1    5
    8  8.1  8.2  8.3  8.4  8.5  8.7  9.4  9.5  9.6  9.7  9.8  9.9   10 10.5 10.6
    1    2    2    3    1    1    1    2    1    1    3    2    2    1    1    1
 10.8 10.9   11 11.1 11.2 11.3 11.4 11.5 11.7 11.8 11.9   12 12.1 12.3 12.5 12.6
    2    1    1    1    2    2    1    1    2    1    1    2    2    1    4    2
 12.8   13 13.3 13.5 13.6 13.8 14.2 14.8 15.5 15.6 15.7 15.8   16 16.4 16.5 16.9
    1    2    1    1    1    2    1    1    2    2    1    1    3    1    3    2
 17.1 17.8 17.9   18 18.2 18.4 18.5 18.6 18.7   19 19.4 19.5   20 20.7 20.9 21.1
    2    1    1    1    1    2    1    1    2    1    1    1    1    1    2    1
 21.2 21.3 21.6 21.7 22.2 22.3 22.5 22.6 22.9   23 23.2 23.6 23.8 23.9 24.4 24.8
    1    1    1    2    1    1    1    1    1    1    1    2    1    1    1    1
 24.9 25.2 25.5 25.8 25.9 26.1 26.7 27.2 27.8 28.3 28.5 29.4 29.6 30.3 31.9 32.7
    1    2    1    1    1    1    1    1    2    1    1    1    1    1    1    2
 33.5 34.4 35.9 36.9   37 37.6 39.6 40.5 41.1 43.9 44.2 45.6 46.1 46.6   47 50.2
    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 51.3 54.2 56.3 61.3 65.9
    1    1    1    1    1
```

```r
hist(data$DMC,col = 'grey', breaks = 165 ,freq = FALSE)
box()
lines(density(data$DMC),col = 'red')
```




```r
boxplot(data$DMC, main='Boxplot of DMC')
```



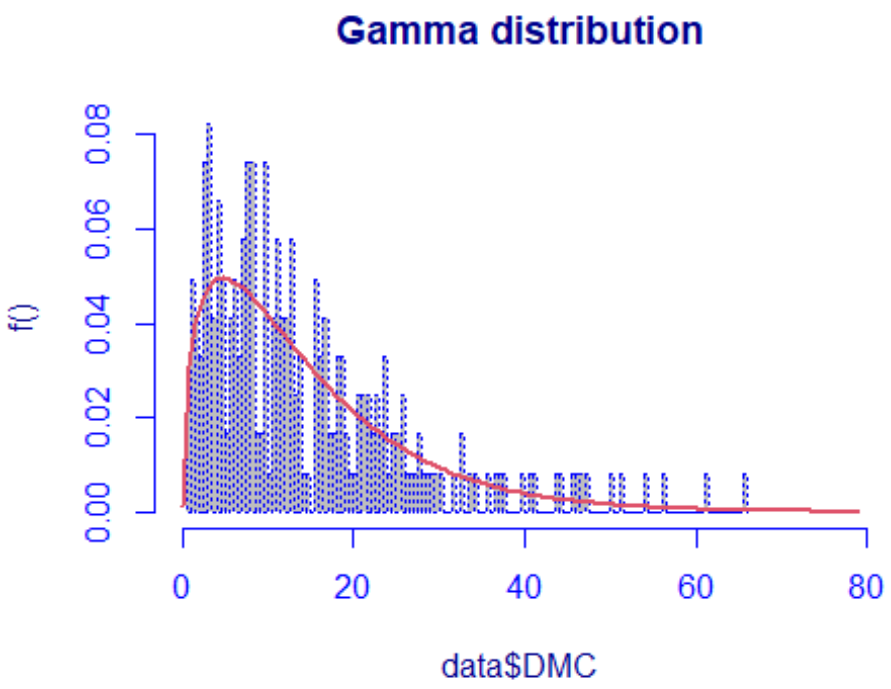
```
By looking at this boxplot we can see how there are some outliers, and we can also
see how the median is quite closer to the first quartile than third.

kurtosis(data$DMC)

 [1] 5.387582

skewness(data$DMC)

 [1] 1.513566
```

By computing the skewness and the kurtosis, it seems that the DMC distribution is highly positive skewed, and the kurtosis' value is greater than one, which means that the distribution is leptokurtic and has heavy tails

Data fitting for DMC:

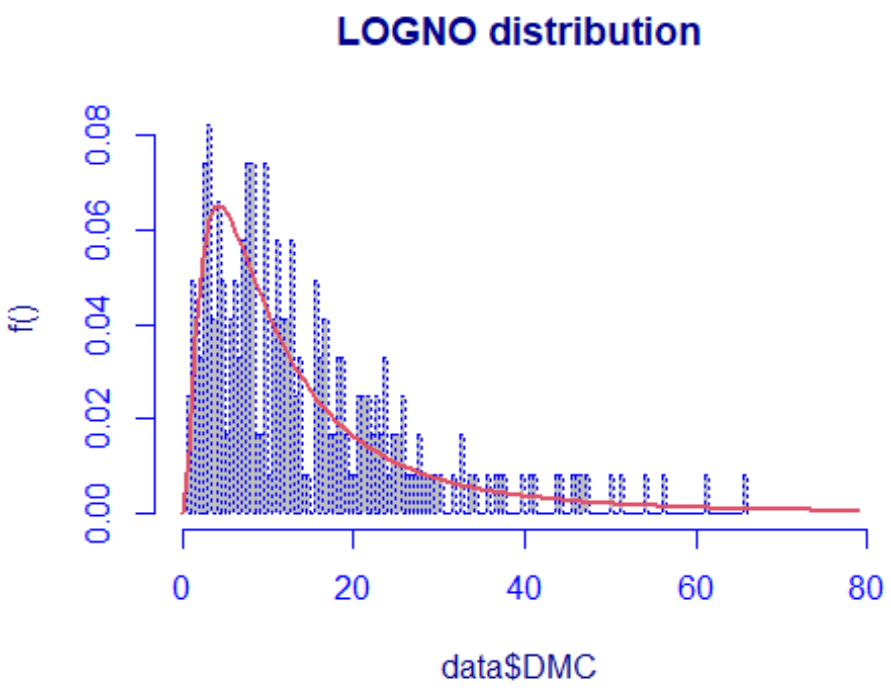
In the following code, we will try to establish a theoretical model to be fitted with the DMC variable, by looking at three different criterions: the Log-likelihood value (which must be maximized), the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC) (which must be minimized).

```
fit.GA = histDist(data$DMC,family = GA , nbins=165, main='Gamma distribution')
```

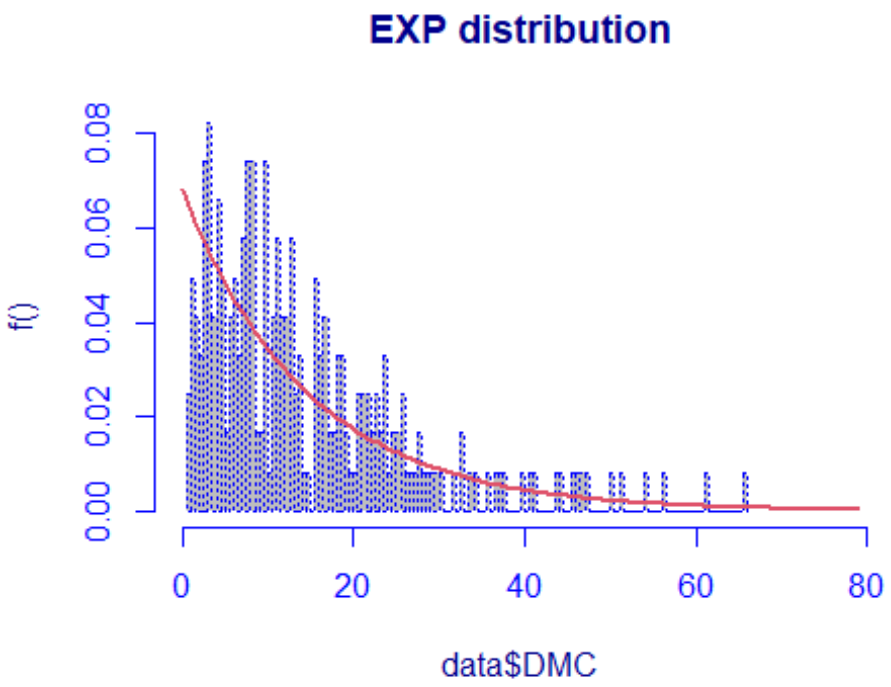



```
logLik(fit.GA)

 'log Lik.' -885.4969 (df=2)

AIC(fit.GA)
```

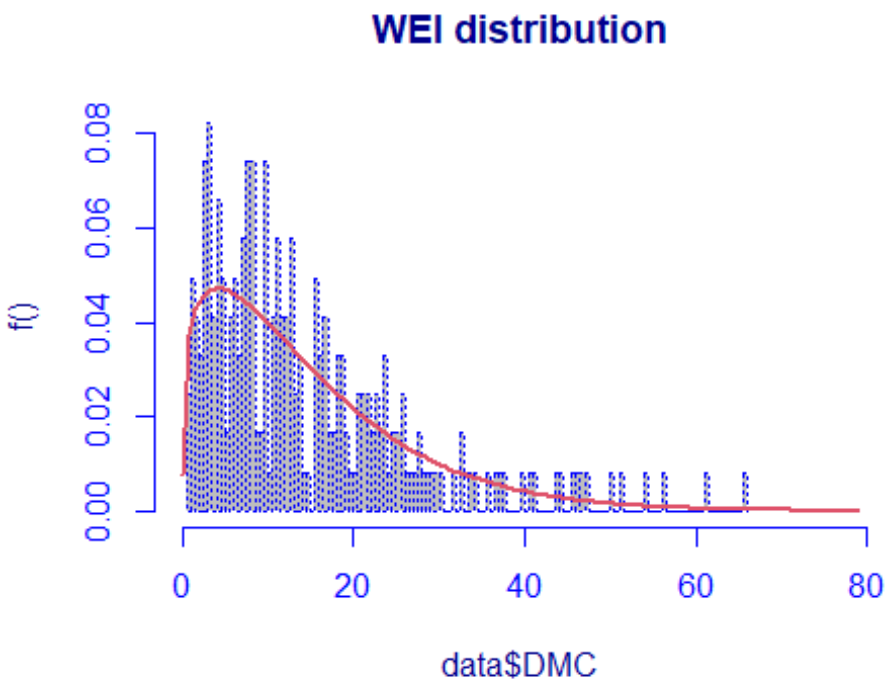
```
 [1] 1774.994

fit.GA$sbc

 [1] 1781.98

fit.LOGNO = histDist(data$DMC,family = LOGNO , nbins=165, main='LOGNO
distribution')
```



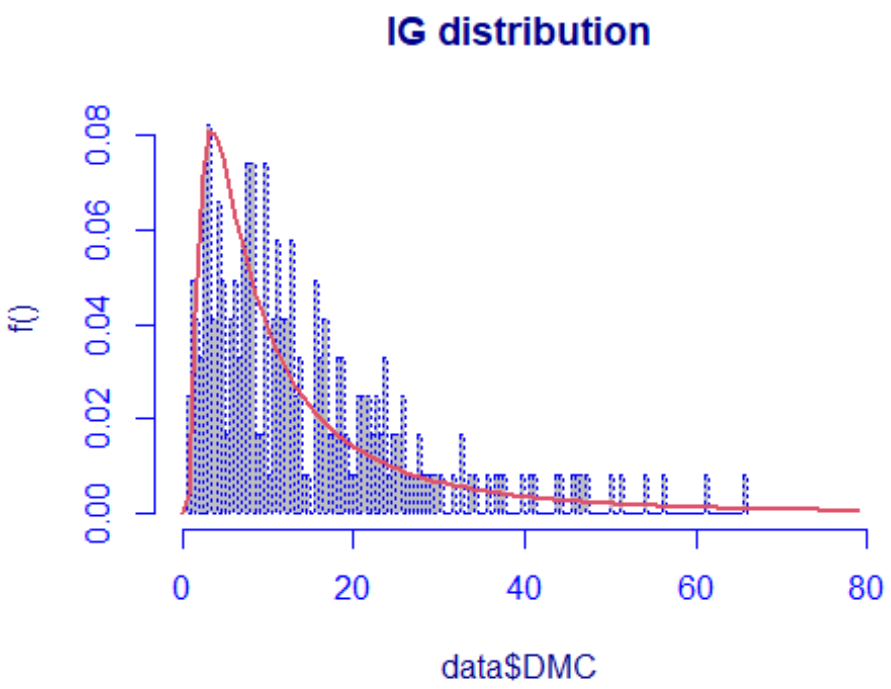

```
logLik(fit.LOGNO)

 'log Lik.' -889.9742 (df=2)

AIC(fit.LOGNO)

 [1] 1783.948

fit.LOGNO$sbc

 [1] 1790.934

fit.EXP = histDist(data$DMC,family = EXP() , nbins=165, main='EXP distribution')
```




```
logLik(fit.EXP)
 'log Lik.' -895.827 (df=1)
AIC(fit.EXP)
 [1] 1793.654
fit.EXP$sbc
 [1] 1797.147
fit.WEI = histDist(data$DMC,family = WEI , nbins=165, main='WEI distribution')
```




```
logLik(fit.WEI)
 'log Lik.' -887.1616 (df=2)
AIC(fit.WEI)
 [1] 1778.323
fit.WEI$sbc
 [1] 1785.309
fit.IG = histDist(data$DMC,family = IG , nbins=165, main='IG distribution')
```




```
logLik(fit.IG)

 'log Lik.' -897.6867 (df=2)

AIC(fit.IG)

 [1] 1799.373

fit.IG$sbc

 [1] 1806.36

DMC.fitted =
matrix(c(logLik(fit.GA),+AIC(fit.GA),+fit.GA$sbc,+logLik(fit.LOGNO),+AIC(fit.LOGNO
),+fit.LOGNO$sbc,+logLik(fit.EXP),+AIC(fit.EXP),+fit.EXP$sbc,+logLik(fit.WEI),+AIC
(fit.WEI),+fit.WEI$sbc,+logLik(fit.IG),AIC(fit.IG),fit.IG$sbc),nrow=5,ncol=3,byrow
=TRUE)
colnames(DMC.fitted) = c('Loglikelihood','AIC','BIC')
rownames(DMC.fitted) = c('GA','LOGNO','EXP','WEI','IG')
DMC.fitted

       Loglikelihood      AIC      BIC
 GA        -885.4969 1774.994 1781.980
 LOGNO     -889.9742 1783.948 1790.934
 EXP       -895.8270 1793.654 1797.147
 WEI       -887.1616 1778.323 1785.309
 IG        -897.6867 1799.373 1806.360

By looking at the matrix we can say that GAMMA  is the best model, because it has
lowest level of AIC and BIC.
```

8)DC: Drought Code from the FWI, it is a continuous variable and has support [0: ,∞]. It refers to the presence of flammable materials in the deep soil.

```
unique(data$DC)
```

```
  [1]   7.6   7.1   6.9  14.2  22.2  30.5  38.3  38.8  46.3  54.3  61.4  17.0
 [13]   7.8   7.4   8.0  16.0  27.1  31.6  39.5  47.7  55.8  63.8  71.8  80.3
 [25]  88.5  84.4  92.8   8.6   8.3   9.2  18.5  27.9  37.0  40.4  49.8   9.3
 [37]  18.7  27.7  37.2  22.9  25.5  34.1  43.1  52.8  62.1  71.5  79.9  71.3
 [49]  79.7  88.7  98.6 108.5 117.8 127.0 136.0 145.7  10.2  10.0  19.8  29.7
 [61]  39.1  48.6  47.0  57.0  67.0  77.0  75.1  85.1  94.7  92.5  90.4 100.7
 [73] 110.9 120.9 130.6 141.1 151.3 161.5 171.3 181.3 190.6 200.2 210.4 220.4
 [85] 180.4   8.7   7.5   7.0  15.7  24.0  32.2  30.1   8.4   8.9  16.6   7.3
 [97]  24.3  33.1  41.3  49.3  57.9  41.4  30.4  15.2   7.7  16.3  24.9   8.8
[109]   8.2  15.4  17.6  26.3  28.9  14.7  22.5  37.8  18.4  25.6  34.5  43.3
[121]  52.4  36.7   8.5  17.8  27.3  36.8  46.4  45.1  35.4   9.7   9.9   9.5
[133]  19.4  10.4  24.1  42.3  51.6  61.1  71.0  80.6  90.1  99.0  56.6  15.9
[145]  19.7  28.3  37.6  47.2  57.1  67.2  10.5  21.4  32.1  42.7  52.5   9.1
[157]   9.8  20.2  30.9  41.5  55.5  54.2  65.1  76.4  86.8  96.8 107.0 117.1
[169] 127.5 137.7 147.7 157.5 167.2 177.3 166.0 149.2 159.1 168.2  26.6  17.7
[181]  26.1  25.2  33.4  50.2  59.2  63.3  77.8  86.0  88.0  97.3 106.3 115.6
[193]  28.1  36.1  44.5   7.9  16.5
```

```
length(unique(data$DC))
```

```
 [1] 197
```

By looking at the length we can see that our variable has 197 different modalities.

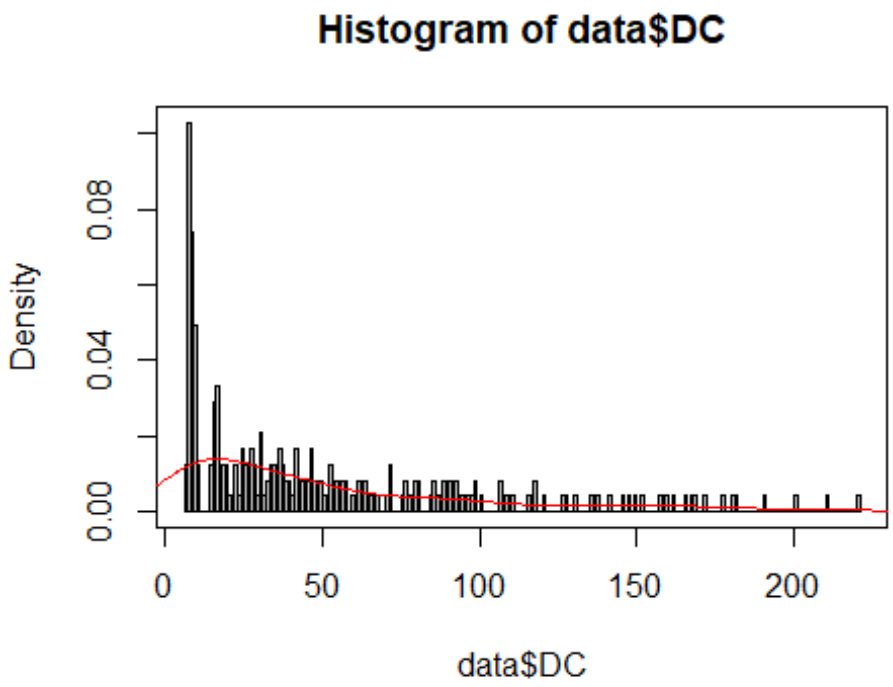
Now let’s compute frequency’s distribution and plot it into a hist to get a better visual rappresentantion.

```
table(data$DC)
```

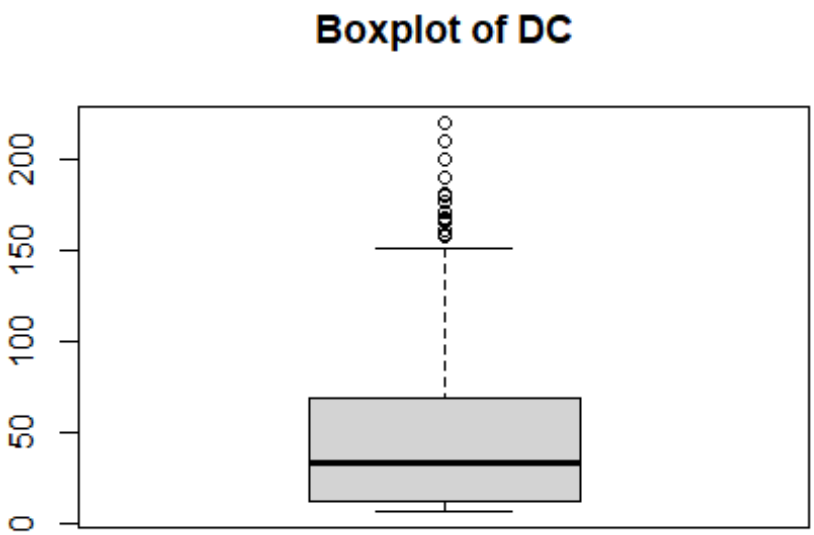
```
  6.9     7   7.1   7.3   7.4   7.5   7.6   7.7   7.8   7.9     8   8.2   8.3
    1     2     1     2     2     4     4     2     4     1     5     4     4
  8.4   8.5   8.6   8.7   8.8   8.9   9.1   9.2   9.3   9.5   9.7   9.8   9.9
    4     2     1     1     1     1     2     2     1     2     1     1     1
   10  10.2  10.4  10.5  14.2  14.7  15.2  15.4  15.7  15.9    16  16.3  16.5
    2     1     1     1     1     2     2     1     2     1     1     2     1
 16.6    17  17.6  17.7  17.8  18.4  18.5  18.7  19.4  19.7  19.8  20.2  21.4
    2     3     1     1     1     1     1     1     1     1     1     1     1
 22.2  22.5  22.9    24  24.1  24.3  24.9  25.2  25.5  25.6  26.1  26.3  26.6
    1     1     1     1     1     2     1     1     1     1     1     1     1
 27.1  27.3  27.7  27.9  28.1  28.3  28.9  29.7  30.1  30.4  30.5  30.9  31.6
    1     1     1     1     1     1     1     1     1     1     2     1     1
 32.1  32.2  33.1  33.4  34.1  34.5  35.4  36.1  36.7  36.8    37  37.2  37.6
    1     1     2     1     1     2     2     1     1     1     1     1     1
 37.8  38.3  38.8  39.1  39.5  40.4  41.3  41.4  41.5  42.3  42.7  43.1  43.3
    1     1     1     1     1     1     1     1     2     1     1     1     1
 44.5  45.1  46.3  46.4    47  47.2  47.7  48.6  49.3  49.8  50.2  51.6  52.4
```

```
    1     2     2     1     1     1     1     1     1     1     1     1     1
 52.5  52.8  54.2  54.3  55.5  55.8  56.6    57  57.1  57.9  59.2  61.1  61.4
    1     1     1     1     1     1     1     1     1     1     1     1     1
 62.1  63.3  63.8  65.1    67  67.2    71  71.3  71.5  71.8  75.1  76.4    77
    1     1     1     1     1     1     1     1     1     1     1     1     1
 77.8  79.7  79.9  80.3  80.6  84.4  85.1    86  86.8    88  88.5  88.7  90.1
    1     1     1     1     1     1     1     1     1     1     1     1     1
 90.4  92.5  92.8  94.7  96.8  97.3  98.6    99 100.7 106.3   107 108.5 110.9
    1     1     1     1     1     1     1     1     1     1     1     1     1
115.6 117.1 117.8 120.9   127 127.5 130.6   136 137.7 141.1 145.7 147.7 149.2
    1     1     1     1     1     1     1     1     1     1     1     1     1
151.3 157.5 159.1 161.5   166 167.2 168.2 171.3 177.3 180.4 181.3 190.6 200.2
    1     1     1     1     1     1     1     1     1     1     1     1     1
210.4 220.4
    1     1
```

```
hist(data$DC,col = 'grey', breaks = 197 ,freq = FALSE)
box()
lines(density(data$DC),col = 'red')
```




```
boxplot(data$DC, main='Boxplot of DC')
```



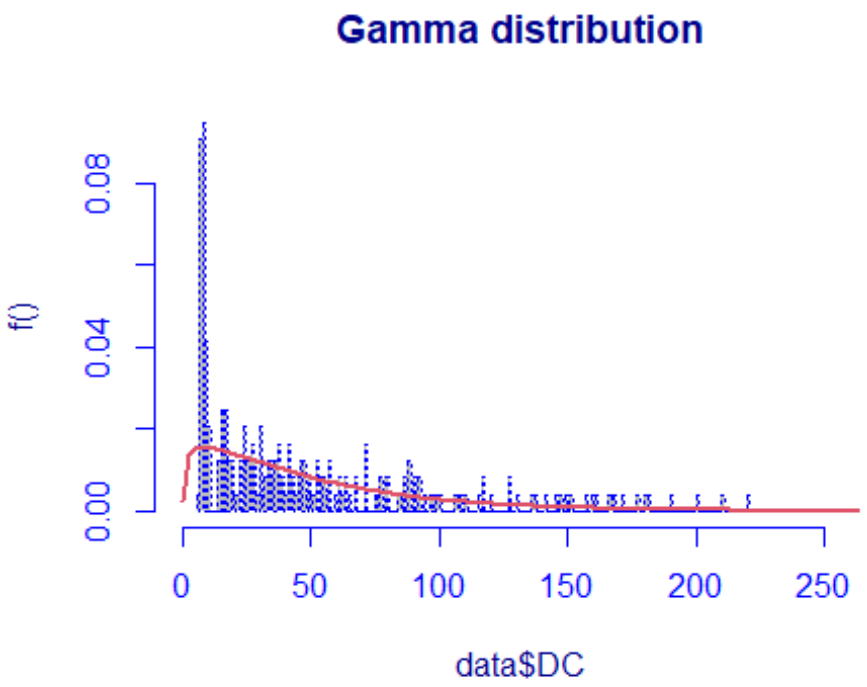

```
By looking at the boxplot we can see that we have the presence of outliers. It is
also useful to underline the fact that we have first quartile closer to the median
than third.

kurtosis(data$DC)

 [1] 4.539216

skewness(data$DC)

 [1] 1.464349
```

By computing the skewness and the kurtosis, it seems that the DC distribution is highly positive skewed, and the kurtosis' value is greater than one, which means that the distribution is leptokurtic and has heavy tails.

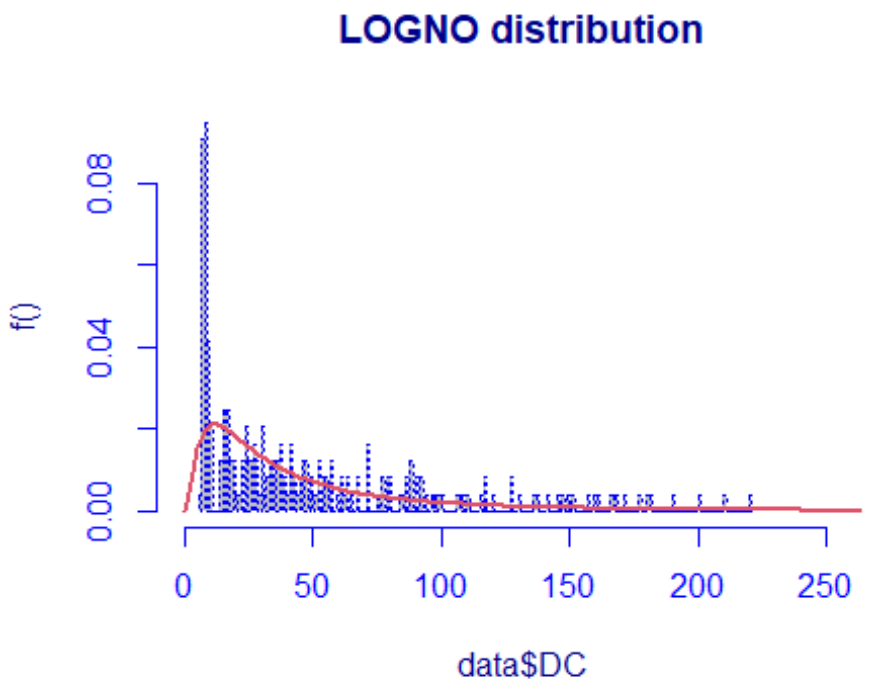
DATA FITTING FOR DC:

In the following code, we will try to establish a theoretical model to be fitted with the DC variable, by looking at three different criterions: the Log-likelihood value (which must be maximized), the Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC) (which must be minimized).

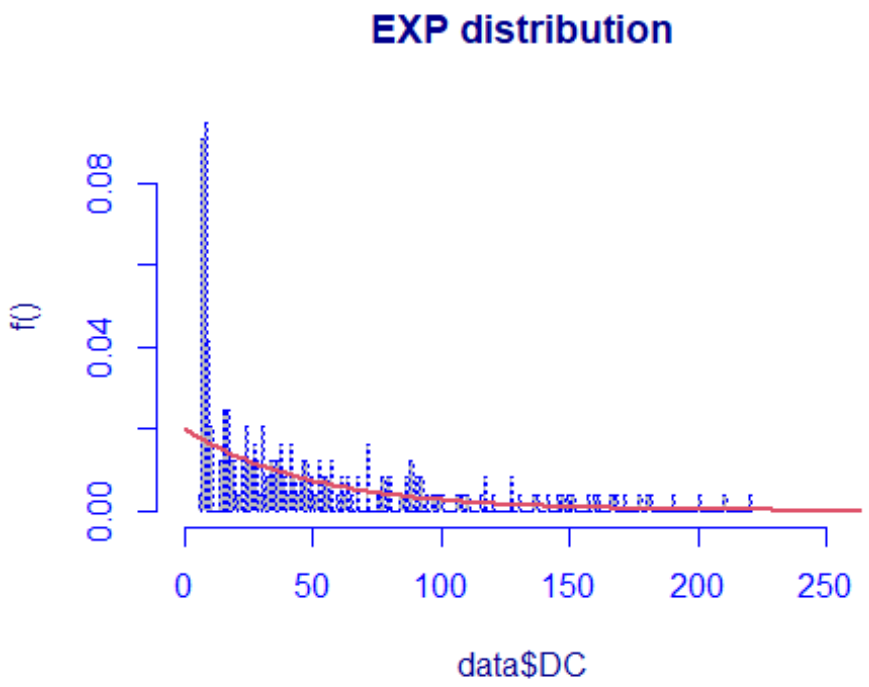
```
fit.GA = histDist(data$DC,family = GA , nbins=197, main='Gamma distribution')
```



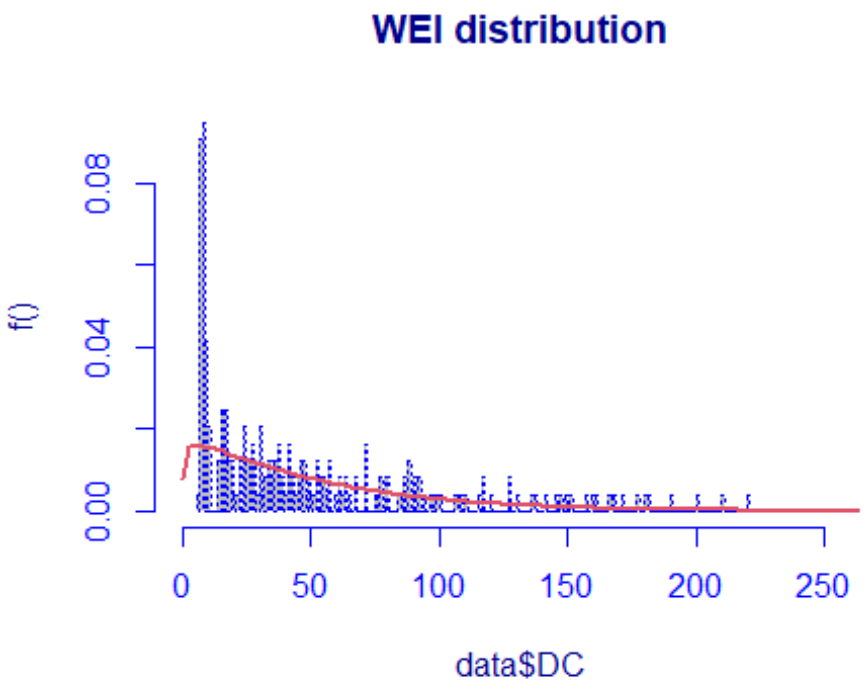

```
logLik(fit.GA)

 'log Lik.' -1187.859 (df=2)

AIC(fit.GA)

 [1] 2379.719

fit.GA$sbc

 [1] 2386.705

fit.LOGNO = histDist(data$DC,family = LOGNO , nbins=197, main='LOGNO
distribution')
```



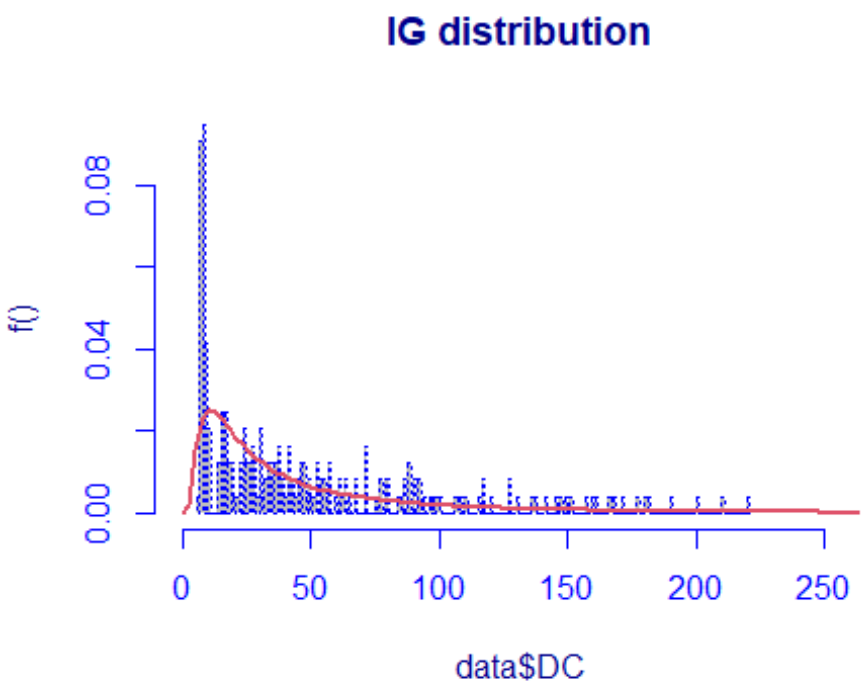

```
logLik(fit.LOGNO)
 'log Lik.' -1178.843 (df=2)
AIC(fit.LOGNO)
 [1] 2361.685
fit.LOGNO$sbc
 [1] 2368.671
fit.EXP = histDist(data$DC,family = EXP , nbins=197, main='EXP distribution')
```




```
logLik(fit.EXP)
 'log Lik.' -1190.84 (df=1)
AIC(fit.EXP)
 [1] 2383.679
fit.EXP$sbc
 [1] 2387.173
fit.WEI = histDist(data$DC,family = WEI , nbins=197, main='WEI distribution')
```




```
logLik(fit.WEI)
 'log Lik.' -1189.342 (df=2)
AIC(fit.WEI)
 [1] 2382.684
fit.WEI$sbc
 [1] 2389.671
fit.IG = histDist(data$DC,family = IG , nbins=197, main='IG distribution')
```

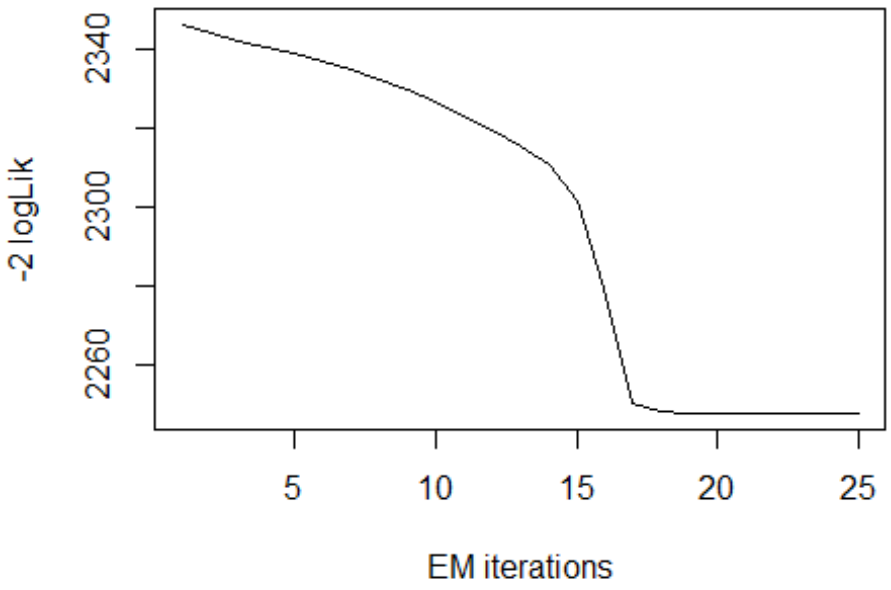



```
logLik(fit.IG)

 'log Lik.' -1172.041 (df=2)

AIC(fit.IG)

 [1] 2348.082

fit.IG$sbc

 [1] 2355.068

DC.fitted =
matrix(c(logLik(fit.GA),+AIC(fit.GA),+fit.GA$sbc,+logLik(fit.LOGNO),+AIC(fit.LOGNO
),+fit.LOGNO$sbc,+logLik(fit.EXP),+AIC(fit.EXP),+fit.EXP$sbc,+logLik(fit.WEI),+AIC
(fit.WEI),+fit.WEI$sbc,+logLik(fit.IG),AIC(fit.IG),fit.IG$sbc),nrow=5,ncol=3,byrow
=TRUE)
colnames(DC.fitted) = c('Loglikelihood','AIC','BIC')
rownames(DC.fitted) = c('GA','LOGNO','EXP','WEI','IG')
DC.fitted

       Loglikelihood      AIC      BIC
 GA        -1187.859 2379.719 2386.705
 LOGNO     -1178.843 2361.685 2368.671
 EXP       -1190.840 2383.679 2387.173
 WEI       -1189.342 2382.684 2389.671
 IG        -1172.041 2348.082 2355.068

By looking at this matrix we can see how IG  gets the best values for AIC and BIC.
```

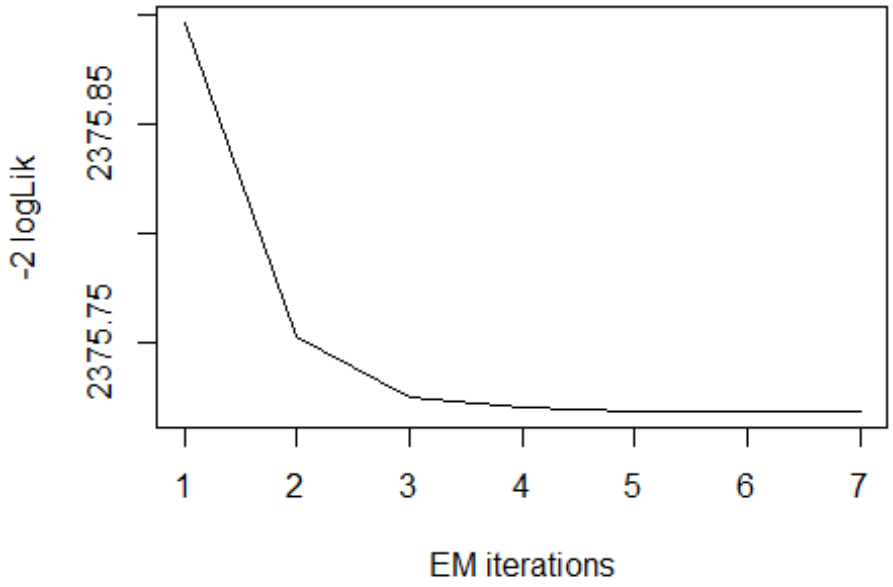
MIXTURE OF 2 GAMMAS DISTRIBUTIONS

For a further and more complete analysis we are going to compare the result of the previous fitting analysis with the outcome of fitting mixture of two GAMMA distributions. The purpose of this analysis is to assess which kind of fitting approach yield the best result. Note that the algorithm will be repeated 5 times in order to have a more stable result.

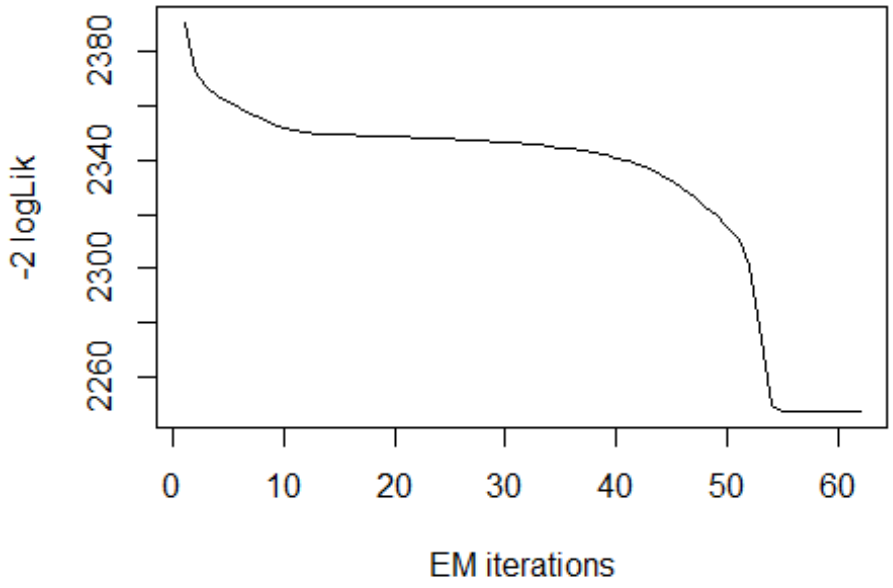
```
fit.GA.2 <- gamlssMXfits(n = 5, data$DC~1, family = GA, K = 2, data = NULL)
```



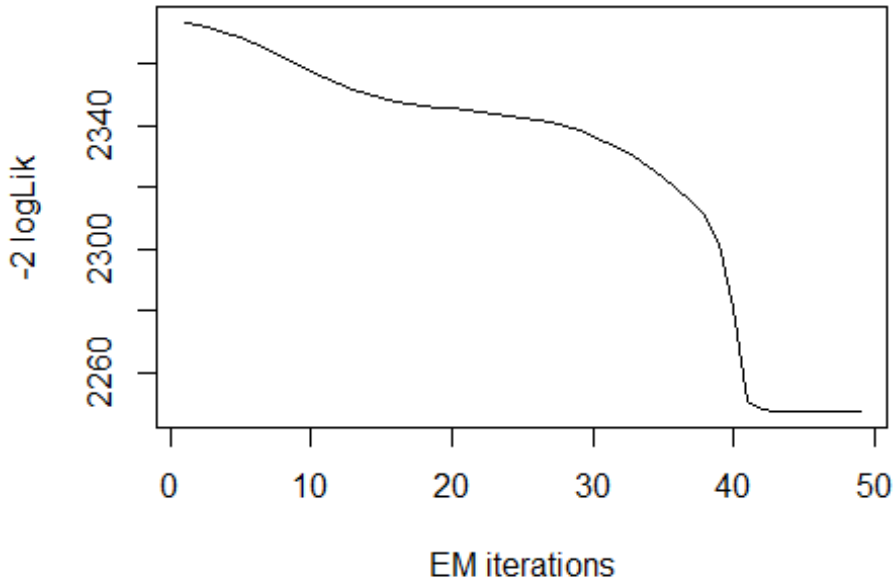

```
model= 1
```



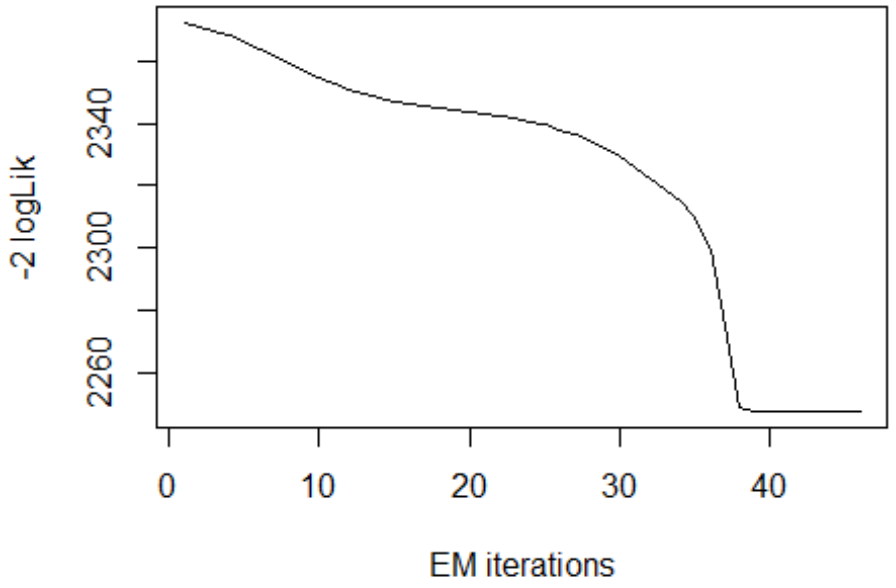

```
model= 2
```




```
model= 3
```

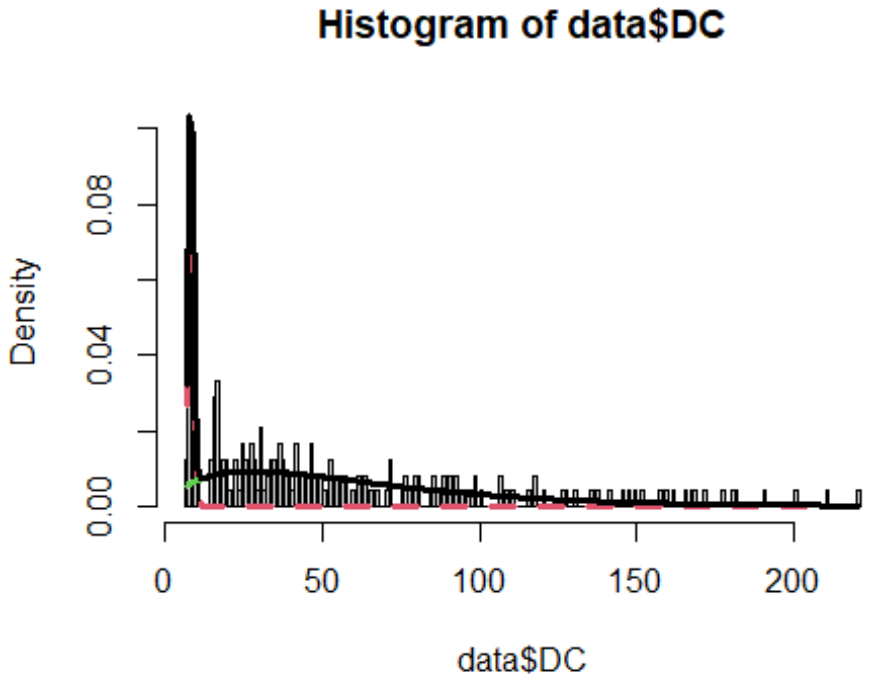



```
model= 4
```




```
 model= 5

logLik(fit.GA.2)
```

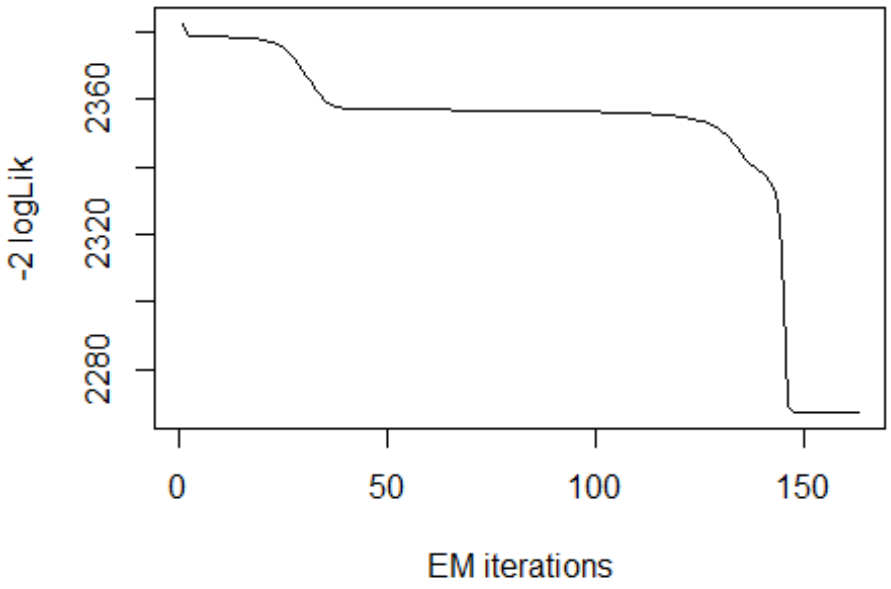
```
 'log Lik.' -1123.795 (df=5)

fit.GA.2$prob

 [1] 0.2207653 0.7792347

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

hist(data$DC, breaks = 197,freq = FALSE)
lines(seq(min(data$DC),max(data$DC),length=length(data$DC)),fit.GA.2[["prob"]][1]*
dGA(seq(min(data$DC),max(data$DC),length=length(data$DC)), mu = mu.hat1, sigma =
sigma.hat1),lty=2,lwd=3,col=2)
lines(seq(min(data$DC),max(data$DC),length=length(data$DC)),fit.GA.2[["prob"]][2]*
dGA(seq(min(data$DC),max(data$DC),length=length(data$DC)), mu = mu.hat2, sigma =
sigma.hat2),lty=2,lwd=3,col=3)
lines(seq(min(data$DC),max(data$DC),length=length(data$DC)),

fit.GA.2[["prob"]][1]*dGA(seq(min(data$DC),max(data$DC),length=length(data$DC)),
mu = mu.hat1, sigma = sigma.hat1) +

fit.GA.2[["prob"]][2]*dGA(seq(min(data$DC),max(data$DC),length=length(data$DC)),
mu = mu.hat2, sigma = sigma.hat2),
      lty = 1, lwd = 3, col = 1)
```



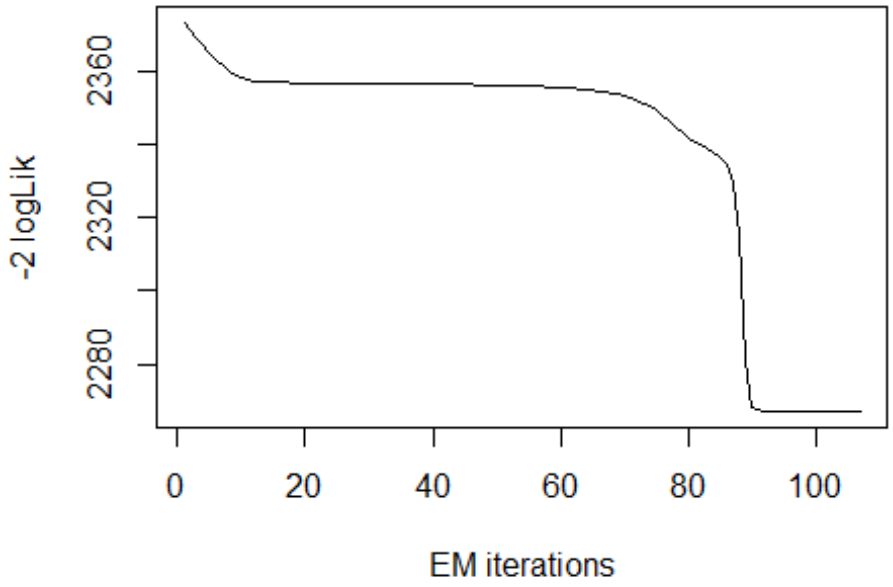
In the last graph the black segment is the overall mixture of the two gamma distributions. The first Gamma distribution represents 22% of the mixture while the second one accounts for the 78%.

We can also do the same analysis with a mixture of 2 WEIBULL distributions.

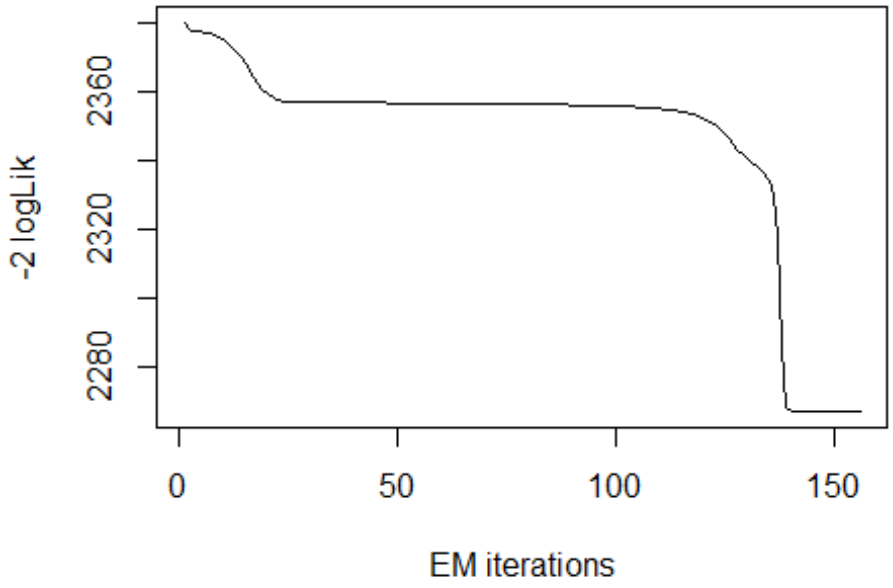
```
fit.WEI.2 <- gamlssMXfits(n = 5, data$DC~1, family = WEI, K = 2, data = NULL)
```



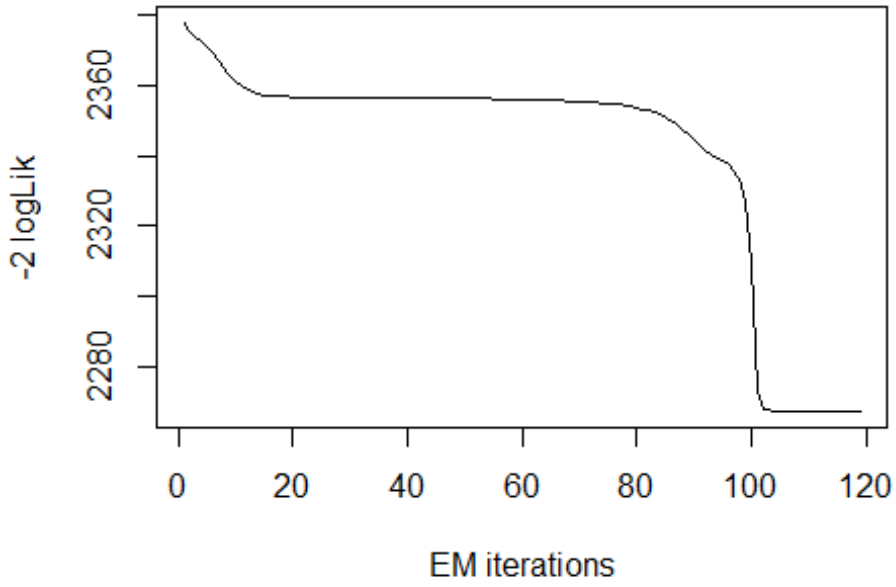

```
model= 1
```



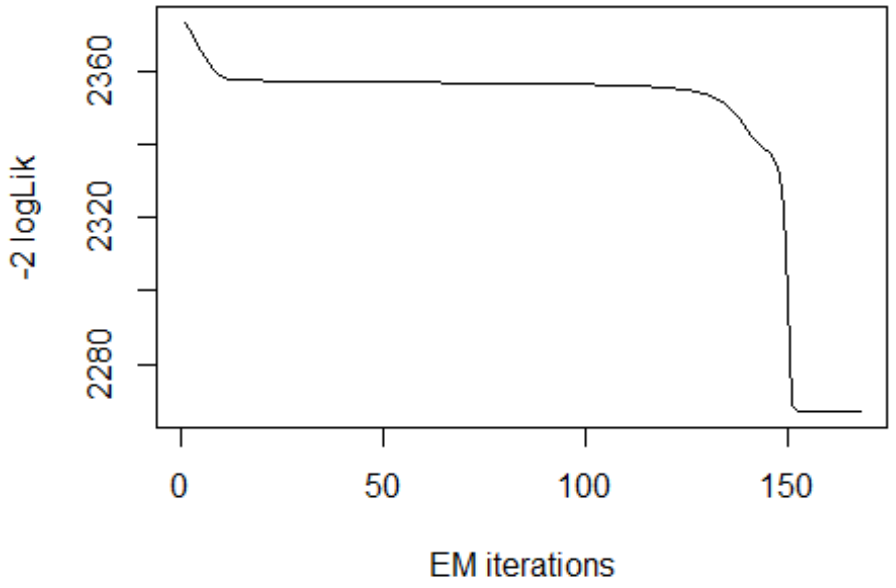

```
model= 2
```




```
model= 3
```

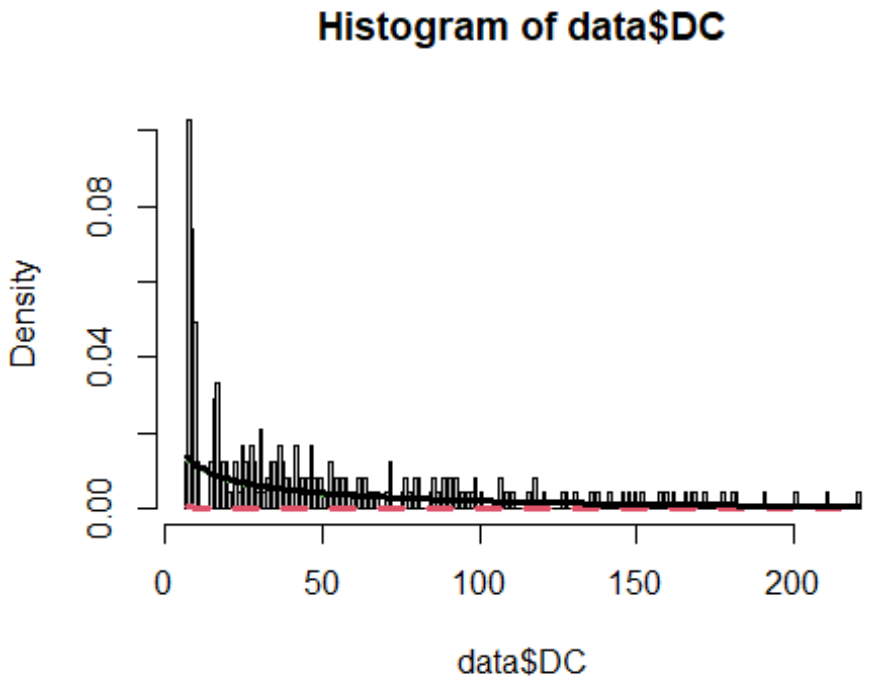



```
model= 4
```




```
 model= 5

logLik(fit.WEI.2)
```

```
 'log Lik.' -1133.583 (df=5)

fit.WEI.2$prob

 [1] 0.2115044 0.7884956

mu.hat1 <- exp(fit.WEI.2[["models"]][[1]][["mu.coefficients"]])
sigma.hat1 <- exp(fit.WEI.2[["models"]][[1]][["sigma.coefficients"]])
mu.hat2 <- exp(fit.WEI.2[["models"]][[2]][["mu.coefficients"]])
sigma.hat2 <- exp(fit.WEI.2[["models"]][[2]][["sigma.coefficients"]])

hist(data$DC, breaks = 197,freq = FALSE)
lines(seq(min(data$DC),max(data$DC),length=length(data$DC)),fit.WEI.2[["prob"]][1]
*dGA(seq(min(data$DC),max(data$DC),length=length(data$DC)), mu = mu.hat1, sigma =
sigma.hat1),lty=2,lwd=3,col=2)
lines(seq(min(data$DC),max(data$DC),length=length(data$DC)),fit.WEI.2[["prob"]][2]
*dGA(seq(min(data$DC),max(data$DC),length=length(data$DC)), mu = mu.hat2, sigma =
sigma.hat2),lty=2,lwd=3,col=3)
lines(seq(min(data$DC),max(data$DC),length=length(data$DC)),

fit.WEI.2[["prob"]][1]*dGA(seq(min(data$DC),max(data$DC),length=length(data$DC)),
mu = mu.hat1, sigma = sigma.hat1) +

fit.WEI.2[["prob"]][2]*dGA(seq(min(data$DC),max(data$DC),length=length(data$DC)),
mu = mu.hat2, sigma = sigma.hat2),
      lty = 1, lwd = 3, col = 1)
```



MULTIVARIATE ANALYSIS: PRINCIPAL COMPONENT ANALYSIS

Principal component analysis (PCA) is a procedure used in statistics with two aims:

• first, PCA allows us to summarize the information contained into a dataset from the point of view of the variables.

• the second aim, related to the first one, is to find the best representation of the information contained in the data matrix. PCA is particularly useful when the available variables are highly correlated. Correlation indicates that there is redundancy in the data. Due to this redundancy, PCA can be used to reduce the original variables into a smaller number of new variables explaining most of the variance in the original variables. From a geometrical point of view, each variable of the data matrix represents a different dimension in the space; thus, our main goal is to reduce the original dimensionality of the dataset in order to reach a 3 or less dimensions that can be graphically represented. Moreover, we must precise that the PCA can be applied only to numerical variables.

```
corr = cor(data[,-8])
corr
```

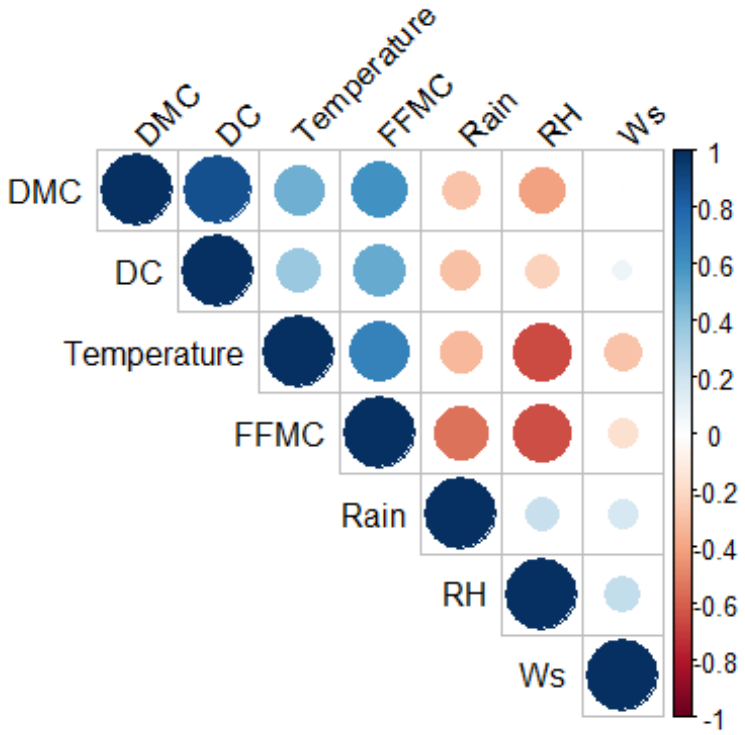
```
             Temperature         RH             Ws       Rain       FFMC
Temperature   1.0000000 -0.6514003 -0.2845098897 -0.3264919  0.6765681
RH           -0.6514003  1.0000000  0.2440483822  0.2223561 -0.6448735
Ws           -0.2845099  0.2440484  1.0000000000  0.1715062 -0.1665483
Rain         -0.3264919  0.2223561  0.1715061807  1.0000000 -0.5439062
FFMC          0.6765681 -0.6448735 -0.1665482728 -0.5439062  1.0000000
DMC           0.4856869 -0.4085192 -0.0007209737 -0.2887729  0.6036076
DC            0.3762835 -0.2269411  0.0791345143 -0.2980231  0.5073967
                     DMC          DC
Temperature  0.4856869230  0.37628353
RH          -0.4085191880 -0.22694112
Ws          -0.0007209737  0.07913451
Rain        -0.2887729260 -0.29802308
FFMC         0.6036076410  0.50739666
DMC          1.0000000000  0.87592466
DC           0.8759246607  1.00000000
```

The PCA helps to visualize in a lower dimensional space the variables of our dataset. Since there are 7 numerical variables a complete multivariate analysis would require analyzing d(d-1)/2 = 7x6/2 = 21 scatterplot. Exactly for this reason, PCA analysis summarizes the information presented into a data matrix which means that it allows to represent only the most representative variables which are those that together, explain the most variability in the original space. PCA is used to extract the important information from a multivariate data set and to express this information as a set of few new variables called principal components (PCs). These new variables correspond to a linear combination of the originals. In order to do so, variables must have some sort of correlation, positive or negative. In fact, Principal Component Analysis would be not useful on a data set with only uncorrelated variable.

So let's take a look at the correlation.

```
library(corrplot)
```

```
corrplot(corr, type='upper',order='hclust',tl.col='black',tl.srt=45)
```



DATA PREPARATION FOR PCA

First, we compute the mean and variance of our dataset:

```
apply(data[-8],2,var)
```

```
 Temperature           RH           Ws         Rain         FFMC          DMC
   13.162670  219.874333     7.903887     4.012837  205.912204  153.587434
          DC
 2272.009994
```

```
apply(data[-8],2,mean)
```

```
 Temperature           RH           Ws         Rain         FFMC          DMC
   32.152263   62.041152   15.493827     0.762963   77.842387   14.680658
          DC
   49.430864
```

```
By computing var and mean we can see how there are some variables that are not so
much comparable, an example is RH that is a percentage and assume big values, so
is usefull in order to perform a PCA to standardize our data by scaling them.
```

```
data.scaled = apply(data[-8],2,scale)
head(data.scaled)
```

```
      Temperature          RH         Ws       Rain       FFMC        DMC
 [1,]  -0.8688614 -0.33997153  0.8914370 -0.3808708 -0.8461805 -0.9102414
 [2,]  -0.8688614 -0.07021453 -0.8870457  0.2680887 -0.9367751 -0.8537581
```

```
 [3,]  -1.6957543  1.34600973  2.3142231  6.1586438 -2.1423802 -0.9828629
 [4,]  -1.9713852  1.81808448 -0.8870457  0.8671282 -3.4316110 -1.0796914
 [5,]  -1.4201233  1.00881347  0.1800439 -0.3808708 -0.9088999 -0.9425176
 [6,]  -0.3175995  0.33442097 -0.5313491 -0.3808708  0.3315493 -0.7165844
              DC
 [1,] -0.8775901
 [2,] -0.8775901
 [3,] -0.8880798
 [4,] -0.8922757
 [5,] -0.7391255
 [6,] -0.5712896
```

```
corr.matrix = cor(data.scaled)
eigen.data = eigen(corr.matrix)
str(eigen.data)
```

The amount of variance retained by each PC is measured by the so-called eigenvalue.

```
 List of 2
  $ values : num [1:7] 3.476 1.333 0.838 0.699 0.332 ...
  $ vectors: num [1:7, 1:7] 0.429 -0.383 -0.128 -0.301 0.476 ...
  - attr(*, "class")= chr "eigen"
```

```
colnames(eigen.data$vectors) = c('PC1','PC2','PC3','PC4','PC5','PC6','PC7')
rownames(eigen.data$vectors) = c('Temperature','RH','Ws','Rain','FFMC','DMC','DC')
eigen.data
```

```
 eigen() decomposition
 $values
 [1] 3.4762530 1.3331429 0.8383805 0.6987611 0.3320977 0.2198356 0.1015293
```

The eigen value for each of our 7 components

```
 $vectors
                     PC1         PC2         PC3        PC4           PC5
 Temperature  0.4287232  0.24224011 -0.24585384  0.1046560  0.8102427837
 RH          -0.3831901 -0.31235291  0.43416435 -0.3099363  0.5462918399
 Ws          -0.1277729 -0.64099701 -0.13269689  0.7331696  0.1085414212
 Rain        -0.3011609 -0.09353486 -0.84656399 -0.2581006  0.0360748013
 FFMC         0.4761081  0.07276284  0.07048901  0.2577245 -0.1101748071
 DMC          0.4342002 -0.39579637 -0.10436461 -0.3048576 -0.1408900768
 DC           0.3835743 -0.51203540  0.03028991 -0.3598236  0.0008507796
                     PC6         PC7
 Temperature  0.17118077 -0.01797069
 RH          -0.38687704 -0.15143720
 Ws           0.07166787 -0.02707992
 Rain        -0.33718285  0.04767811
 FFMC        -0.82663699  0.03418802
 DMC          0.10021872 -0.72186665
 DC           0.09392675  0.67192040
```

In the following code, we will put the computed eigen vectors of the first two Principal Components into a "phi" matrix. In the matrix are displayed the value of the three PCs for each variable. We picked first 2 PCs thanks to the Kaiser's rule.

```
phi = eigen.data$vectors[,1:2]
phi = phi
row.names(phi) = c('Temperature','RH','Ws','Rain','FFMC','DMC','DC')
colnames(phi) = c('PC1','PC2')
phi
```

```
                    PC1         PC2
 Temperature  0.4287232  0.24224011
 RH          -0.3831901 -0.31235291
 Ws          -0.1277729 -0.64099701
 Rain        -0.3011609 -0.09353486
 FFMC         0.4761081  0.07276284
 DMC          0.4342002 -0.39579637
 DC           0.3835743 -0.51203540
```

PROPORTION OF VARIANCE EXPLAINED (PVE) and CUMULATIVE PVE

According to the PVE, the first PC explains 50% of the variability, the second one explains 19% of the variability . According to the CPVE we can retail the first two principal components because they explain 69% of the variability of our data.

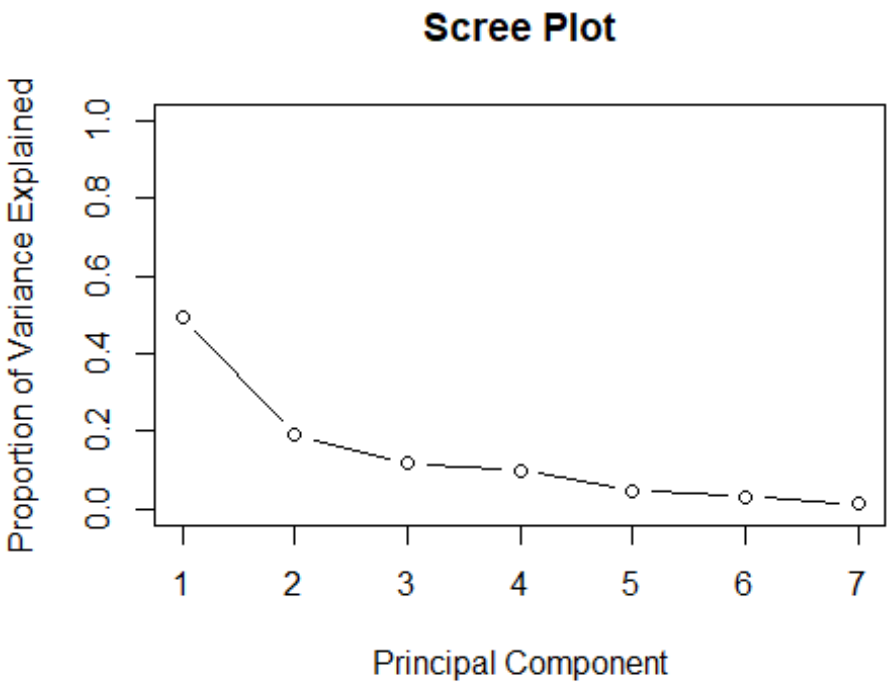
```
PVE = eigen.data$values/sum(eigen.data$values)
round(PVE,3)
```

```
 [1] 0.497 0.190 0.120 0.100 0.047 0.031 0.015
```

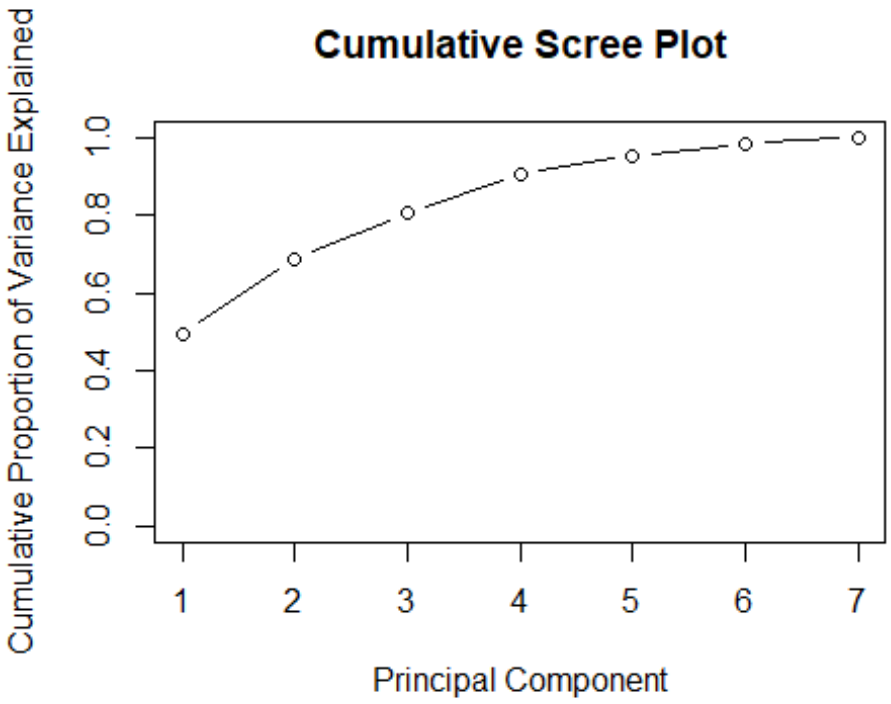
```
cumsum(PVE)
```

```
 [1] 0.4966076 0.6870565 0.8068252 0.9066482 0.9540907 0.9854958 1.0000000
```

```
plot(PVE,xlab = 'Principal Component', ylab = 'Proportion of Variance
Explained',main = 'Scree Plot',ylim = c(0,1),type = 'b')
```




```
plot(cumsum(PVE),xlab = 'Principal Component',ylab = 'Cumulative Proportion of
Variance Explained', main = 'Cumulative Scree Plot',ylim = c(0,1),type = 'b')
```

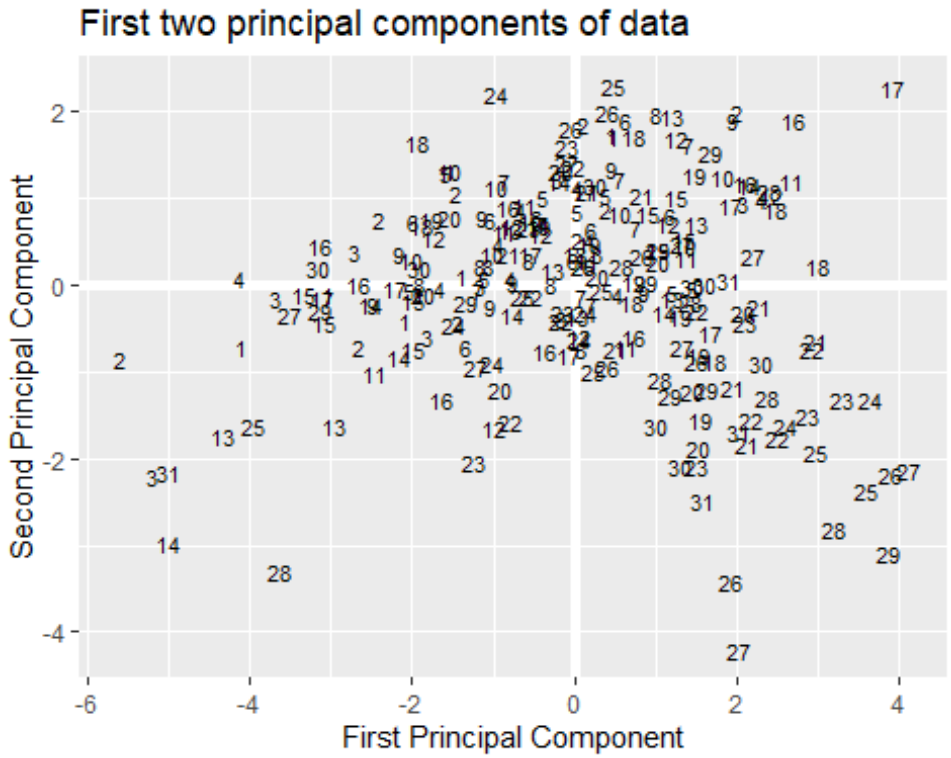



```
PC1 = data.scaled %*% phi[,1]
PC2 = data.scaled %*% phi[,2]
```

```
PC = data.frame(data5[1],PC1,PC2)
head(PC)
```

```
  day        PC1         PC2
1   1 -1.3761468  0.10799124
2   2 -1.4663222  1.07408577
3   3 -5.1806327 -2.20281153
4   4 -4.1345293  0.07657530
5   5 -2.0291939 -0.05353024
6   6 -0.4541335  0.79509166
```

```
ggplot(PC,aes(PC1,PC2),loadings =
TRUE)+modelr::geom_ref_line(h=0)+modelr::geom_ref_line(v=0)+geom_text(aes(label=da
y), size=3)+xlab('First Principal Component')+ ylab('Second Principal Component')
+ ggtitle('First two principal components of data')
```

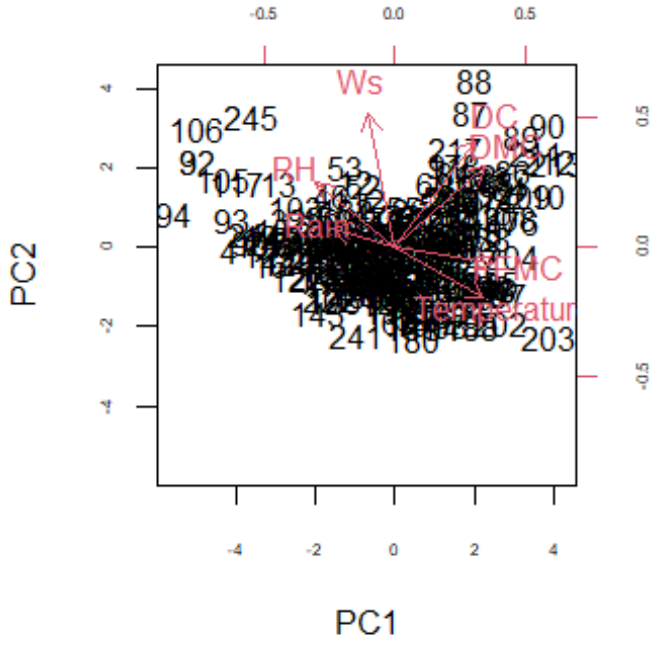



```
pr.out = prcomp(data[,-8], scale = TRUE)
pr.out$rotation[,1:2]
```

```
                   PC1         PC2
Temperature  0.4287232 -0.24224011
RH          -0.3831901  0.31235291
Ws          -0.1277729  0.64099701
Rain        -0.3011609  0.09353486
FFMC         0.4761081 -0.07276284
DMC          0.4342002  0.39579637
DC           0.3835743  0.51203540
```

Principal Components Biplot: The biplot will allows us to visualize the computed scores and the original variable in the first two principal components' space.

```
biplot(pr.out,cex.axis = 0.5,scale = 0)
```



In order to interpret correctly the biplot result, we have to say that this graphical representation gives us information about the correlation of the original variables in the principal components space. Indeed, in this graphical representation the angles made by different variables, which are represented by narrows, indicates the correlation between variables.

CLUSTER ANALYSIS AND CLUSTER VALIDATION

The cluster analysis is one of the most important statistical tools to discover knowledge about multidimensional data. The aim of the cluster analysis is to find clusters of similar units inside the data set. When we want to perform a cluster analysis there are a lot of factors that must be taken in consideration. First, the CA can be performed with different type of Clustering techniques: The Agglomerative and Divisive Hierarchical, The Partitional and the Model-based Technique. However, most of the Clustering Techniques cannot be done without starting from the Distance Matrix. Indeed, in Clustering Analysis we want at the same time to maximize the similarity within

the cluster and the dissimilarity between clusters. In doing so, the first step is undoubtedly the computation of the Distance Matrix (D). During the construction of D, we can use different kind of measuring techniques, the most famous are: the Euclidian distance and the Manhattan distance (City block). Both measuring techniques have different peculiarities that make them fit more to certain dataset respect to others. In general, we can say that the E.D. and the M.D are specifical cases of the so called Minkosky Distance. The Euclidian tends to be more affected by outliers due to its formula based on the square of the value while the Manhattan Distance is more robust respect with outliers, due to the module in its formula. Once set the measuring method that will be applied to our CA, we must consider and select the most advantageous Clustering Technique to our dataset taking in consideration that the CTs perform in different way and yield to different results due to their peculiar algorithms and the dataset they are applied on. It is therefore clear that the best clustering result can be obtained only after a series of trials according to the best combination of distances, clustering methods and number of clusters. This process of trials and errors is called cluster validation. Cluster validation is therefore defined as the set of statistical tools used to assess the quality of the results of the Cluster Analysis. This statistical process is made by different steps:

- Assessing cluster tendency

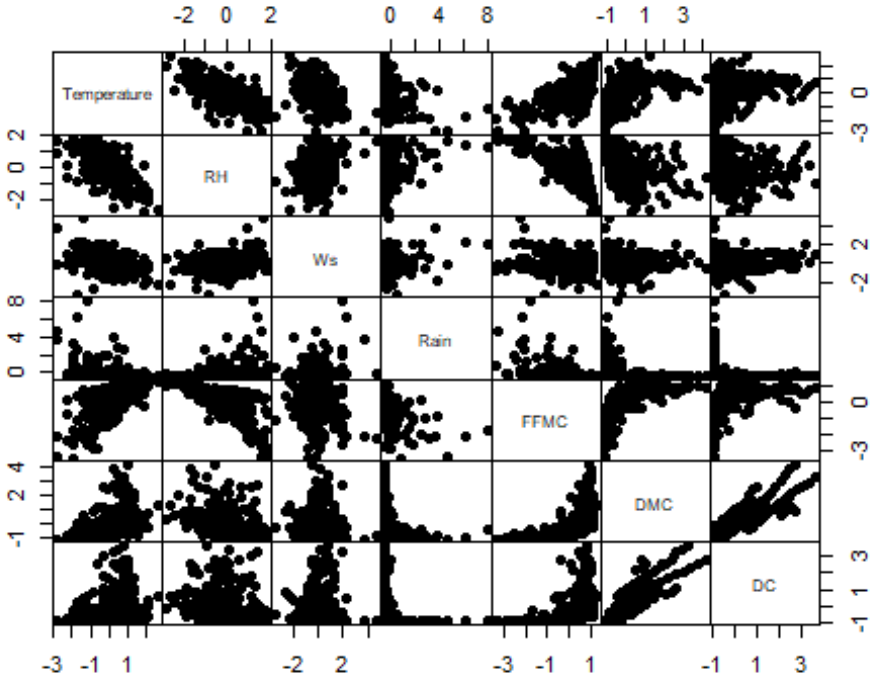
- Determining the best number of clusters

- Cluster validation statistics.

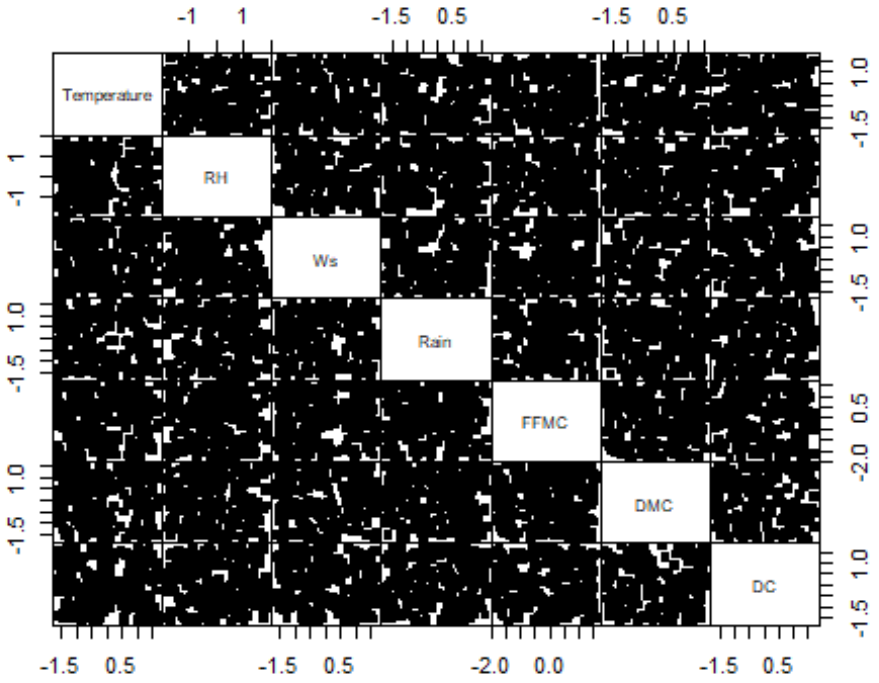
Assessing clustering tendency

In order to measure the clustering tendency of our dataset we would need another to compare with. This new dataset randomly generated will be our benchmark and starting from it, we will rate the clustering tendency of our data. It will have the same minimum, maximum and number of points. From here and after we will use a randomly generate dataset as benchmark in order to compare the result of our dataset with it and confirm certain results

```
data.scaled = scale(data[,-8],scale=TRUE)
pairs(data.scaled, gap =0,pch=16)
```

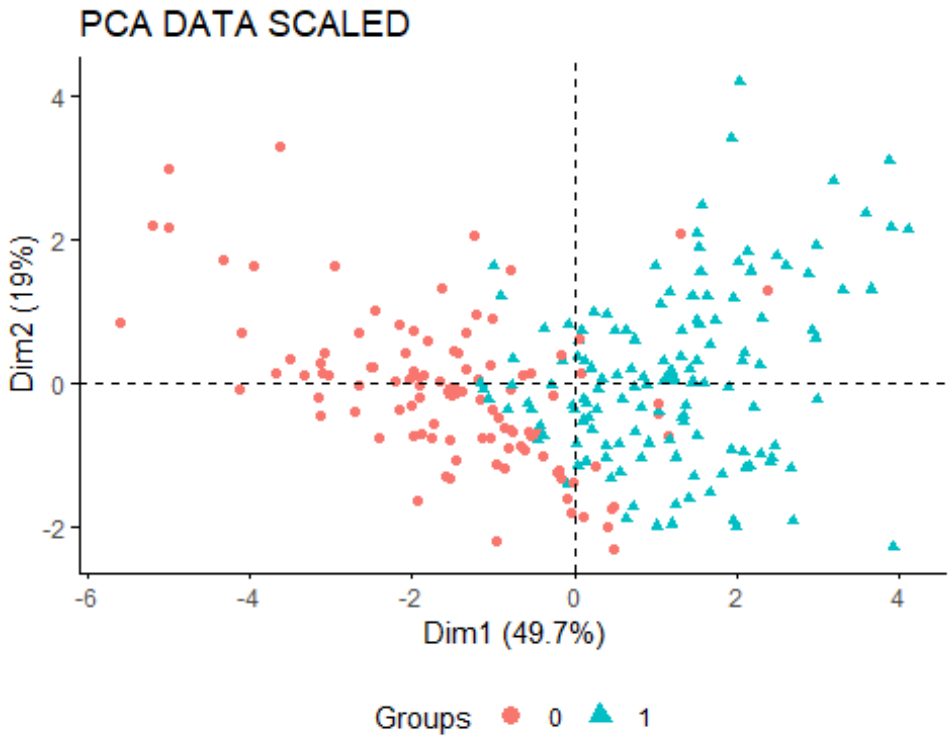



```
random_data = apply(data.scaled,2, function(x){runif(length(x),min(x),max(x))})
random_data = as.data.frame(random_data)
random_data = scale(random_data)
pairs(random_data,gap=0,pch=15)
```

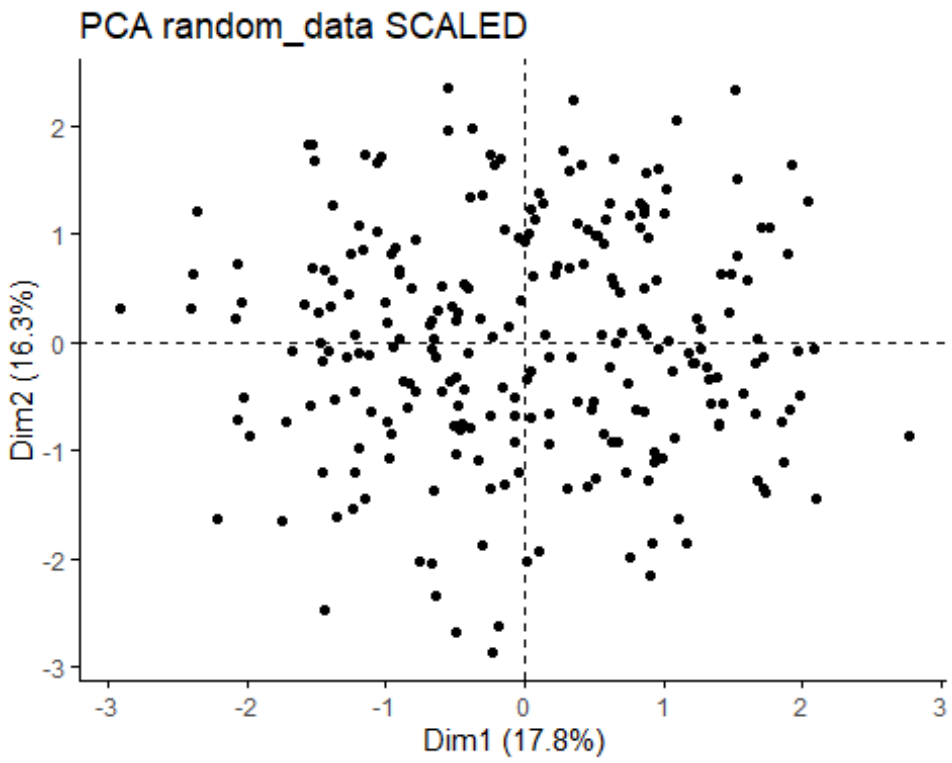


As we can see from the graphical representation, the original dataset presents a grouping propensity, while the same grouping tendency cannot be seen in the random generated set of data. In the following codes, we will plot both the mean-centered data and the random ones in the PCs space:

```
fviz_pca_ind(prcomp(data.scaled),habillage = data$Classes,   title = 'PCA DATA
SCALED',geom = 'point', ggtheme=theme_classic(),legend='bottom')
```




```
fviz_pca_ind(prcomp(random_data),title = 'PCA random_data SCALED',geom = 'point',
ggtheme=theme_classic(),legend='bottom')
```



Clustering tendency can be evaluated through two main indicators: a statistical one, the Hopkins index and another one, based more on a visual and graphic result, the VAT algorithm. As we did previously, we are going to use the random dataset as benchmark in order to compare the two result and analyze the outcomes:

HOPKINS INDEX

```
set.seed(123)
hopkins(data.scaled,n=nrow(data.scaled)-1)
```

```
$H
[1] 0.1627094
```

The closer the Hopkins statistics is to 0, the more the dataset has clustering tendency. We can state that 0.16 is a good result to assess the clustering tendency of our dataset.

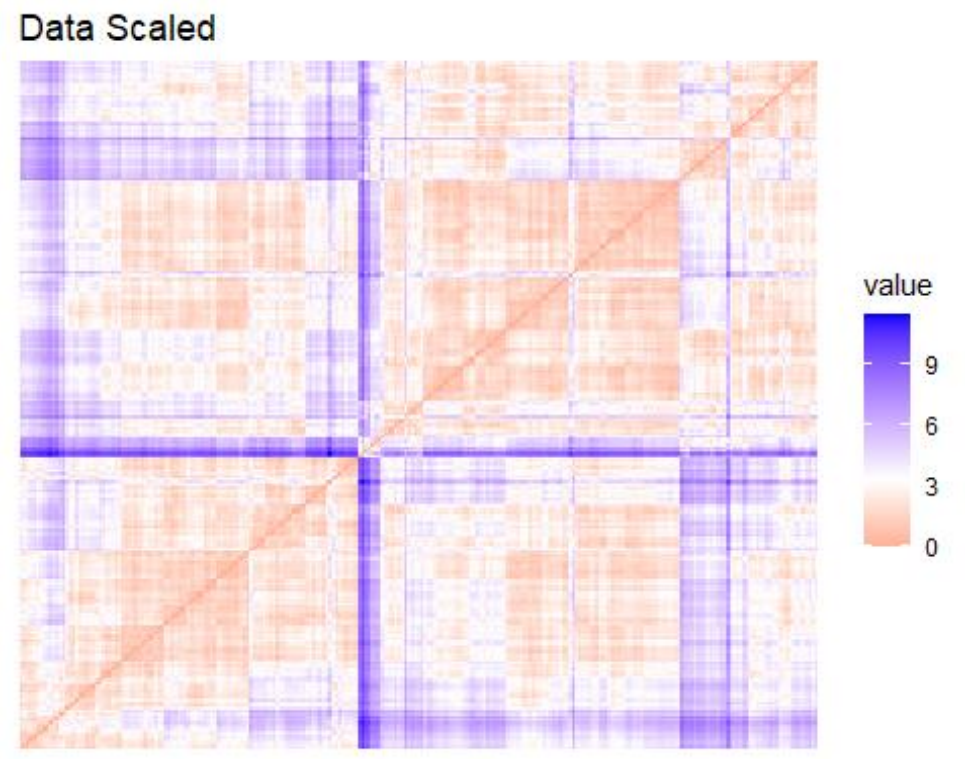
```
set.seed(123)
hopkins(random_data,n=nrow(random_data)-1)
```

```
$H
[1] 0.4990544
```

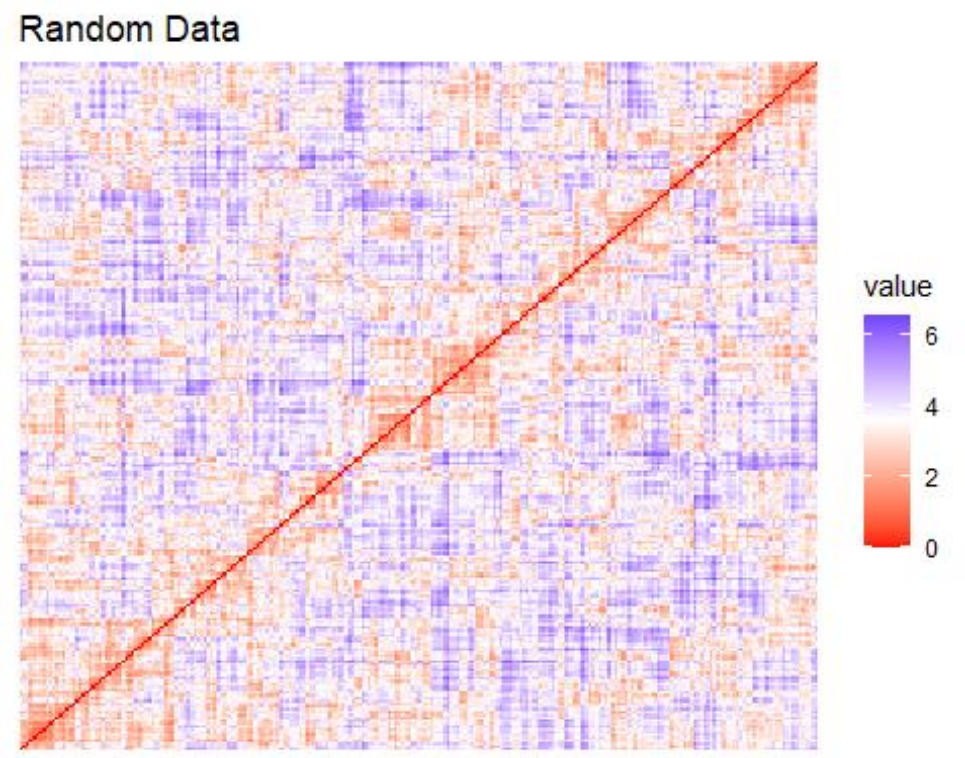
As it was expected, the dataset randomly generated shows a much higher Hopkins's value.

VAT ANALYSIS (Visual method)

```
fviz_dist(dist(data.scaled),show_labels = FALSE)+ labs(title = 'Data Scaled')
```




```
fviz_dist(dist(random_data),show_labels = FALSE)+ labs(title = 'Random Data')
```



The VAT algorithm is based on the ordered dissimilarity matrix computed on the observations of the two datasets. In particular, the colours are proportional to the level of dissimilarity between the observations. In the two representations the red colour indicates a higher similarity between observations while the blue colour indicates a lower similarity between units. The VAT algorithm confirms what was already pointed out from the outcome of the Hopkin's index.

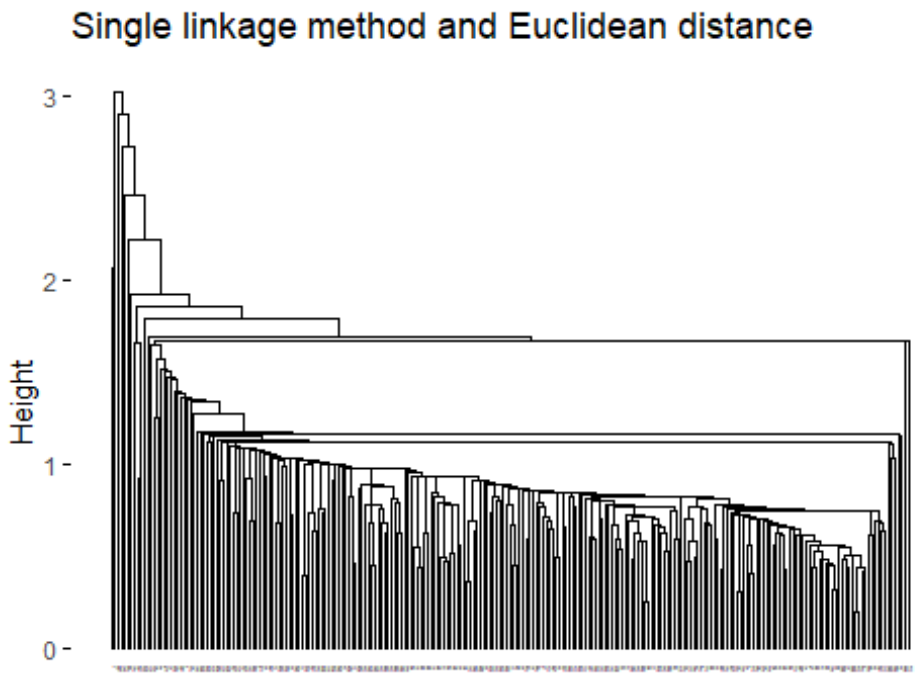
Hard clustering approach

Agglomerative Hierarchical Method

Now, we can proceed with the cluster analysis. The first clustering algorithm that will be applied is the agglomerative hierarchical clustering, that is a "bottom-up" approach: each observation starts in its own cluster (leaf), and pairs of clusters are merged as one moves up the hierarchy.

As stated previously, every cluster analysis must start from the computation of the distance/dissimilarity matrix. The dissimilarity matrix will be computed using both the Euclidean and the Manhattan distance. The analysis will begin with different agglomerative hierarchical approaches using different linkage method: **single, complete, average, ward and centroid.**

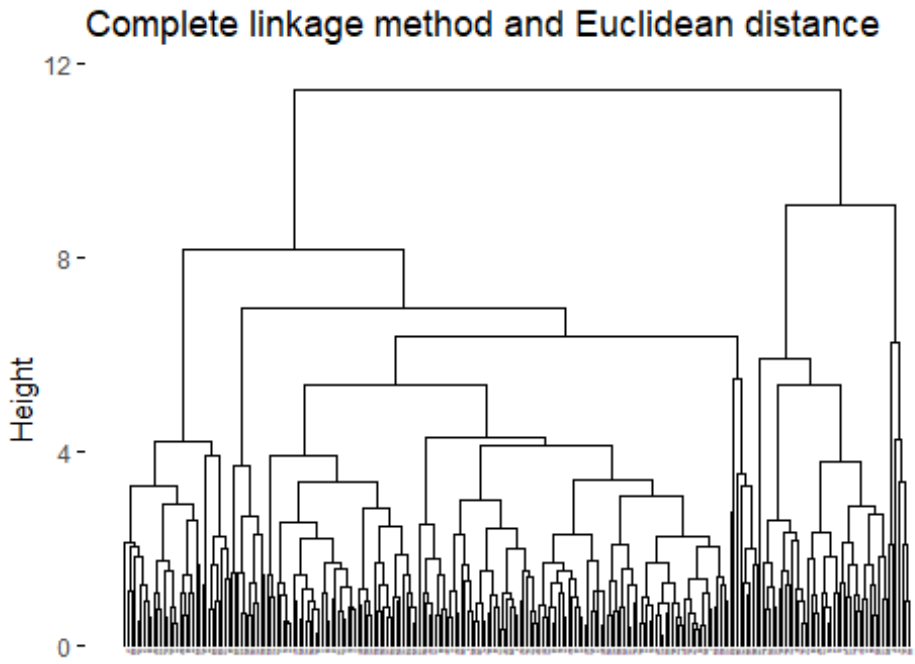
```
res.dist = dist(data.scaled,method = 'euclidean')
res.hc.es = hclust(d = res.dist,method = 'single')
fviz_dend(res.hc.es,cex = 0,main = 'Single linkage method and Euclidean distance')
```




```
res.coph = cophenetic(res.hc.es)
cor(res.dist,res.coph)
```

```
 [1] 0.6814561
```

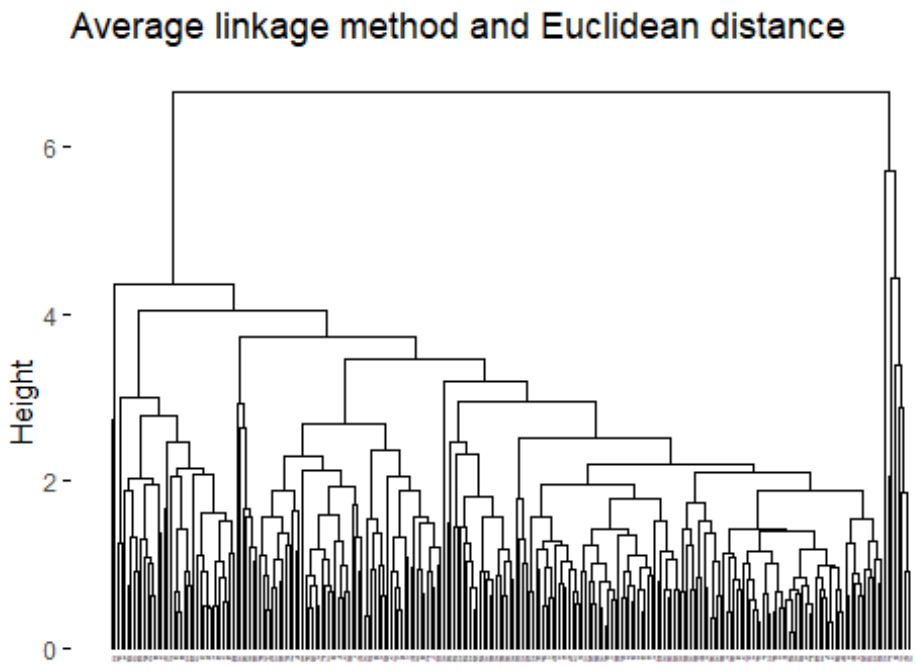
```
res.dist = dist(data.scaled,method = 'euclidean')
res.hc.ec = hclust(d = res.dist,method = 'complete')
fviz_dend(res.hc.ec,cex = 0,main = 'Complete linkage method and Euclidean
distance')
```




```
res.coph = cophenetic(res.hc.ec)
cor(res.dist,res.coph)
```

```
 [1] 0.5813788
```

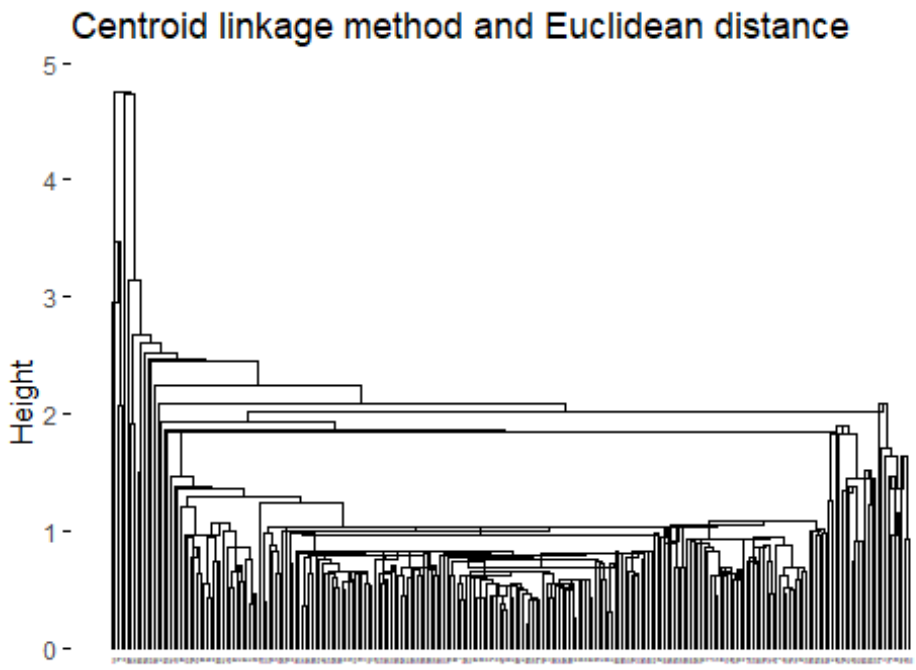
```
res.dist = dist(data.scaled,method = 'euclidean')
res.hc.ea = hclust(d = res.dist,method = 'average')
fviz_dend(res.hc.ea,cex = 0,main = 'Average linkage method and Euclidean
distance')
```




```
res.coph = cophenetic(res.hc.ea)
cor(res.dist,res.coph)
```

```
 [1] 0.7355965
```

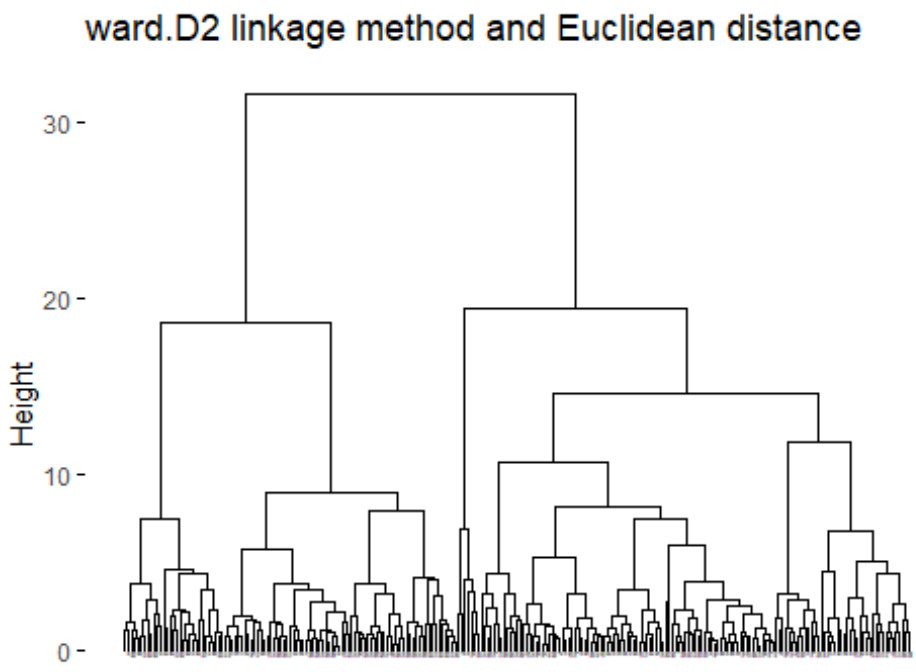
```
res.dist = dist(data.scaled,method = 'euclidean')
res.hc.ec = hclust(d = res.dist,method = 'centroid')
fviz_dend(res.hc.ec,cex = 0,main = 'Centroid linkage method and Euclidean
distance')
```




```
res.coph = cophenetic(res.hc.ec)
cor(res.dist,res.coph)
```

```
 [1] 0.7159735
```

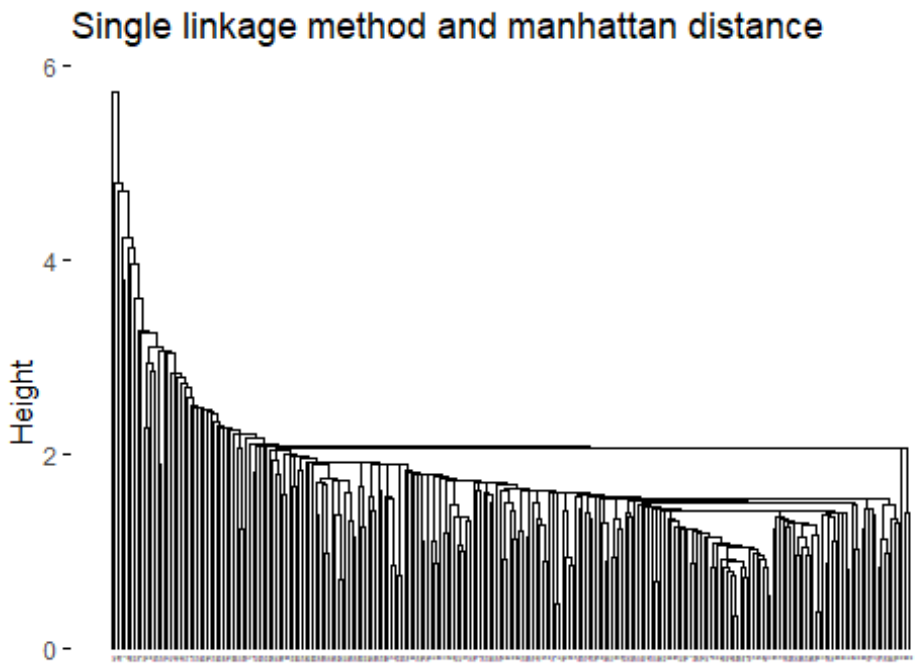
```
res.dist = dist(data.scaled,method = 'euclidean')
res.hc.ew = hclust(d = res.dist,method = 'ward.D2')
fviz_dend(res.hc.ew,cex = 0,main = 'ward.D2 linkage method and Euclidean
distance')
```




```
res.coph = cophenetic(res.hc.ew)
cor(res.dist,res.coph)
```

```
 [1] 0.4520908
```

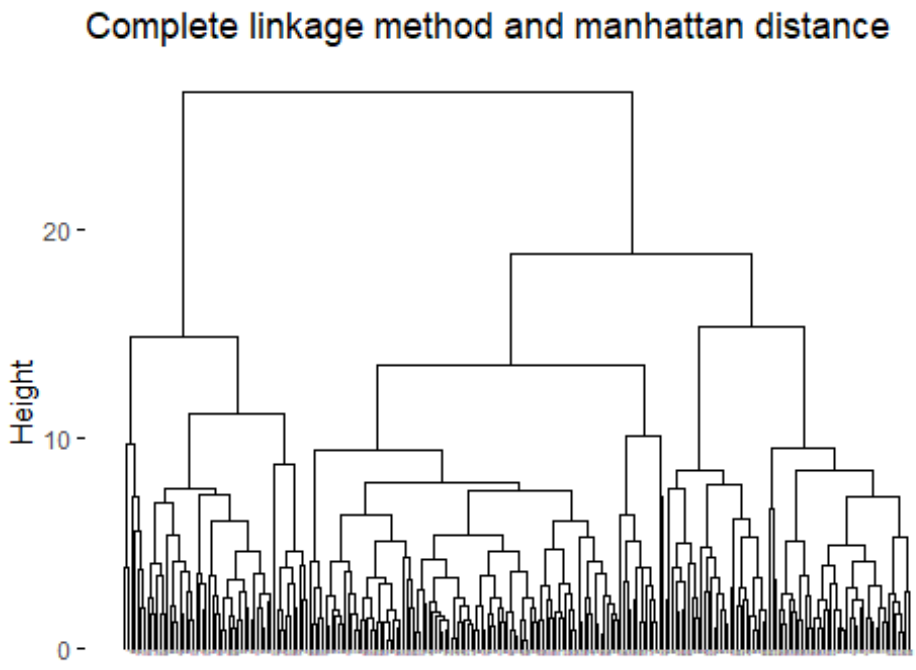
```
res.dist = dist(data.scaled,method = 'manhattan')
res.hc.ms = hclust(d = res.dist,method = 'single')
fviz_dend(res.hc.ms,cex = 0,main = 'Single linkage method and manhattan distance')
```




```
res.coph = cophenetic(res.hc.ms)
cor(res.dist,res.coph)
```

```
 [1] 0.5853402
```

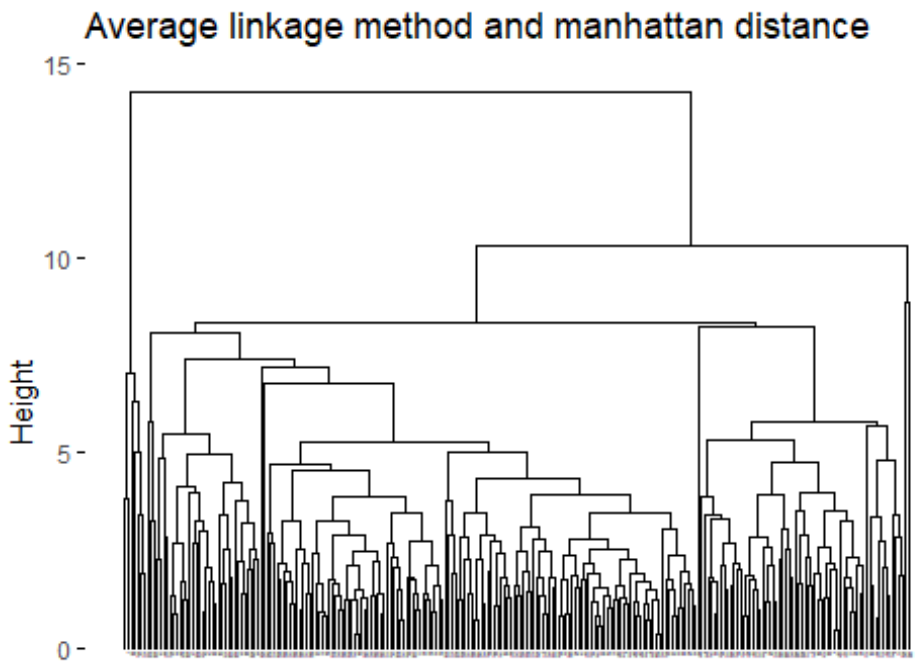
```
res.dist = dist(data.scaled,method = 'manhattan')
res.hc.mc = hclust(d = res.dist,method = 'complete')
fviz_dend(res.hc.mc,cex = 0,main = 'Complete linkage method and manhattan
distance')
```




```
res.coph = cophenetic(res.hc.mc)
cor(res.dist,res.coph)
```

```
 [1] 0.5544152
```

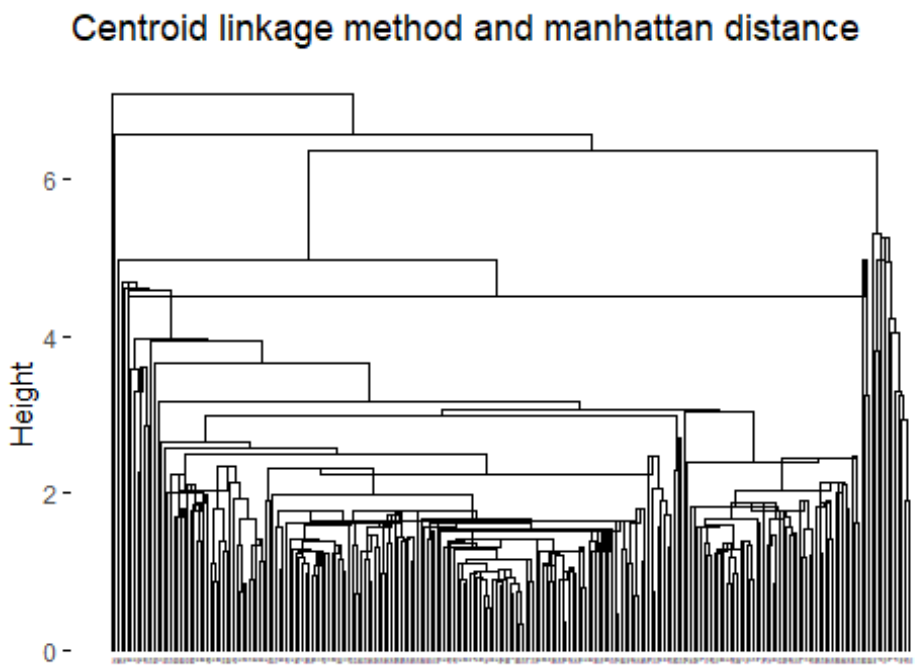
```
res.dist = dist(data.scaled,method = 'manhattan')
res.hc.ma = hclust(d = res.dist,method = 'average')
fviz_dend(res.hc.ma,cex = 0,main = 'Average linkage method and manhattan
distance')
```




```
res.coph = cophenetic(res.hc.ma)
cor(res.dist,res.coph)
```

```
 [1] 0.7015384
```

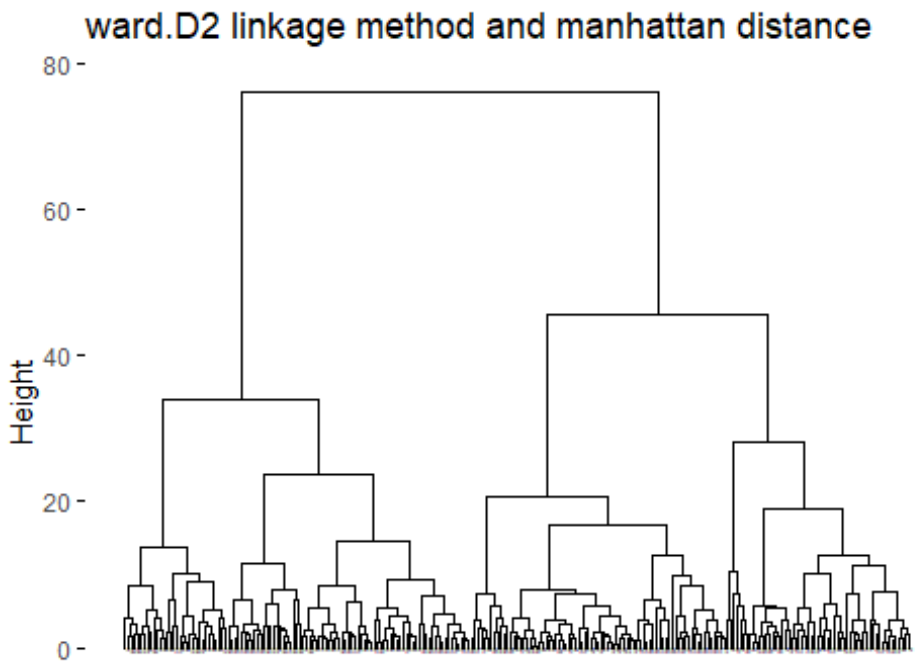
```
res.dist = dist(data.scaled,method = 'manhattan')
res.hc.mc = hclust(d = res.dist,method = 'centroid')
fviz_dend(res.hc.mc,cex = 0,main = 'Centroid linkage method and manhattan
distance')
```




```
res.coph = cophenetic(res.hc.mc)
cor(res.dist,res.coph)
```

```
 [1] 0.6527741
```

```
res.dist = dist(data.scaled,method = 'manhattan')
res.hc.mw = hclust(d = res.dist,method = 'ward.D2')
fviz_dend(res.hc.mw,cex = 0,main = 'ward.D2 linkage method and manhattan
distance')
```




```
res.coph = cophenetic(res.hc.mw)
cor(res.dist,res.coph)
```

```
 [1] 0.4615538
```

```
grid.arrange(fviz_nbclust(data.scaled, hcut, method="silhouette",
hc_metric="euclidean",hc_method= "single"),
fviz_nbclust(data.scaled, hcut, method="silhouette",
hc_metric="manhattan",hc_method= "single"),ncol=2,nrow=1)
```

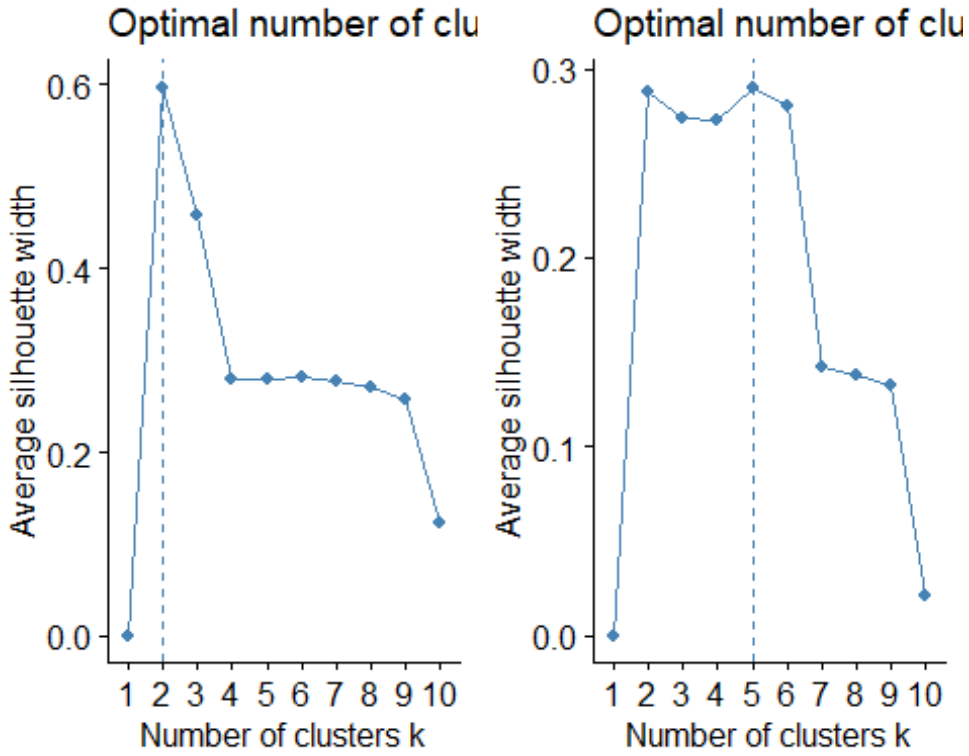
In order to select the best combination of agglomerative hierarchical clustering approach we will analyze the correlation between the cophenetic distances and the original distances. The closer to 1 is the value, the more accurately is the clustering solution. Also, the cophenetic dissimilarity or cophenetic distance of two units is a measure of how similar those two units have to be in order to be grouped into the same cluster. Geometrically speaking, the cophenetic distance between two units is the height of the dendrogram where the two branches that include the two units merge into a single branch (height of the fusion). Therefore, according to this correlation index, the following list of models can be considered as the best Agglomerative Hierarchical Clustering approaches for our dataset. However, since the majority of our variables have outliers, we will proceed in further analysis taking in consideration methods with either Euclidean or Manhattan distance. In fact, despite that the Euclidean approaches have the highest scores, we know that this kind of distance can be strongly influenced by the presence of outliers yielding to misleading results.

In this second step, we will be estimating the optimal number of clusters by which cut the dendrograms. We will be using two methods to determine the optimal number of K: direct methods like the Elbow method and the Silhouette method, and statistical methods like the Gap Statistic:

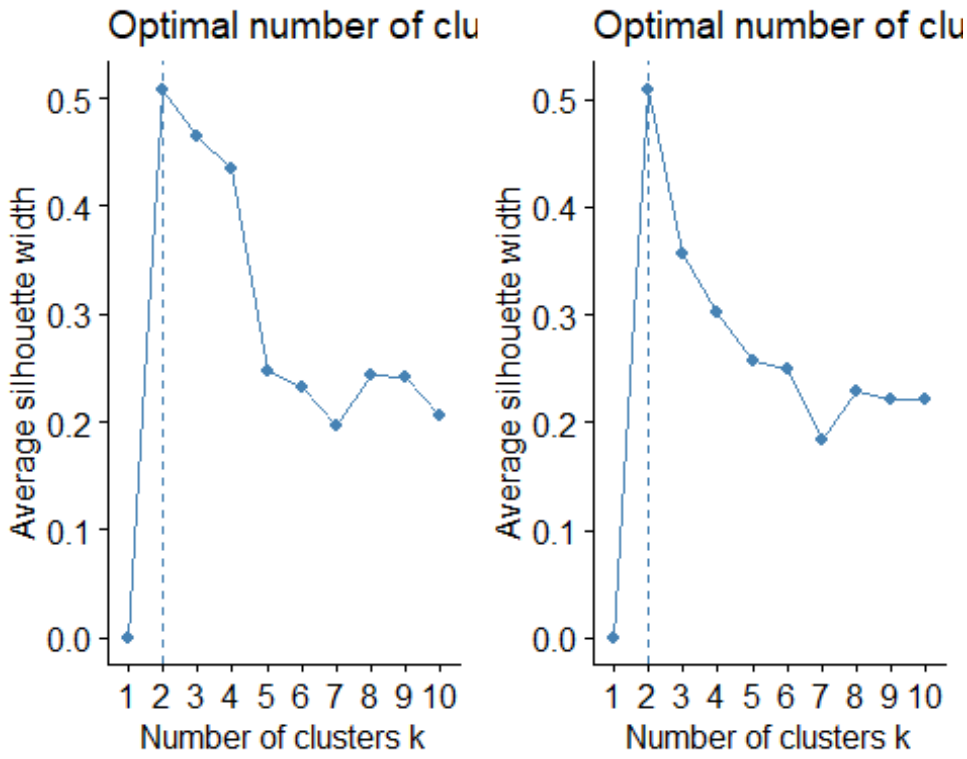
• Elbow method roughly measures the quality of a clustering by determining how compact clusters are in terms of within-cluster sum of squares (WSS), a classical measure of compactness or cohesion.

• The average silhouette method roughly measures the quality of a clustering by determining how well each unit lies within its cluster. It assumes values from -1 (units badly matched, perfect assignment for the neighboring cluster) to 1 (perfect assignment).

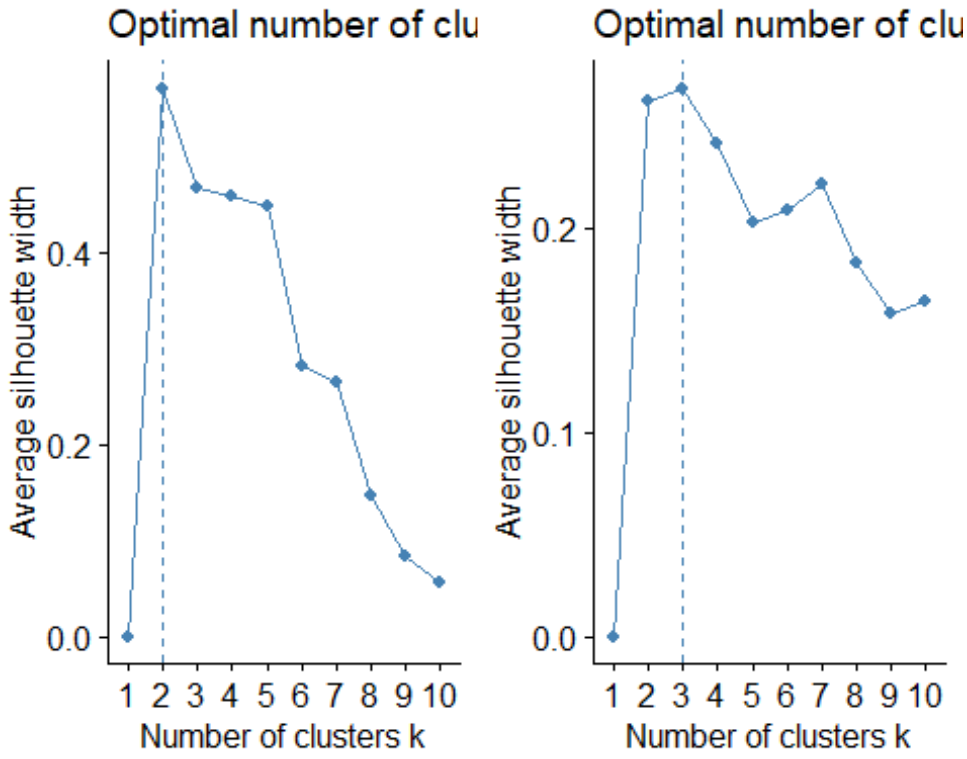
• The Gap statistic provides a statistical procedure to formalize the heuristic elbow method.



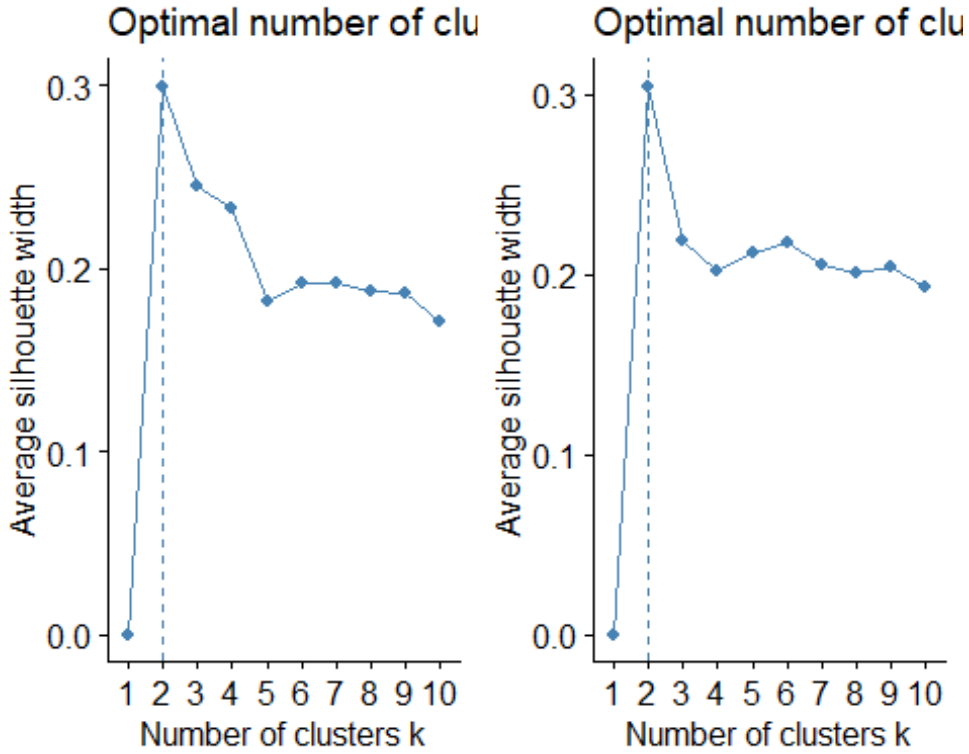

```
grid.arrange(fviz_nbclust(data.scaled, hcut, method="silhouette",
hc_metric="euclidean",hc_method= "average"),
fviz_nbclust(data.scaled, hcut, method="silhouette",
hc_metric="manhattan",hc_method= "average"),ncol=2,nrow=1)
```



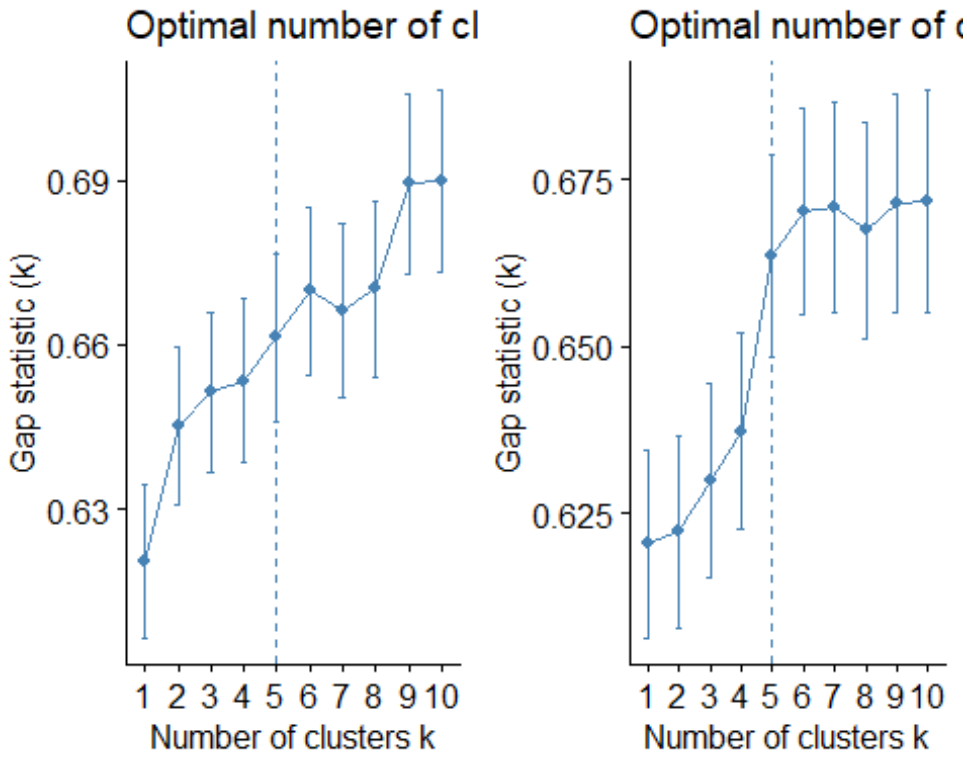

```
 grid.arrange(fviz_nbclust(data.scaled, hcut, method =
"silhouette",hc_metric="euclidean", hc_method= "centroid"),
 fviz_nbclust(data.scaled, hcut, method = "silhouette",hc_metric="euclidean",
hc_method= "ward.D2"),ncol=2,nrow=1)
```



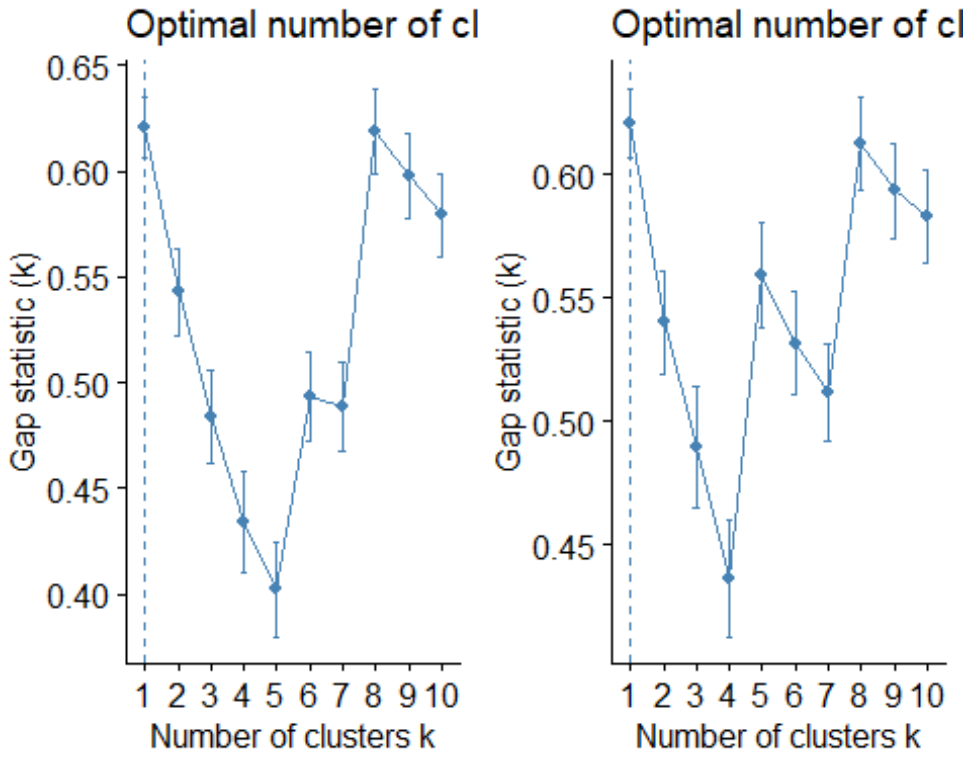
```
grid.arrange(fviz_nbclust(data.scaled, hcut, method="silhouette",
hc_metric="euclidean",hc_method= "complete"), fviz_nbclust(data.scaled, hcut,
method="silhouette", hc_metric="manhattan",hc_method= "complete"),ncol=2,nrow=1)
```



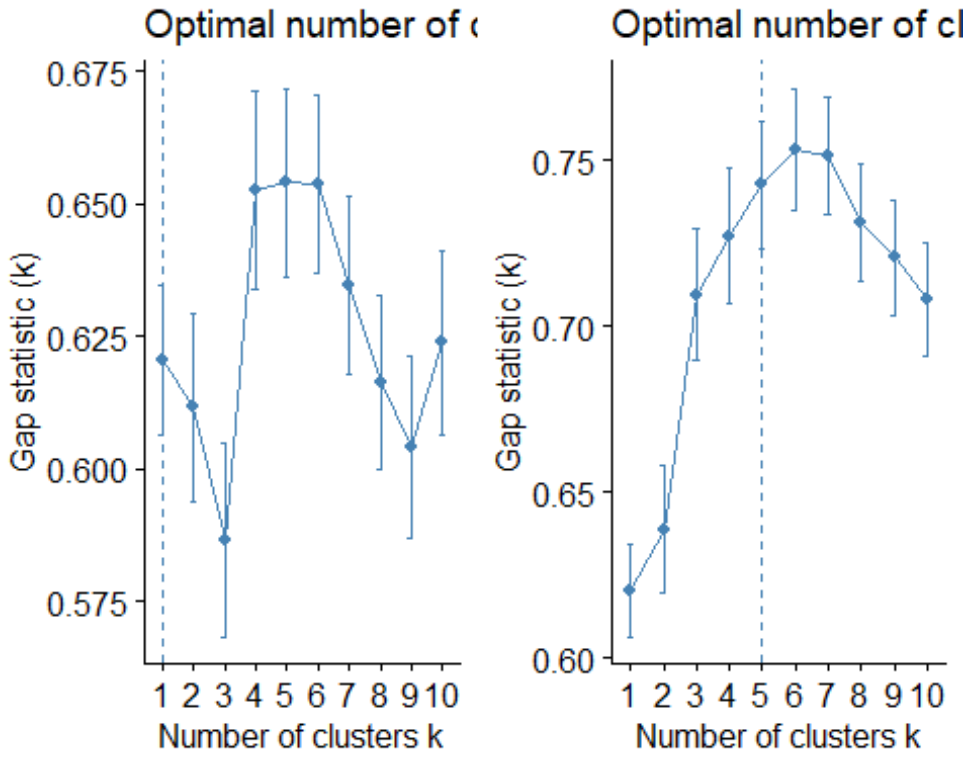

```
grid.arrange(fviz_nbclust(data.scaled, hcut, method = "gap_stat", nboot =
500,hc_metric="euclidean", hc_method= "single"),
 fviz_nbclust(data.scaled, hcut, method = "gap_stat", nboot =
500,hc_metric="manhattan", hc_method= "single"),ncol=2,nrow=1)
```



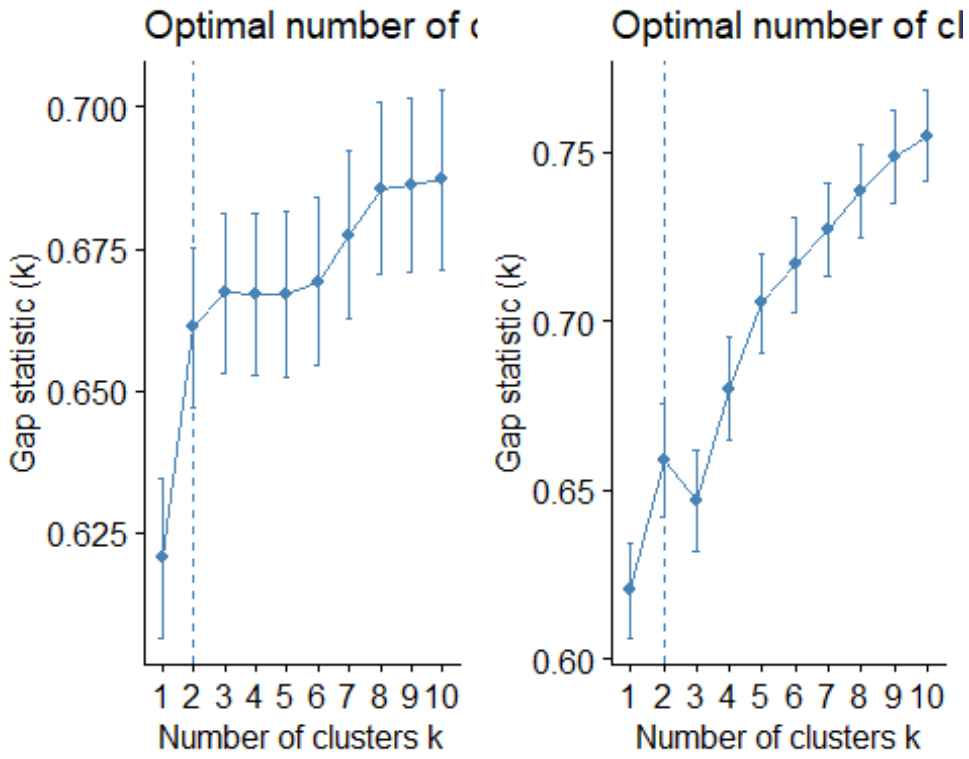

```
grid.arrange(fviz_nbclust(data.scaled, hcut, method = "gap_stat", nboot =
500,hc_metric="euclidean", hc_method= "average"),
fviz_nbclust(data.scaled, hcut, method = "gap_stat", nboot =
500,hc_metric="manhattan", hc_method= "average"),ncol=2,nrow=1)
```



```
grid.arrange(fviz_nbclust(data.scaled, hcut, method = "gap_stat", nboot =
500,hc_metric="euclidean", hc_method= "complete"),
 fviz_nbclust(data.scaled, hcut, method = "gap_stat", nboot =
500,hc_metric="manhattan", hc_method= "complete"),ncol=2,nrow=1)
```




```
grid.arrange(fviz_nbclust(data.scaled, hcut, method = "gap_stat", nboot =
500,hc_metric="euclidean", hc_method= "centroid"),
fviz_nbclust(data.scaled, hcut, method = "gap_stat", nboot =
500,hc_metric="euclidean", hc_method= "ward.D2"),ncol=2,nrow=1)
```



By looking at these model, the optimal number of clusters is 2, so i decided to use this in order to continue my analysis.

```
grp.ea = cutree(res.hc.ea,k=2)
table(grp.ea)
```

```
 grp.ea
   1   2
 235   8
```

```
grp.ew = cutree(res.hc.ew,k=2)
table(grp.ew)
```

```
 grp.ew
   1   2
 140 103
```

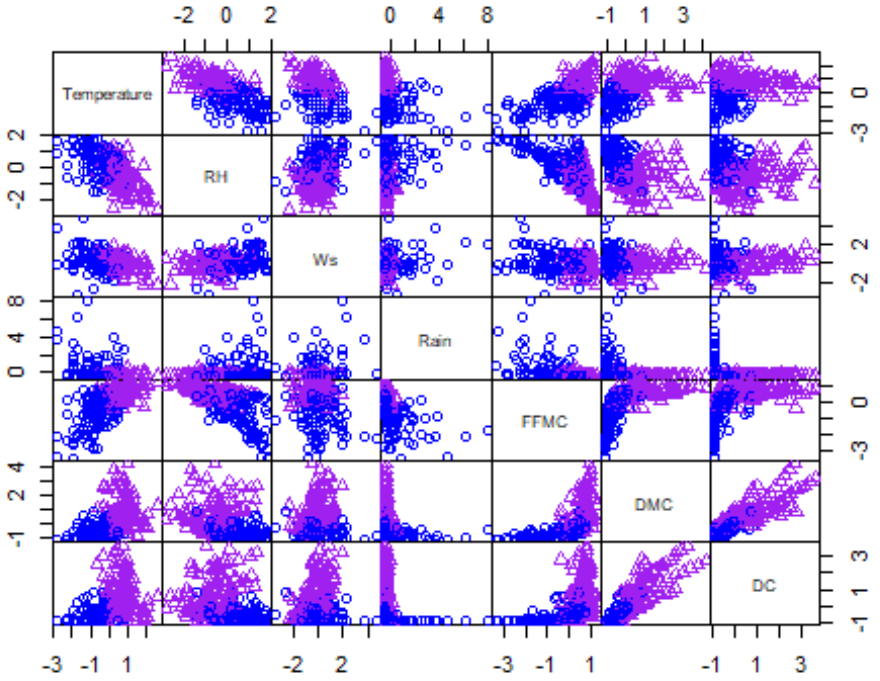
```
grp.ma = cutree(res.hc.ma,k=2)
table(grp.ma)
```

```
 grp.ma
   1   2
 236   7
```

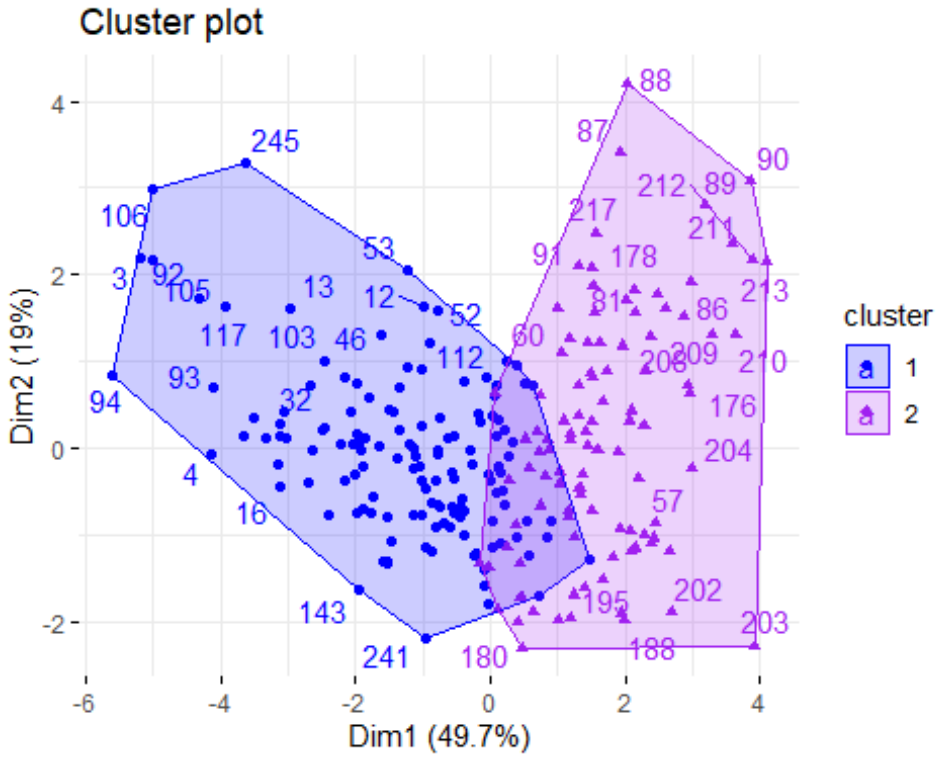
```
table(data$Classes)
```

```
   0   1
 106 137
```

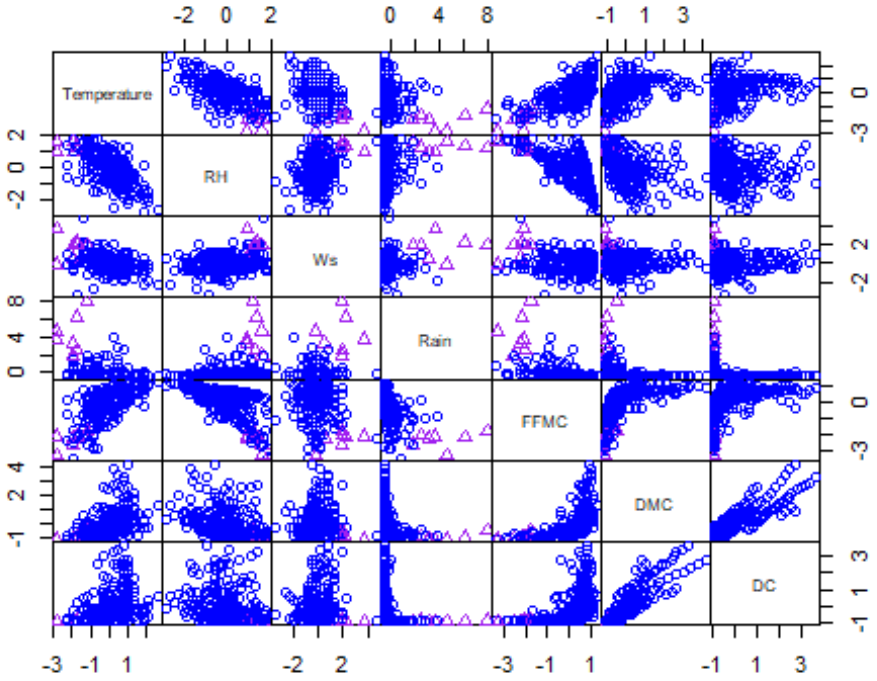
```
pairs(data.scaled,gap=0,pch=grp.ew,col=c('blue','purple')[grp.ew])
```



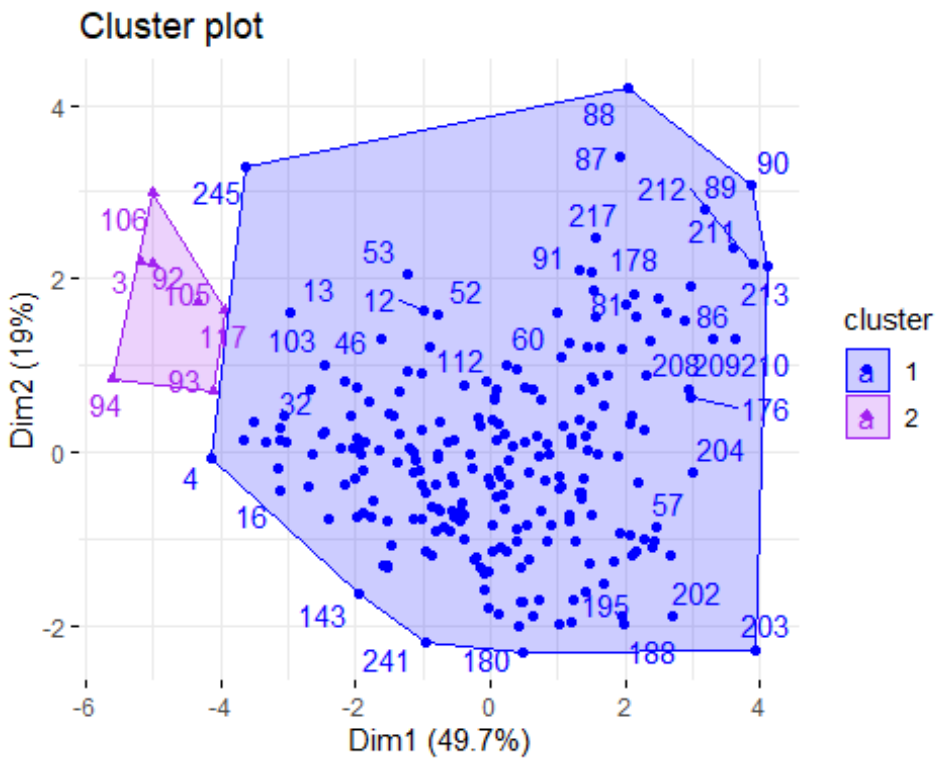

```
fviz_cluster(list(data=data.scaled,cluster =
grp.ew),palette=c('blue','purple'),ellipse.type = 'convex',repel =
TRUE,show.clust.cent = FALSE,ggtheme = theme_minimal())
```



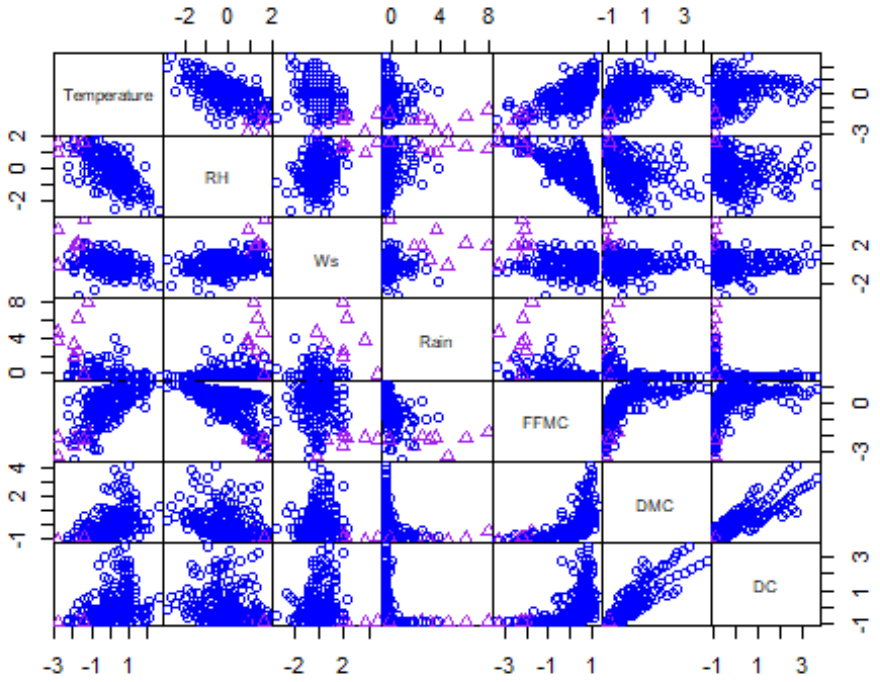
```
pairs(data.scaled,gap=0,pch=grp.ma,col=c('blue','purple')[grp.ma])
```




```
fviz_cluster(list(data=data.scaled,cluster =
grp.ma),palette=c('blue','purple'),ellipse.type = 'convex',repel =
TRUE,show.clust.cent = FALSE,ggtheme = theme_minimal())
```

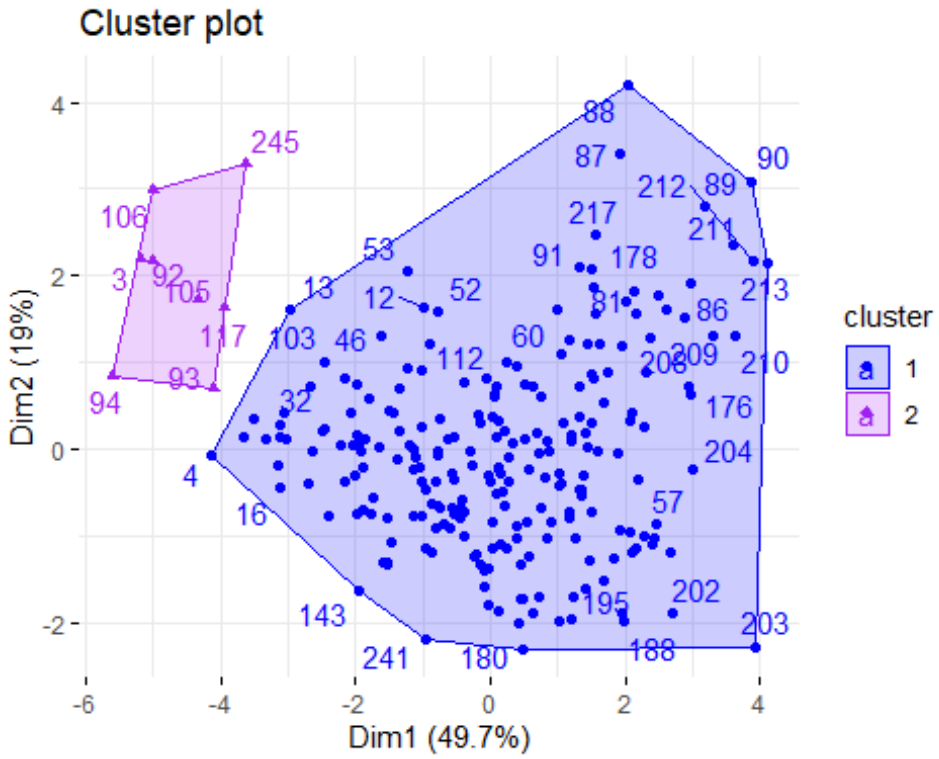



```
pairs(data.scaled,gap=0,pch=grp.ea,col=c('blue','purple')[grp.ea])
```

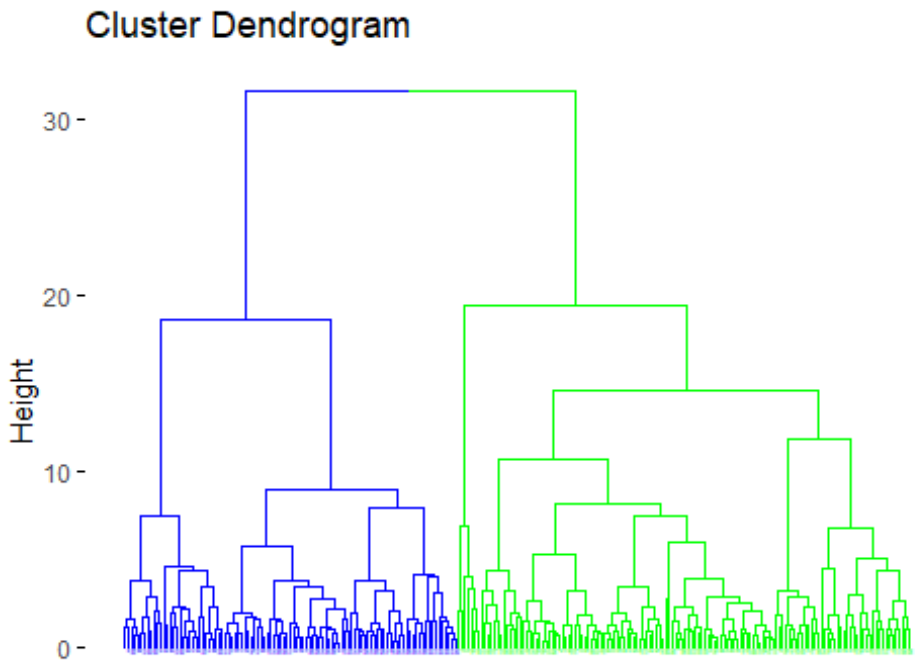



```
fviz_cluster(list(data=data.scaled,cluster =
grp.ea),palette=c('blue','purple'),ellipse.type = 'convex',repel =
TRUE,show.clust.cent = FALSE,ggtheme = theme_minimal())
```

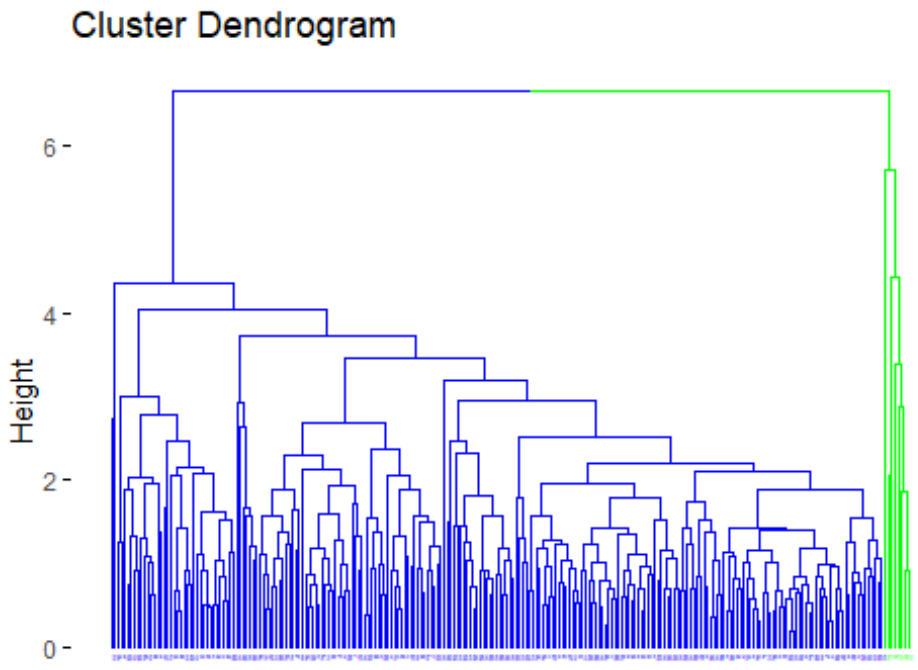
```
Warning: ggrepel: 199 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



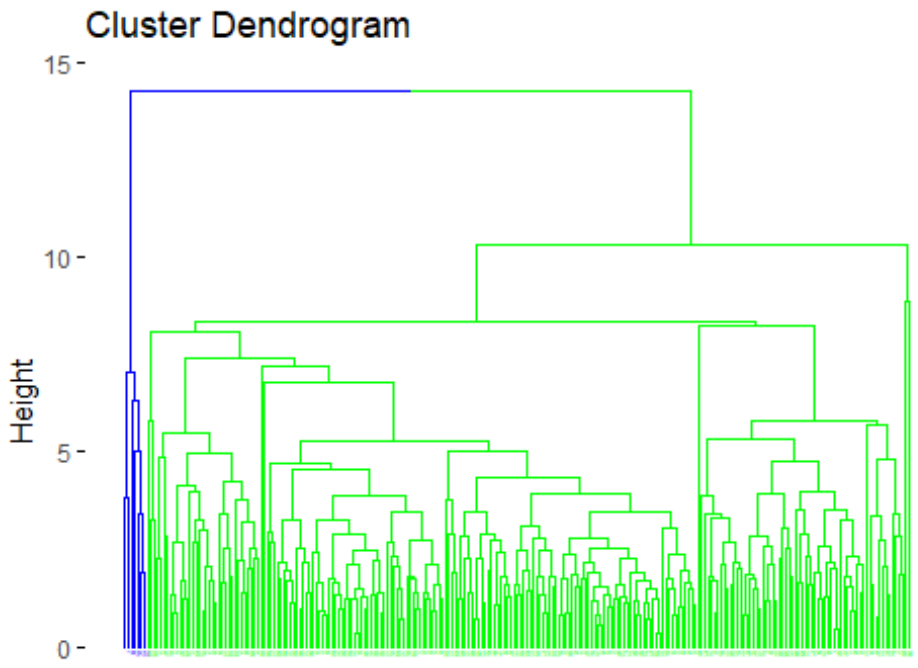

```
fviz_dend(res.hc.ew,k=2,cex = 0,k_colors=c('blue','green'),color_labels_by_k =
TRUE,rect = FALSE)
```



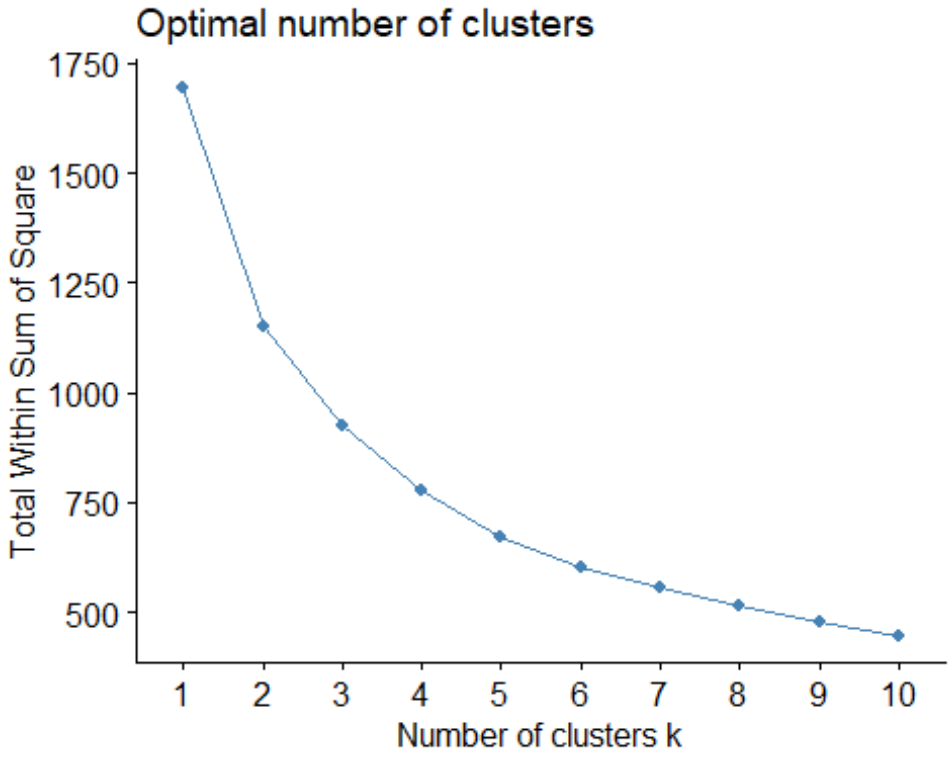
```
fviz_dend(res.hc.ea,k=2,cex = 0,k_colors=c('blue','green'),color_labels_by_k =
TRUE,rect = FALSE)
```



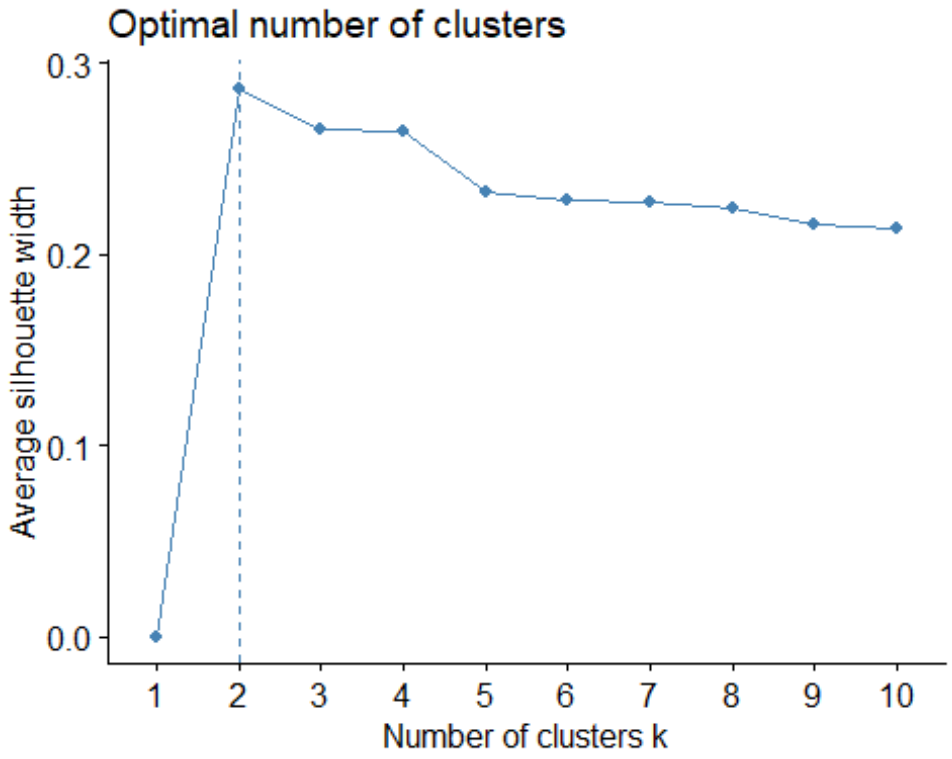

```
fviz_dend(res.hc.ma,k=2,cex = 0,k_colors=c('blue','green'),color_labels_by_k =
TRUE,rect = FALSE)
```



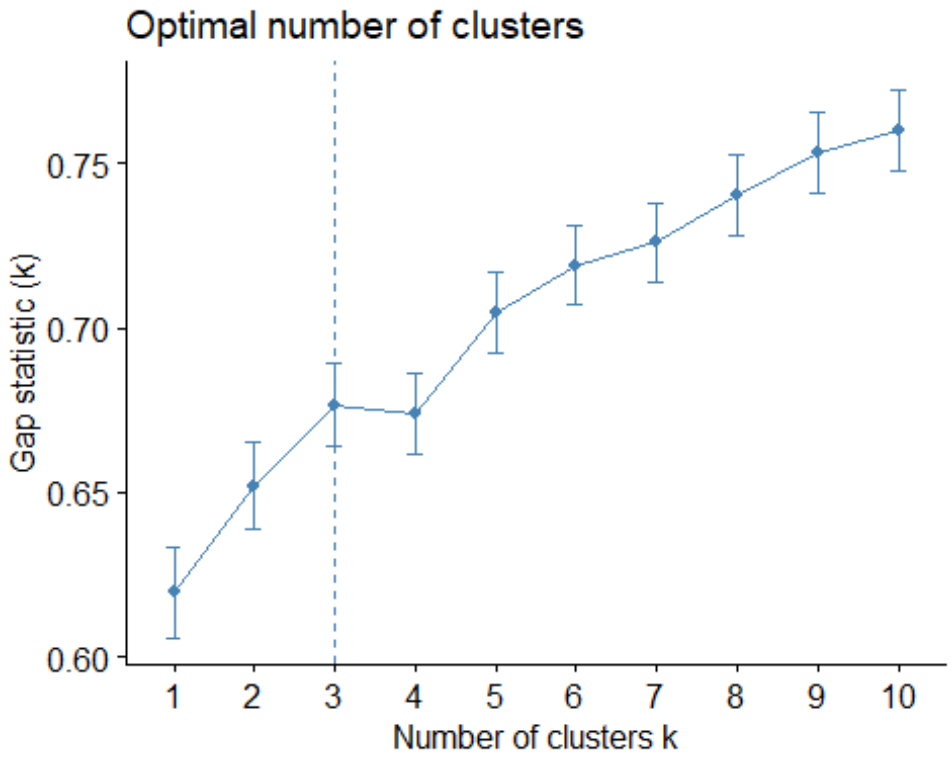
```
fviz_nbclust(data.scaled,kmeans,nstart=25,method='wss')
```




```
fviz_nbclust(data.scaled,kmeans,nstart=25,method='silhouette')
```




```
fviz_nbclust(data.scaled,kmeans,nstart=25,method='gap_stat',nboot=500)
```



Partitional Method

The partitioning or partitional approach is another method used to perform the CA analysis. Also, for this method, we have to look for clusters with a higher separation and a higher cohesion, as a way to obtain the best clustering results. However, differently from the hierarchical approach, here the number of clusters must be specified a priori. It means that different values of K, different metrics as the Euclidean or the Manhattan or different partitional approaches yield to different outcomes. We can distinguish between two different types of Partitioning methods:

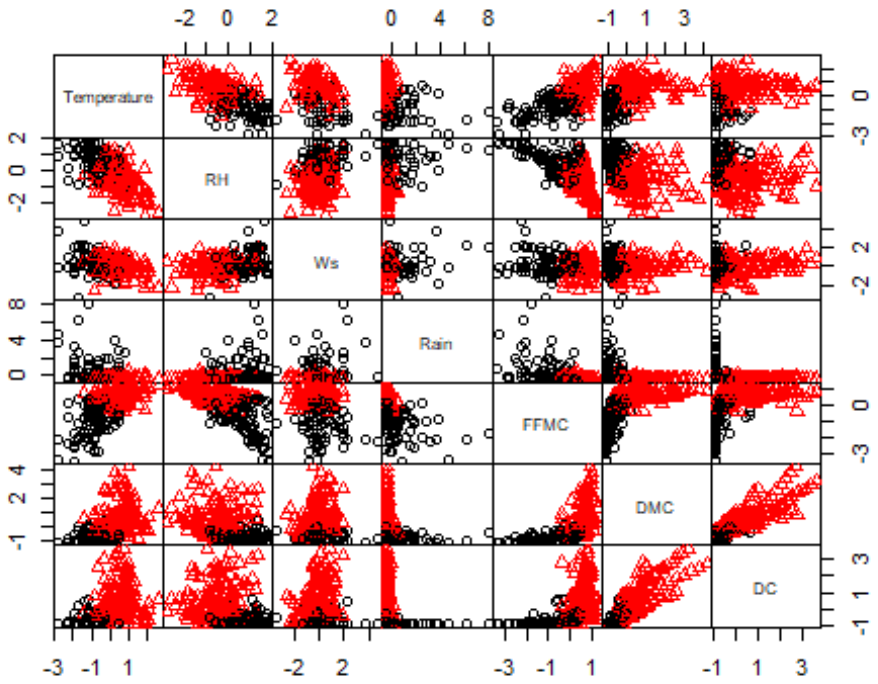
- - the K-means in which each cluster is represented by the sample mean or centroid

- - the K-medoids in which each cluster is represented by one unit within the cluster, the so-called medoid. Since in the K-means the sample mean is not forced to be a unit belonging to the cluster, this method is more sensitive to the presence of the outliers with respect to the K-medoids, in which the clusters are represented by observations within the groups.

K-means

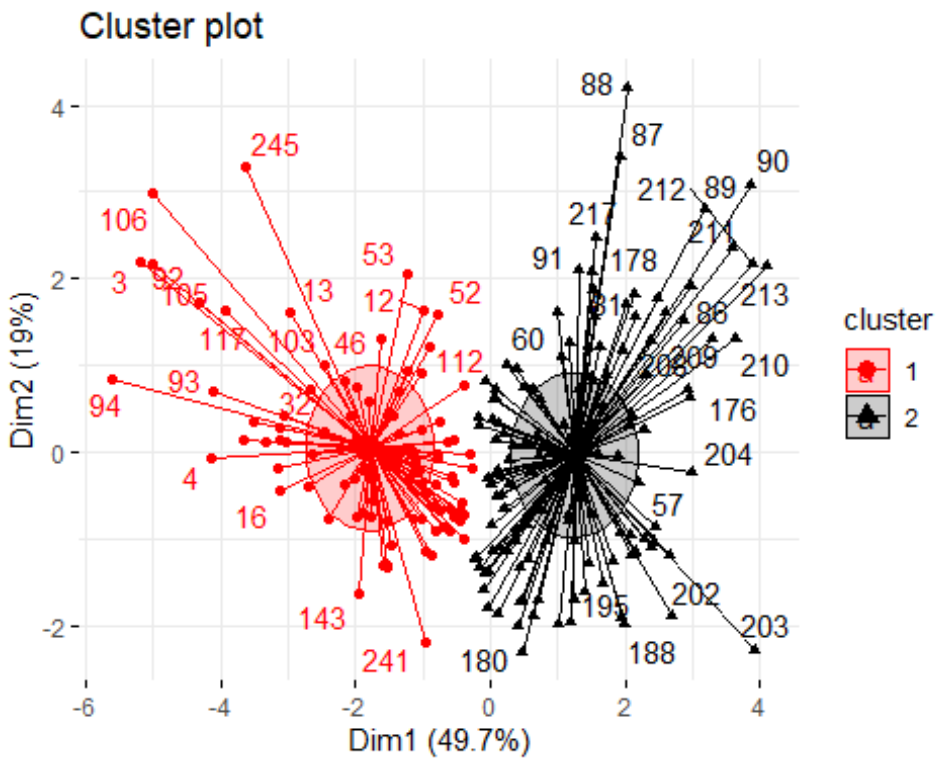
As stated before, the first step of every partitional approach is to decide a priori the value of K. Here follows some indications provided by different indices. However, it is important to keep in mind that both the hierarchical and the partitional approach are based on empirical experimentation rather than formal laws, so all these indications, especially for the WSS and the Silhouette index, must be considered as heuristic approaches to the problem. The most rigorous index is the Gap-stat (third figure) which tries to formalize from a statistical point of view the WSS method.

```
set.seed(123)
km.res1 = kmeans(data.scaled,2)
cl1 = km.res1$cluster
pairs(data.scaled,gap=0,pch=cl1,col = c('black','red')[cl1])
```

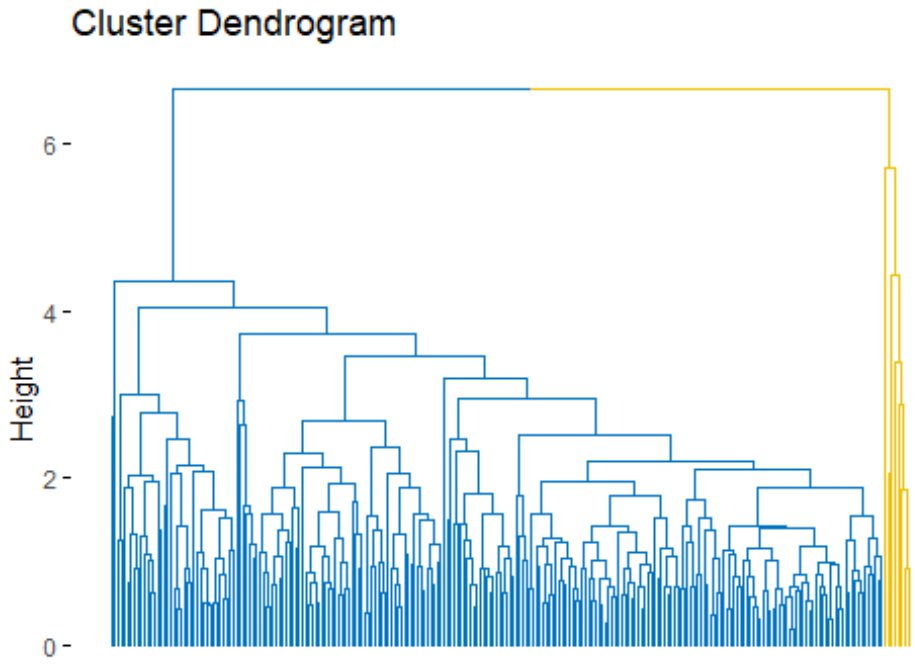



```
fviz_cluster(km.res1,data = data.scaled,palette=c('red','black'),ellipse.type =
'euclid',star.plot=TRUE,repel=TRUE,ggtheme = theme_minimal())
```

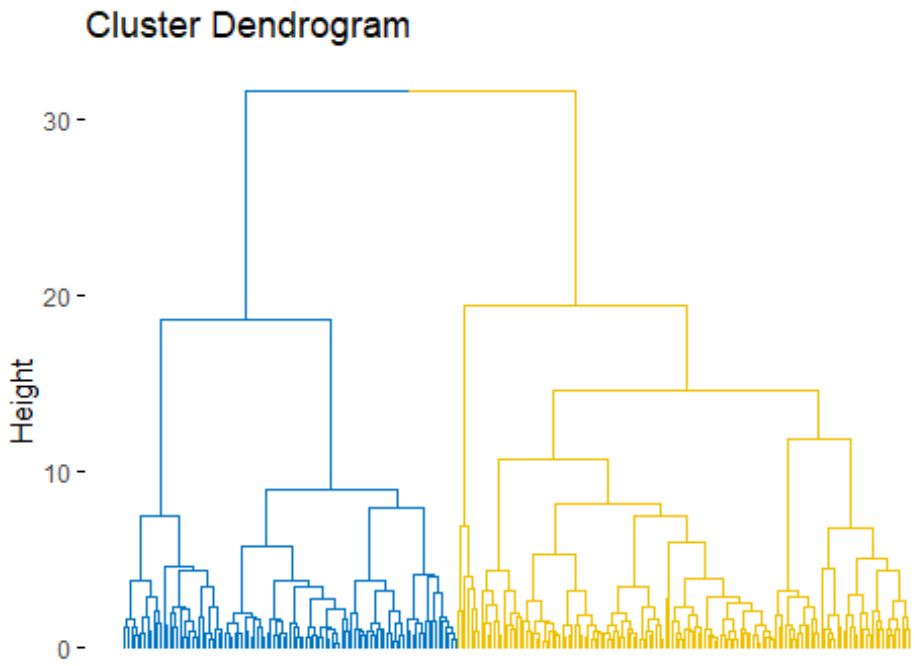
```
 Warning: ggrepel: 199 unlabeled data points (too many overlaps). Consider
 increasing max.overlaps
```



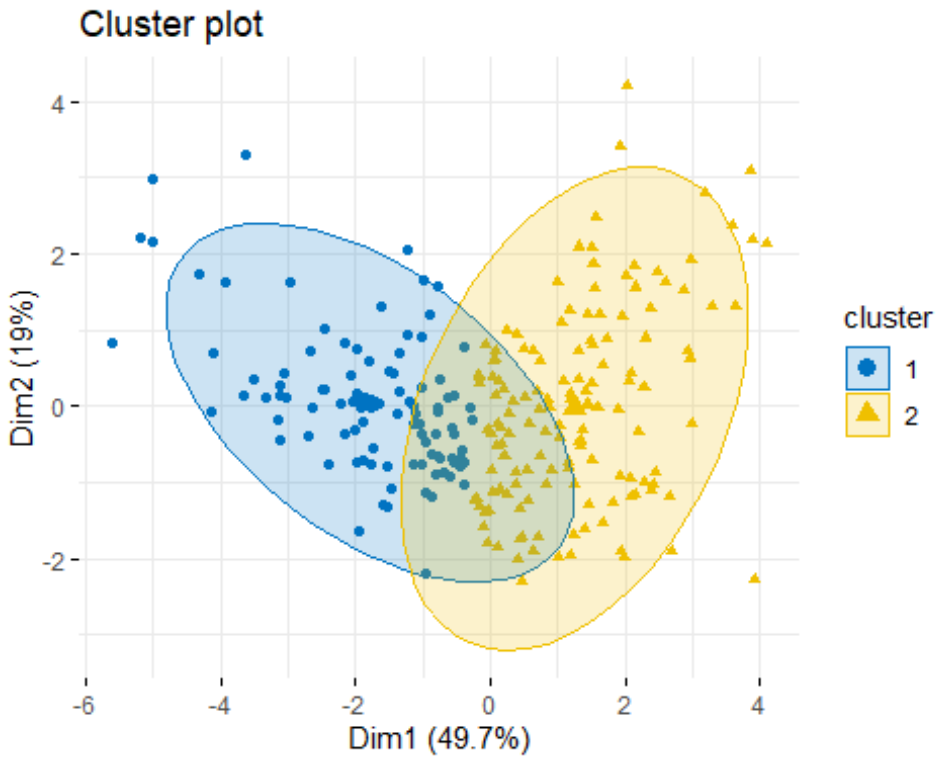

```
hc.res=eclust(data.scaled,'hclust',k=2,hc_metric =
'euclidean',hc_method='average',graph=FALSE)
fviz_dend(hc.res,show_labels = FALSE,palette='jco',as.ggplot=TRUE)
```



```
hc.res2=eclust(data.scaled,'hclust',k=2,hc_metric =
'euclidean',hc_method='ward.D2',graph=FALSE)
fviz_dend(hc.res2,show_labels = FALSE,palette='jco',as.ggplot=TRUE)
```




```
km.res = eclust(data.scaled,'kmeans',k=2,nstart=25,graph=FALSE)
fviz_cluster(km.res,geom='point',ellipse.type = 'norm',palette='jco',ggtheme =
theme_minimal())
```

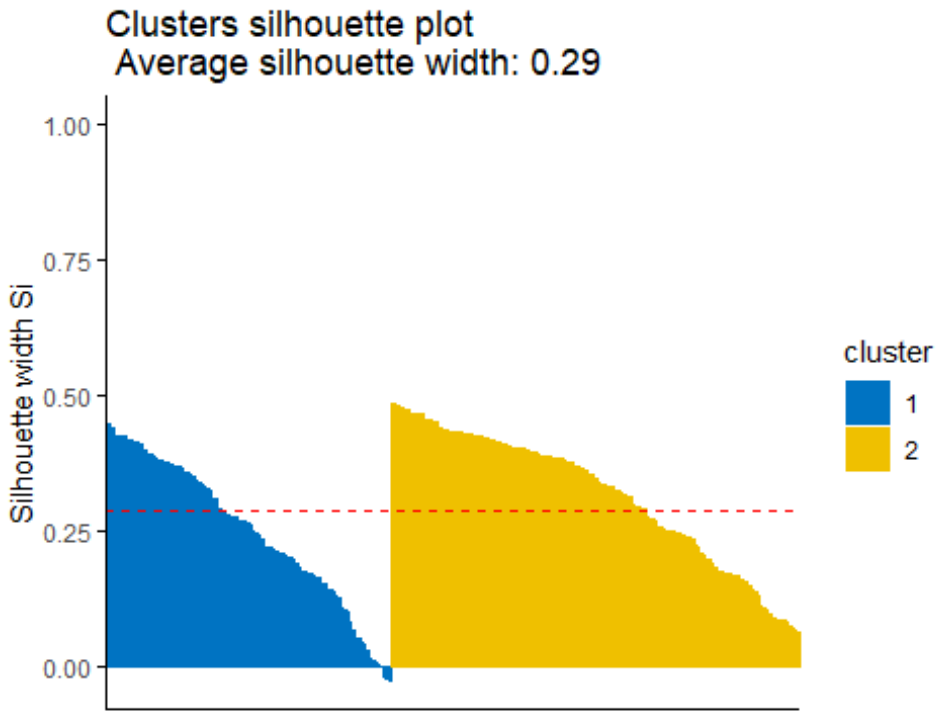



```
fviz_silhouette(km.res,palette='jco',ggtheme=theme_classic())
```

```
  cluster size ave.sil.width
1       1  100          0.25
2       2  143          0.31
```

Cluster validation statistics: Internal validation measures In this part will be evaluated the quality of the clustering results. Generally, this kind of analysis will be conducted by looking both at internal and external validation measures.

The most famous external validation measures are: the confusion matrix, the correct Rand index and the Meila's VI index. The internal validation measures that will be proposed are throughout the report will be: the average silhouette width and the Dunn index.



The silhouette width is a measure of both the separation and the cohesion of the clusters. It can assume values within the range [-1,1]; the closer is this value to -1 or to 0, the worst is the clustering results; the closer is this value to 1 the better is the partitioning.

```
silinfo = km.res$silinfo
names(silinfo)
```

```
 [1] "widths"          "clus.avg.widths" "avg.width"
```

```
head(silinfo$widths[,1:3],10)
```

```
     cluster neighbor sil_width
 121       1        2 0.4484997
 139       1        2 0.4417732
 15        1        2 0.4394607
 32        1        2 0.4264572
 128       1        2 0.4251911
 227       1        2 0.4248688
 101       1        2 0.4240215
 122       1        2 0.4187286
 5         1        2 0.4161553
 141       1        2 0.4146039
```

```
silinfo$clus.avg.widths
```

```
 [1] 0.2467359 0.3140863
```

```
silinfo$avg.width
```

```
 [1] 0.2863701
```

```
km_stats = cluster.stats(dist(data.scaled),km.res$cluster)
km_stats$dunn
```

```
 [1] 0.04124671
```

The Dunn index is another internal validation measures which allows us to understand if the data are well clustered or not; the Dunn index has to be maximized; in this case we have obtained a low value of the Dunn index, which means that the data contain a not so compact and well-separated clusters.

```
pam.res = pam(data,2)
print(pam.res)
```

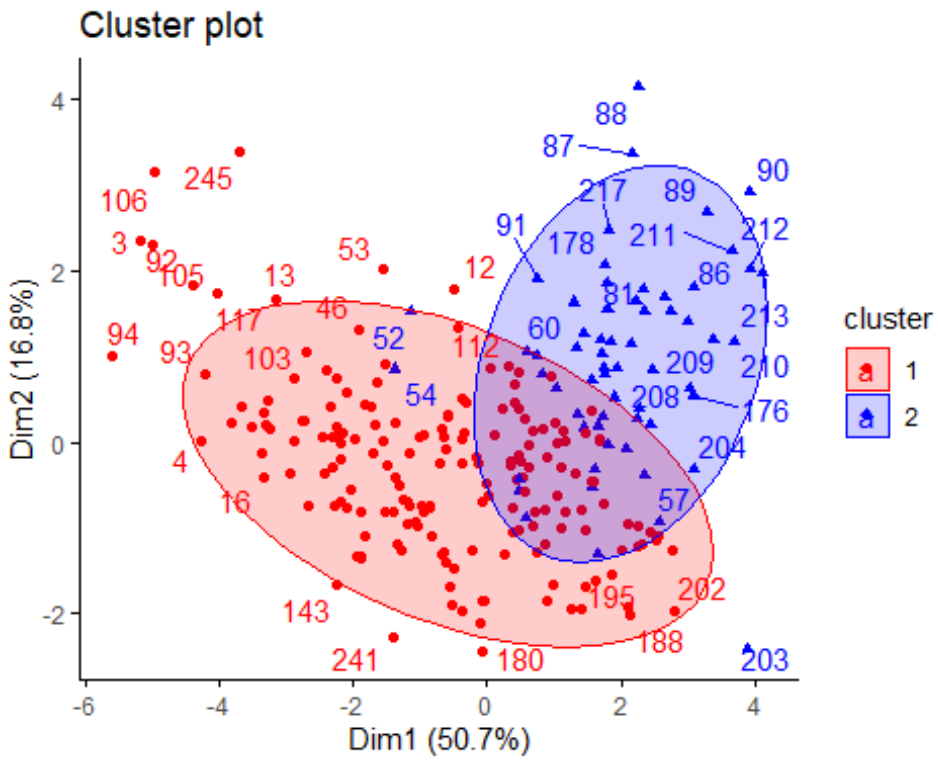
```
 Medoids:
      ID Temperature RH Ws Rain FFMC  DMC    DC Classes
 44   44          34 61 13  0.6 73.9  7.8  22.9       1
 240 236          35 56 14  0.0 89.0 29.4 115.6       2
 Clustering vector:
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
   1   1   1   1   1   2   2   2   2   2   1   1   1   1   1   1   1   1   1   1
  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
   1   1   1   1   1   1   1   1   1   1   2   2   1   2   2   2   2   2   2   2
  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
   2   1   1   1   1   1   1   1   1   1   1   2   2   2   2   2   2   2   2   2
  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
   2   2   2   2   2   2   2   2   2   2   2   1   1   1   1   1   1   1   1   1
 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
 121 122 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163
   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
 164 165 166 167 168 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184
   1   1   1   1   1   1   1   1   1   1   2   2   2   2   1   1   1   1   1   1
 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204
   1   2   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2   2
 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224
   2   2   2   2   2   2   2   2   2   2   2   2   2   1   1   1   1   1   1   1
 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244
   1   1   1   1   1   1   1   1   1   1   2   2   2   2   2   2   1   1   1   1
 245 246 247
   1   1   1
 Objective function:
    build     swap
 32.24359 29.99013

 Available components:
```

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

```
dd = cbind(data,cluster = pam.res$cluster)
head(dd)
```

```
  Temperature RH Ws Rain FFMC DMC   DC Classes cluster
1          29 57 18  0.0 65.7 3.4  7.6       0       1
2          29 61 13  1.3 64.4 4.1  7.6       0       1
3          26 82 22 13.1 47.1 2.5  7.1       0       1
4          25 89 13  2.5 28.6 1.3  6.9       0       1
5          27 77 16  0.0 64.8 3.0 14.2       0       1
6          31 67 14  0.0 82.6 5.8 22.2       1       1
```

```
fviz_cluster(pam.res,palette=c('red','blue'),ellipse.type = 't',repel =
TRUE,ggtheme = theme_classic())
```



External Validation

In order to get a better response for our result, after computing the internal validation we need to perform external validation. In this case we compare results from our cluster analysis to the categorical variable Classes throught 2 indexes: the correct Rand index and the Meila's VI index..

```
table(data$Classes,km.res$cluster)
```

```
   1   2
```

```
   0  85  21
   1  15 122
```

```
Fire = as.numeric(data$Classes)
clust_fire = cluster.stats(d=dist(data),Fire,km.res$cluster)
clust_fire$corrected.rand
```

```
 [1] 0.4929439
```

```
clust_fire$vi
```

```
 [1] 0.831368
```

In particular the second result gave us a good result, so our clustering is quite similar to the distribution of elements in the categorical variable.

CLUSTER VALIDATION WITH clValid package

Using the package clValid it's possible to identify the best clustering algorithms and the optimal number of clusters with a single function.

The internal measures give the following results:

```
clmethods = c('hierarchical','kmeans','pam')
clvalid = clValid(data.scaled,nClust = 2:7,clMethods = clmethods,validation =
'internal')
summary(clvalid)
```

```
 Clustering Methods:
  hierarchical kmeans pam

 Cluster sizes:
  2 3 4 5 6 7

 Validation Measures:
                                    2        3        4        5        6        7

 hierarchical Connectivity    9.7278  10.1373  13.8290  19.1869  34.6865  45.2337
              Dunn            0.2125   0.2125   0.2125   0.2049   0.1077   0.1077
              Silhouette      0.5078   0.4654   0.4343   0.2463   0.2317   0.1960
 kmeans       Connectivity   36.9778  64.0623  76.4024 102.5353 117.0940 130.3643
              Dunn            0.0412   0.0557   0.0445   0.0551   0.0692   0.0696
              Silhouette      0.2864   0.2450   0.2646   0.2297   0.2168   0.2114
 pam          Connectivity   35.6520  74.8968 102.1183 113.4667 136.7107 136.3155
              Dunn            0.0463   0.0210   0.0496   0.0538   0.0417   0.0520
              Silhouette      0.2984   0.2556   0.1955   0.2097   0.1827   0.1938

 Optimal Scores:

              Score  Method       Clusters
 Connectivity 9.7278 hierarchical 2
```

```
 Dunn         0.2125 hierarchical 2
 Silhouette   0.5078 hierarchical 2
```

```
intern = clValid(data.scaled, nClust = 2:10,clMethods = clmethods,validation =
'internal',metric = 'euclidean',method = 'single')
summary(intern)
```

```
 Clustering Methods:
  hierarchical kmeans pam

 Cluster sizes:
  2 3 4 5 6 7 8 9 10

 Validation Measures:
                                    2        3        4        5        6        7
8        9       10

 hierarchical Connectivity    4.6913   7.7631  10.6921  13.4671  15.9849  17.9849
21.1639  25.7758  28.7048
              Dunn            0.2815   0.2703   0.2535   0.2389   0.2400   0.2234
0.2081   0.2125   0.2049
              Silhouette      0.5967   0.4587   0.2784   0.2788   0.2807   0.2771
0.2701   0.2570   0.1230
 kmeans       Connectivity   36.9778  64.0623  76.4024 102.5353 114.2952 113.8956
140.0476 125.3206 138.1651
              Dunn            0.0412   0.0557   0.0445   0.0551   0.0770   0.0871
0.0783   0.0896   0.0577
              Silhouette      0.2864   0.2450   0.2646   0.2297   0.2306   0.2408
0.2171   0.2257   0.1990
 pam          Connectivity   35.6520  74.8968 102.1183 113.4667 136.7107 136.3155
140.1861 150.1075 159.3758
              Dunn            0.0463   0.0210   0.0496   0.0538   0.0417   0.0520
0.0448   0.0665   0.0665
              Silhouette      0.2984   0.2556   0.1955   0.2097   0.1827   0.1938
0.1978   0.1903   0.1902

 Optimal Scores:

              Score  Method       Clusters
 Connectivity 4.6913 hierarchical 2
 Dunn         0.2815 hierarchical 2
 Silhouette   0.5967 hierarchical 2
```

```
intern = clValid(data.scaled, nClust = 2:10,clMethods = clmethods,validation =
'internal',metric = 'euclidean',method = 'complete')
summary(intern)
```

```
 Clustering Methods:
  hierarchical kmeans pam
```

```
 Cluster sizes:
  2 3 4 5 6 7 8 9 10

 Validation Measures:
                                     2        3        4        5        6        7
8        9       10

 hierarchical Connectivity   32.8508  38.8167  58.0698  82.1889  96.0663  99.7579
102.4202 102.7536 105.2210
              Dunn           0.0837   0.0929   0.1073   0.0971   0.0994   0.1045
0.1125   0.1153   0.1155
              Silhouette     0.2991   0.2449   0.2326   0.1820   0.1923   0.1914
0.1877   0.1867   0.1712
 kmeans       Connectivity   36.9778  64.0623  76.4024 105.8222 110.1583 113.4444
121.5440 109.3579 129.5036
              Dunn           0.0412   0.0557   0.0445   0.0722   0.0769   0.0792
0.0896   0.0864   0.0863
              Silhouette     0.2864   0.2450   0.2646   0.2248   0.2393   0.2324
0.2276   0.2321   0.2209
 pam          Connectivity   35.6520  74.8968 102.1183 113.4667 136.7107 136.3155
140.1861 150.1075 159.3758
              Dunn           0.0463   0.0210   0.0496   0.0538   0.0417   0.0520
0.0448   0.0665   0.0665
              Silhouette     0.2984   0.2556   0.1955   0.2097   0.1827   0.1938
0.1978   0.1903   0.1902

 Optimal Scores:

              Score   Method       Clusters
 Connectivity 32.8508 hierarchical 2
 Dunn          0.1155 hierarchical 10
 Silhouette    0.2991 hierarchical 2
```

```
intern = clValid(data.scaled, nClust = 2:10,clMethods = clmethods,validation =
'internal',metric = 'euclidean',method = 'average')
summary(intern)
```

```
 Clustering Methods:
  hierarchical kmeans pam

 Cluster sizes:
  2 3 4 5 6 7 8 9 10

 Validation Measures:
                                     2        3        4        5        6        7
8        9       10

 hierarchical Connectivity    9.7278  10.1373  13.8290  19.1869  34.6865  45.2337
68.0254  69.9032  72.9571
```

```
             Dunn            0.2125   0.2125   0.2125   0.2049   0.1077   0.1077
0.1288   0.1288   0.1288
             Silhouette      0.5078   0.4654   0.4343   0.2463   0.2317   0.1960
0.2437   0.2416   0.2053
 kmeans      Connectivity   36.9778  64.0623  76.4024 102.5353 117.0940 130.3643
134.2651 132.5679 141.6302
             Dunn            0.0412   0.0557   0.0445   0.0551   0.0692   0.0696
0.0578   0.0783   0.0786
             Silhouette      0.2864   0.2450   0.2646   0.2297   0.2168   0.2114
0.2177   0.2093   0.2206
 pam         Connectivity   35.6520  74.8968 102.1183 113.4667 136.7107 136.3155
140.1861 150.1075 159.3758
             Dunn            0.0463   0.0210   0.0496   0.0538   0.0417   0.0520
0.0448   0.0665   0.0665
             Silhouette      0.2984   0.2556   0.1955   0.2097   0.1827   0.1938
0.1978   0.1903   0.1902

 Optimal Scores:

             Score  Method       Clusters
 Connectivity 9.7278 hierarchical 2
 Dunn         0.2125 hierarchical 2
 Silhouette   0.5078 hierarchical 2
```

```
intern = clValid(data.scaled, nClust = 2:10,clMethods = clmethods,validation =
'internal',metric = 'euclidean',method = 'ward')
```

```
 Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
 Il metodo "ward" è stato rinominato in "ward.D"; nota il nuovo "ward.D2"
```

```
summary(intern)
```

```

 Clustering Methods:
  hierarchical kmeans pam

 Cluster sizes:
  2 3 4 5 6 7 8 9 10

 Validation Measures:
                                  2        3        4        5        6        7
8        9       10

 hierarchical Connectivity   19.1659  34.6655  65.6000  73.0750  89.5008  99.7627
108.2520 123.3179 132.9020
             Dunn            0.0912   0.0912   0.0745   0.0790   0.0784   0.1036
0.1036   0.0948   0.0948
             Silhouette      0.3019   0.2505   0.1917   0.2060   0.1449   0.1598
0.1736   0.1800   0.1838
 kmeans      Connectivity   36.9778  67.4258  90.1563 104.9202 136.4552 133.1532
134.2651 143.2000 153.6135
```

```
              Dunn            0.0412   0.0646   0.0408   0.0414   0.0543   0.0789
0.0578   0.0680   0.0680
              Silhouette      0.2864   0.2631   0.2419   0.2322   0.1967   0.2089
0.2177   0.2164   0.2132
 pam          Connectivity   35.6520  74.8968 102.1183 113.4667 136.7107 136.3155
140.1861 150.1075 159.3758
              Dunn            0.0463   0.0210   0.0496   0.0538   0.0417   0.0520
0.0448   0.0665   0.0665
              Silhouette      0.2984   0.2556   0.1955   0.2097   0.1827   0.1938
0.1978   0.1903   0.1902

 Optimal Scores:

              Score   Method       Clusters
 Connectivity 19.1659 hierarchical 2
 Dunn          0.1036 hierarchical 7
 Silhouette    0.3019 hierarchical 2
```

```
intern = clValid(data.scaled, nClust = 2:10,clMethods = clmethods,validation =
'internal',metric = 'manhattan',method = 'single')
summary(intern)
```

```

 Clustering Methods:
  hierarchical kmeans pam

 Cluster sizes:
  2 3 4 5 6 7 8 9 10

 Validation Measures:
                                    2        3        4        5        6        7
8        9       10

 hierarchical Connectivity    2.9290   6.4357   9.7897  12.9948  15.9238  18.8528
20.8528  23.9817  27.1218
              Dunn            0.2158   0.1805   0.1768   0.1832   0.1792   0.1712
0.1648   0.1559   0.1420
              Silhouette      0.2794   0.2602   0.2609   0.2767   0.2559   0.0791
0.0753   0.0590  -0.0178
 kmeans       Connectivity   35.5202  68.6266  78.5048 112.4996 126.5718 129.6758
141.6631 143.2147 162.0306
              Dunn            0.0449   0.0535   0.0536   0.0594   0.0607   0.0890
0.0335   0.0887   0.0639
              Silhouette      0.3359   0.2645   0.2698   0.2363   0.2234   0.2362
0.2259   0.2129   0.2020
 pam          Connectivity   46.0063  69.7333  88.6159 139.9187 159.8421 169.3571
196.8738 214.1155 220.9790
              Dunn            0.0414   0.0618   0.0565   0.0549   0.0538   0.0595
0.0628   0.0628   0.0628
              Silhouette      0.3430   0.2554   0.2336   0.1773   0.1840   0.1856
0.1758   0.1692   0.1710
```

```
 Optimal Scores:

             Score  Method       Clusters
 Connectivity 2.9290 hierarchical 2
 Dunn         0.2158 hierarchical 2
 Silhouette   0.3430 pam          2
```

```
intern = clValid(data.scaled, nClust = 2:10,clMethods = clmethods,validation =
'internal',metric = 'manhattan',method = 'average')
summary(intern)
```

```
 Clustering Methods:
  hierarchical kmeans pam

 Cluster sizes:
  2 3 4 5 6 7 8 9 10

 Validation Measures:
                                   2        3        4        5        6        7
8        9       10

 hierarchical Connectivity    8.7179  14.7758  14.7758  42.7409  45.5270  53.3167
72.5563  75.6353  79.0714
              Dunn            0.1221   0.1221   0.1221   0.0998   0.0998   0.1009
0.1280   0.1280   0.1280
              Silhouette      0.4913   0.3214   0.2799   0.2808   0.2621   0.1832
0.2189   0.2001   0.1974
 kmeans       Connectivity   35.5202  68.6266  78.5048 112.4996 123.4127 135.7921
150.6762 157.1909 161.5587
              Dunn            0.0449   0.0535   0.0536   0.0594   0.0857   0.0575
0.0846   0.0846   0.0859
              Silhouette      0.3359   0.2645   0.2698   0.2363   0.2432   0.2275
0.2020   0.2042   0.2054
 pam          Connectivity   46.0063  69.7333  88.6159 139.9187 159.8421 169.3571
196.8738 214.1155 220.9790
              Dunn            0.0414   0.0618   0.0565   0.0549   0.0538   0.0595
0.0628   0.0628   0.0628
              Silhouette      0.3430   0.2554   0.2336   0.1773   0.1840   0.1856
0.1758   0.1692   0.1710

 Optimal Scores:

             Score  Method       Clusters
 Connectivity 8.7179 hierarchical 2
 Dunn         0.1280 hierarchical 8
 Silhouette   0.4913 hierarchical 2
```

```
intern = clValid(data.scaled, nClust = 2:10,clMethods = clmethods,validation =
'internal',metric = 'manhattan',method = 'complete')
summary(intern)


 Clustering Methods:
  hierarchical kmeans pam

 Cluster sizes:
  2 3 4 5 6 7 8 9 10

 Validation Measures:
                                    2        3        4        5        6        7
8        9       10

 hierarchical Connectivity   30.3270  56.1472  73.7190  82.3258  93.4619 110.2044
111.0044 114.4405 121.5052
              Dunn            0.0815   0.0669   0.0694   0.0764   0.0918   0.1019
0.1055   0.1077   0.1090
              Silhouette      0.3371   0.2366   0.2191   0.2163   0.2267   0.2144
0.2119   0.2121   0.1988
 kmeans       Connectivity   35.5202  68.6266  97.7452 111.7381 123.4127 131.7020
142.9234 146.3595 155.2151
              Dunn            0.0449   0.0535   0.0441   0.0814   0.0857   0.0861
0.0914   0.0914   0.0836
              Silhouette      0.3359   0.2645   0.2460   0.2272   0.2432   0.2300
0.2101   0.2089   0.2070
 pam          Connectivity   46.0063  69.7333  88.6159 139.9187 159.8421 169.3571
196.8738 214.1155 220.9790
              Dunn            0.0414   0.0618   0.0565   0.0549   0.0538   0.0595
0.0628   0.0628   0.0628
              Silhouette      0.3430   0.2554   0.2336   0.1773   0.1840   0.1856
0.1758   0.1692   0.1710

 Optimal Scores:

              Score  Method       Clusters
 Connectivity 30.327 hierarchical 2
 Dunn          0.109 hierarchical 10
 Silhouette    0.343 pam          2

stab = clValid(data.scaled, nClust = 2:10,clMethods = clmethods,validation =
'stability',metric = 'euclidean',method = 'single')
summary(stab)


 Clustering Methods:
  hierarchical kmeans pam

 Cluster sizes:
  2 3 4 5 6 7 8 9 10
```

```
 Validation Measures:
                         2      3      4      5      6      7      8      9     10

 hierarchical APN  0.0023 0.0018 0.0064 0.0053 0.0064 0.0093 0.0128 0.0116 0.0155
              AD   3.2906 3.2431 3.2322 3.1869 3.1544 3.1407 3.1197 3.0600 3.0485
              ADM  0.0427 0.0295 0.0667 0.0530 0.0693 0.0679 0.0933 0.1110 0.1304
              FOM  0.9946 0.9925 0.9860 0.9843 0.9845 0.9829 0.9816 0.9726 0.9736
 kmeans       APN  0.1138 0.2263 0.3173 0.3008 0.2918 0.3288 0.3407 0.3578 0.4139
              AD   2.8625 2.7495 2.5964 2.4173 2.3036 2.2563 2.1879 2.1286 2.1129
              ADM  0.4106 0.9594 0.9837 0.8567 0.6836 0.8176 0.8487 0.7892 0.9059
              FOM  0.8768 0.8758 0.8100 0.7972 0.7660 0.7649 0.7543 0.7480 0.7507
 pam          APN  0.1246 0.2381 0.2477 0.3691 0.3917 0.3610 0.4199 0.3910 0.3951
              AD   2.8815 2.6682 2.4823 2.4608 2.3616 2.2350 2.2163 2.1296 2.0744
              ADM  0.4080 0.6744 0.6521 0.9172 0.9356 0.8324 0.9632 0.8878 0.8760
              FOM  0.8735 0.8387 0.8015 0.7931 0.7764 0.7782 0.7645 0.7467 0.7375

 Optimal Scores:

     Score  Method       Clusters
 APN 0.0018 hierarchical 3
 AD  2.0744 pam          10
 ADM 0.0295 hierarchical 3
 FOM 0.7375 pam          10
```

```
stab = clValid(data.scaled, nClust = 2:10,clMethods = clmethods,validation =
'stability',metric = 'euclidean',method = 'average')
summary(stab)
```

```
 Clustering Methods:
  hierarchical kmeans pam

 Cluster sizes:
  2 3 4 5 6 7 8 9 10

 Validation Measures:
                         2      3      4      5      6      7      8      9     10

 hierarchical APN  0.0097 0.0443 0.0641 0.1755 0.1077 0.1555 0.2098 0.2388 0.2821
              AD   3.2276 3.1904 3.1615 3.1373 2.9526 2.8799 2.6283 2.5891 2.5474
              ADM  0.1546 0.2533 0.3149 0.6189 0.7670 0.7761 1.0847 1.0699 1.0907
              FOM  0.9737 0.9696 0.9577 0.9407 0.9403 0.9201 0.8852 0.8810 0.8593
 kmeans       APN  0.1138 0.2395 0.2644 0.2726 0.3102 0.3766 0.2952 0.3697 0.2837
              AD   2.8625 2.7566 2.5269 2.3914 2.3010 2.3076 2.1239 2.1232 1.9944
              ADM  0.4106 1.0034 0.7608 0.7948 0.8574 1.0402 0.7730 0.8598 0.6997
              FOM  0.8768 0.8740 0.8191 0.8102 0.7545 0.7360 0.7452 0.7467 0.7282
 pam          APN  0.1246 0.2381 0.2477 0.3691 0.3917 0.3610 0.4199 0.3910 0.3951
              AD   2.8815 2.6682 2.4823 2.4608 2.3616 2.2350 2.2163 2.1296 2.0744
              ADM  0.4080 0.6744 0.6521 0.9172 0.9356 0.8324 0.9632 0.8878 0.8760
              FOM  0.8735 0.8387 0.8015 0.7931 0.7764 0.7782 0.7645 0.7467 0.7375
```

```
 Optimal Scores:

     Score  Method       Clusters
 APN 0.0097 hierarchical 2
 AD  1.9944 kmeans       10
 ADM 0.1546 hierarchical 2
 FOM 0.7282 kmeans       10
```

```
stab = clValid(data.scaled, nClust = 2:10,clMethods = clmethods,validation =
'stability',metric = 'euclidean',method = 'complete')
summary(stab)
```

```
 Clustering Methods:
  hierarchical kmeans pam

 Cluster sizes:
  2 3 4 5 6 7 8 9 10

 Validation Measures:
                        2      3      4      5      6      7      8      9     10

 hierarchical APN  0.1340 0.2855 0.3090 0.3281 0.3523 0.3790 0.4167 0.4487 0.4397
              AD   3.1958 3.0556 2.8460 2.7358 2.6873 2.5881 2.5225 2.4574 2.3583
              ADM  0.8884 1.0171 1.1113 1.1256 1.1664 1.0951 1.0605 1.0628 1.0116
              FOM  0.9459 0.9273 0.8980 0.8742 0.8707 0.8115 0.8007 0.7920 0.7864
 kmeans       APN  0.1394 0.2359 0.2548 0.2733 0.2523 0.3026 0.3581 0.4663 0.4002
              AD   2.8813 2.7365 2.5160 2.3674 2.2627 2.2360 2.2091 2.2098 2.0949
              ADM  0.4945 0.9448 0.7232 0.7241 0.6798 0.8341 0.8052 0.9772 0.8968
              FOM  0.8796 0.8646 0.8110 0.7949 0.7674 0.7620 0.7434 0.7447 0.7428
 pam          APN  0.1246 0.2381 0.2477 0.3691 0.3917 0.3610 0.4199 0.3910 0.3951
              AD   2.8815 2.6682 2.4823 2.4608 2.3616 2.2350 2.2163 2.1296 2.0744
              ADM  0.4080 0.6744 0.6521 0.9172 0.9356 0.8324 0.9632 0.8878 0.8760
              FOM  0.8735 0.8387 0.8015 0.7931 0.7764 0.7782 0.7645 0.7467 0.7375

 Optimal Scores:

     Score  Method Clusters
 APN 0.1246 pam    2
 AD  2.0744 pam    10
 ADM 0.4080 pam    2
 FOM 0.7375 pam    10
```

```
stab = clValid(data.scaled, nClust = 2:10,clMethods = clmethods,validation =
'stability',metric = 'euclidean',method = 'ward')

summary(stab)
```

```
 Clustering Methods:
```

```
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6 7 8 9 10

Validation Measures:
                        2      3      4      5      6      7      8      9     10

hierarchical APN  0.2391 0.2746 0.3337 0.3842 0.3752 0.3815 0.3823 0.3556 0.3629
             AD   3.0212 2.7600 2.6011 2.4997 2.4023 2.3276 2.2332 2.1268 2.0575
             ADM  0.8695 0.8858 0.9429 0.9784 0.9576 0.9886 0.9534 0.8571 0.8390
             FOM  0.8623 0.8568 0.7948 0.7848 0.7810 0.7733 0.7552 0.7372 0.7152
kmeans       APN  0.1138 0.2290 0.1694 0.2201 0.3380 0.3396 0.3762 0.3078 0.3248
             AD   2.8625 2.6760 2.4354 2.3167 2.3049 2.2069 2.1461 2.0218 1.9706
             ADM  0.4106 0.7624 0.5444 0.5737 0.8689 0.8270 0.8723 0.7192 0.7183
             FOM  0.8768 0.8513 0.7877 0.7797 0.7708 0.7551 0.7498 0.7323 0.7112
pam          APN  0.1246 0.2381 0.2477 0.3691 0.3917 0.3610 0.4199 0.3910 0.3951
             AD   2.8815 2.6682 2.4823 2.4608 2.3616 2.2350 2.2163 2.1296 2.0744
             ADM  0.4080 0.6744 0.6521 0.9172 0.9356 0.8324 0.9632 0.8878 0.8760
             FOM  0.8735 0.8387 0.8015 0.7931 0.7764 0.7782 0.7645 0.7467 0.7375

Optimal Scores:

    Score  Method Clusters
APN 0.1138 kmeans 2
AD  1.9706 kmeans 10
ADM 0.4080 pam    2
FOM 0.7112 kmeans 10
```

```
stab = clValid(data.scaled, nClust = 2:10,clMethods = clmethods,validation =
'stability',metric = 'manhattan',method = 'single')
summary(stab)
```

```

Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6 7 8 9 10

Validation Measures:
                        2      3      4      5      6      7      8      9     10

hierarchical APN  0.0047 0.0093 0.0128 0.0076 0.0098 0.0122 0.0151 0.0128 0.0244
             AD   7.0426 7.0153 6.9306 6.7449 6.6513 6.6172 6.5963 6.5432 6.5433
             ADM  0.0427 0.0828 0.1086 0.0786 0.0771 0.0918 0.1019 0.0926 0.1534
             FOM  0.9973 0.9965 0.9922 0.9900 0.9844 0.9784 0.9790 0.9790 0.9757
kmeans       APN  0.1138 0.2379 0.2606 0.2687 0.3121 0.3136 0.3153 0.3461 0.3928
             AD   5.7602 5.5180 5.1317 4.8138 4.6784 4.5149 4.3524 4.2405 4.2067
             ADM  0.4106 0.9673 0.7800 0.7851 0.7406 0.7678 0.7665 0.8149 0.8576
             FOM  0.8768 0.8675 0.8153 0.7971 0.7752 0.7600 0.7550 0.7514 0.7514
```

```
 pam          APN  0.1180 0.1552 0.2918 0.3692 0.4048 0.3982 0.4165 0.4477 0.4689
              AD   5.7947 5.1791 5.0211 4.9216 4.7081 4.4907 4.3678 4.3145 4.2199
              ADM  0.3728 0.4347 0.7307 0.9647 0.9264 0.8697 0.8510 0.9265 0.9387
              FOM  0.8808 0.8406 0.8103 0.7826 0.7567 0.7610 0.7311 0.7424 0.7418

 Optimal Scores:

     Score  Method       Clusters
 APN 0.0047 hierarchical 2
 AD  4.2067 kmeans       10
 ADM 0.0427 hierarchical 2
 FOM 0.7311 pam          8
```

```
stab = clValid(data.scaled, nClust = 2:10,clMethods = clmethods,validation =
'stability',metric = 'manhattan',method = 'average')
summary(stab)
```

```
 Clustering Methods:
  hierarchical kmeans pam

 Cluster sizes:
  2 3 4 5 6 7 8 9 10

 Validation Measures:
                         2      3      4      5      6      7      8      9     10

 hierarchical APN  0.0097 0.0613 0.1422 0.1334 0.1960 0.2651 0.1600 0.1885 0.2074
              AD   6.7302 6.6782 6.6551 6.2909 5.8095 5.7044 5.3121 5.1914 5.0978
              ADM  0.1303 0.2980 0.5162 1.1638 0.9205 1.0211 0.9292 0.8697 0.8342
              FOM  0.9657 0.9659 0.9602 0.9448 0.9032 0.9033 0.9027 0.8894 0.8737
 kmeans       APN  0.1138 0.2225 0.3154 0.2099 0.2818 0.3301 0.3719 0.3368 0.3341
              AD   5.7602 5.5062 5.2285 4.6567 4.5897 4.4469 4.3995 4.2379 4.0759
              ADM  0.4106 0.9353 0.9180 0.5478 0.7772 0.7933 0.9397 0.8791 0.7927
              FOM  0.8768 0.8731 0.8234 0.7865 0.7587 0.7510 0.7667 0.7389 0.7276
 pam          APN  0.1180 0.1552 0.2918 0.3692 0.4048 0.3982 0.4165 0.4477 0.4689
              AD   5.7947 5.1791 5.0211 4.9216 4.7081 4.4907 4.3678 4.3145 4.2199
              ADM  0.3728 0.4347 0.7307 0.9647 0.9264 0.8697 0.8510 0.9265 0.9387
              FOM  0.8808 0.8406 0.8103 0.7826 0.7567 0.7610 0.7311 0.7424 0.7418

 Optimal Scores:

     Score  Method       Clusters
 APN 0.0097 hierarchical 2
 AD  4.0759 kmeans       10
 ADM 0.1303 hierarchical 2
 FOM 0.7276 kmeans       10
```

```
stab = clValid(data.scaled, nClust = 2:10,clMethods = clmethods,validation =
'stability',metric = 'manhattan',method = 'complete')
summary(stab)
```

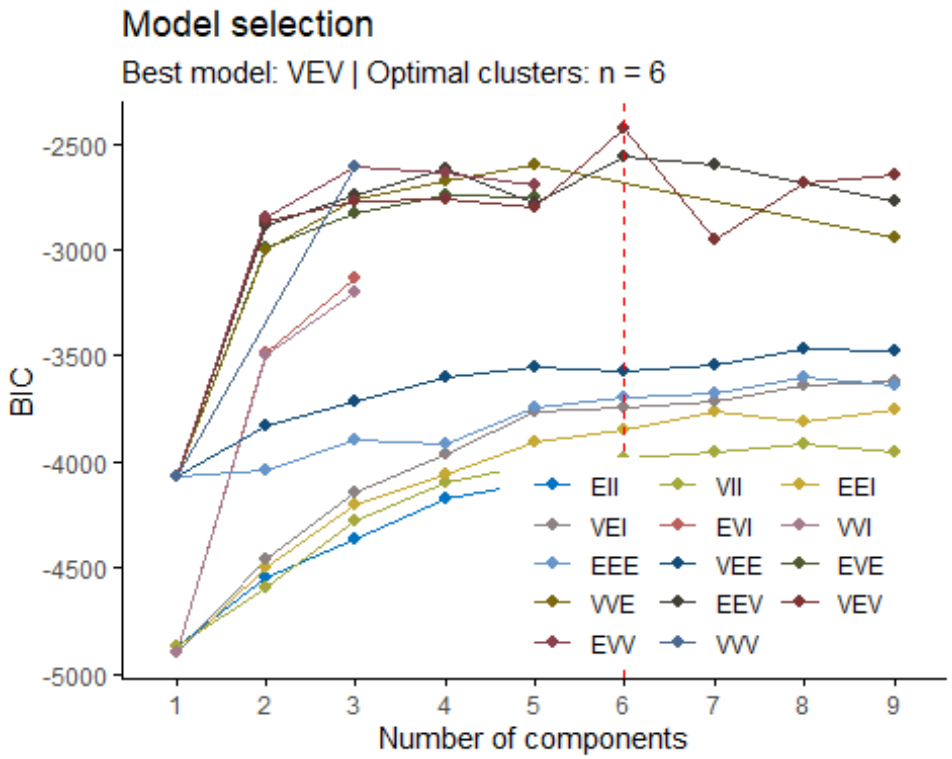
```
Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6 7 8 9 10

Validation Measures:
                      2      3      4      5      6      7      8      9     10

hierarchical APN  0.2018 0.3265 0.3094 0.3249 0.3593 0.3989 0.4358 0.4739 0.5005
             AD   6.2489 5.7776 5.5781 5.1977 5.0209 4.8401 4.7721 4.6917 4.6221
             ADM  0.7039 1.1494 1.1923 1.0248 1.0504 1.0402 1.0649 1.0987 1.1068
             FOM  0.9108 0.8717 0.8585 0.8376 0.8193 0.7862 0.7829 0.7797 0.7669
kmeans       APN  0.1088 0.2870 0.3166 0.2603 0.2349 0.3704 0.3704 0.4213 0.3603
             AD   5.7431 5.4701 5.1971 4.7262 4.4387 4.4674 4.3388 4.2986 4.0932
             ADM  0.3806 0.9822 1.0717 0.6589 0.5766 0.8770 0.8781 0.9483 0.8226
             FOM  0.8733 0.8372 0.8002 0.7796 0.7411 0.7466 0.7451 0.7393 0.7479
pam          APN  0.1180 0.1552 0.2918 0.3692 0.4048 0.3982 0.4165 0.4477 0.4689
             AD   5.7947 5.1791 5.0211 4.9216 4.7081 4.4907 4.3678 4.3145 4.2199
             ADM  0.3728 0.4347 0.7307 0.9647 0.9264 0.8697 0.8510 0.9265 0.9387
             FOM  0.8808 0.8406 0.8103 0.7826 0.7567 0.7610 0.7311 0.7424 0.7418

Optimal Scores:

    Score  Method Clusters
APN 0.1088 kmeans 2
AD  4.0932 kmeans 10
ADM 0.3728 pam    2
FOM 0.7311 pam    8
```

MODEL BASED CLUSTERING

The algorithms used until now, which are k-means, k-medoids and hierarchical clustering, aren't based on formal models, so it's necessary to use Model based clustering. This method supposes that data are generated by a mixture of models, usually one for each cluster.

To apply model-based clustering it's necessary to use the function mclust() from the mclust package. This method uses a Gaussian mixture with 14 models given by the combination of three constraints about the clusters, which are volume, shape and orientation.

```
mod = Mclust(data.scaled)

fviz_mclust(mod, 'BIC',palette='jco')
```




```
summary(mod)

 ----------------------------------------------------
 Gaussian finite mixture model fitted by EM algorithm
 ----------------------------------------------------

 Mclust VEV (ellipsoidal, equal shape) model with 6 components:

  log-likelihood   n  df       BIC       ICL
       -707.5614 243 185 -2431.339 -2446.107

 Clustering table:
  1  2  3  4  5  6
 20 60 13 61 56 33
```

```
This model suggest us to divide our dataset in 6 clusters, cluster 1 with 20
obs,cluster 2 with 60 obs, cluster 3 with 13 obs etc.
```

The model has a log-likelihood of -707.56 and uses 185 parameters, it has a BIC of -2431.339 and an ICL, which is the Integrated Completed Likelihood, of -2446.

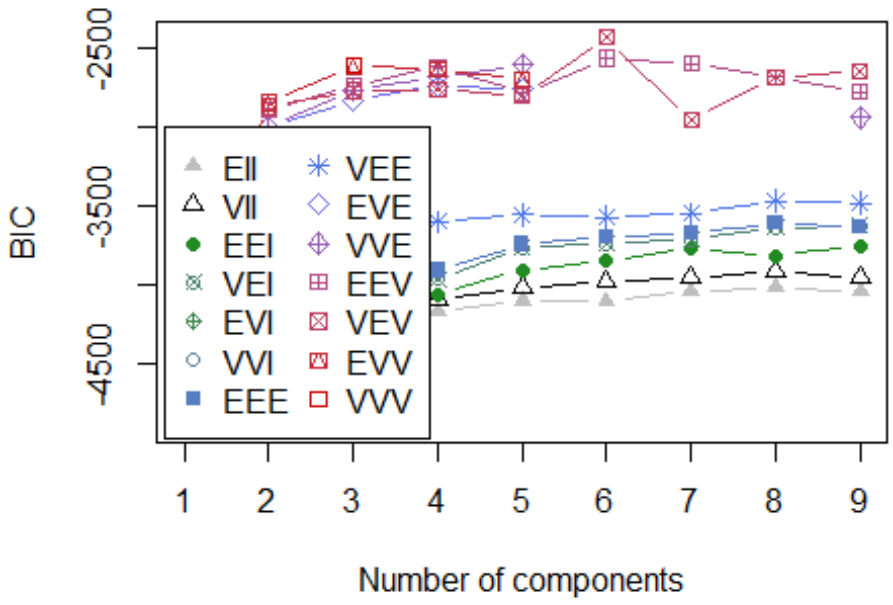
In Model Based Clustering, the BIC has to be maximized, because it implies a better trade-off between parsimony (the number of parameters) and accuracy (the fit of the model), and using the following command it's possible to see the three combination which had the highest BIC.

```
summary(mod$BIC)
```

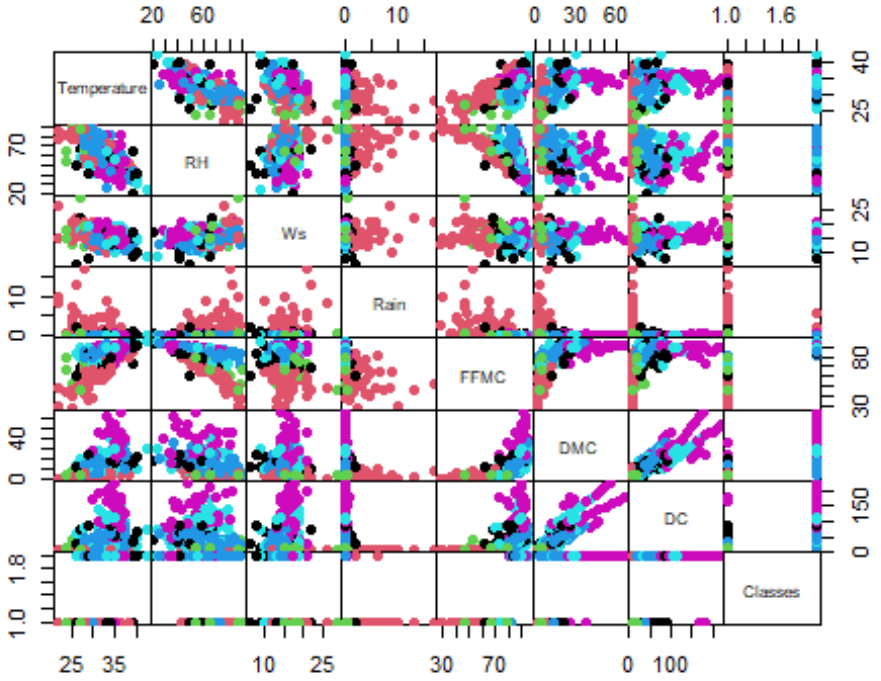
```
Best BIC values:
             VEV,6      EEV,6     EEV,7
BIC      -2431.339 -2561.5950 -2597.331
BIC diff     0.000  -130.2558  -165.992
```

It's possible, using the function fviz_mclust() to represent all the 14 possible models. On the horizontal axis there is the number of components, on the vertical axis is displayed the BIC, and the graph also shows the optimal number of clusters, which is 6.

```
plot(mod,what='BIC',ylim=range(mod$BIC,na.rm =
TRUE),legendArgs=list(x='bottomleft') )
```




```
mod$G

 [1] 6

pairs(data,gap=0,pch=16,col = mod$classification)

we also can give a pairs() rappresentation of what we said before.
```



Now we can compare this new cluster division with the categorical variable Classes and perform an External Validation

```
table(data$Classes,mod$classification)
```

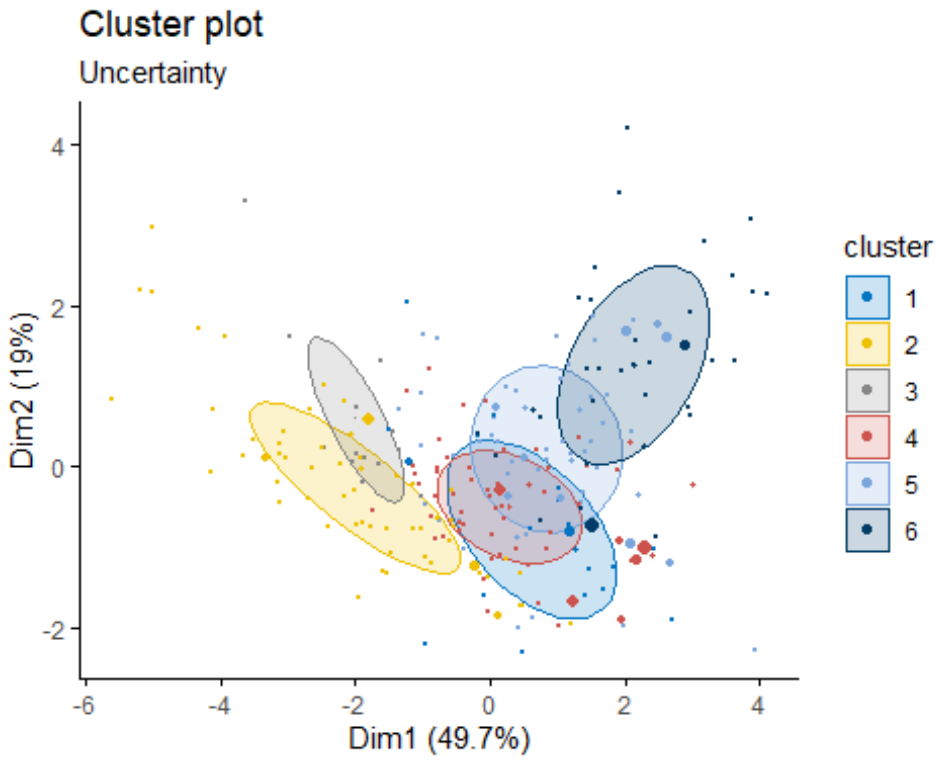
```
   1  2  3  4  5  6
0 12 57 13 10 10  4
1  8  3  0 51 46 29
```

```
adjustedRandIndex(data$Classes,mod$classification)
```

```
 [1] 0.2132234
```

This value is not so good as the previous one that we had, so maybe the best thing to do is to use k=2 clusters.

```
fviz_mclust(mod,'uncertainty',palette='jco')
```


Cluster plot
Uncertainty
Dim2 (19%)
Dim1 (49.7%)
4
2
0
-2
-6
-4
-2
0
2
4
cluster
1
2
3
4
5
6