MOST STREAMED SPOTIFY SONGS 2023

REPORT OF GIUSEPPE LEONARDI,

1000065630

University of Catania

LM-DATA

Data Analysis

Professor Antonio Punzo

Year 2023/24

CONTENTS

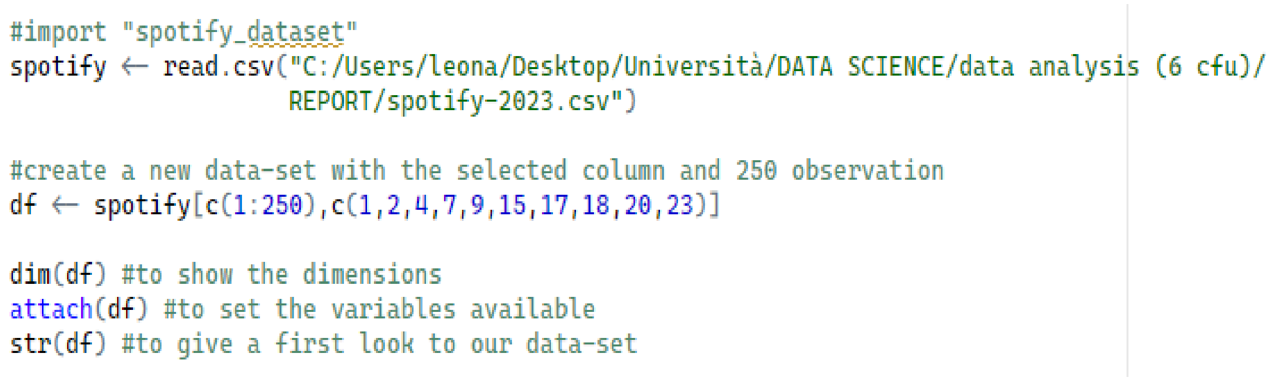
Introduction

The dataset that is object of the study contains a comprehensive list of the most famous songs of 2023 as listed on Spotify. It provides insights into each song's attributes, popularity, and presence on various music platforms. It comes from the community of scientists and data-analyzer "Kaggle", that provides dataset for free. The original dataset is composed by twenty-four variables and 953 observations (that corresponds to the name of the songs).

The starting point of the analysis is to drop some attributes and observation in which we are not interested. In this way, we can concentrate on the most important observation, to try to find pattern and relevant information. Due to this choice, I decided to maintain 250 observations and ten columns. the columns are the following:

- track_name
- artist.s_name
- released_year
- in_spotify_playlists
- streams
- bpm
- mode
- danceability
- energy
- liveness

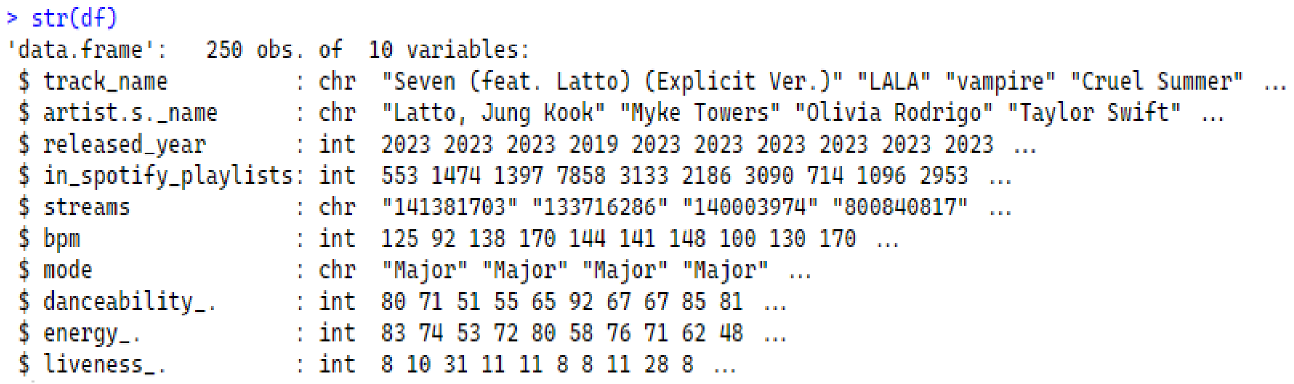
About the first three there is not so much to say in a preliminary way. For the others, we use the command view() to look the raw data. If we want to do a much better analysis, we can use the specific command:



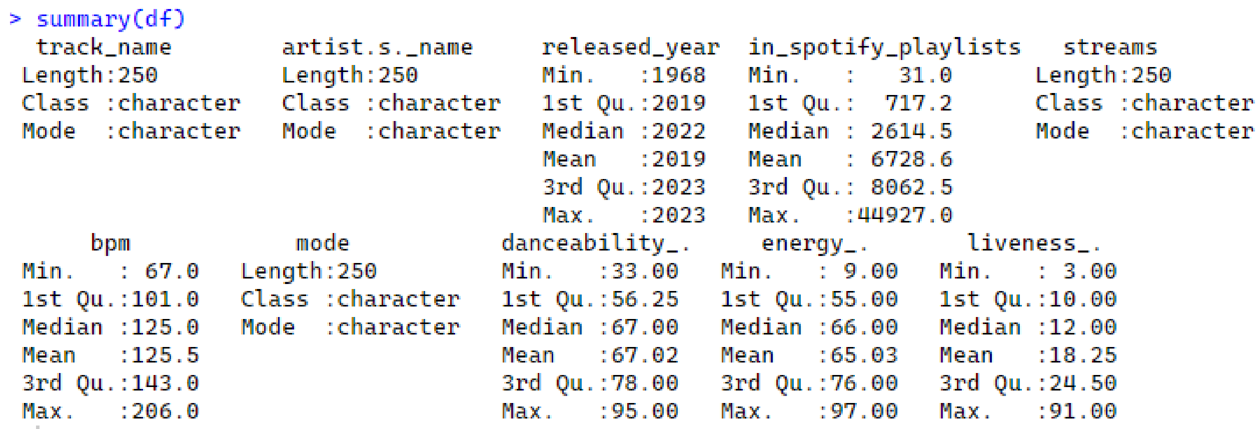
```
#import "spotify_dataset"
spotify <- read.csv("C:/Users/leona/Desktop/Università/DATA SCIENCE/data analysis (6 cfu)/
                    REPORT/spotify-2023.csv")

#create a new data-set with the selected column and 250 observation
df <- spotify[c(1:250),c(1,2,4,7,9,15,17,18,20,23)]

dim(df) #to show the dimensions
attach(df) #to set the variables available
str(df) #to give a first look to our data-set
```

The command dim() tells us that there are 250 row and ten column and attach() makes the variable findable without a specific calling of the dataset. Do not forget to use set.seed() to the make the result always replicable.

Another two essential functions of R are str() and summary().



```
> str(df)
'data.frame':	250 obs. of  10 variables:
 $ track_name          : chr  "Seven (feat. Latto) (Explicit Ver.)" "LALA" "vampire" "Cruel Summer" ...
 $ artist.s._name      : chr  "Latto, Jung Kook" "Myke Towers" "Olivia Rodrigo" "Taylor Swift" ...
 $ released_year       : int  2023 2023 2023 2019 2023 2023 2023 2023 2023 2023 ...
 $ in_spotify_playlists: int  553 1474 1397 7858 3133 2186 3090 714 1096 2953 ...
 $ streams             : chr  "141381703" "133716286" "140003974" "800840817" ...
 $ bpm                 : int  125 92 138 170 144 141 148 100 130 170 ...
 $ mode                : chr  "Major" "Major" "Major" "Major" ...
 $ danceability_.      : int  80 71 51 55 65 92 67 67 85 81 ...
 $ energy_.            : int  83 74 53 72 80 58 76 71 62 48 ...
 $ liveness_.          : int  8 10 31 11 11 8 8 11 28 8 ...
```

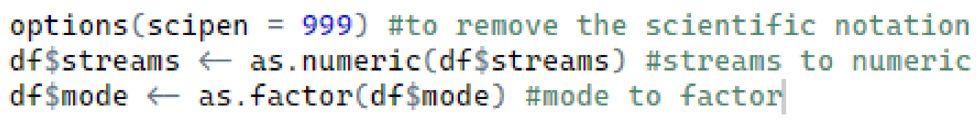


```
> summary(df)
  track_name         artist.s._name     released_year  in_spotify_playlists   streams         
 Length:250         Length:250         Min.   :1968   Min.   :   31.0      Length:250        
 Class :character   Class :character   1st Qu.:2019   1st Qu.:  717.2      Class :character  
 Mode  :character   Mode  :character   Median :2022   Median : 2614.5      Mode  :character  
                                       Mean   :2019   Mean   : 6728.6                        
                                       3rd Qu.:2023   3rd Qu.: 8062.5                        
                                       Max.   :2023   Max.   :44927.0                        
      bpm             mode           danceability_.     energy_.      liveness_.   
 Min.   : 67.0   Length:250         Min.   :33.00   Min.   : 9.00   Min.   : 3.00  
 1st Qu.:101.0   Class :character   1st Qu.:56.25   1st Qu.:55.00   1st Qu.:10.00  
 Median :125.0   Mode  :character   Median :67.00   Median :66.00   Median :12.00  
 Mean   :125.5                      Mean   :67.02   Mean   :65.03   Mean   :18.25  
 3rd Qu.:143.0                      3rd Qu.:78.00   3rd Qu.:76.00   3rd Qu.:24.50  
 Max.   :206.0                      Max.   :95.00   Max.   :97.00   Max.   :91.00  
```

Str() shows, for first, the dimension of the dataset, then for each variable, shows the type of the variables and the first observations. Summary(), instead, return the principal measures for each variable. As expected, there is no variable with negative value. The **support** of this variables is [0,∞], but the last three variables have support [0,1], because they are expressed in percentage. In fact, I multiplied these variables per 0.01 and renamed it to make them simpler.

```
> df <- df %>%
+   rename(danceability=danceability_.,energy=energy_.,liveness=liveness_.) %>%
+   mutate(across(c(danceability, energy, liveness), ~ . * 0.01))
>         view(df)
```

It is possible to see that there are some variables classified in the wrong way. So, we need to transform the type of this attribute. After this work, it is possible to go on with the **Univariate Analysis**.



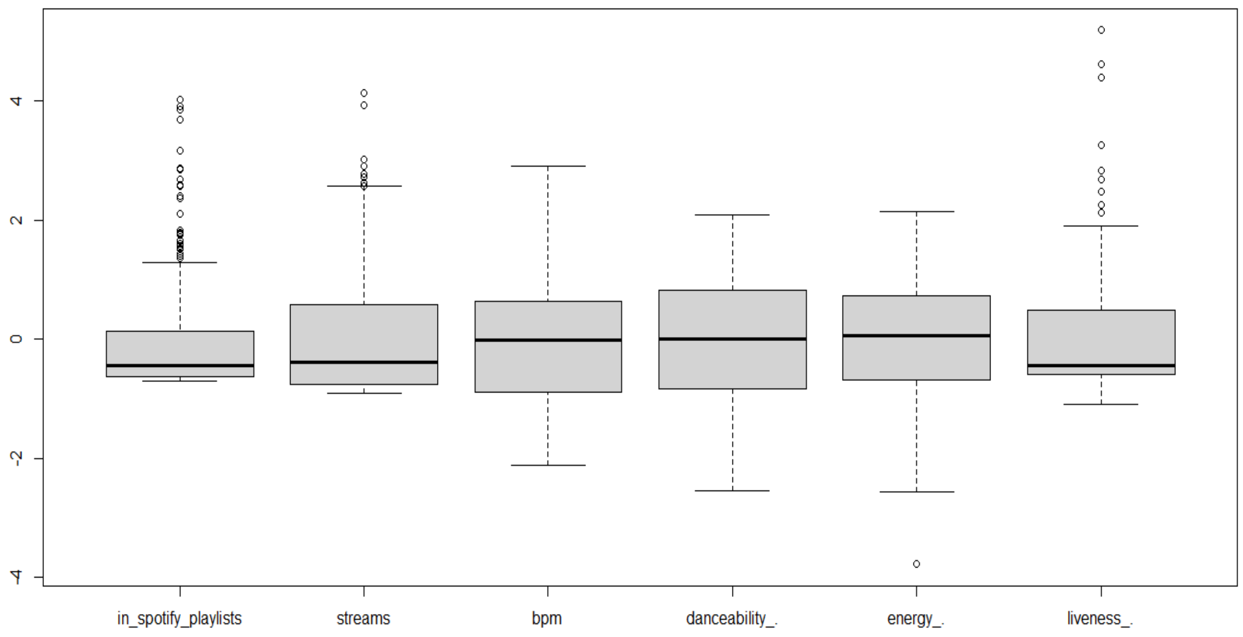
```
options(scipen = 999) #to remove the scientific notation
df$streams <- as.numeric(df$streams) #streams to numeric
df$mode <- as.factor(df$mode) #mode to factor
```

In this way is possible to do inference also on them, to see if the streams are in some relation with other variables or there is a preference for a specific mode. But we will study in deep this idea in the following chapter.

After this, a useful way to go on, is to create a dataframe with only numeric variables and standardize it, to have all the variables on the same scale.

```
df1 <- df[,c(4,5,6,8,9,10)]
scaled_df <- apply(df1,2,scale)
```

In this way is possible to the see the boxplot of each variable at the same time.



In the following chapter we will analyze one variable at time, trying to also understand the behavior of this outlier.

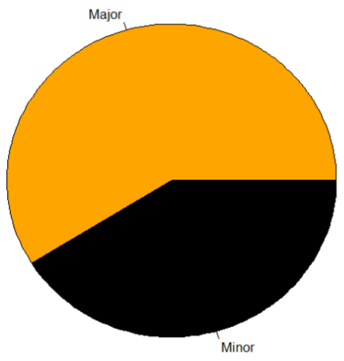
Univariate Analysis

Let us now look specifically at each single variable, to study their behavior. The starting variable that I choose for the univariate study is the only categorical one present in the dataset, i.e., ***mode***.

1.Mode

A song in a major key will revolve around the chords and notes of the major scale. A song in a minor key will, similarly, revolve around the chords and notes of the minor key.

```
> table(mode)
mode
Major Minor
  147   103
> freqMode <- table(mode)/length(mode)*100
> freqMode
mode
Major Minor
 58.8  41.2
> pie(freqMode, col = c("orange","black"))
```

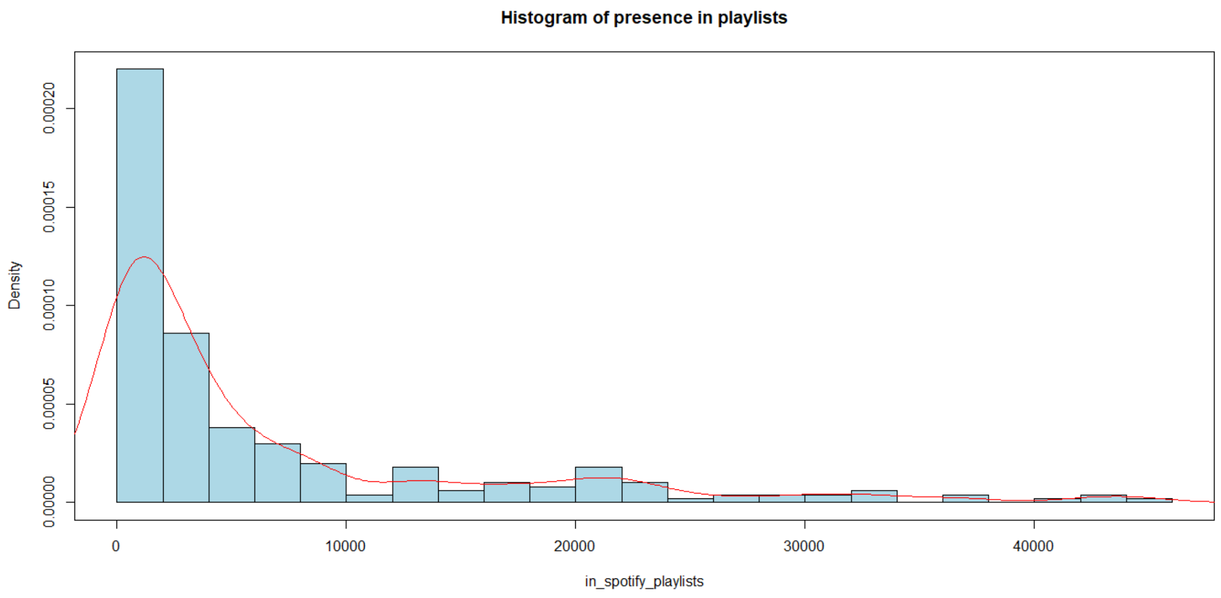


There is a prevalence of songs in major key.

2.In_spotify_playlist

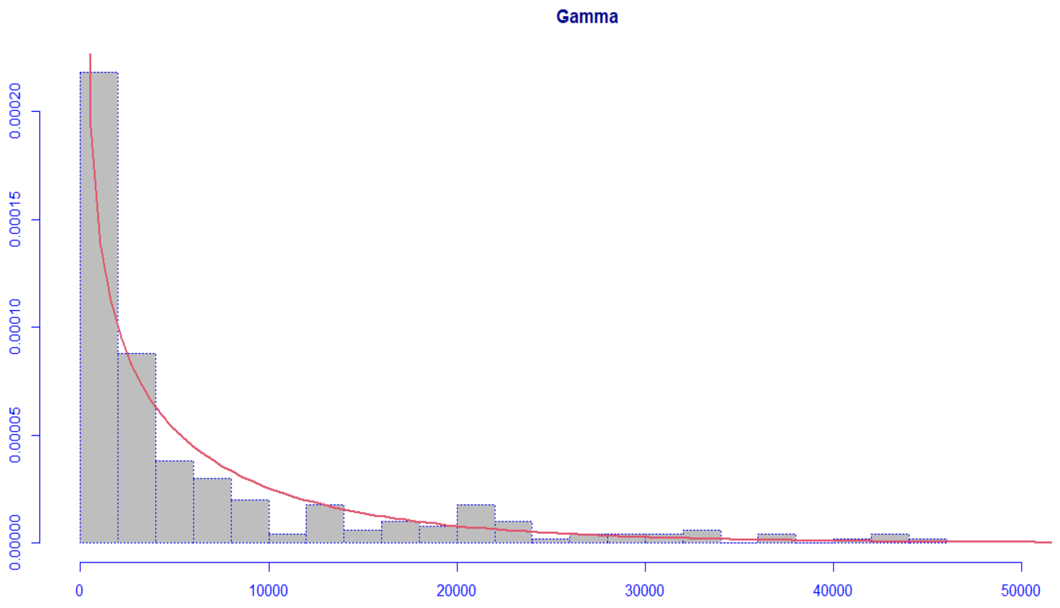
This variable counts in how many playlists of Spotify the song is present and return an integer, so it is a discrete variable, but it's possible to use it like a continuous one. We can extract the most essential information of it:

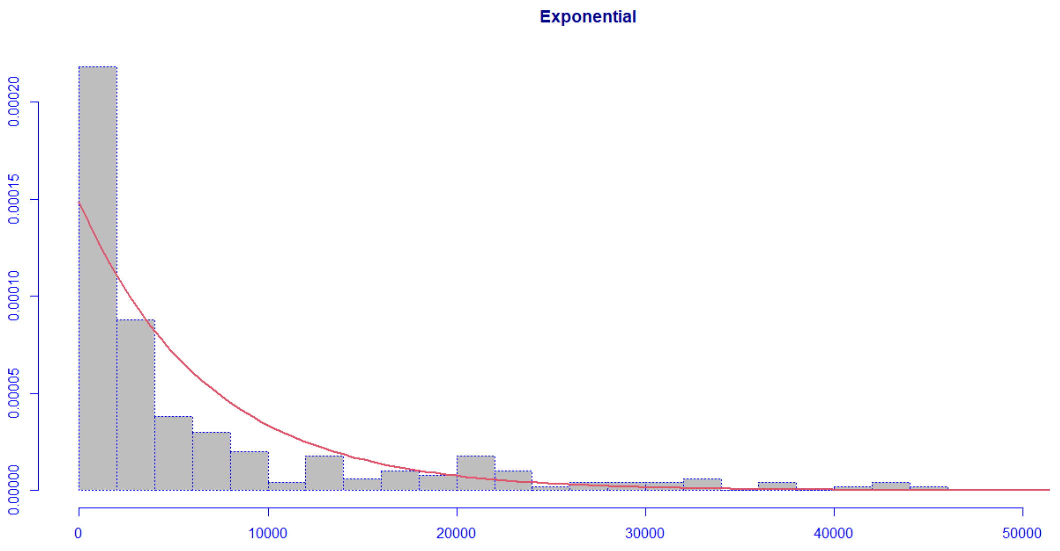
```
> sd(in_spotify_playlists)
[1] 9480.479                          #to compute the standard deviation
> skewness(in_spotify_playlists)
[1] 1.981623                          #positive skew: tail on the right
> kurtosis(in_spotify_playlists)
[1] 3.537535                          #leptokurtic distribution

> hist(in_spotify_playlists, breaks = 30, col="lightblue",freq = FALSE,main = "His
togram of presence in playlists")
> box()
> lines(density(in_spotify_playlists), col="red")
```

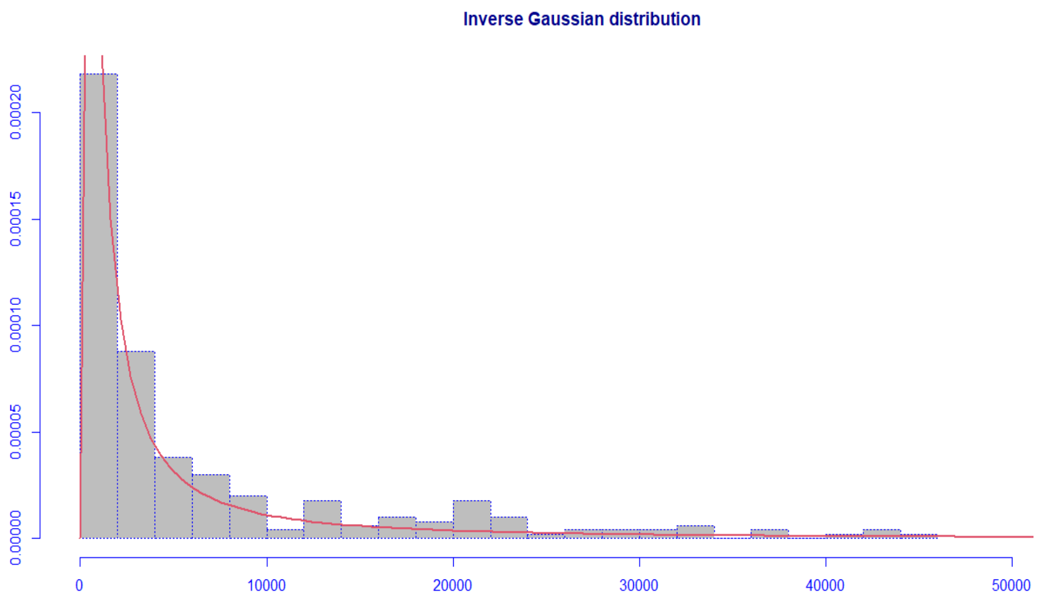


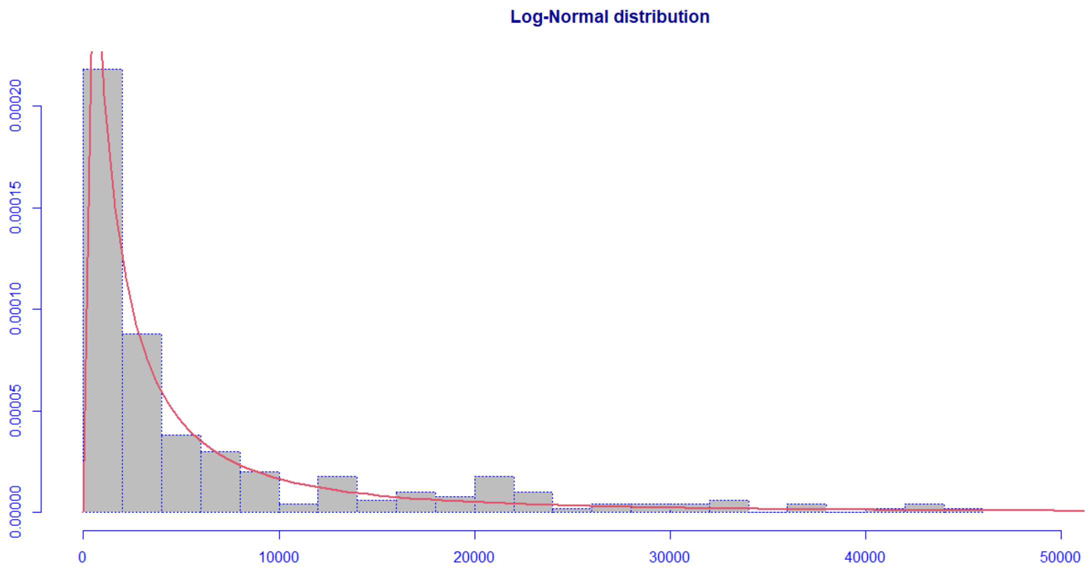
From this starting plot, I have tried to understand which is the best Distribution that could represent the data. This is an important choice to describe and summarize the complex behavior of the variable. Based on the support of in_spotify_playlists and on the barplot above, I have tried to fit the model with five Distribution: Gamma, Exponential, Inverse Gaussian, Log-Normal, Weibull.

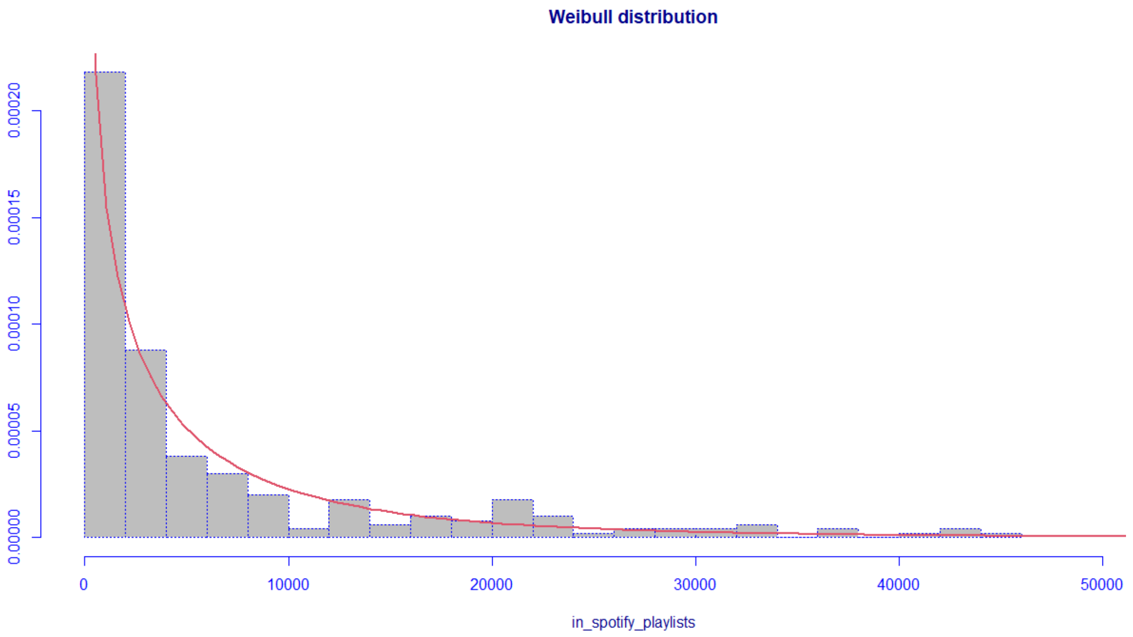
```
> fit.GA <- histDist(in_spotify_playlists, family=GA, nbins=30,
main="Gamma")
> fit.EXP <- histDist(in_spotify_playlists, family=EXP, nbins = 30,
main="Exponential")
> fit.IG <- histDist(in_spotify_playlists, family=IG, nbins = 30,
main="Inverse Gaussian distribution")
> fit.LN <- histDist(in_spotify_playlists, family=LOGNO, nbins = 30,
main="Log-Normal distribution")
> fit.WEI <- histDist(in_spotify_playlists, family=WEI, nbins = 30,
main="Weibull distribution")
```






Inverse Gaussian distribution
0.00000
0.00005
0.00010
0.00015
0.00020
0
10000
20000
30000
40000
50000


Log-Normal distribution
0.00000
0.00005
0.00010
0.00015
0.00020
0
10000
20000
30000
40000
50000


Weibull distribution
0.00000
0.00005
0.00010
0.00015
0.00020
0
10000
20000
30000
40000
50000
in_spotify_playlists

To compare (and then choose) the distribution that fits well our data, we use the **Akaike's information criterion** (**AIC**) and the **Schwartz's Bayesian Criterion** (**SBC**) for each distribution and then save the result of the test in a dataframe:

```
> data.frame(row.names = c("Gamma","Exponential", "Inverse Gaussian", "Log-Normal"
, "Weibull"), AIC=c(AIC(fit.GA), AIC(fit.EXP), AIC(fit.IG), AIC(fit.LN), AIC(fit.W
EI)), SBC=c(fit.GA$sbc, fit.EXP$sbc, fit.IG$sbc, fit.LN$sbc, fit.WEI$sbc))
                      AIC      SBC
Gamma            4854.587 4861.630
Exponential      4909.061 4912.582
Inverse Gaussian 4872.930 4879.973
Log-Normal       4832.693 4839.735
Weibull          4844.132 4851.175
> LR.test(fit.WEI,fit.LN)
 Likelihood Ratio Test for nested GAMLSS models.
 (No check whether the models are nested is performed).

       Null model: deviance= 4840.132 with  2 deg. of freedom
 Alternative model: deviance= 4828.693 with  2 deg. of freedom

 LRT = 11.4392 with 0 deg. of freedom and p-value= 0
```

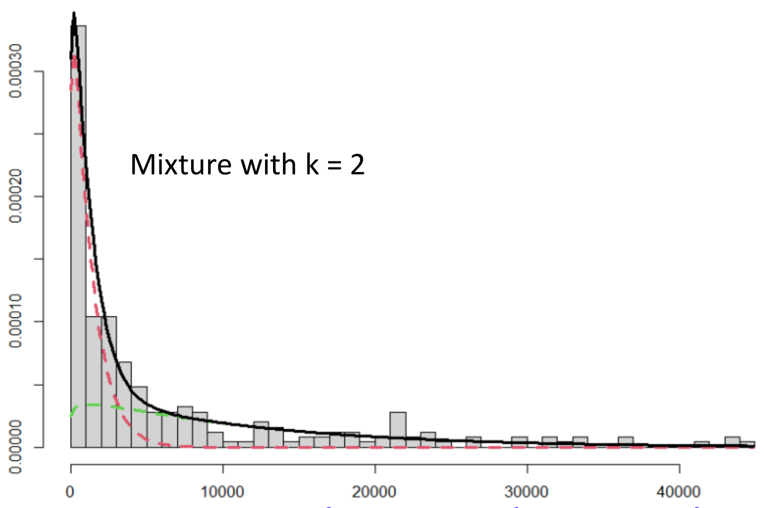
The two indexes above are information criteria, and they are used for comparing statistical models, to evaluate which one fits better our data. For both, the rule is to choose the distribution that has the smallest value of this index. And in our case the best-fit is the ***Log-Normal Distribution***.

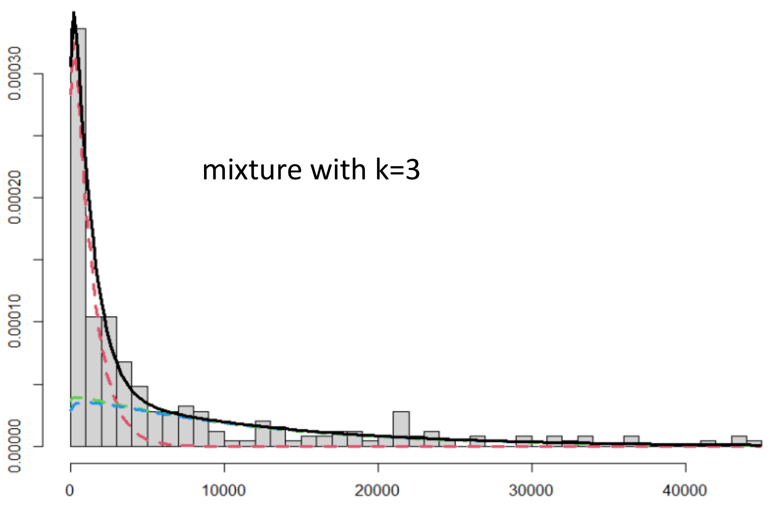
To improve this conclusion, I decided to also perform a Likelihood-ratio test between the two distributions with the smallest value of the index. As null Hypothesis, the Weibull is the model that fits better, and on the other side we the hypothesis that is the log-normal the best model for our data. Looking at the p-value, we can reject the null hypothesis and we can conclude that the best statistical model for in_spotify_playlists is the **Log-Normal**.

Another way to follow is the **log-normal mixture distribution**, often used to model positive-valued data that are contaminated; that is, most of the values appear to come from a single lognormal distribution, but a few “outliers” appear. I have tried to do this first with the mix of two lognormal distribution (k=2), then I have tried to do the same but with three distributions.

```
> #MIXTURE MODEL (LOGNO WITH K=2)
> fit.LOGNO.2 <- gamlssMX(formula = in_spotify_playlists~1, family = LOGNO, K = 2,
data = NULL)

> # estimate of mu and sigma in groups
> mu.hat1 <- exp(fit.LOGNO.2[["models"]][[1]][["mu.coefficients"]])
> sigma.hat1 <- exp(fit.LOGNO.2[["models"]][[1]][["sigma.coefficients"]])
> mu.hat2 <- exp(fit.LOGNO.2[["models"]][[2]][["mu.coefficients"]])
> sigma.hat2 <- exp(fit.LOGNO.2[["models"]][[2]][["sigma.coefficients"]])

>#plot
> hist(in_spotify_playlists, breaks = 50,freq = FALSE)
> lines(seq(min(in_spotify_playlists),max(in_spotify_playlists),length=length(in_s
potify_playlists)),fit.LOGNO.2[["prob"]][1]*dGA(seq(min(in_spotify_playlists),max(
in_spotify_playlists),length=length(in_spotify_playlists)), mu = mu.hat1, sigma =
sigma.hat1),lty=2,lwd=3,col=2)

> lines(seq(min(in_spotify_playlists),max(in_spotify_playlists),length=length(in_s
potify_playlists)),fit.LOGNO.2[["prob"]][2]*dGA(seq(min(in_spotify_playlists),max(
in_spotify_playlists),length=length(in_spotify_playlists)), mu = mu.hat2, sigma =
sigma.hat2),lty=2,lwd=3,col=3)

> lines(seq(min(in_spotify_playlists),max(in_spotify_playlists),length=length(in_s
potify_playlists)),fit.LOGNO.2[["prob"]][1]*dGA(seq(min(in_spotify_playlists),max(
in_spotify_playlists),length=length(in_spotify_playlists)), mu = mu.hat1, sigma =
sigma.hat1) +fit.LOGNO.2[["prob"]][2]*dGA(seq(min(in_spotify_playlists),max(in_spo
tify_playlists),length=length(in_spotify_playlists)), mu = mu.hat2, sigma = sigma.
hat2),lty = 1, lwd = 3, col = 1)
```

The code is the same in the case with k = 3, with the only difference in that we must calculate another two parameters. I decided to report only the plot and the dataframe where the AIC and the BIC get evaluated.





```
> data.frame(row.names=c('K=2','K=3'),AIC=c(fit.LOGNO.2$aic,fit.LOGNO.3$aic),
BIC=c(fit.LOGNO.2$sbc,fit.LOGNO.3$sbc))
          AIC      SBC
K=2 4825.735 4843.343
K=3 4831.740 4859.912
```

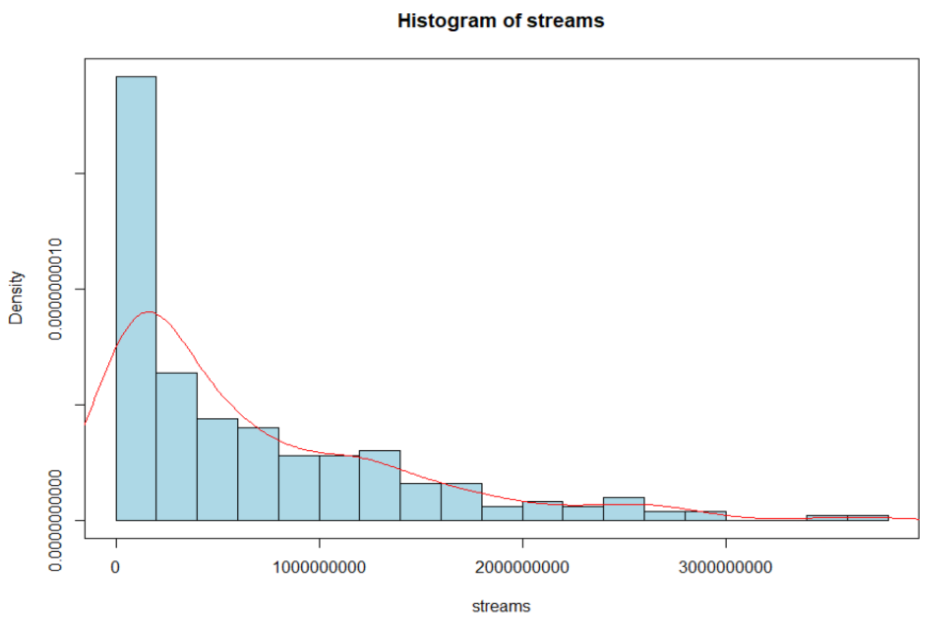
In this case, the model with two distributions fits better our variable. As we can see, the AIC of the mixture model is less than the AIC of the lognormal, but the BIC is less in the not-mixture type. So, for simplicity, I have decided to use the Log-normal distribution to explain this data.

3.streams

It represents the total number of streams per song on Spotify. It's an integer value that assume unique values.

```
> summary(streams)
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
      2762  109944015  380385350  667594078 1088473226 3703895074
> sd(streams)
[1] 735314878
> skewness(streams)
[1] 1.525543
> kurtosis(streams)
[1] 5.110072
```

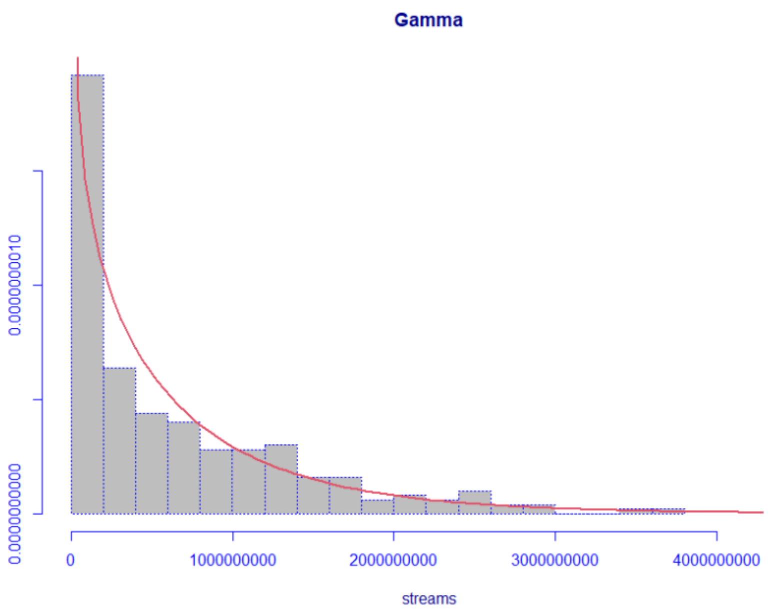
this first approach tells us that it's a variable that have big variation, a right skewed tail and it is a leptokurtic curve. The graphical representation will give us an image clearer of the situation:

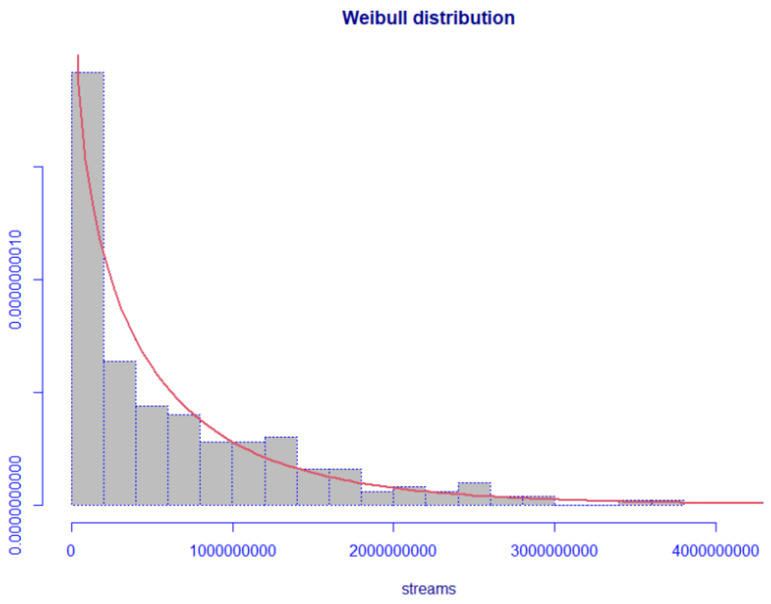


The plots respect the prevision that we did. According to the distribution of the data, we search for a distribution that could fit better our data.

```
> fit.GA <- histDist(streams, family=GA, nbins=20, main="Gamma")
> fit.EXP <- histDist(streams, family=EXP, nbins = 20, main="Exponential")
> fit.IG <- histDist(streams, family=IG, nbins = 20, main="Inverse Gaussian distri
bution")
> fit.LN <- histDist(streams, family=LOGNO, nbins = 20, main="Log-Normal distribut
ion")
> fit.WEI <- histDist(streams, family=WEI, nbins = 20, main="Weibull distribution"
)
```

To be concise, I only report the graphs of the two best distributions: the GAMMA and the WEIBULL.





We need to evaluate the the AIC and the BIC:

```
> data.frame(row.names = c("Gamma","Exponential", "Inverse Gaussian", "Log-Normal",
"Weibull"),    AIC=c(AIC(fit.GA),    AIC(fit.EXP),    AIC(fit.IG),    AIC(fit.LN),
AIC(fit.WEI)), SBC=c(fit.GA$sbc, fit.EXP$sbc, fit.IG$sbc, fit.LN$sbc, fit.WEI$sbc))
```

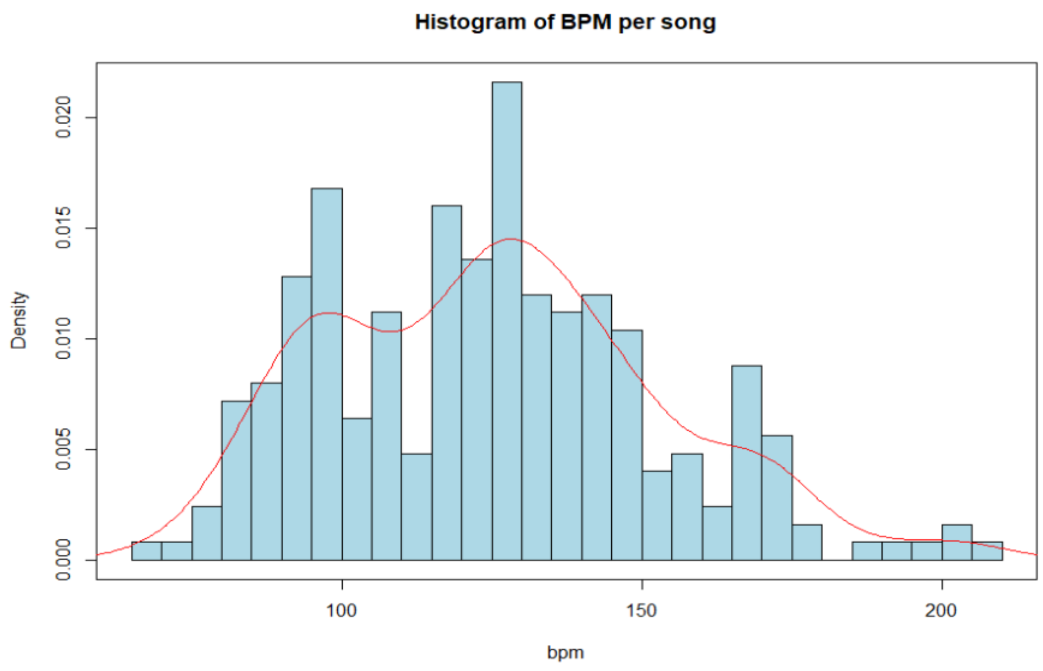
```
                      AIC      SBC
Gamma             10651.33 10658.38
Exponential       10661.60 10665.12
Inverse Gaussian 12012.70 12019.75
Log-Normal        10701.85 10708.89
Weibull           10650.99 10658.03
```

As we see, the Weibull has the less values for these indexes, so we choose this model to represent this variable.

4.bpm

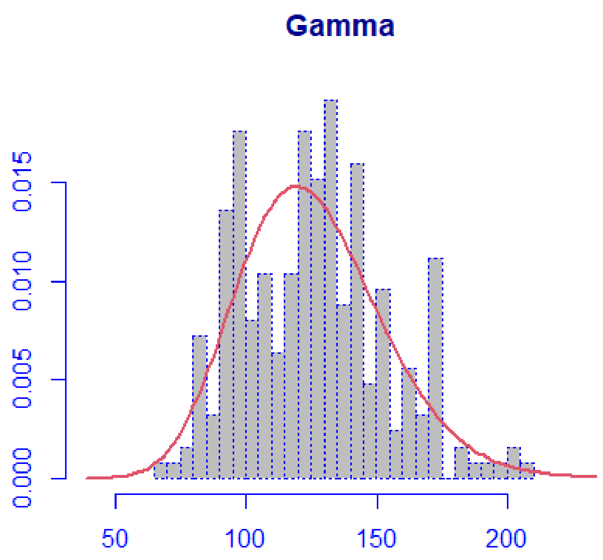
```
> summary(bpm)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   67.0   101.0   125.0   125.5   143.0   206.0 
> sd(bpm)
[1] 27.62909
> skewness(bpm)
[1] 0.4088577
> kurtosis(bpm)
[1] 2.793877
> hist(bpm, breaks = 30, col="lightblue",freq = FALSE, main = "Histogram of presen
ce in playlists")
> box()
> lines(density(bpm), col="red")
```

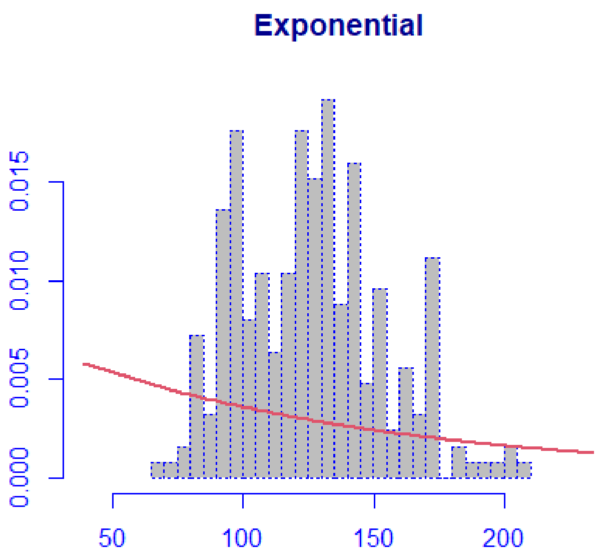
The BPM is a continuous variable with a small standard error. As we know, in the real life, the BPM range goes from 40 to 300 (for hard metal songs) so the measurement results coherent. The skewness is near to zero, so there isn't the presence of a tail on one side and the kurtosis indicate that the curve is asymmetric.

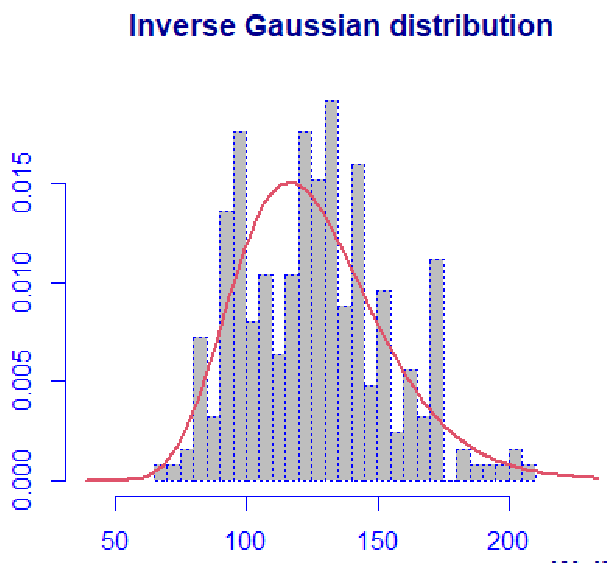


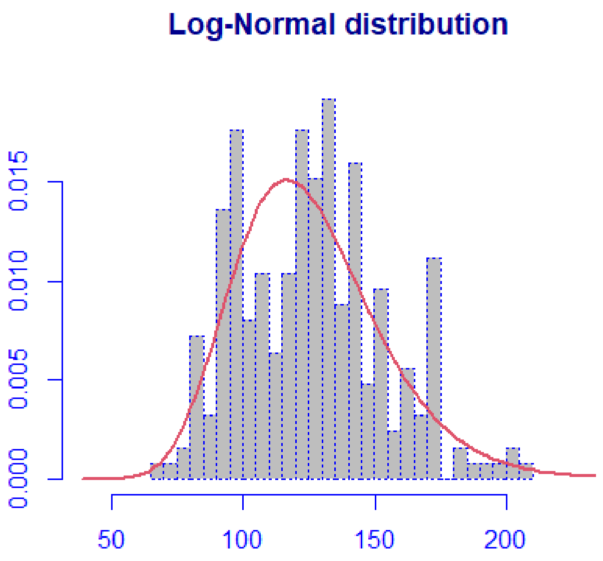
In this case, the plot of the variable follows another trend, with peaks that suggest that we need a mixture model. But let's start with searching one model to fit the data.

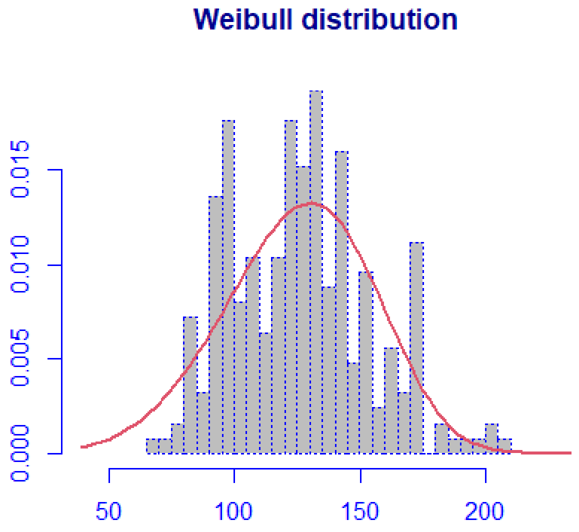
```
> fit.GA <- histDist(bpm, family=GA, nbins=30, main="Gamma")
> fit.EXP <- histDist(bpm, family=EXP, nbins = 30, main="Exponential")
> fit.IG <- histDist(bpm, family=IG, nbins = 30, main="Inverse Gaussian distributi
on")
> fit.LN <- histDist(bpm, family=LOGNO, nbins = 30, main="Log-Normal distribution"
)
> fit.WEI <- histDist(bpm, family=WEI, nbins = 30, main="Weibull distribution")
```











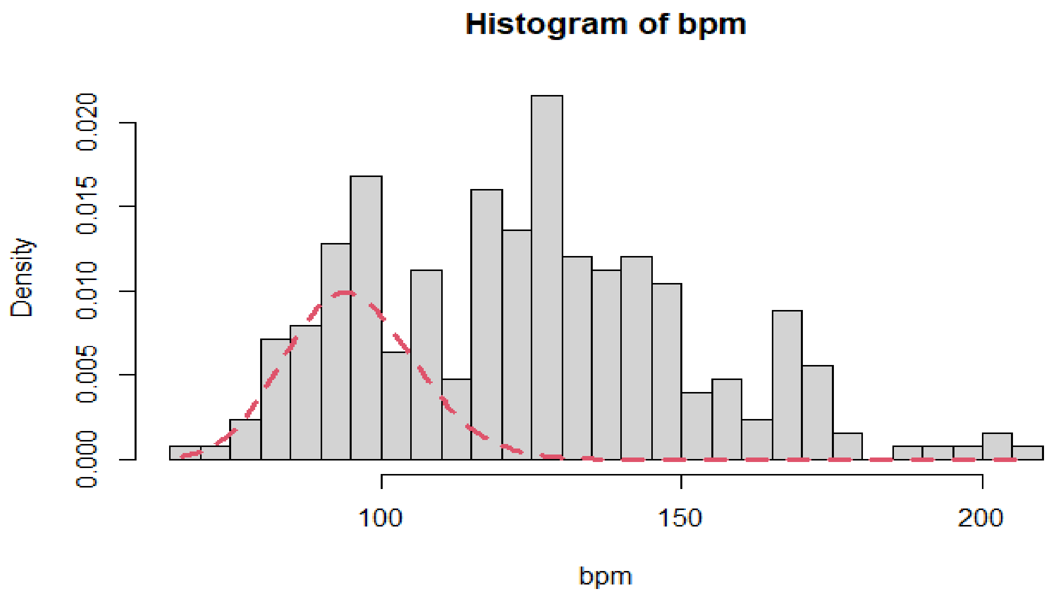
```
> data.frame(row.names = c("Gamma","Exponential", "Inverse Gaussian", "Log-Normal"
, "Weibull"), AIC=c(AIC(fit.GA), AIC(fit.EXP), AIC(fit.IG), AIC(fit.LN), AIC(fit.W
EI)), SBC=c(fit.GA$sbc, fit.EXP$sbc, fit.IG$sbc, fit.LN$sbc, fit.WEI$sbc))
                      AIC      SBC
Gamma            2362.075 2369.118
Exponential      2918.280 2921.802
Inverse Gaussian 2361.691 2368.734
Log-Normal       2362.171 2369.214
Weibull          2385.535 2392.578
> LR.test(fit.WEI,fit.IG)
 Likelihood Ratio Test for nested GAMLSS models.
 (No check whether the models are nested is performed).

       Null model: deviance= 2381.535 with  2 deg. of freedom
 Altenative model: deviance= 2357.691 with  2 deg. of freedom

 LRT = 23.84437 with 0 deg. of freedom and p-value= 0
```

The result of the HistDist() and of the likelihood ratio test are irrefutable: the best fit for the BPM is the inverse-Gaussian. In fact, the p-value is 0, so we refuse the hypothesis where is the Weibull the best model to fit. Following this way, I've tried to fit BPM with a mixture of Inverse Gaussian distributions. I've started first from k=2 and then k=3.

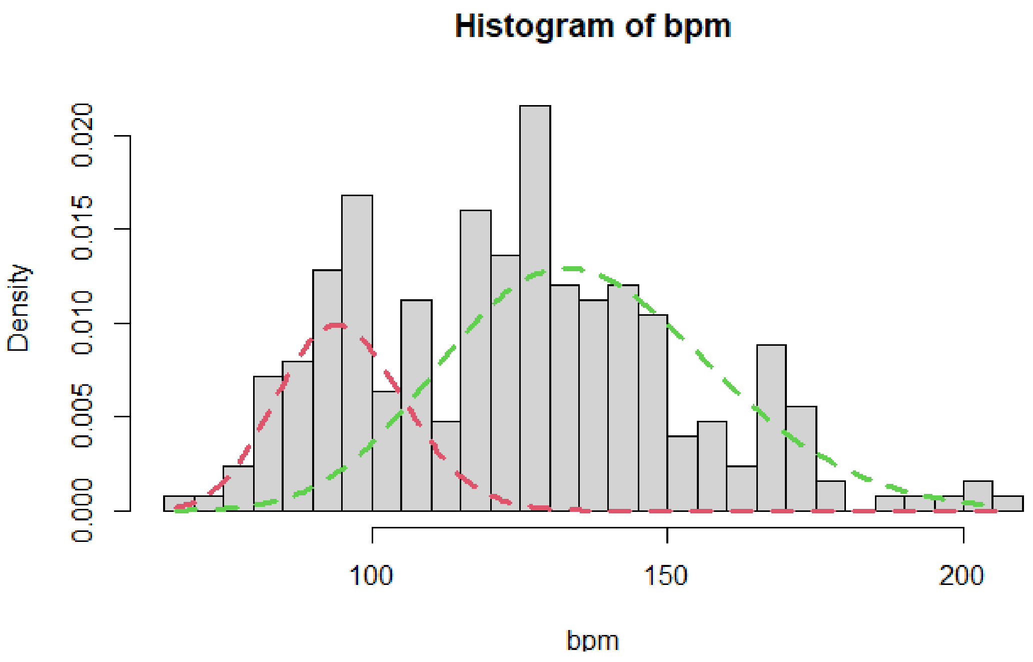
```
> #MIXTURE MODEL (LOGNO WITH K=2)
> set.seed(123)
> fit.IG.2 <- gamlssMX(formula = bpm~1, family = GA, K = 2, data = NULL)
>
> # estimate of mu and sigma in group 1
> mu.hat1 <- exp(fit.IG.2[["models"]][[1]][["mu.coefficients"]])
> sigma.hat1 <- exp(fit.IG.2[["models"]][[1]][["sigma.coefficients"]])
>
> # estimate of mu and sigma in group 2
> mu.hat2 <- exp(fit.IG.2[["models"]][[2]][["mu.coefficients"]])
> sigma.hat2 <- exp(fit.IG.2[["models"]][[2]][["sigma.coefficients"]])
```

Now I add the lines in the plot to show the two curve and their summation.

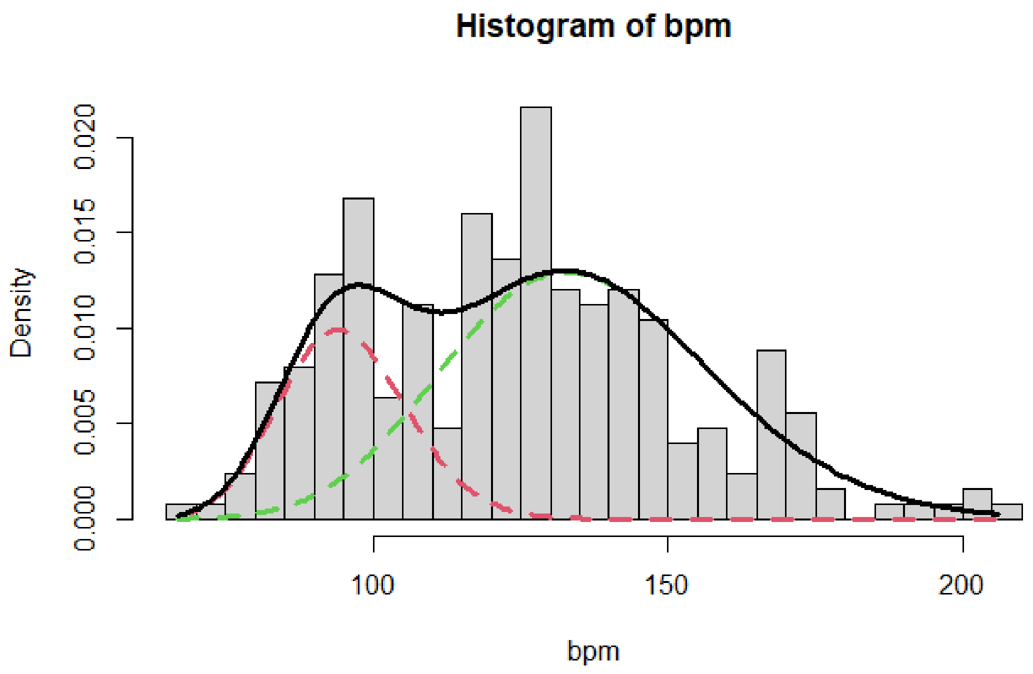
```
> hist(bpm, breaks = 20,freq = FALSE)
>  lines(seq(min(bpm),max(bpm),length=length(bpm)),fit.IG.2[["prob"]][1]*dGA(seq(mi
n(bpm),max(bpm),length=length(bpm)), mu = mu.hat1, sigma = sigma.hat1),lty=2,lwd=3
,col=2)
```



```
> lines(seq(min(bpm),max(bpm),length=length(bpm)),fit.IG.2[["prob"]][2]*dGA(seq(mi
n(bpm),max(bpm),length=length(bpm)), mu = mu.hat2, sigma = sigma.hat2),lty=2,lwd=3
,col=3)
```



```
> lines(seq(min(bpm),max(bpm),length=length(bpm)),fit.IG.2[["prob"]][1]*dGA(seq(mi
n(bpm),max(bpm),length=length(bpm)), mu = mu.hat1, sigma = sigma.hat1)+fit.IG.2[["
prob"]][2]*dGA(seq(min(bpm),max(bpm),length=length(bpm)), mu = mu.hat2, sigma = si
gma.hat2),lty = 1, lwd = 3, col = 1)
```

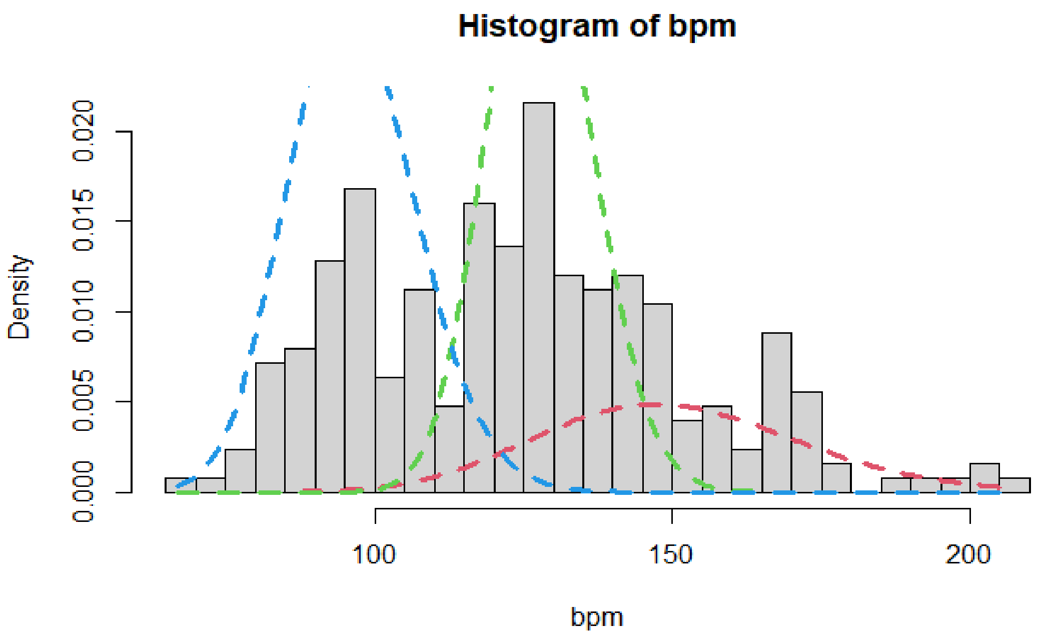


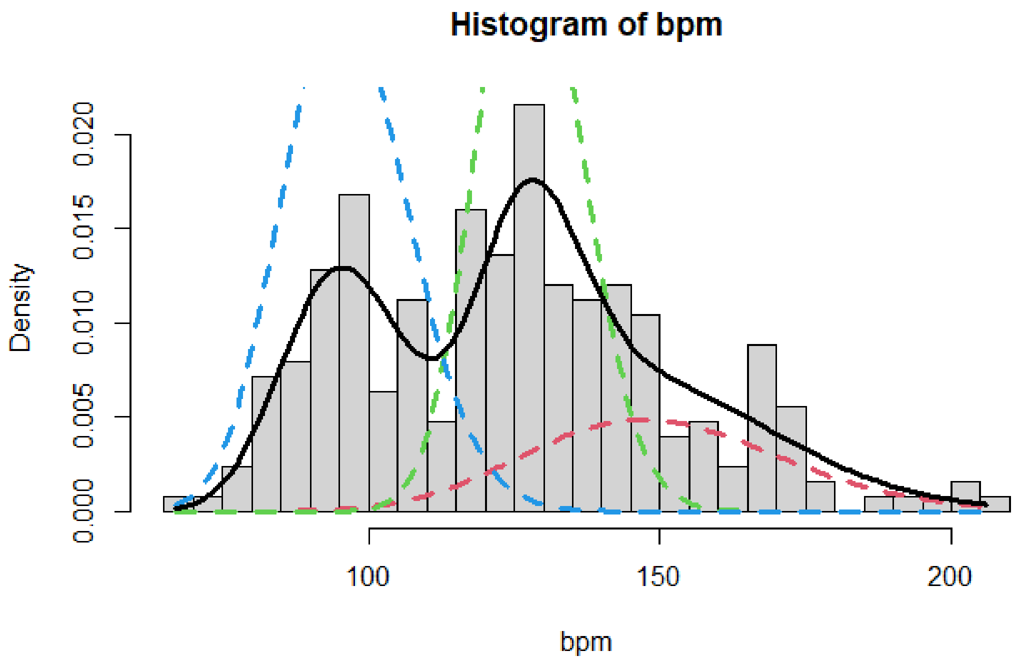
Now we try to fit the model with three distributions (k=3) and then we compare the BIC and the AIC.

```
> set.seed(123)
> fit.IG.3 <- gamlssMX(formula = bpm~1, family = GA, K = 3, data = NULL)
```

The code is the same as before, with the only difference that we calculate two parameters more,   f or the third distribution, and we must add another line on the plot that I have colored in blue.

```
> # estimate of mu and sigma in group 3
> mu.hat3 <- exp(fit.IG.3[["models"]][[3]][["mu.coefficients"]])
> sigma.hat3 <- exp(fit.IG.3[["models"]][[3]][["sigma.coefficients"]])
```





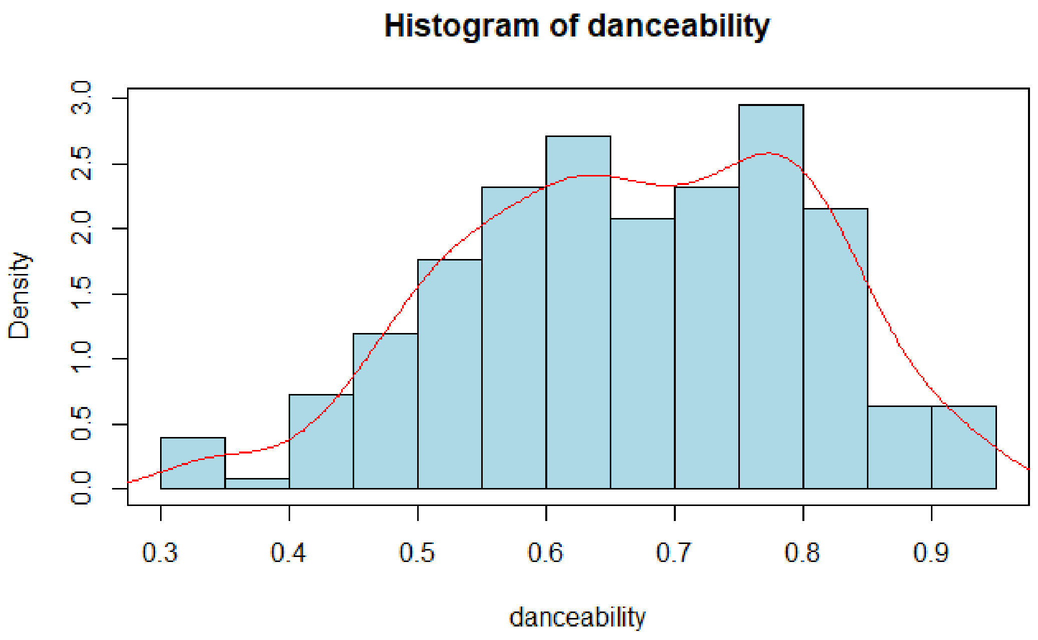
```
> data.frame(row.names=c('K=2','K=3'),AIC=c(fit.IG.2$aic,fit.IG.3$aic),SBC=c(fit.I
G.2$sbc,fit.IG.3$sbc))
         AIC      SBC
K=2 2358.008 2375.615
K=3 2354.333 2382.505
```

In this case I've choose to fit the variable with the mixture of three Inverse-Gaussian Models.

5.danceability

This variable, like the following two, is in percentage. Danceability express how much a song is suitable for dancing. The support is [0,1] and its principal measurements are:

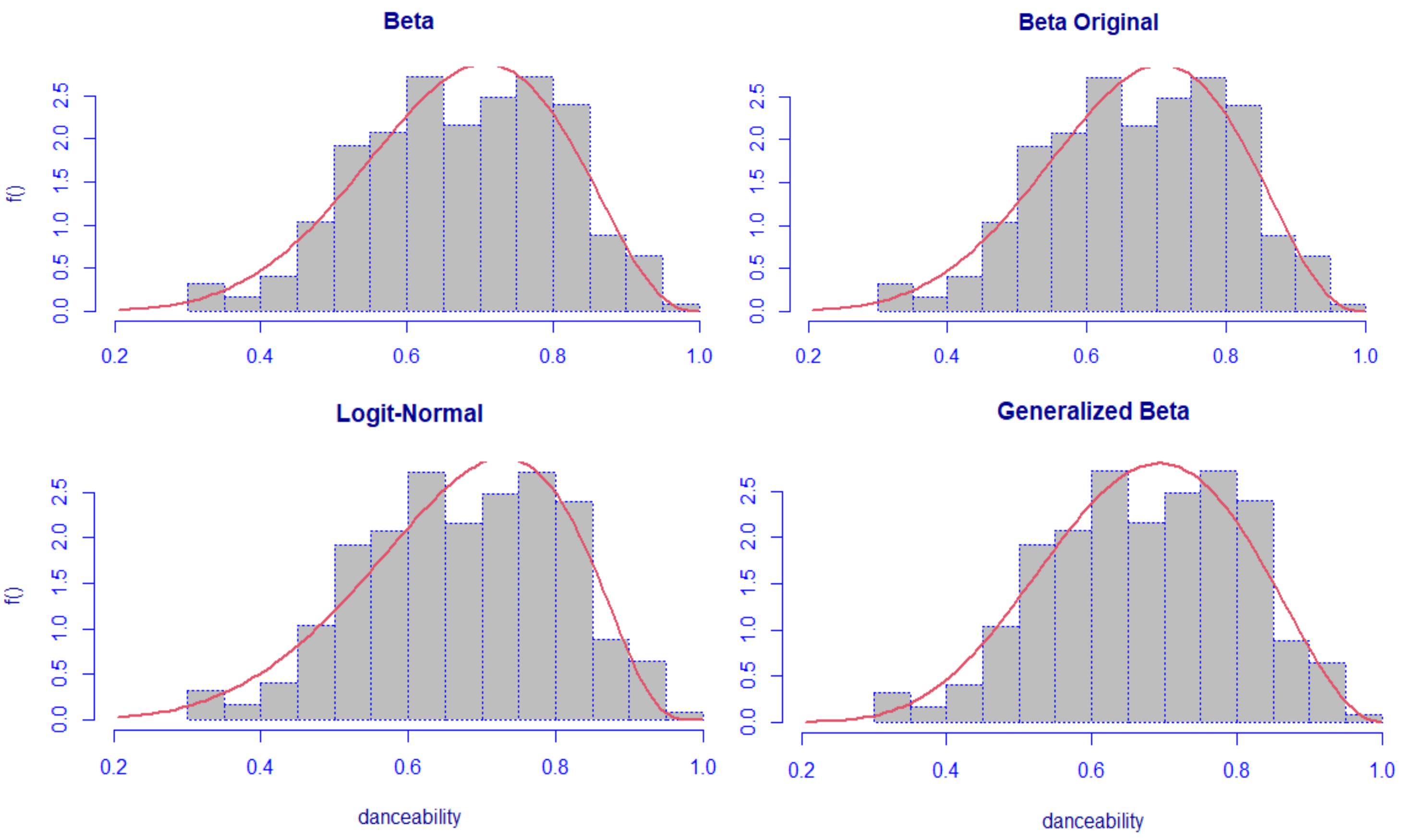
```
> round(summary(danceability),2)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   0.33    0.56    0.67    0.67    0.78    0.95 
> sd(danceability)
[1] 0.133651
> skewness(danceability)
[1] -0.2344608
> hist(danceability, col="lightblue",freq = FALSE, main = "Histogram of danceabili
ty")
> box()
> lines(density(danceability), col="red")
```



It does not follow the support of the precedent variables, so is necessary to confront the observed data with other distributions, to find a theoretical model that fits well this data. Another important aspect is the support of this variable. It can assume value between 0 and 1, so we need a model that respect this criteria and work on the same support. Models like this are:

1. Beta distribution
2. Beta original distribution
3. Logit-normal distribution
4. Generalized beta type 1

```
> fit.BE <- histDist(danceability, family=BE, nbins=10, main="Beta")
> fit.BEo <- histDist(danceability, family=BEo, nbins = 10, main="Beta Original")
> fit.LOGITNO <- histDist(danceability, family=LOGITNO, nbins = 10, main="Logit-No
rmal")
> fit.GB1 <- histDist(danceability, family=GB1, nbins = 10, main="Generalized Beta
")
```

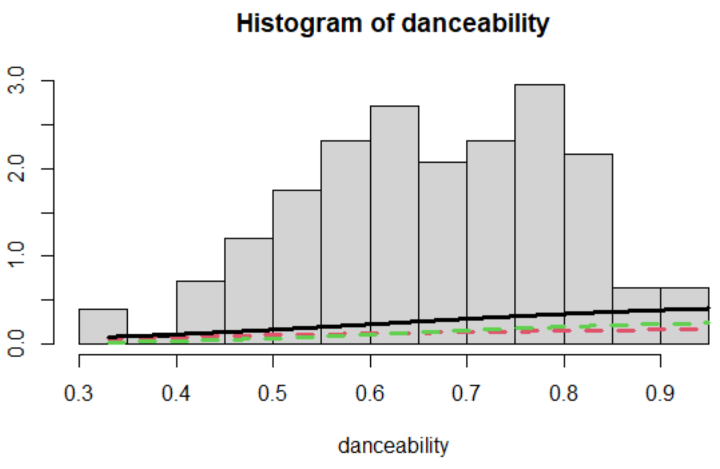


The models are very similar, so we look to the value of the bic to choose the better model.

```
> data.frame(row.names = c("Beta","Beta Or", "Gen Beta", "logit Norm"), AIC=c(AIC(
fit.BE), AIC(fit.BEo), AIC(fit.GB1), AIC(fit.LOGITNO)), SBC=c(fit.BE$sbc, fit.BEo$
sbc, fit.GB1$sbc, fit.LOGITNO$sbc))
                 AIC       SBC
Beta       -304.1682 -297.1253
Beta Or    -304.1682 -297.1253
Gen Beta   -301.5034 -287.4175
logit Norm -300.4637 -293.4208
```

in this case, the beta model and the original one has the same value, so we choose the Beta.
It is unnecessary to execute a likelihood-ratio test. I have tried also to perform a mixture model on the variable, but it does not work well. So finally, we choose the Beta model to explain our data.

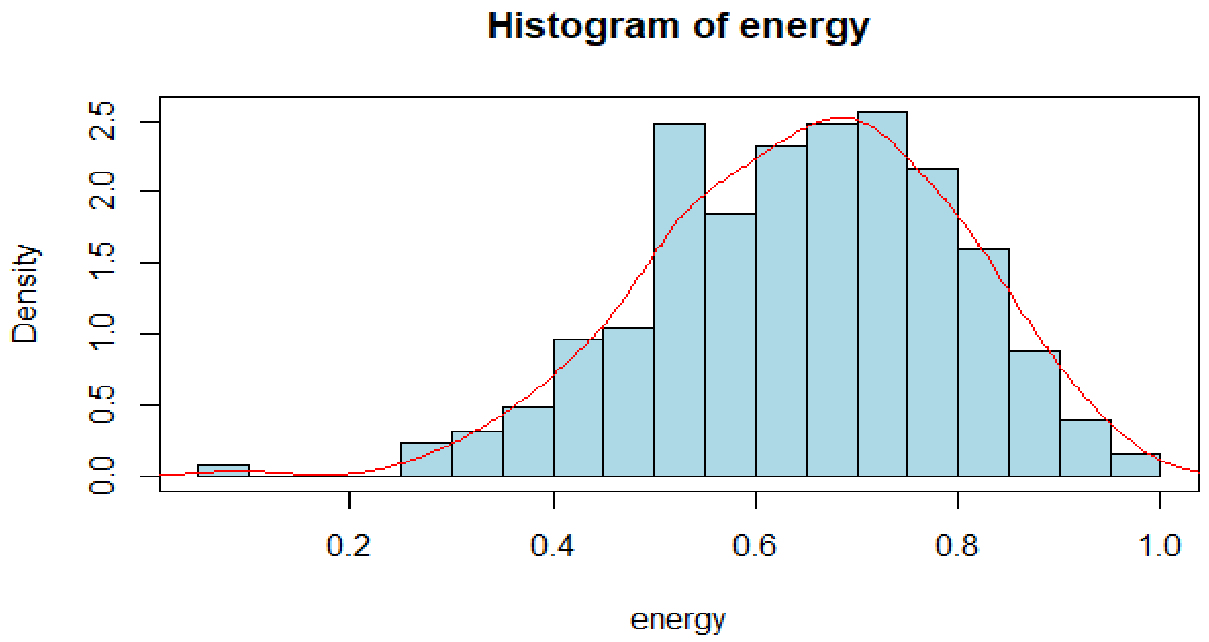
```
> fit.LOGITNO.2$aic
[1] -300.674
> fit.LOGITNO.2$sbc
[1] -283.0667
```



6.energy

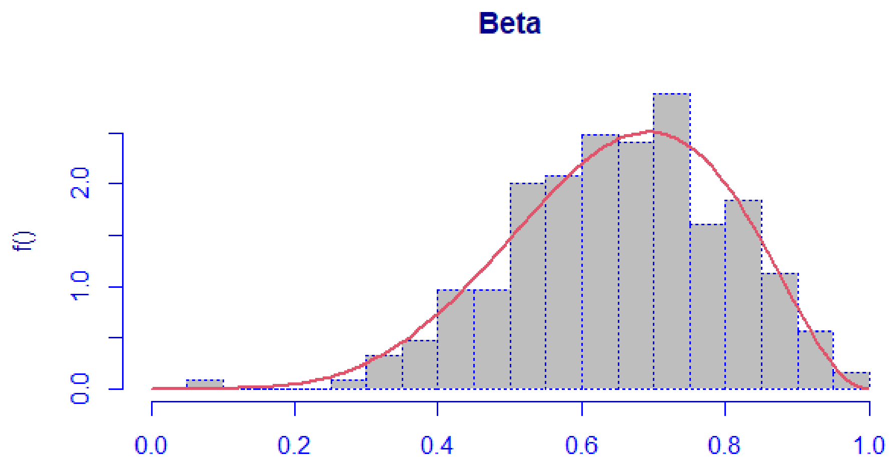
```
> round(summary(energy),2)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.09    0.55    0.66    0.65    0.76    0.97
> sd(energy)
[1] 0.1483927
> skewness(energy)
[1] -0.359721
> kurtosis(energy)
[1] 3.083078
```

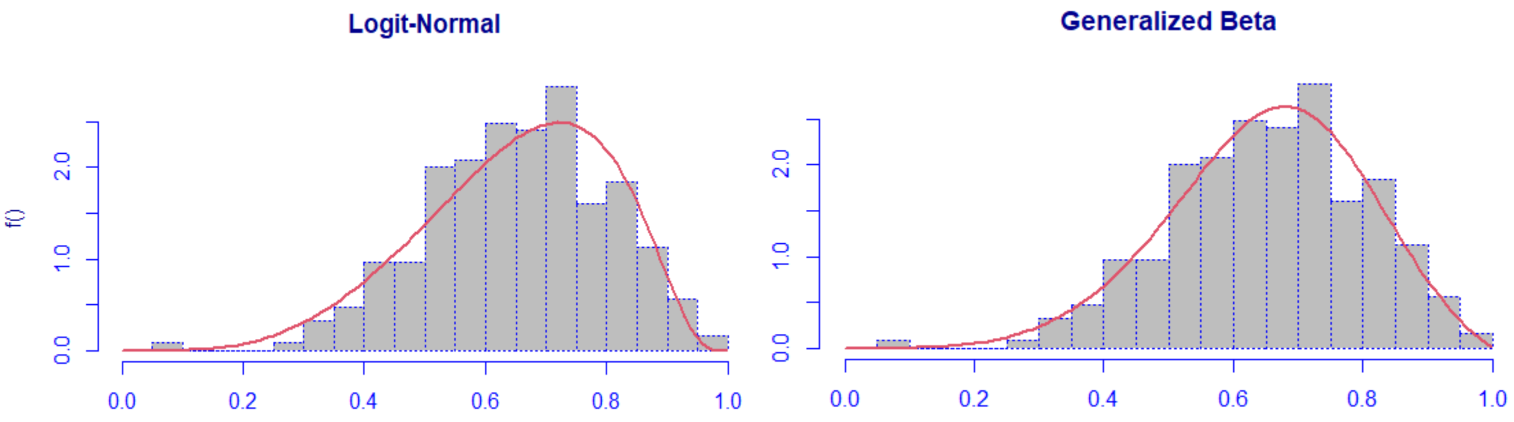
This variable is like “danceability”, in fact it presents a tail on the left and the kurtosis index indicat e that is present some asymmetry. All these conclusions are synthetized on the plot belove:



Watching at the form of the distribution and on its support, I have used the follow distribution:

```
> #FITTING MODEL
> fit.BE <- histDist(energy, family=BE, nbins=15, main="Beta")
> fit.LOGITNO <- histDist(energy, family=LOGITNO, nbins = 15, main="Logit-Normal")
> fit.GB1 <- histDist(energy, family=GB1, nbins = 15, main="Generalized Beta")
```

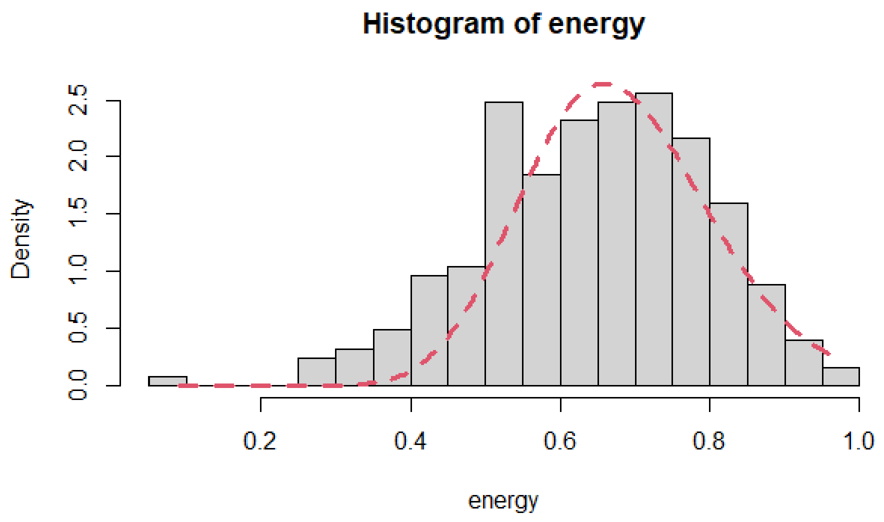




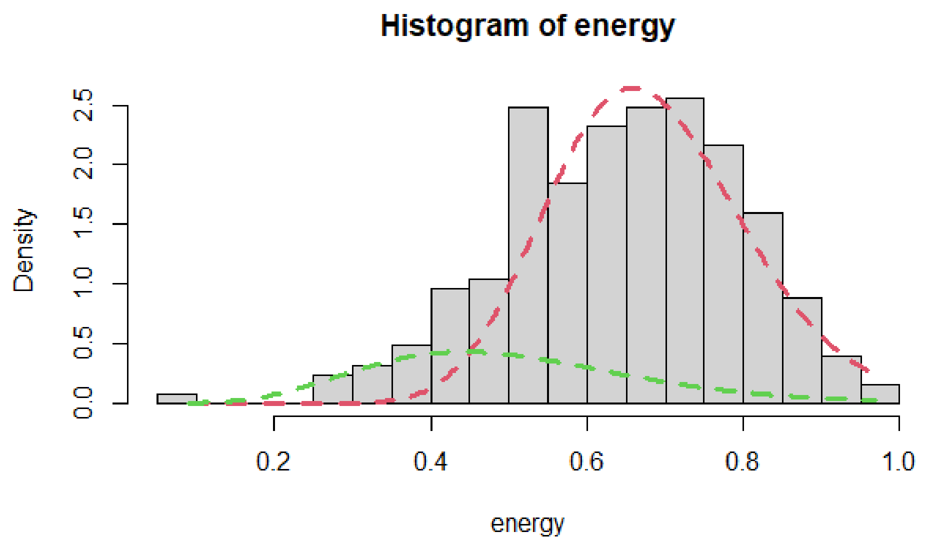
```
> data.frame(row.names = c("Beta", "Gen Beta", "logit Norm"), AIC=c(AIC(fit.BE), A
IC(fit.GB1), AIC(fit.LOGITNO)), SBC=c(fit.BE$sbc, fit.GB1$sbc, fit.LOGITNO$sbc))
                 AIC       SBC
Beta       -245.7739 -238.7310
Gen Beta   -244.1881 -230.1023
logit Norm -238.6389 -231.5960
```

as we see the beta distribution is again the better to explain our data. In facts, the beta distribution is "famous" thanks to its ability to adapt to many observed data distributions. We could try to use a gamma distribution (knowing that the support is not respected). Then we will evaluate the results of this analysis.

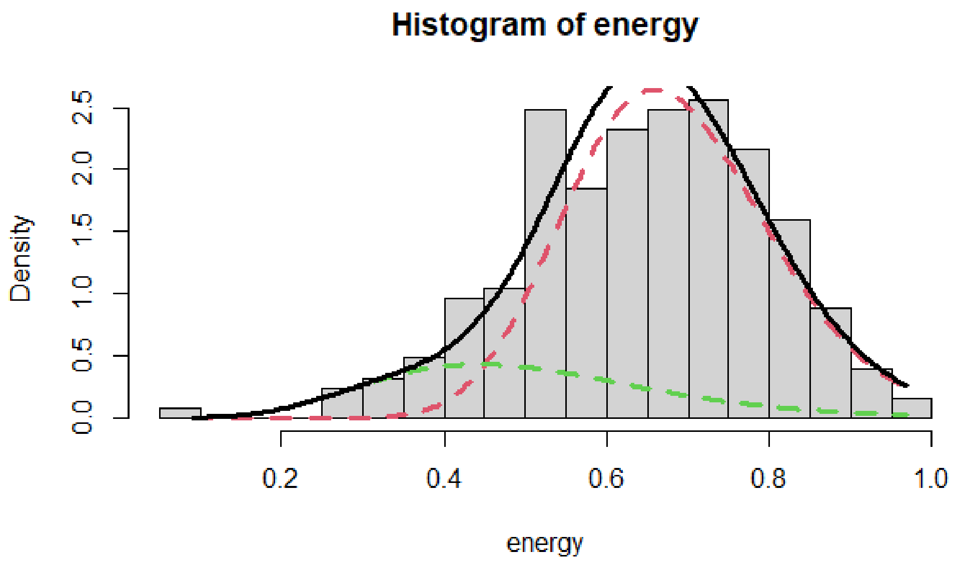
```
> #MIXTURE MODEL (GA WITH K=2)
> fit.GA.2 <- gamlssMX(formula = energy~1, family = GA, K = 2, data = NULL)
> # estimate of mu and sigma in group 1
> mu.hat1 <- exp(fit.GA.2[["models"]][[1]][["mu.coefficients"]])
> sigma.hat1 <- exp(fit.GA.2[["models"]][[1]][["sigma.coefficients"]])
> # estimate of mu and sigma in group 2
> mu.hat2 <- exp(fit.GA.2[["models"]][[2]][["mu.coefficients"]])
> sigma.hat2 <- exp(fit.GA.2[["models"]][[2]][["sigma.coefficients"]])
> hist(energy, breaks = 20,freq = FALSE)
>  lines(seq(min(energy),max(energy),length=length(energy)),fit.GA.2[["prob"]][1]*d
GA(seq(min(energy),max(energy),length=length(energy)), mu = mu.hat1, sigma = sigma
.hat1),lty=2,lwd=3,col=2)
```



```
>lines(seq(min(energy),max(energy),length=length(energy)),fit.GA.2[["prob"]][2]*dG
A(seq(min(energy),max(energy),length=length(energy)),   mu   =   mu.hat2,   sigma   =
sigma.hat2),lty=2,lwd=3,col=3)
```



```
> lines(seq(min(energy),max(energy),length=length(energy)),
+        fit.GA.2[["prob"]][1]*dGA(seq(min(energy),max(energy),length=length(energy
)), mu = mu.hat1, sigma = sigma.hat1) +
+          fit.GA.2[["prob"]][2]*dGA(seq(min(energy),max(energy),length=length(ener
gy)), mu = mu.hat2, sigma = sigma.hat2),
                        +       lty = 1, lwd = 3, col = 1)
```



```
> fit.GA.2$aic
[1] -228.8938
> fit.GA.2$sbc
[1] -211.2865
```

As we can see from the value of BIC and AIC the mixture model doesn't fit well our data, so I confirm the coiche of use the Beta Distribution to fits our data.

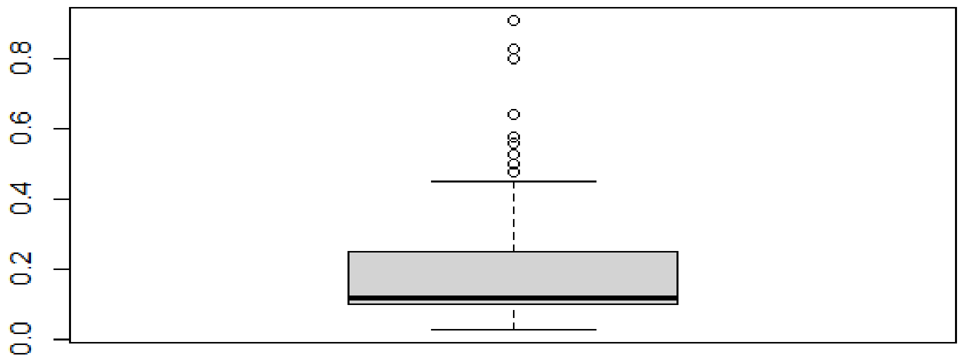
7.liveness

This variable indicates the percentage of presence of live performed elements in the song.

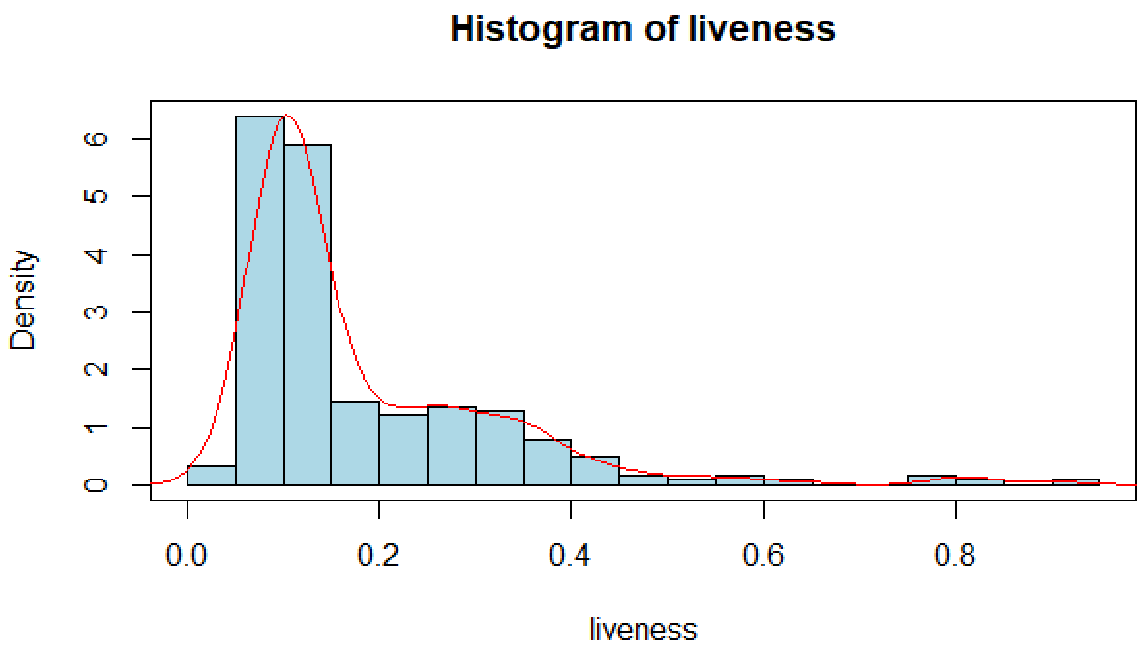
```
> round(summary(liveness),2)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   0.03    0.10    0.12    0.18    0.24    0.91 
> sd(liveness)
[1] 0.1402481
> skewness(liveness)
[1] 2.278155
> kurtosis(liveness)
[1] 9.619432
```

The positive skewness indicates the presence of a right-tail in this distribution and the high kurtosis represent the asymmetry that characterize the distribution. First, we plot the boxplot, then we look at the histogram, as usual.

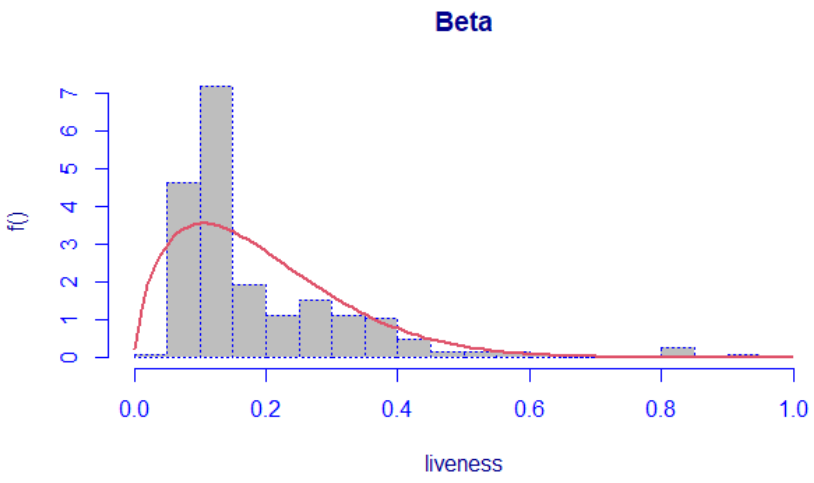
```
> boxplot(liveness)
```



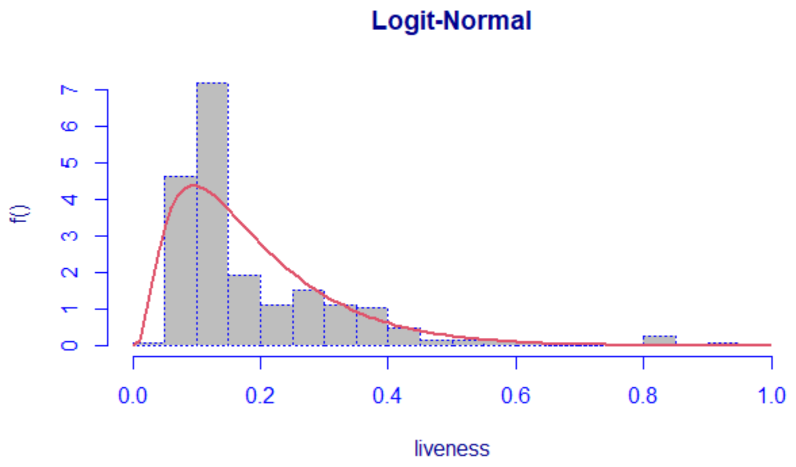
```
> hist(liveness,breaks = 20, col="lightblue",freq = FALSE, main = "Histogram of li
veness")
> box()
> lines(density(liveness), col="red")
```



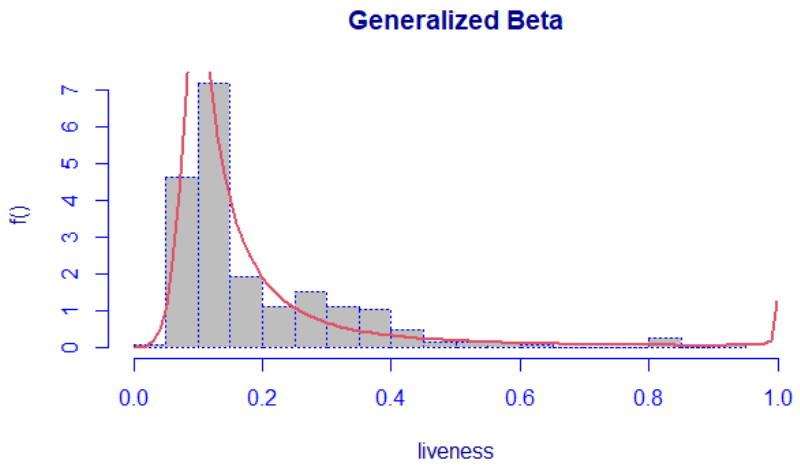
```
> #FITTING MODEL
> fit.BE <- histDist(liveness, family=BE, nbins=20, main="Beta")
```



```
> fit.LOGITNO <- histDist(liveness, family=LOGITNO, nbins = 20, main="Logit-Normal
")
```



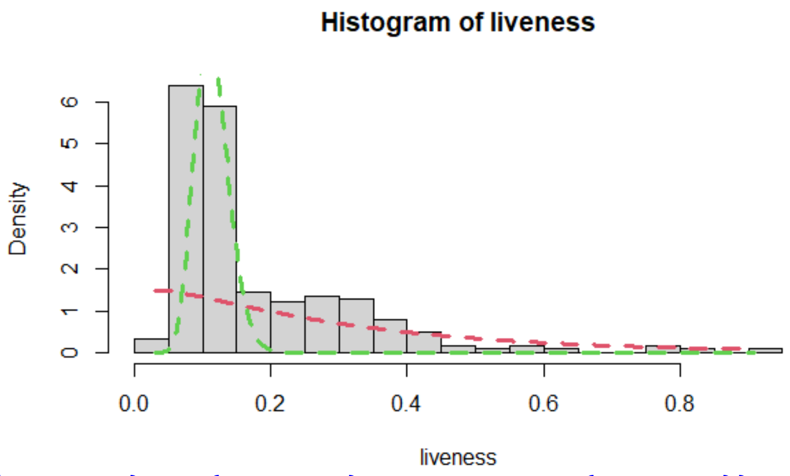
```
> fit.GB1 <- histDist(liveness, family=GB1, nbins = 20, main="Generalized Beta")
```



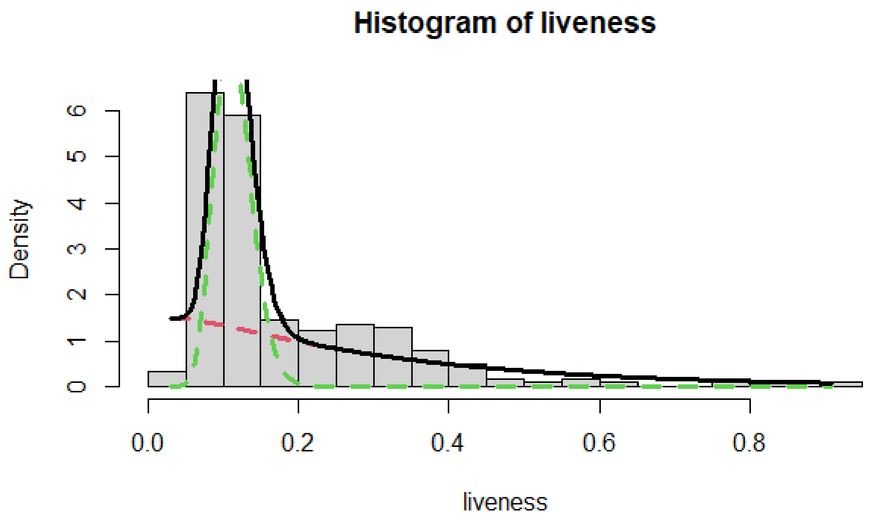
```
> data.frame(row.names = c("Beta", "Gen Beta", "logit Norm"), AIC=c(AIC(fit.BE), A
IC(fit.GB1), AIC(fit.LOGITNO)), SBC=c(fit.BE$sbc, fit.GB1$sbc, fit.LOGITNO$sbc))
                 AIC       SBC
Beta       -390.7374 -383.6944
Gen Beta   -523.1259 -509.0401
logit Norm -443.7046 -436.6617
```

the best model is the Generalized Beta, but we could try to fit our data with a Logit-Normal mixture model.

```
> #MIXTURE MODEL (GA WITH K=2)
> fit.LOGITNO.2 <- gamlssMX(formula = liveness~1, family = LOGITNO, K = 2, data =
NULL)
> # estimate of mu and sigma in group 1
> mu.hat1 <- exp(fit. LOGITNO.2[["models"]][[1]][["mu.coefficients"]])
> sigma.hat1 <- exp(fit.LOGITNO.2[["models"]][[1]][["sigma.coefficients"]])
> # estimate of mu and sigma in group 2
> mu.hat2 <- exp(fit.LOGITNO.2[["models"]][[2]][["mu.coefficients"]])
> sigma.hat2 <- exp(fit.LOGITNO.2[["models"]][[2]][["sigma.coefficients"]])
> hist(liveness, breaks = 20,freq = FALSE)
> lines(seq(min(liveness),max(liveness),length=length(liveness)),fit.LOGITNO.2[["p
rob"]][1]*dGA(seq(min(liveness),max(liveness),length=length(liveness)), mu = mu.ha
t1, sigma = sigma.hat1),lty=2,lwd=3,col=2)
> lines(seq(min(liveness),max(liveness),length=length(liveness)),fit.LOGITNO.2[["p
rob"]][2]*dGA(seq(min(liveness),max(liveness),length=length(liveness)), mu = mu.ha
t2, sigma = sigma.hat2),lty=2,lwd=3,col=3)
```



```
> lines(seq(min(liveness),max(liveness),length=length(liveness)),
+       fit.LOGITNO.2[["prob"]][1]*dGA(seq(min(liveness),max(liveness),length=leng
th(liveness)), mu = mu.hat1, sigma = sigma.hat1) +
+         fit.LOGITNO.2[["prob"]][2]*dGA(seq(min(liveness),max(liveness),length=le
ngth(liveness)), mu = mu.hat2, sigma = sigma.hat2),
+       lty = 1, lwd = 3, col = 1)
```



```
> fit.GA.2$aic
[1] -542.7351
> fit.GA.2$sbc
[1] -525.1278
```

As we can see from the plot and from the indexes, a mixture of two Logit-Normal models is the best fit for the variable “Liveness”.

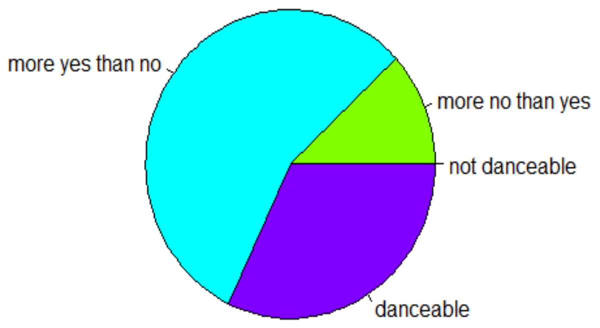
bonus

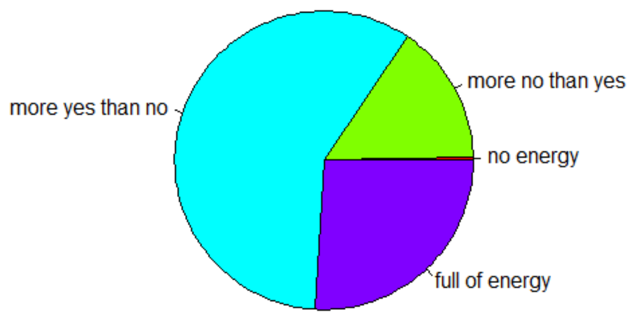
To have a better visualization about this last three variables, I decided to divide them into four (1/4 or 100/4) classes, which represent the “quantity” of the variable present in the song.

```
> danceability_fact <- cut(danceability, breaks = c(0.001,0.25,0.50,0.75,1), label
s = c("not danceable","more no than yes","more yes than no","danceable"), include.
lowest = TRUE)
> energy_fact <- cut(energy, breaks = c(0.001,0.25,0.50,0.75,1), labels = c("no en
ergy","more no than yes","more yes than no","full of energy"), include.lowest = TR
UE)
> liveness_fact <- cut(liveness, breaks = c(0.001,0.25,0.50,0.75,1), labels = c("n
o live element","more no than yes","more yes than no","live"), include.lowest = TR
UE)
> df <- cbind(df, dance_f = danceability_fact, energy_f = energy_fact, live_f = li
veness_fact)
```

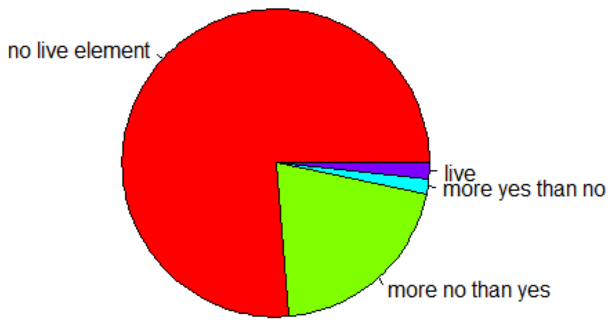
So, I created a “transitory” variable (%_fact), with names for every class and then I’ve added they on the dataset df. Now it’s possible to treat them as a factor, so we can put this data in a pie chart, for example.

```
> table(dance_f)
dance_f
   not danceable more no than yes more yes than no        danceable
               0               30              140               80
> table(energy_f)
energy_f
       no energy more no than yes more yes than no   full of energy
               1               38              146               65
> table(live_f)
live_f
 no live element more no than yes more yes than no             live
             191               51                4                4
>pie(table(dance_f), main = "Danceability", col = rainbow(length(table(dance_f))))
>pie(table(energy_f), main = "Energy", col = rainbow(length(table(energy_f))))
 >pie(table(live_f), main = "Live Elements", col = rainbow(length(table(live_f))))
```





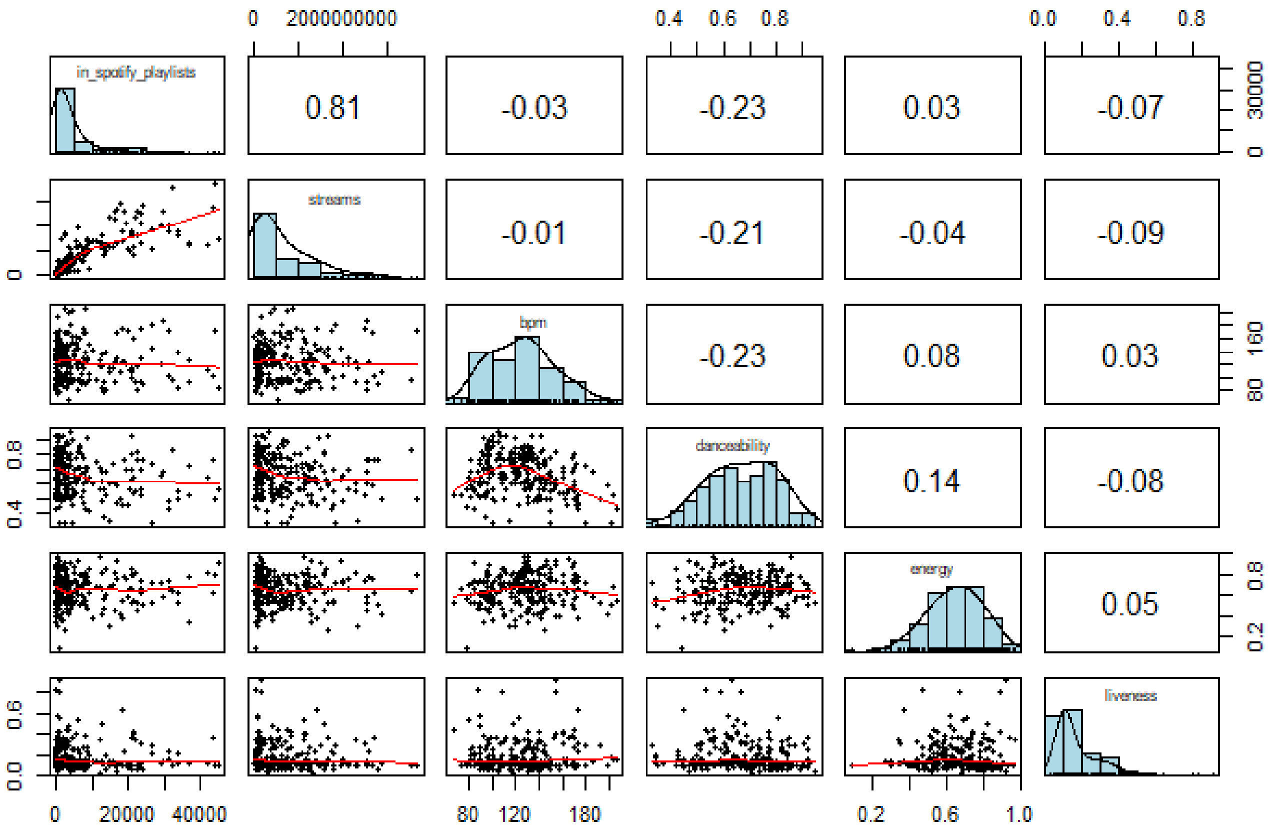
1) DANCEABILITY
2) ENERGY
3) LIVENESS



Multivariate Analysis

To start the multivariate analysis, it's necessary to define and evaluate the correlation between the variable. It is possible in two ways: or calculating the correlation matrix, or with a function that create a corrgram, with the synthesis of the most important information that we need.

```
> df_cor <- df[,c(4,5,6,8,9,10)]
> plot <- pairs.panels(df_cor, hist.col="lightblue", density = TRUE, ellipses = FA
LSE)
```



This is a preliminary analysis of the data in the original space, which aim is to understand if it would be useful to run a Principal Component Analysis and a Cluster one on this dataset.

In the upper triangle of the matrix there are the coefficients of correlation, which are used to understand if PCA is useful or not, while in the lower triangle there are the scatterplots of data and on the main diagonal there is the non-parametric density distribution of the data, both used to understand if CA could be useful.

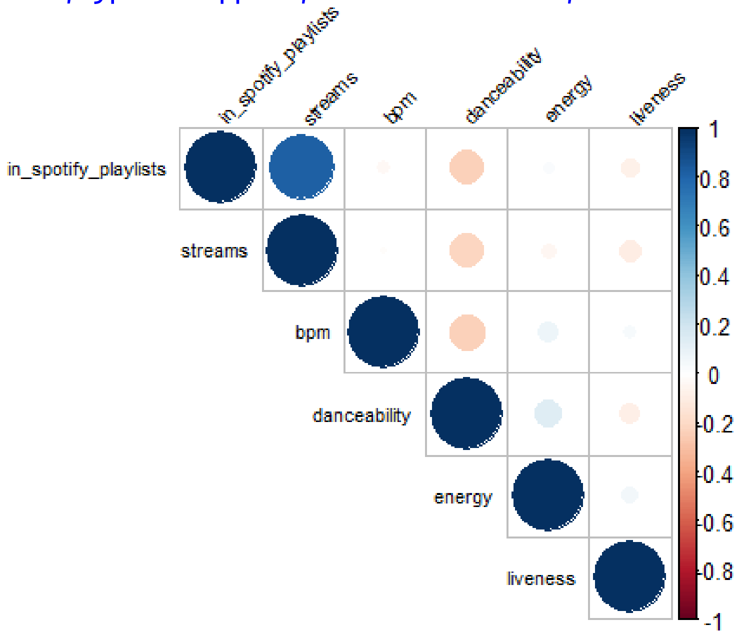
if we look at the plot, we can see that there is a high correlation *in_spotify_playlists* and *streams*, which is 0.81, while there is no strong correlation between other variables. This means that a PCA in this dataset could be not useful, but we execute in anyways.

As regard to the diagonal panels, they are used to understand if the single variable is useful for clustering: if the density line is bimodal, the relative variable could be useful for clustering, otherwise not. In this case, each variable separately considered is not useful to see clusters, but, if we look at the pairwise of variables, it is useful. In fact, from the scatterplots of the data in the lower triangle we can see that there are some clusters, but the situation is not even so clear. So, we need to investigate and try our possibility.

The pairs of scatter plots from where we see this are the ones related to the variables *energy and danceability, energy and bpm, danceability and bpm, in_spotify_playlists and streams.*

The same conclusions are visible in another type of matrix, findable in the package “corrplot”:

```
> cor_matrix <- cor(df_cor)
> corrplot(cor_matrix,type = "upper",tl.col = "black", tl.srt = 45,tl.cex =0.7)
```



Principal Component Analysis

PCA is based on the variance of the data, and if variables have very different scales, those with larger scales will have a greater influence on the total variance. By standardizing the data, you ensure that each variable contributes fairly to the overall variance. So, we now start to use the database “scaled_df” that contains the same variables as our “df”, but all on the same scale, with the operation of standardization.

```
> summary(scaled_df)
 in_spotify_playlists    streams             bpm              danceability
 Min.   :-0.7065      Min.   :-0.9079   Min.   :-2.11849   Min.   :-2.545735
 1st Qu.:-0.6341      1st Qu.:-0.7584   1st Qu.:-0.88790   1st Qu.:-0.806129
 Median :-0.4340      Median :-0.3906   Median :-0.01926   Median :-0.001796
 Mean   : 0.0000      Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.000000
 3rd Qu.: 0.1407      3rd Qu.: 0.5724   3rd Qu.: 0.63223   3rd Qu.: 0.821243
 Max.   : 4.0292      Max.   : 4.1293   Max.   : 2.91244   Max.   : 2.093213
     energy             liveness
 Min.   :-3.7757   Min.   :-1.0875
 1st Qu.:-0.6758   1st Qu.:-0.5884
 Median : 0.0655   Median :-0.4458
 Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 0.7394   3rd Qu.: 0.4455
 Max.   : 2.1546   Max.   : 5.1871
```

To find the principal components, we use the “eigen-decomposition”, that is applied to the covariance matrix of the standardized data. The eigenvalues indicate the variance for each of the five principal components and the eigenvectors matrix indicates the loadings for each Principal component, i.e. the weight of each original variable in each principal component. The eigenvalues are shown below:

```
> df_cov <- cov(scaled_df)
> df_eigen <- eigen(df_cov)
> (df_eigen$values)
[1] 1.9339223 1.2264274 1.0647215 0.9635568 0.6290598 0.1823122
```

The eigenvectors matrix of the first three principal component is:

```
> (phi <- df_eigen$vectors[,1:3])
            [,1]        [,2]        [,3]
[1,] -0.66247063  0.13988364 -0.14624384
[2,] -0.66340805  0.14007255 -0.08165793
[3,] -0.04239546 -0.67747505 -0.16773555
[4,]  0.32808055  0.58275858 -0.25419248
[5,]  0.06088532  0.02268107 -0.91230594
[6,]  0.08877596 -0.40214154 -0.21654921
> phi <- -phi
> row.names(phi) <- c("in_spotify_playlist", "streams", "bpm", "danceability","ene
rgy","liveness")
> colnames(phi) <- c("PC1", "PC2","PC3")
> phi
                            PC1         PC2        PC3
in_spotify_playlist  0.66247063 -0.13988364 0.14624384
streams              0.66340805 -0.14007255 0.08165793
bpm                  0.04239546  0.67747505 0.16773555
danceability        -0.32808055 -0.58275858 0.25419248
energy              -0.06088532 -0.02268107 0.91230594
liveness            -0.08877596  0.40214154 0.21654921
```

So, the numbers above represent the weight of every variable in each principal component. In order of importance (in absolute weight), watching the loadings we have:

- For the first vector, heavy and equal weights on **streams** (0.66) and **in_spotify_playlist** (0.66) and slightly less weight on **danceability** (0.32). much less weight is placed for the other variables. This suggest us that the PC1 is a measure of how much the songs are played.
- The second loadings vector place heavy weights on **bpm** (0.67), **danceability** (0.68) and **liveness** (0.40). so, it is maybe can be a measure of this variables.
- The third have a very strong weight on **energy** (0.91), and less weights on other variables.

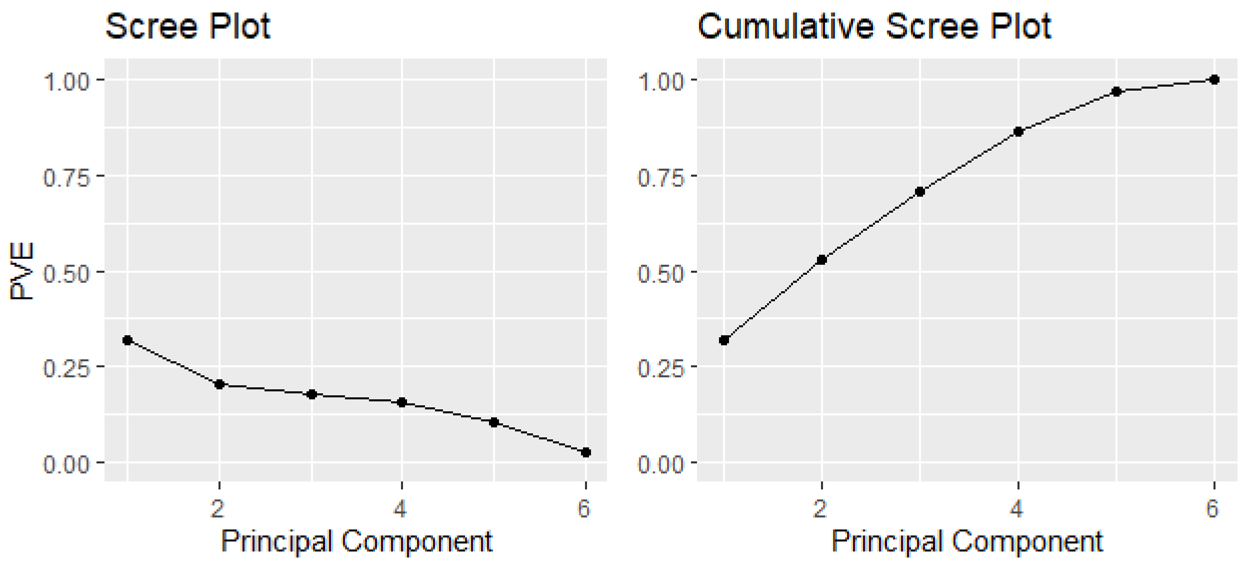
Now , we must decide the appropriate number **q** of principal components to use, basing our choice on criteria like the Kaiser's Rule, the scree plot and the cumulative proportion of variance explained.

The main rule for the <u>cumulative proportion of variance explained</u> is to extract the first q PCs which can explain a substantial proportion of variance, so retain as many PCs as needed to explain at least the 80% of the total variance.

```
> PVE <- df_eigen$values/sum(df_eigen$values)
> cumsum(round(PVE, 3))
[1] 0.322 0.526 0.703 0.864 0.969 0.999
```

As we can see from the results above (and how I already saw the beginning of the PCA), every PC does not explain so much variance. So, in this case, we must use minimum four PC to retain a good quantity of variability (86%). Already five components will not reduce so much the dimension of the variables. So, four PC are preferred. Let us see if this conclusion is the same with the scree plot. The scree plot suggests selecting q, on the x-axis, as the value to the left of the point where the curve becomes flat, namely to the left of the point at which the elbow occurs.

```
> # scree plot
> PVEplot <- qplot(c(1:6), PVE) +
+   geom_line() +
+   xlab("Principal Component") +
+   ylab("PVE") +
+   ggtitle("Scree Plot") +
+   ylim(0, 1)
> PVEplot
> # Cumulative PVE plot
> cumPVE <- qplot(c(1:6), cumsum(PVE)) +
+   geom_line() +
+   xlab("Principal Component") +
+   ylab(NULL) +
+   ggtitle("Cumulative Scree Plot") +
+   ylim(0,1)
> grid.arrange(PVEplot, cumPVE, ncol = 2)
> grid.arrange(PVEplot, cumPVE, ncol = 2)
```



As we can see from the Scree Plot, the elbow appears a little in q=4 and we can see better this in q=5.

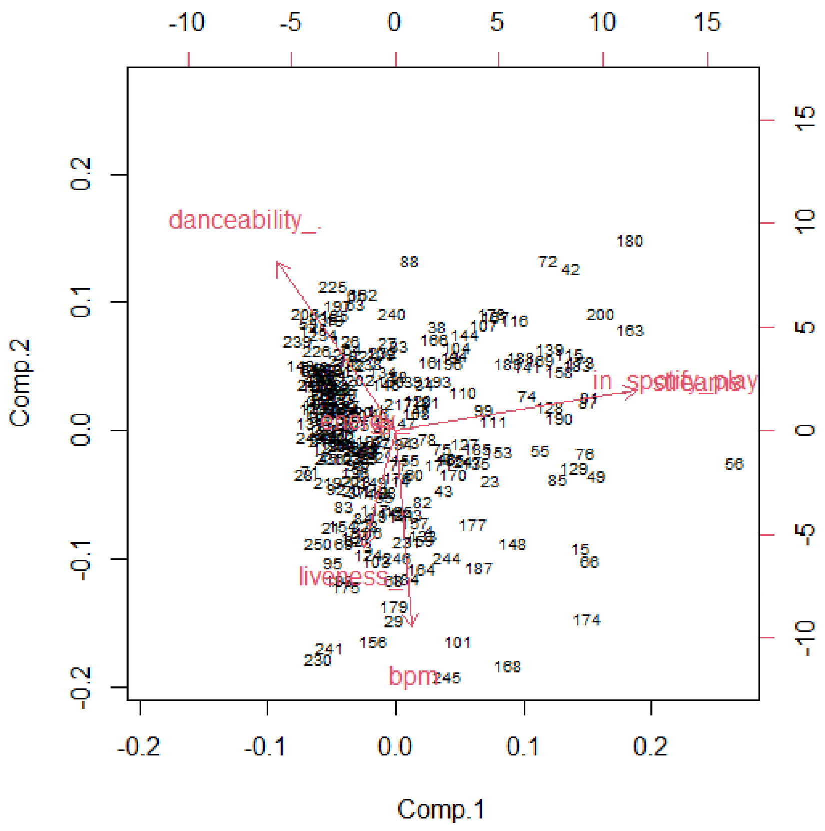
```
> head(round(phi[,1:4], digits = 2))
                       PC1   PC2  PC3   PC4
in_spotify_playlist   0.66 -0.14 0.15 -0.08
streams               0.66 -0.14 0.08 -0.05
bpm                   0.04  0.68 0.17  0.50
danceability         -0.33 -0.58 0.25  0.02
energy               -0.06 -0.02 0.91  0.12
liveness             -0.09  0.40 0.22 -0.85
```

Now, recalling the loading matrix, and deeming to take a level of correlation strictly greater than 0.40 in absolute value) as significant one, it is possible to interpret:

- The **PC1** as a measure the “famousness” of a song on spotify, related to how it is listened and in how many playlists it is.

- The **PC2** of the rhythm of a song, related to the *bpm* and to the presence of *live element*.
- The **PC3** the energy that a specific song communicate.
- The **PC4** explain the song that do not use live element in their arrangement.

```
biplot(princomp(scaled_df))
```



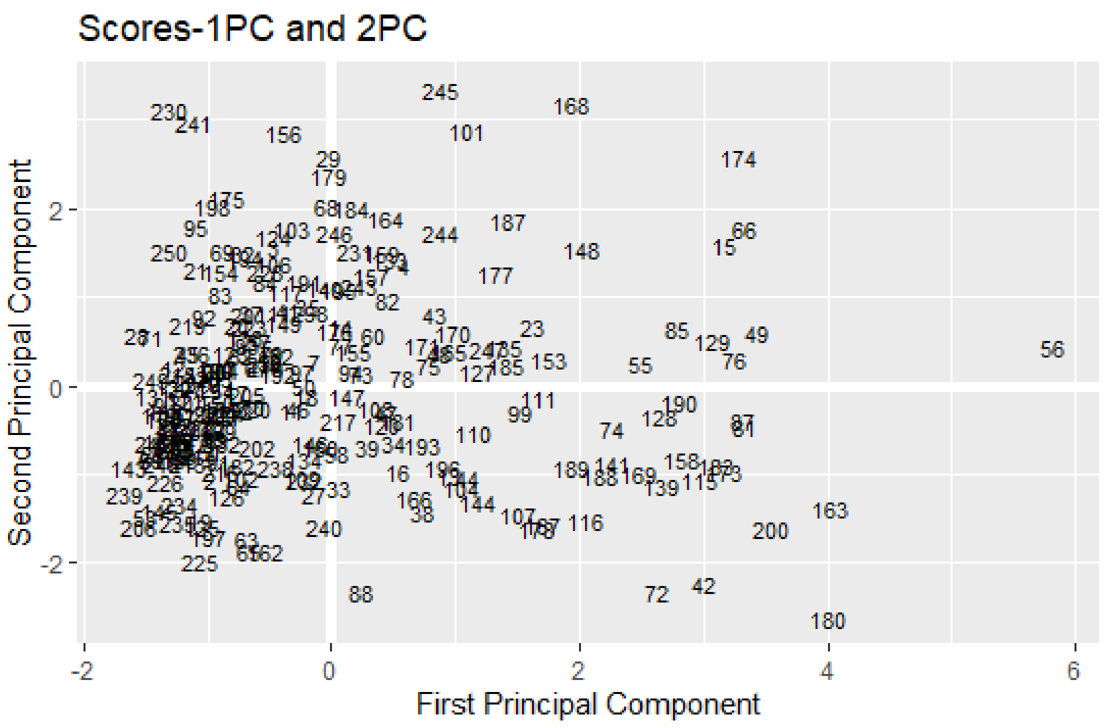
First, it is clearly noticeable how **playlists** and **streams** are two variables strongly and positively corr elated since the arrows representing them are basically overlapped, meaning that when the presen ce in playlists increases/decreases there is also an increase/decrease of the same entity in streams (and vice versa). Both are positively associated with PC1, as we can see that both the arrows are alm ost parallel to the bottom axis representing PC1, and orthogonal with respect to PC2. The two featu res **liveness** and **BPM** are more likely to be negatively associated with PC2. It is then observable ho w liveness, bpm and danceability are uncorrelated with streams, forming with it an angle very close to 90°. Finally, the **danceability** feature is positively associated with PC2. In conclusion the first prin cipal component can be viewed as a measure of **popularity** of a track's sound. On the other hand, t he second principal component primarily measures the danceability of a song, with negative correl ation with the bpm and the live element.

Now we can compute the principal component scores – which are the projections of the two hundr ed data points along the directions defined by loading vectors. With it is possible to make some con siderations about each specific song of the sample dataset, with reference to the original variables.

```
> # Create the data frame with Principal Component scores.
> PC <- data.frame(ID = row.names(df), PC1, PC2, PC3)
> head(PC)
  ID        PC1        PC2        PC3
1  1 -1.2344776 -0.7089080  1.0364702
2  2 -0.9824782 -1.0666786  0.1558803
3  3 -0.4674159  1.5674989 -0.9125397
4  4  0.5798288  1.3540096  0.3901562
```

```
5   5 -0.5174802  0.4327300  0.7861885
6   6 -1.2496272 -0.8337645 -0.1452530
```

```
> ggplot(PC, aes(PC1, PC2)) + modelr::geom_ref_line(h = 0) + modelr::geom_ref_line
(v = 0) + geom_text(aes(label = ID), size = 3) + xlab("First Principal Component")
+ ylab("Second Principal Component") + ggtitle("Scores-1PC and 2PC")
```



In the image above we can see the score-plot of each observation, related to the first two Principal component. For example:

- the song 56, that corresponds to "Blinding Lights – The Weeknd", is very famous, watching the high value of PC1 (in fact, it was streamed because it was streamed 3703895074 times and added to 43899 playlists). The near-zero value of PC2 suggests that is rhythm is on average, with BPM high and low liveness (or vice versa). In fact, the song has 171 BPM and very low liveness (only 0.09).
- the song 174, "Demons" by Imagine Dragon, in terms of PC1, seems to be a famous song with a high value of live elements and BPM.

As we see, there is a loss of information not negligible using two Principal Component and the price of using four PC to explain six variables seems to be so much, also considering that the PCA in this case does not explain well our data.

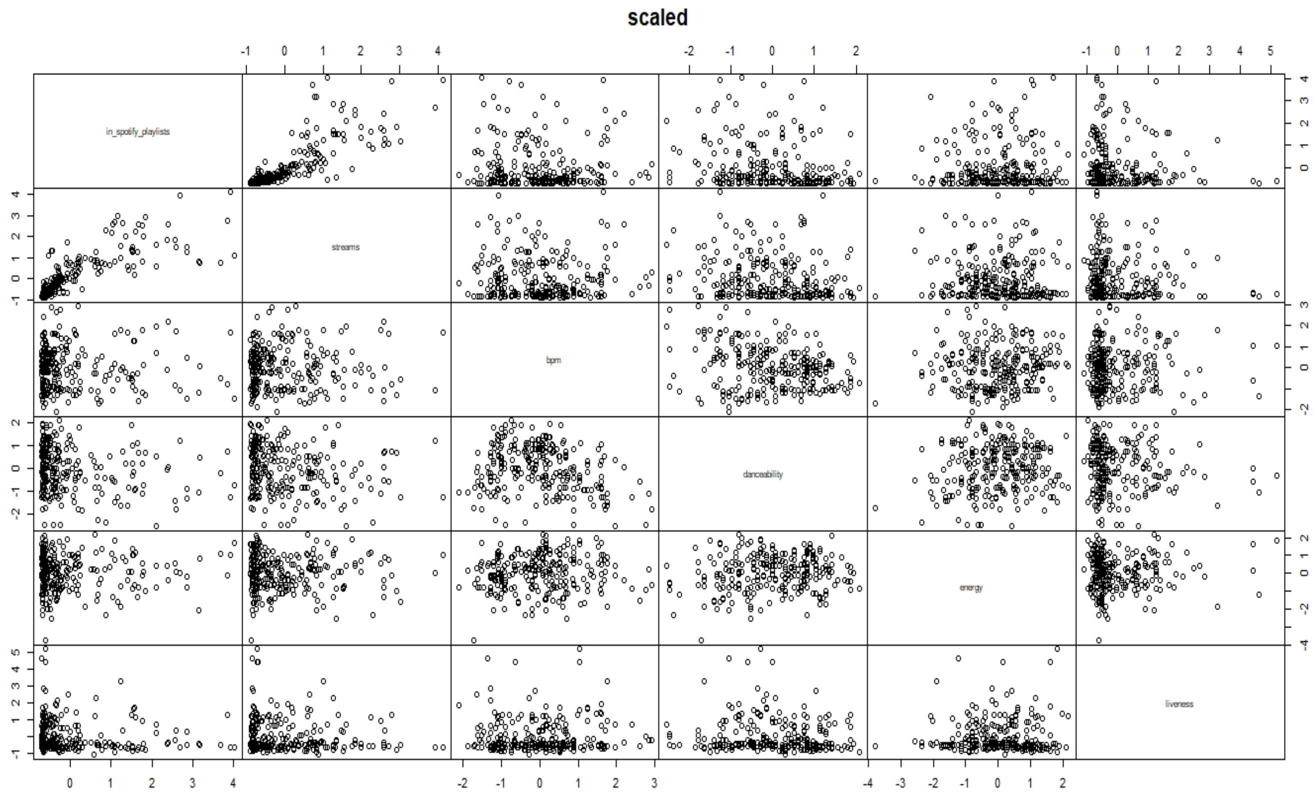
Cluster Analysis

The goal of Cluster Analysis (CA) is to organize a set of objects into groups, or clusters, in such a way that items in the same cluster are more similar to each other than they are to items in other clusters. As for PCA, we use for our analysis the scaled version of the dataset. Before applying any clustering method on our data, it is important to evaluate whether the dataset contains meaningful clusters or not.

1.Assessing of Clustering Tendency

There are different techniques to assess the cluster tendency, like Hopkins method. Let's visualize the standardized data to assess whether they contain meaningful clusters.

```
> pairs(scaled_df, pch = 21, gap = 0)
```

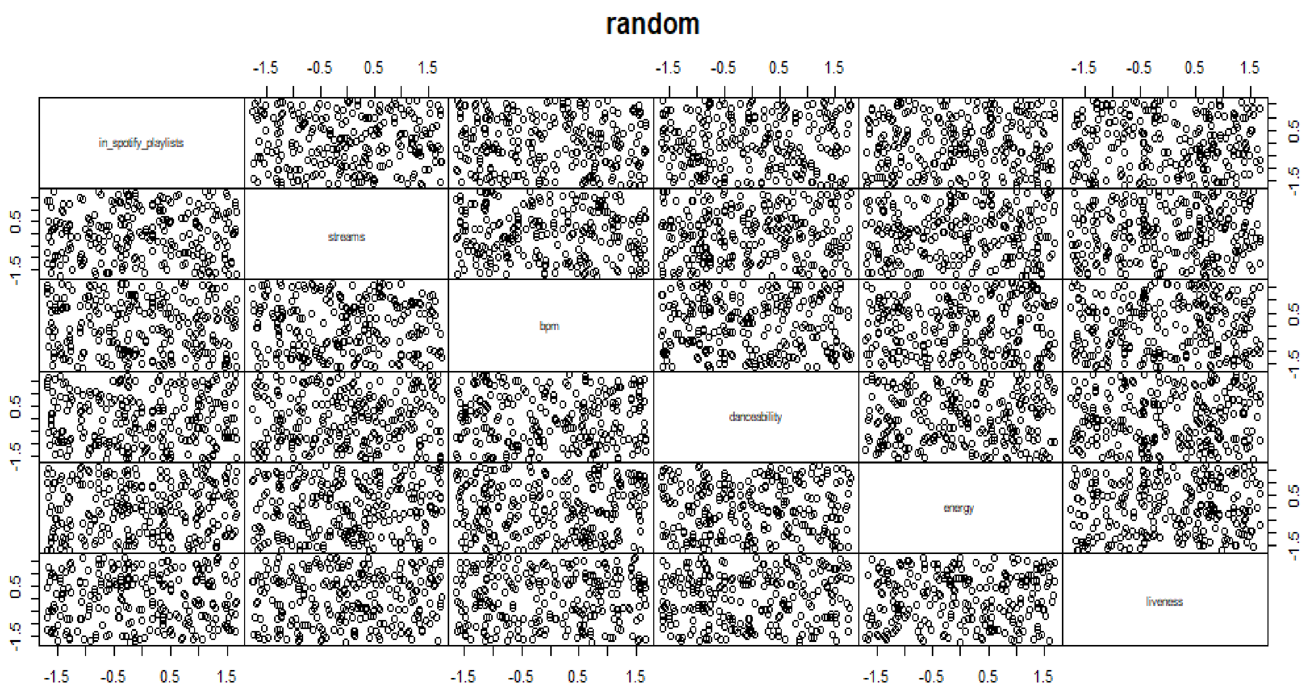


The pairwise plots compares each variable against every other variable; we can see that there isn't a "tendency" in the plot. To have a comparison, we can create a random dataset, generated from the original dataframe. Then we standardize it. We can see the scatterplots matrix uncorrelated.

```
> random_df = apply(df[c(4,5,6,8,9,10)], 2, function(x){runif(length(x), min(x), (
max(x)))})
> random_df = as.data.frame(random_df)
> scaled_random_df = scale(random_df)
```

In this way, with apply(), I've created a random_df, with the variables, dimension and max and min of the original dataframe. Then I plotted this matrix. In our case, all the points are concentrated on the bottom-left part.

```
> pairs(scaled_random_df, pch = 21, gap = 0, main = "random")
```



As the result are not so clear, we can try the Hopkins Statistic:

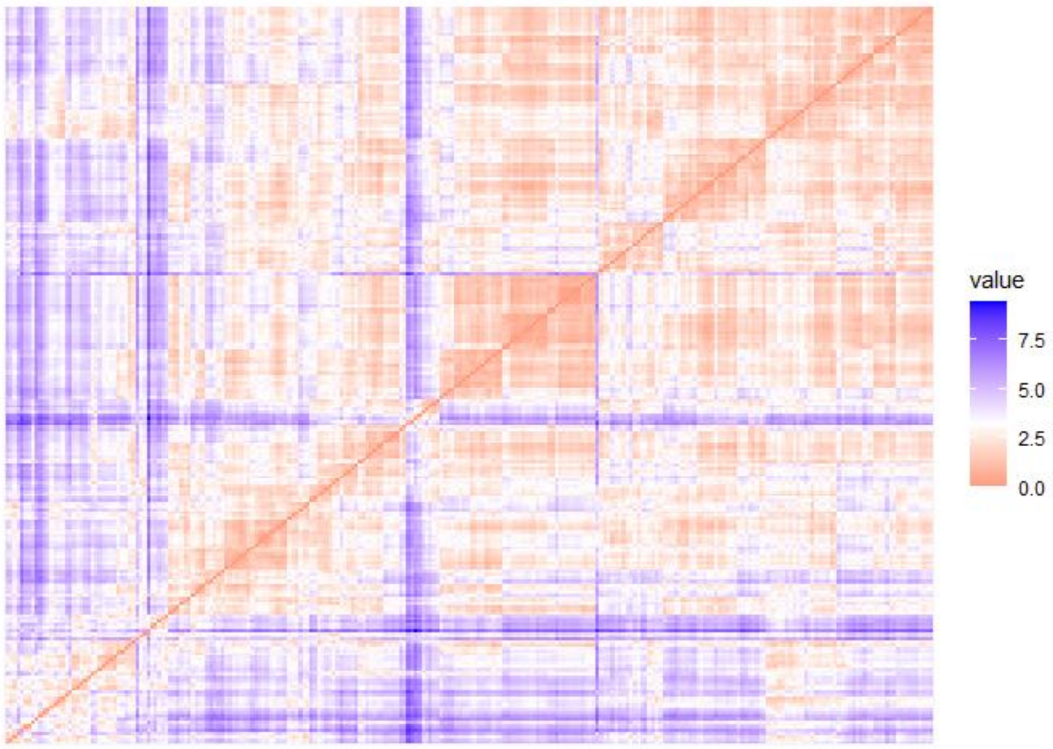
```
> hopkins(scaled_df)
$H
[1] 0.2617802
> hopkins(scaled_random_df)
$H
[1] 0.4949213
```

The Hopkins statistic is a method to assess cluster tendency. It measures the tendency of a dataset to form clusters by comparing the distances between randomly selected points and the distances b etween data points in the dataset. A higher Hopkins statistic value suggests a higher tendency to cl uster. As saw on R: Calculated values 0-0.3 indicate regularly spaced data. Values around 0.5 indicat e random data. Values 0.7-1 indicate clustered data. So, the Hopkins defines this dataset regularly s paced. Another algorithm that can confirm the cluster tendency is the visual assessment of cluster t endency (VAT). It computes the distance matrix, which is a measure of dissimilarities between units , and reorder it so that the similar units are closer to each other. I've calculated it with the Euclidian distance and with the Manhattan.

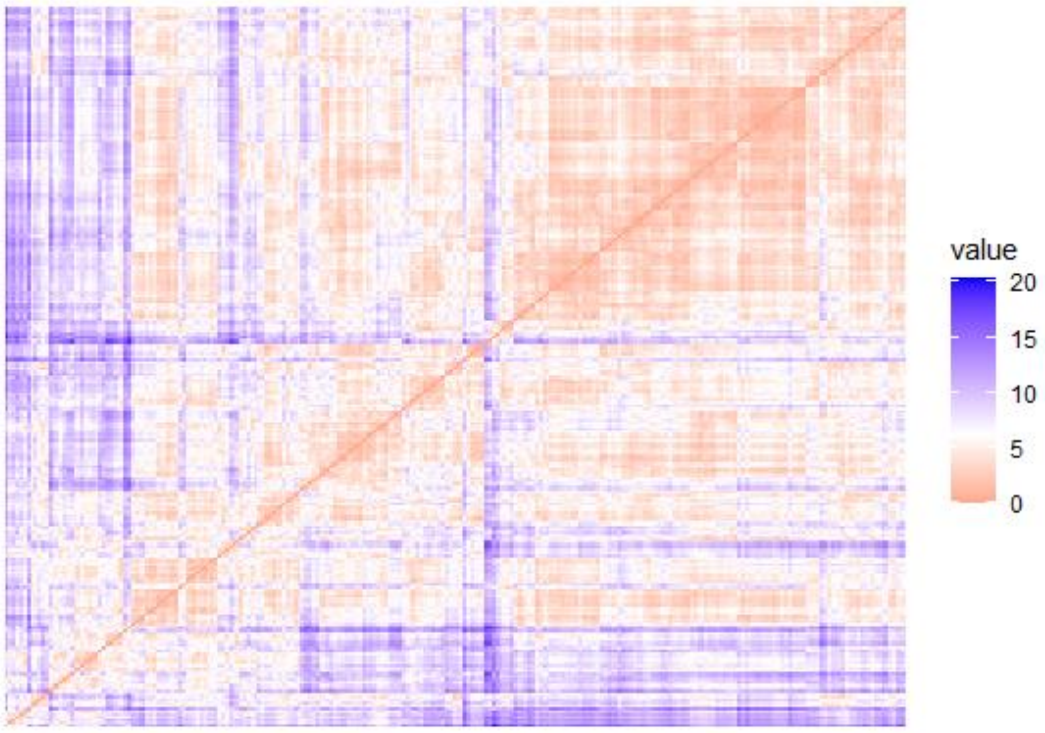
```
> dist.eucl <- dist(scaled_df, method = "euclidean")

> round(as.matrix(dist.eucl)[1:5, 1:5], 2)

     1    2    3    4    5
1 0.00 1.51 3.42 2.85 1.39
2 1.51 0.00 3.04 3.27 2.00
3 3.42 3.04 0.00 2.53 2.56
4 2.85 3.27 2.53 0.00 1.56
5 1.39 2.00 2.56 1.56 0.00

> dist.man <- dist(scaled_df, method = "manhattan")
> round(as.matrix(dist.man)[1:3, 1:3], 2)
```

```
  1    2    3
1 0.00 2.72 6.39
2 2.72 0.00 6.09
3 6.39 6.09 0.00
```

```
> fviz_dist(dist.eucl)
```



```
> fviz_dist(dist.man, show_labels=F)
```



The red color denotes low dissimilarity between units, the blue one high dissimilarity. The clustering tendency in the standardized dataset's dissimilarity is not so high. While we are not sure about the presence of a clustering structure, the analysis continues.

2.Determining Optimal Numbers of Clusters

The optimal number of clusters is somehow subjective and depends on the clustering method used. There are two methods to determine the optimal number of clusters: Direct methods (Elbow and Average Silhouette) and Gap statistic.

The **Elbow method** 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.

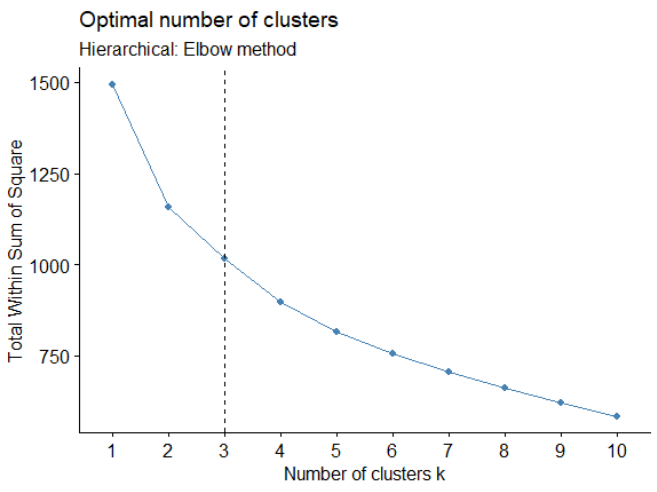
The **Average Silhouette method** roughly measures the quality of a clustering by determining how well each unit lies within its cluster. A high average silhouette width indicates a good clustering.

The **Gap Statistic** provides a statistical procedure to formalize the heuristic elbow method.

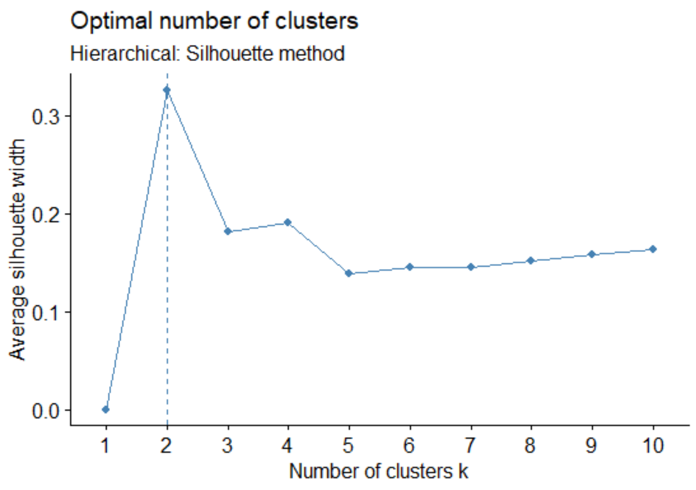
Now, we can run some clustering-algorithm and use the method above to evaluate the number of clusters.

2.1.For Hierarchical Clustering

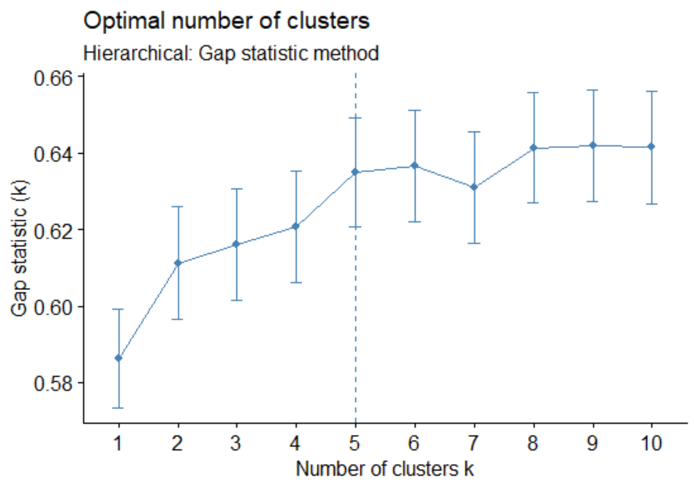
```
> fviz_nbclust(scaled_df, hcut, method = "wss") + geom_vline(xintercept = 3, linet
ype = 2) + labs(subtitle = "Hierarchical: Elbow method")
```



```
> fviz_nbclust(scaled_df, hcut, method = "silhouette") + labs(subtitle = "Hierarch
ical: Silhouette method")
```



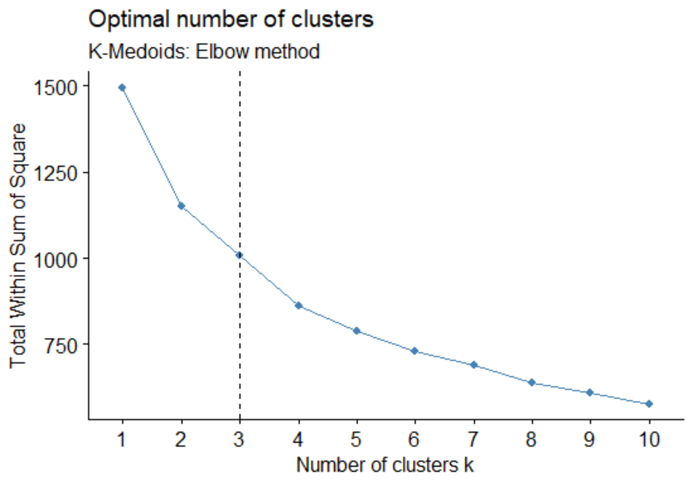
```
> fviz_nbclust(scaled_df, hcut, method = "gap_stat", nboot = 500) + labs(subtitle
= "Hierarchical: Gap statistic method")
```



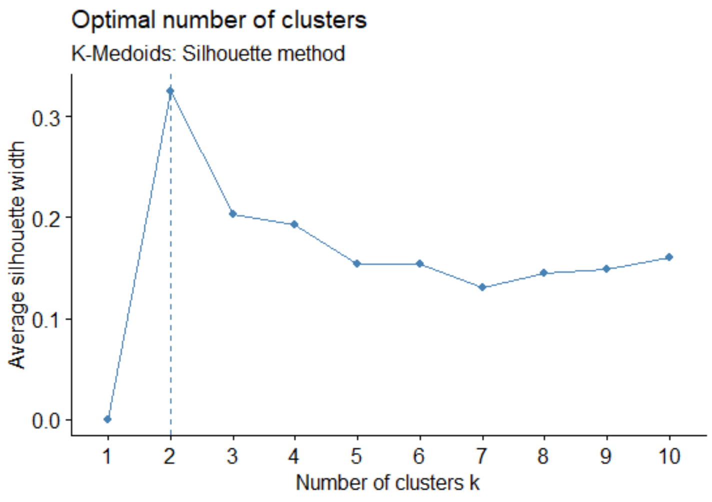
So, the suggested solutions are **2 clusters** for Elbow method, **3** for Silhouette method and **5** for Gap statistic method.

2.2.For K-Medoids

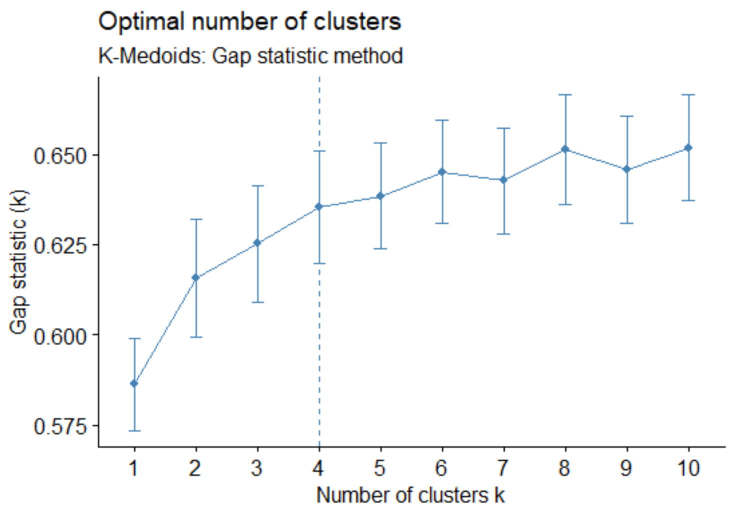
```
> fviz_nbclust(scaled_df, cluster::pam, method = "wss") + geom_vline(xintercept =
3, linetype = 2) + labs(subtitle = "K-Medoids: Elbow method")
```



```
> fviz_nbclust(scaled_df, cluster::pam, method = "silhouette") + labs(subtitle = "
K-Medoids: Silhouette method")
```



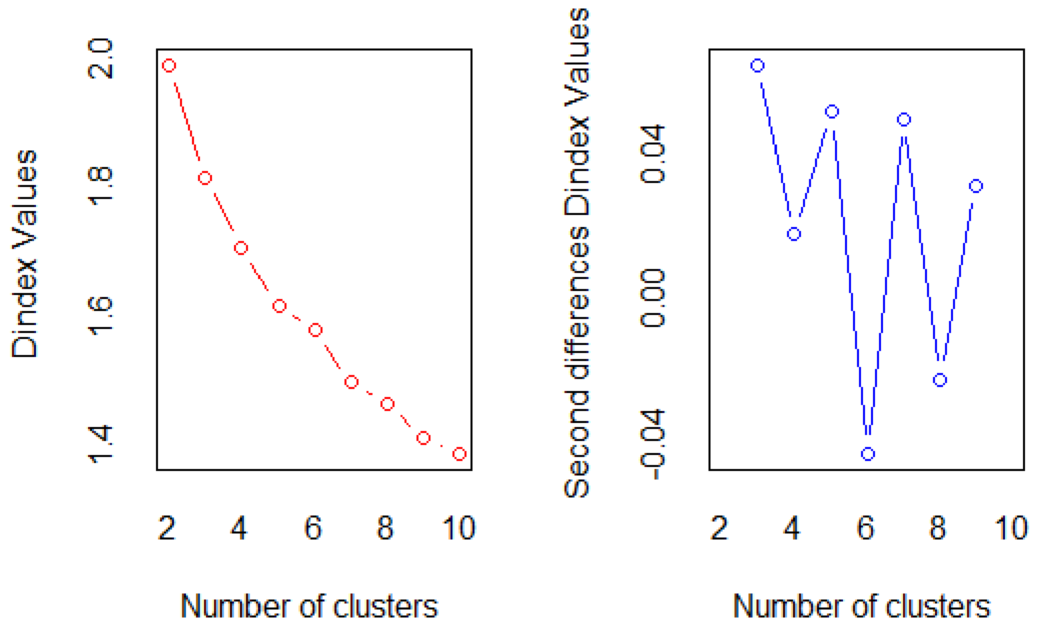
```
> fviz_nbclust(scaled_df, cluster::pam, method = "gap_stat", nboot = 500) + labs(s
ubtitle = "K-Medoids: Gap statistic method")
```



With k-medoids, the best number of clusters founded with the elbow method is **3**, with the silhouette method **2**, and for the Gap Statistic is **4**.

2.3.For K-Means

```
> fviz_nbclust(NbClust(scaled_df, distance = "euclidean", min.nc = 2, max.nc = 10,
method = "kmeans"))
*** : The Hubert index is a graphical method of determining the number of clusters
In the plot of Hubert index, we seek a significant knee that corresponds to a sign
ificant increase of the value of the measure i.e the significant peak in Hubert in
dex second differences plot.

*** : The D index is a graphical method of determining the number of clusters. In
the plot of D index, we seek a significant knee (the significant peak in Dindex se
cond differences plot) that corresponds to a significant increase of the value of
the measure.

*******************************************************************
* Among all indices:
* 9 proposed 2 as the best number of clusters
* 4 proposed 3 as the best number of clusters
* 1 proposed 4 as the best number of clusters
* 5 proposed 5 as the best number of clusters
* 1 proposed 6 as the best number of clusters
* 2 proposed 7 as the best number of clusters
* 2 proposed 10 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  2


*******************************************************************
```



In this way, I have tried to calculate the optimal number of clusters in the data. As we see from the result, it is not so clear to understand which is the best number of clusters, but as we have seen, the number 2 for clustering has chosen many times in each clustering algorithm. So, k = 2.

3.hierarchical Clustering

The Strategies for hierarchical clustering fall into two categories:

- **Agglomerative**: Each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- **Divisive**: All observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

To decide which clusters should be combined (for agglomerative), or where a cluster should be split (for divisive), a measure of dissimilarity between sets of observations is required. In most methods of hierarchical clustering, this is achieved by use of an appropriate distance d, between single observations of the data set, and a linkage criterion, which specifies the dissimilarity of sets as a function of the pairwise distances of observations in the sets.

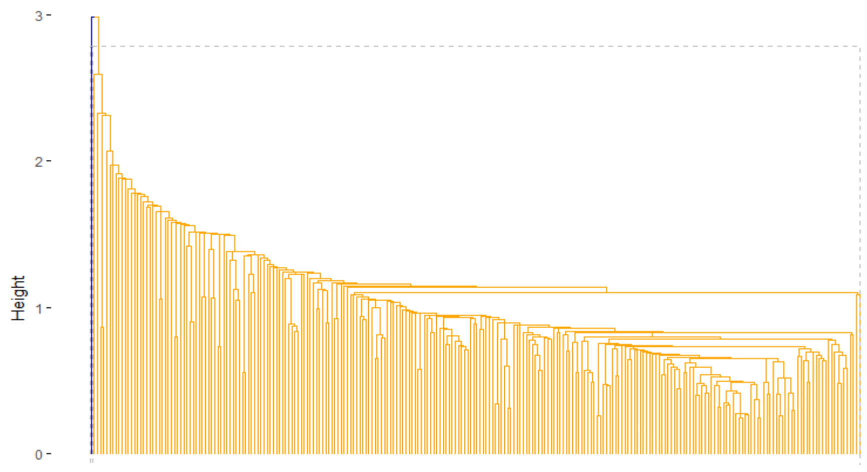
There are several distance measures and linkage method but, in this study, I decided to use ***Euclidian and Manhattan*** as distance measures. The linkage method that I will use are: ***Single Linkage, Average, Centroid and WARD's***.

```
> #""""AGGLOMERATIVE METHOD, dissimilarity matrix"""""""
> d.euclidean <- dist(scaled_df, method = "euclidean")
> d.manhattan <- dist(scaled_df, method = "manhattan")
```

3.1.Agglomerative

Single Linkage – Euclidian

```
> eu.single = hclust(d = d.euclidean, method = "single")
> fviz_dend(eu.single, k = 2, cex = 0.5, k_colors = c("#2E9FDF", "#FC4E07"), color
_labels_by_k = TRUE,rect = TRUE, show_labels = FALSE, main="Single linkage method
- Euclidean")
```

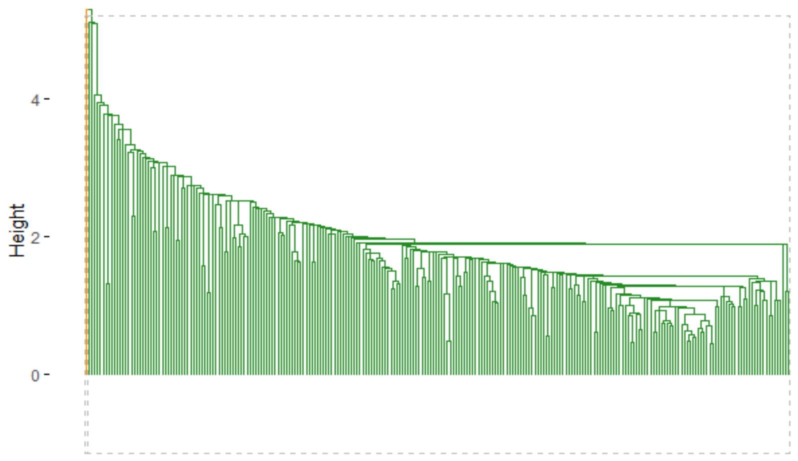


```
> cor(d.euclidean, cophenetic(eu.single))
[1] 0.7346091
> grp <- cutree(eu.single, k = 2 )
> table(grp)
grp
  1   2
248   2
```

The high correlation value between the original distances and the cophenetic distances indicates that the method preserves the original distances between units. But, with k = 2, it's found a cluster with only two observations. Also, with k=3, the single linkage method doesn't find any interesting cluster. We can try the same linkage method with another way to measure distances.

Single Linkage – Manhattan

```
> fviz_dend(man.single,k=2, cex = 0.5, k_colors = c("darkorange","forestgreen"), c
olor_labels_by_k = TRUE,rect = TRUE, show_labels = FALSE, main="Single linkage met
hod - Manhattan")
> cor(d.manhattan, cophenetic(man.single))
[1] 0.7346201
> grp <- cutree(man.single, k = 2 )
> head(grp, n = 6)
1 2 3 4 5 6
1 1 1 1 1 1
> table(grp)
grp
  1   2
248   2
```

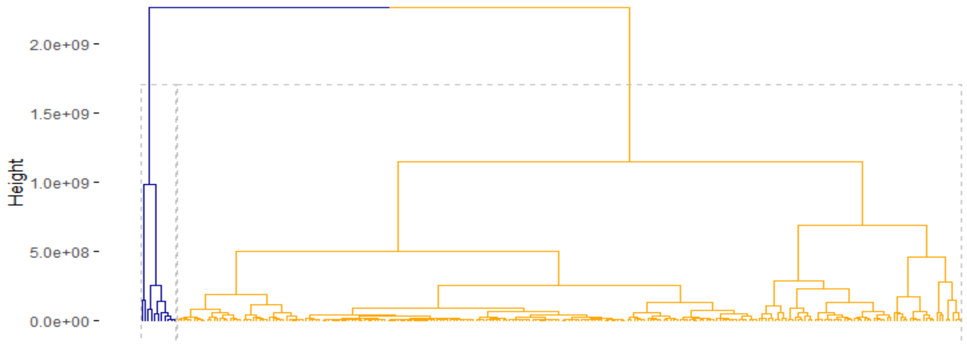


```
> cor(d.manhattan, cophenetic(man.single))
[1] 0.7450027
```

As first, there are not any interesting result.

Average Linkage – Euclidian

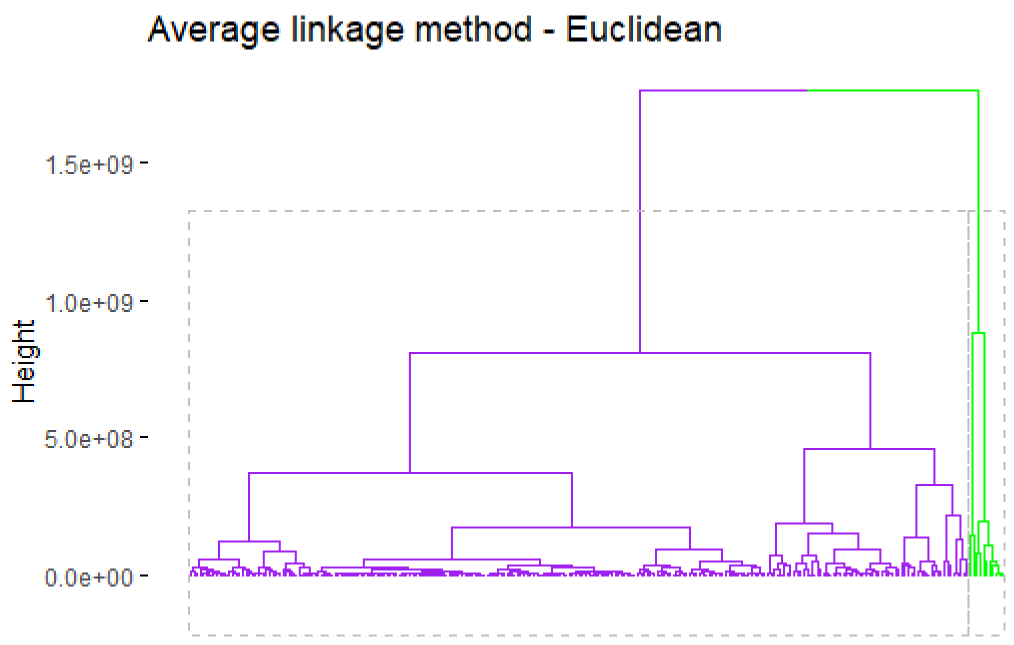
```
> eu.av <- hclust(d = d.euclidean, method = "average")

> fviz_dend(eu.av, k = 2, cex = 0.5, k_colors = c("darkblue", "orange"), color_lab
els_by_k = TRUE,rect = TRUE, show_labels = FALSE, main="Average linkage method - E
uclidean")
```



```
> cor(d.euclidean, cophenetic(eu.av))
[1] 0.7543888

> grp <- cutree(eu.av, k = 2 )

> table(grp)
grp
  1   2
239  11
```

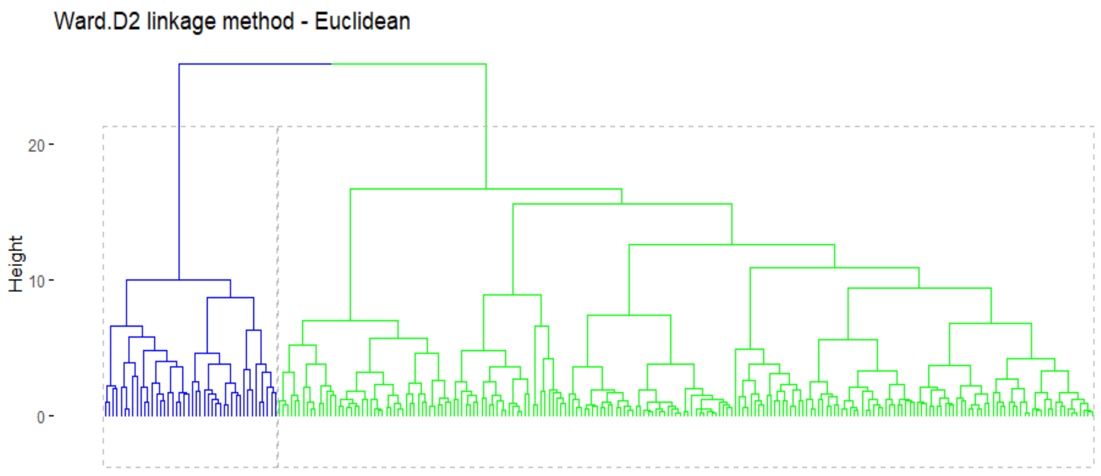
Centroids – Euclidian

```
> eu.cen <- hclust(d = d.euclidean, method = "centroid")

> fviz_dend(eu.cen, k = 2, cex = 0.5, k_colors = c("purple","green"), color_labels
_by_k = TRUE,rect = TRUE, show_labels = FALSE, main="Average linkage method - Eucl
idean")

> cor(d.euclidean, cophenetic(eu.cen))
[1] 0.7522038

> grp <- cutree(eu.cen, k = 2 )

> head(grp, n = 6)
1 2 3 4 5 6
1 1 1 1 1 1

> table(grp)
grp
  1   2
239  11
```



WARD.D2 – Euclidean

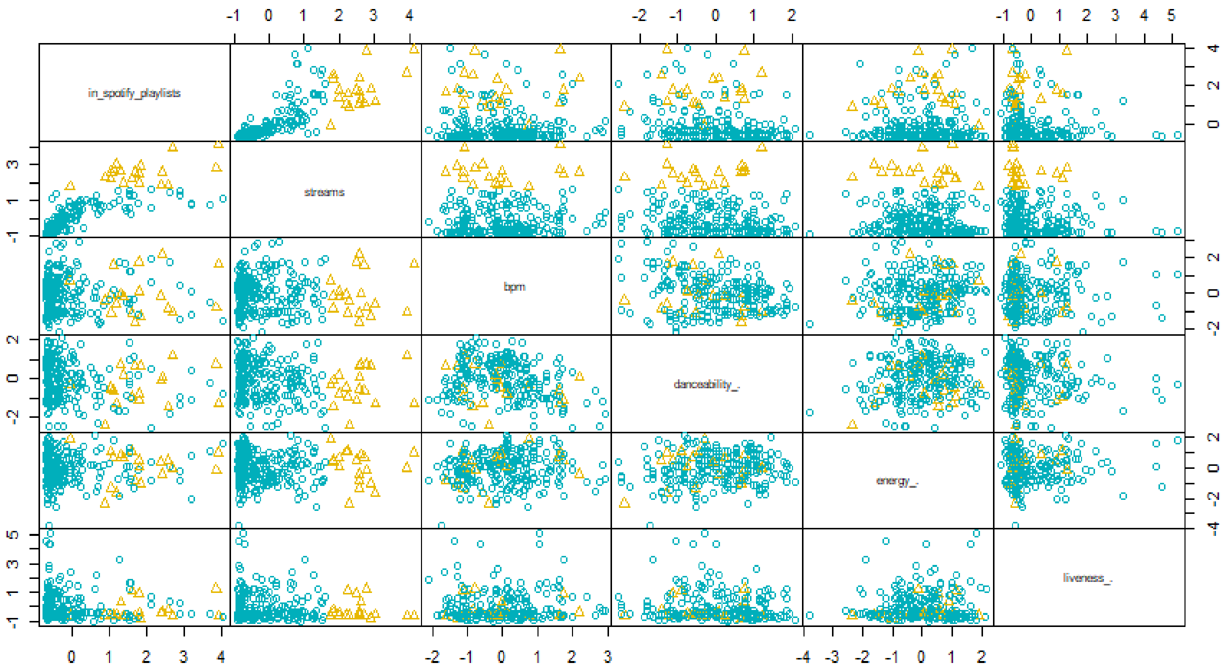
```
> eu.war <- hclust(d = d.euclidean, method = "ward.D2")
> fviz_dend(eu.war, k = 2, cex = 0.5, k_colors = c("blue","green"), color_labels_b
y_k = TRUE,rect = TRUE, show_labels = FALSE, main="Ward.D2 linkage method - Euclid
ean")
```



```
> cor(d.euclidean, cophenetic(eu.war))
[1] 0.7724329
> grp <- cutree(eu.war, k = 2 )
> table(grp)
grp
  1   2
184  66
```

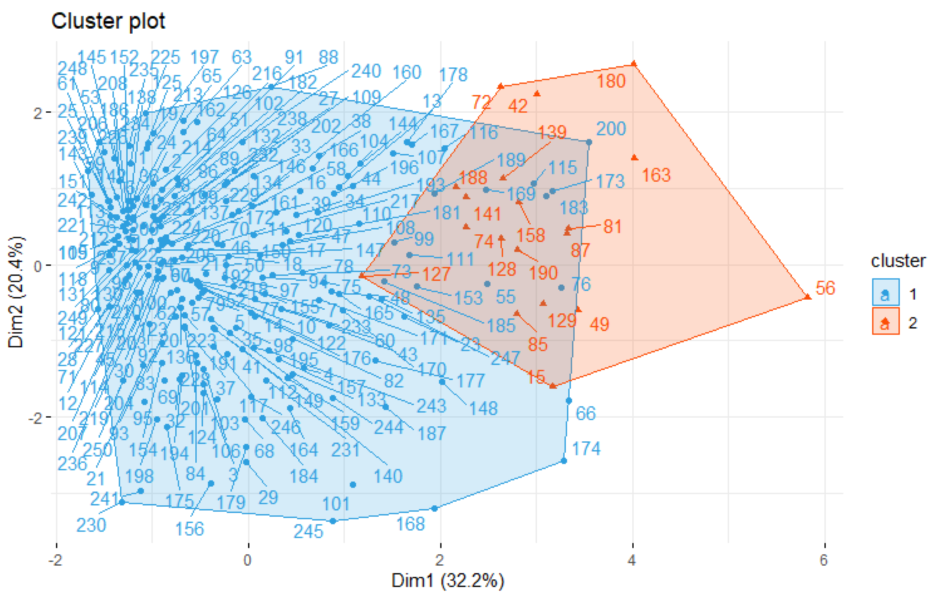
Now we have an interesting result. Two cluster with 184 and 66 observations with a high correlatio n between the dissimilarity matrix and the cophenetic distance.

```
> pairs(scaled_df, gap=0, pch=grp, col=c("#00AFBB", "#E7B800")[grp])
```



With the command above we have the possibility to show, in the original space, the two clusters composed with the Euclidian distance and the WARD.D2 linkage method. The lightblue circles are from the first cluster, yellow triangle from the others. We can also see that from each couple of variables the grade of separation between the clusters. In fact, the variable "Streams", paired with the other variables, it shows the division of the clusters better.

So, to the Agglomerative Method with k=2, the WARD.D2 linkage method is preferred because it have the biggest correlation between the cophenetic matrix and the dissimilarity matrix. So, the original distance is preserved and there is a good division between cluster.



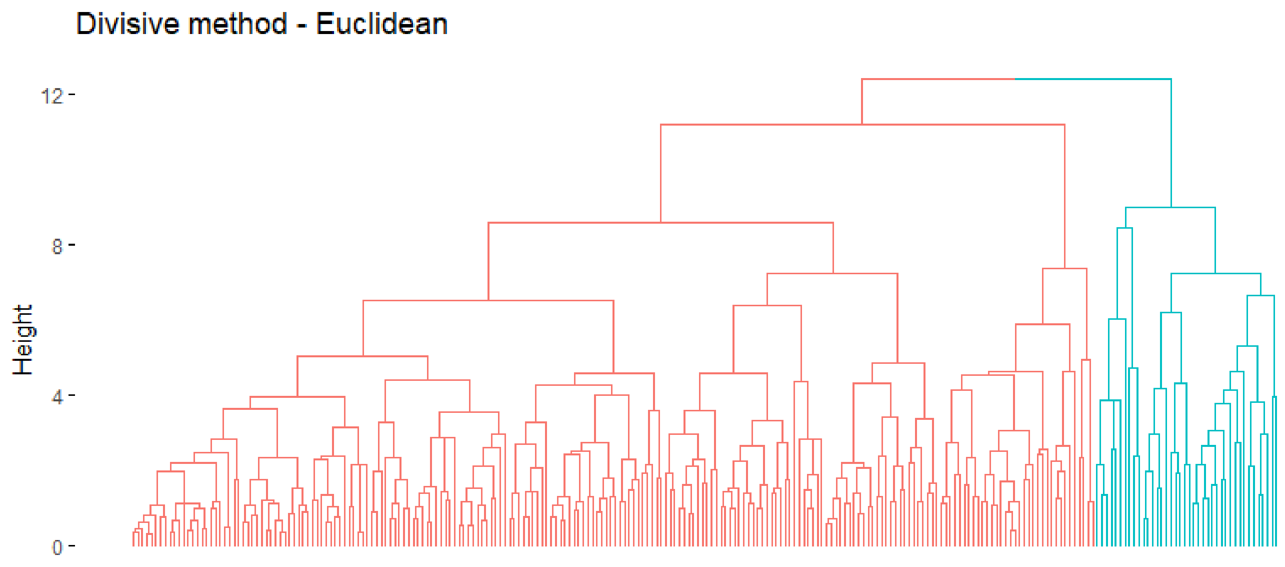
```
> fviz_cluster(list(data = numeric_df, cluster = grp),
+              palette = c("#2E9FDF", "#FC4E07"),
+              ellipse.type = "convex", # Concentration ellipse
+              repel = TRUE, # Avoid label overplotting (slow)
+              show.clust.cent = FALSE, ggtheme = theme_minimal())
```

We can also see the two clusters in the Principal Component space, where there's a loss of information and variability, so we also lose the separation between the clusters.

3.2.Divisive

DIANA (Divisive Analysis clustering) is a hierarchical clustering algorithm used for partitioning data into clusters.

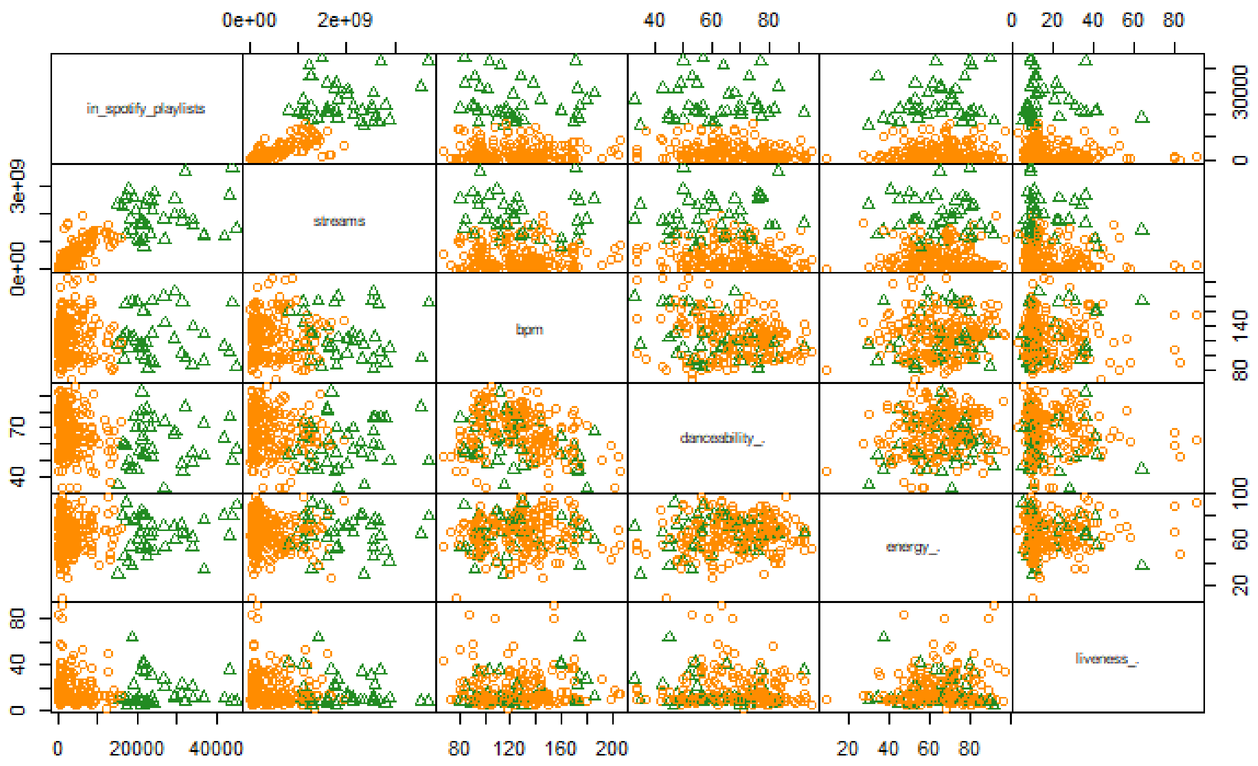
```
> res.diana <- diana(x = numeric_df, stand = TRUE, metric = "euclidean")
> fviz_dend(res.diana, cex = 0.6, k = 2, main = "Divisive method - Euclidean", sho
w_labels = FALSE)
```



```
> cor(d.euclidean, cophenetic(res.diana))
[1] 0.4671074
> grp <- cutree(res.diana, k = 2)
> table(grp)
grp
  1   2
210  40
```

In this case there's a low correlation, so the distances in the real world aren't respected so much. Anyway, this algorithm found 210 observations in the first group and 40 in the second. So, we can see the difference between the pairplots of DIANA and AGGLOMERATIVE. In fact, the Diana's pairplot shows more separation between the clusters but we know that the distance is not respected so much. As tell us the correlation index of 0.47, lower than the one obtained with the agglomerative method (WARD.D2, 0.77).

```
> pairs(numeric_df, gap=0,cex = 1, pch=grp, col=c("darkorange", "forestgreen")[grp
])
```

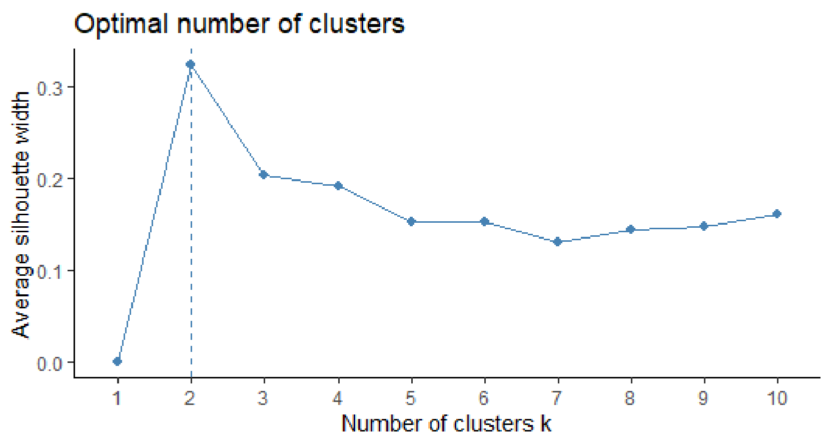


In this case the variable "In_spotify_playlists" seems to show better the division between the cluster and in general there's so much division then the other pairwise. But, since the correlation is bigger in the ward.d2 method, the preferred way is the Agglomerative Method with Euclidean Distance and Ward.D2 linkage method.

4.PAM

PAM, which stands for Partitioning Around Medoids, is a clustering algorithm that belongs to the family of partitioning methods. The distinctive feature of PAM is that it uses medoids instead of centroids to represent clusters. The centroid of a cluster is the geometric midpoint of all points in the cluster. It is the mean of the coordinates of all points. The medoid of a cluster is one of the points in the cluster that minimizes the sum of distances between itself and the other points in the cluster. It is a more robust choice than the centroid.

```
> fviz_nbclust(scaled_df, pam, method = "silhouette")+
+   theme_classic()
```



We recall the silhouette method, and we see that k = 2 is the best number of clusters, so we proceed in this way.

```
> pam.res <- pam(x = scaled_df, k = 2, diss = F, metric = "euclidean")
> print(pam.res)
Medoids:
      ID in_spotify_playlists    streams        bpm danceability_.    energy_. liv
eness_.
[1,] 220           -0.3976162 -0.3625083 -0.3088050      0.3723129 -0.40621935  0.
1246363
[2,] 190            1.8011125  1.9958312  0.1255199     -0.7500130  0.06550186 -0.
4457815
Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 2 2 1 1
 [46] 1 1 1 2 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 2 1 2 1 1 1 1 2 1 1
1 2 1 2 1 1 1
 [91] 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 2 1 1 2 2 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 2 2
2 1 1 1 1 1 2
[136] 1 1 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 2 2 2 1 1 1 2
2 1 1 2 2 1 2
[181] 1 1 2 1 2 1 2 2 2 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1
[226] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
Objective function:
   build     swap
2.077467 2.062884

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

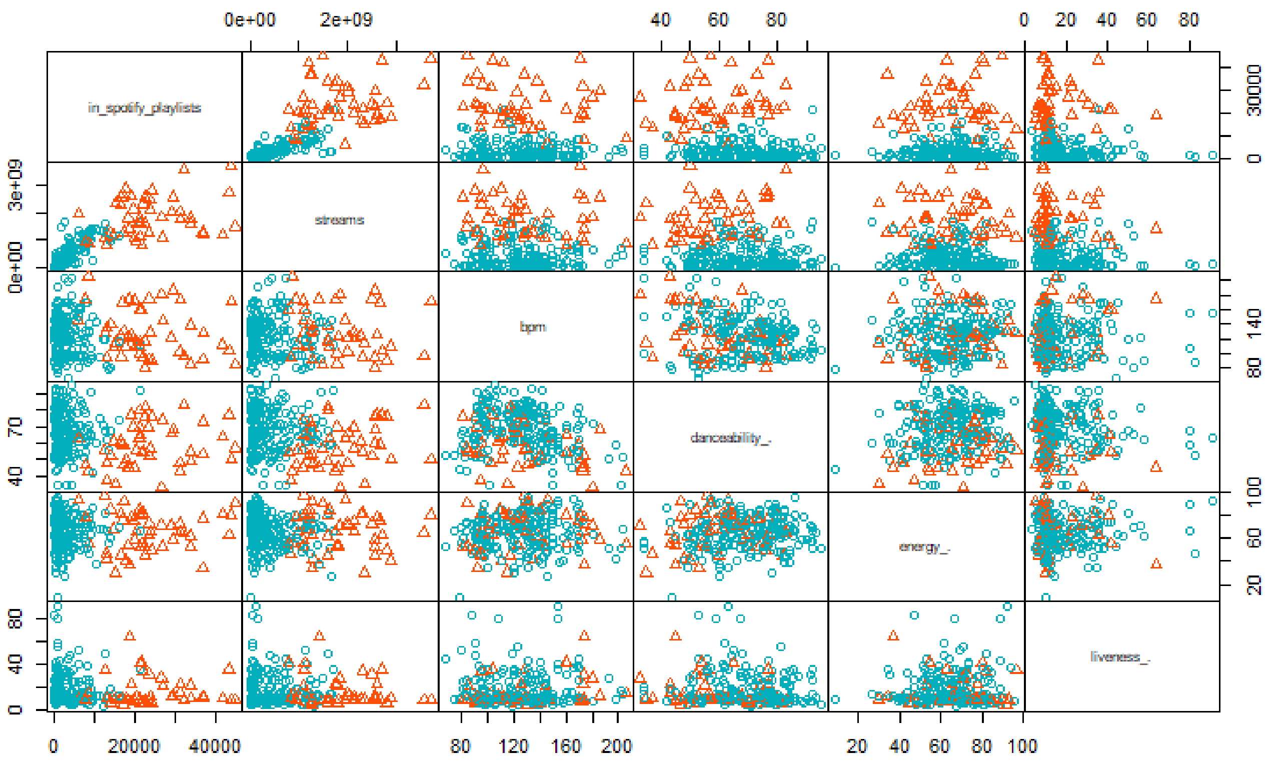
Recalling the print() function, we see that the medoid for cluster One is the observation 220 and for cluster Two is the 190th. We can also see the values of the variables relative to each medoid.

```
> pam.res$clusinfo
     size max_diss   av_diss diameter separation
[1,]  203 5.743895 1.985660 8.616839  0.7375849
[2,]   47 4.723217 2.396426 6.879154  0.7375849
```

Here above we have the result of this clustering method: the first cluster have 203 observation and a di ameter of 8.62. the second cluster have 47 observations and a diameter of 6.87. there is a good separat ion value but it's not so much. Let's plot this result.
Looking at the scatterplot below, between pairs of variables we can understand which are more useful to identify clusters. For example, looking at the scatterplot between danceability and streams we can see that the separation between clusters is good enough, although some observations overlap. Instead, if we look at the scatterplot between BPM and energy, we see a high overlap between clusters.

```
> cl <- pam.res$clustering
> pairs(numeric_df, gap=0, pch=cl, col=c("#00AFBB", "#FC4E07")[cl])
```

The first rows put in cl the observations in each cluster and the second shows the result in a plot.

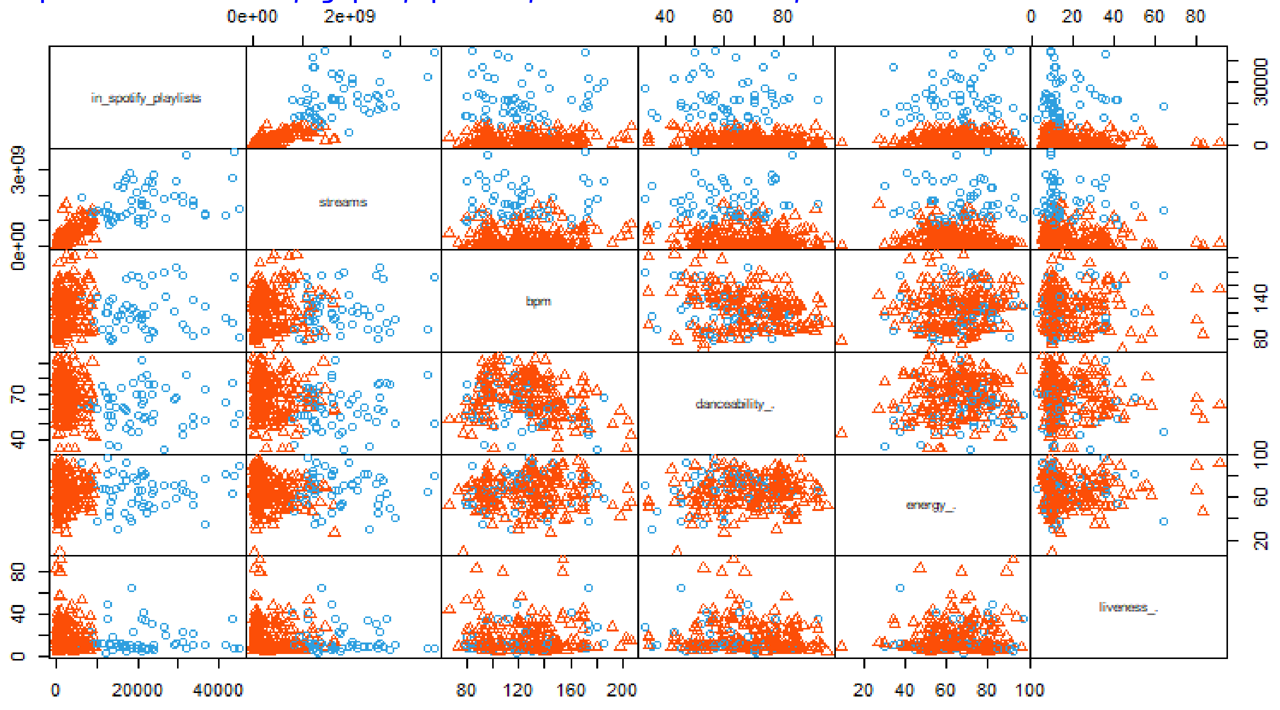


5.k-Means

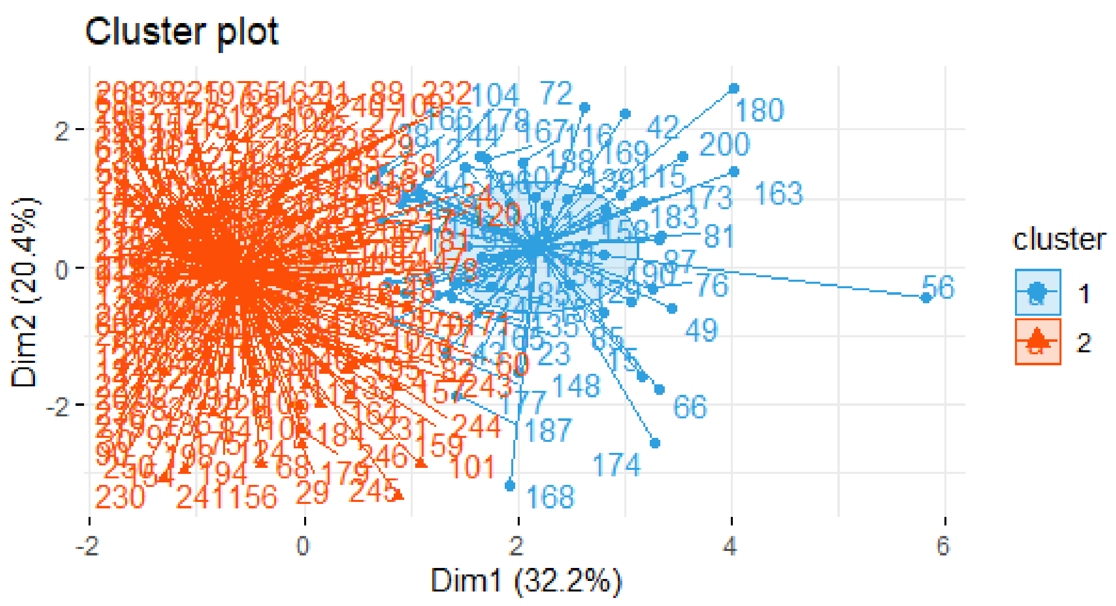
We already seen what a K-Means is. Now we use this algorithm to see the result of this computation.

```
> km.res <- kmeans(scaled_df, 2, nstart = 25)
> print(km.res)
K-means clustering with 2 clusters of sizes 56, 194

Cluster means:
  in_spotify_playlists    streams         bpm danceability_.    energy_.  liveness
_.
1            1.5889961  1.4716277 -0.12137312     -0.4547344  0.03662097 -0.112188
95
2           -0.4586793 -0.4247998  0.03503554      0.1312635 -0.01057100  0.032384
44

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1
2 2 2 1 1 1 2
 [46] 2 2 2 1 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 1 1 1 2 2 2 2 1 2 2
2 1 2 1 2 2 2
 [91] 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 1 2 2 1 1 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 1 1
1 2 2 2 2 2 1
[136] 2 2 2 1 2 1 2 2 1 2 2 2 1 2 2 2 2 1 2 2 2 2 1 2 2 2 2 1 2 1 1 1 1 1 2 2 2 1
1 2 2 1 1 2 1
[181] 2 2 1 2 1 2 1 1 1 1 2 2 1 2 2 1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2
[226] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2

Within cluster sum of squares by cluster:
[1] 317.6492 820.8634
 (between_SS / total_SS =  23.8 %)

Available components:
```

```
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "be
tweenss"
[7] "size"         "iter"         "ifault"

> # Mean of the original variables for each cluster
> aggregate(numeric_df, by=list(cluster=km.res$cluster), mean)
  cluster in_spotify_playlists    streams      bpm danceability_. energy_. livenes
s_.
1       1            21793.036 1749703842 122.1786       60.94643 65.57143   16.67
857
2       2             2380.093  355232497 126.5000       68.77835 64.87113   18.70
619
> # Adding the point classifications to the original data
> dd <- cbind(numeric_df, cluster = km.res$cluster)
> head(dd)
  in_spotify_playlists   streams bpm danceability_. energy_. liveness_. cluster
1                  553 141381703 125             80       83          8       2
2                 1474 133716286  92             71       74         10       2
3                 1397 140003974 138             51       53         31       2
4                 7858 800840817 170             55       72         11       2
5                 3133 303236322 144             65       80         11       2
6                 2186 183706234 141             92       58          8       2
> km.res$size
[1]  56 194
> round(km.res$centers,2)
  in_spotify_playlists streams   bpm danceability_. energy_. liveness_.
1                 1.59    1.47 -0.12          -0.45     0.04      -0.11
2                -0.46   -0.42  0.04           0.13    -0.01       0.03
> cl <- km.res$cluster
> pairs(numeric_df, gap=0, pch=cl, col=c("#2E9FDF", "#FC4E07")[cl])
```



```
> fviz_cluster(km.res,
+              data = numeric_df,
+              palette = c("#2E9FDF", "#FC4E07"),
+              ellipse.type = "euclid", # Concentration ellipse
+              star.plot = TRUE, # Add segments from centroids to items
+              repel = TRUE, # Avoid label overplotting (slow)
+              ggtheme = theme_minimal()
```



The procedure to do a k-Means clustering Method is the same as for k-medoids, and we can see the difference between this two partitioning methods. In the k-Means case, we have a cluster with 194 observations and one with 56 observations. From a graphical point of view, we can see that the variables “in_spotify_playlists” and “views” are the best to show separation between clusters.

To choose the best algorithm and so the clusters that are present in the data, we have to go trought the validiation measure.

6.Cluster Validation

Cluster validation is used to evaluate the goodness of clustering results. It can be categorized into 3 classes: internal, external and relative cluster validation. The internal cluster validation allows us to estimate the optimal number of clusters and to select the appropriate clustering algorithm, by means of two indices: the Silhouette Width and the Dunn index. the R cLValid() function allows to compare simultaneously multiple clustering algorithms using both “internal” and “stability” cluster validation measures.

```
> clust.methods <- c("hierarchical","kmeans","pam")
> internal <- clValid(scaled_df,nClust = 2:6,clMethods = clust.methods, validation
= "internal")
> summary(internal)
Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6

Validation Measures:
                                  2        3        4        5        6

hierarchical Connectivity    9.8421   9.8421  40.7278  46.5119  51.4532
             Dunn            0.1987   0.1987   0.1344   0.1519   0.1519
             Silhouette      0.4289   0.3661   0.3154   0.2567   0.2340
kmeans       Connectivity   51.3639  92.5306 103.3897 123.6750 144.3048
             Dunn            0.0538   0.0682   0.0889   0.0905   0.0914
             Silhouette      0.1341   0.2271   0.2320   0.2102   0.1986
```

```
pam          Connectivity   46.4044   98.1460 129.2329 165.1937 171.4417
             Dunn            0.0856    0.0732   0.0531   0.0404   0.0738
             Silhouette      0.3249    0.2035   0.1925   0.1536   0.1537

Optimal Scores:

             Score  Method       Clusters
Connectivity 9.8421 hierarchical 2
Dunn         0.1987 hierarchical 2
Silhouette   0.4289 hierarchical 2

> stability <- clValid(scaled_df,nClust = 2:6, clMethods = clust.methods, validati
on = "stability")
> summary(stability)
Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6

Validation Measures:
                       2      3      4      5      6

hierarchical APN  0.0080 0.0250 0.0451 0.0894 0.1056
             AD   3.1770 3.1424 3.0473 2.9384 2.8698
             ADM  0.1615 0.1573 0.8058 0.6530 0.5618
             FOM  1.0008 1.0003 0.9982 0.9791 0.9778
kmeans       APN  0.2715 0.2343 0.2493 0.2632 0.3470
             AD   3.1362 2.7776 2.5823 2.5016 2.4941
             ADM  0.9489 0.8043 0.6625 0.7643 0.9537
             FOM  0.9833 0.9568 0.9244 0.9168 0.9190
pam          APN  0.2503 0.3251 0.2957 0.3987 0.3736
             AD   3.0154 2.8355 2.6150 2.5723 2.4709
             ADM  0.7542 0.9256 0.7544 0.9605 0.9042
             FOM  0.9959 0.9182 0.9203 0.9188 0.9149

Optimal Scores:

    Score  Method       Clusters
APN 0.0080 hierarchical 2
AD  2.4709 pam          6
ADM 0.1573 hierarchical 3
FOM 0.9149 pam          6
```

To summarize: looking at the Internal measures, K = 2 in the hierarchical method is established as number of patterns that is more likely to be found within the dataset. For the stability measures, the best method are hierarchical (k=2 and k=6) and PAM (K=6). Then, if we must choose one clustering method, surely, we have to use the agglomerative method (ward.d2) or the PAM, but also the k-means return satisfactory results in terms of cohesion and separation.

Fuzzy Clustering

Fuzzy clustering is a variant of traditional clustering that allows for the assignment of data points to multiple clusters with varying degrees of membership. In traditional clustering, each data point is assigned to exactly one cluster, making it a "hard" assignment. Fuzzy clustering, on the other hand, introduces the concept of partial membership, meaning that a data point can belong to multiple clusters simultaneously, and the degree of membership is expressed as a probability distribution.

while traditional clustering provides a clear assignment of data points to clusters, fuzzy clustering introduces a more flexible and probabilistic approach, allowing for partial memberships and capturing the inherent ambiguity in some datasets.

For this report I choose to use two algorithms:

- Fuzzy K-Means (FKM);
- Fuzzy K-Medoids (FKMed);

1.FKM

```
> fkm <- FKM(X = numeric_df, m = 1.2, RS = 50, stand = 1, index = "SIL.F")
The default value k=2:6 has been set
The default value alpha=1 has been set for computing SIL.F
> summary(fkm)

 Fuzzy clustering object of class 'fclust'

 Number of objects:
 250

 Number of clusters:
 2

 Cluster sizes:
Clus 1 Clus 2
    58    192

 Clustering index values:
SIL.F k=2 SIL.F k=3 SIL.F k=4 SIL.F k=5 SIL.F k=6
0.5039650 0.4047346 0.4104765 0.3885113 0.3502547

 Closest hard clustering partition (first 24 object):
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  2
1  22  23  24
  2   2   2   2   2   2   2   2   2   2   2   2   1   2   1   2   2   2   2   2
2   2   1   2

 Cluster memberships:
  Clus 1 (First 25 objects)
 [1] "13"  "15"  "23"  "38"  "42"  "43"  "44"  "48"  "49"  "55"  "56"  "66"  "72"
"74"  "75"
[16] "76"  "81"  "85"  "87"  "99"  "104" "107" "110" "111" "115"
 Clus 2 (First 25 objects)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "14" "16" "17" "1
8" "19" "20"
[19] "21" "22" "24" "25" "26" "27" "28"

Number of objects with unclear assignment (maximal membership degree <0.5):
 0

 Membership degree matrix (rounded, first 10 objects):
   Clus 1 Clus 2
1    0.00   1.00
2    0.00   1.00
3    0.00   1.00
4    0.17   0.83
5    0.00   1.00
6    0.00   1.00
7    0.00   1.00
8    0.00   1.00
9    0.00   1.00
10   0.01   0.99

 Cluster summary:
      Cl.size Min.memb.deg. Max.memb.deg. Av.memb.deg. N.uncl.assignm.
Clus 1      58          0.50             1         0.92               0
Clus 2     192          0.51             1         0.96               0

 Euclidean distance matrix for the prototypes (rounded):
      Clus 1
Clus 2   2.76

> Hraw(fkm$X, fkm$H)
      in_spotify_playlists    streams      bpm danceability_. energy_. liveness_.
Clus 1            20740.083 1700450791 124.0193       60.51794 64.84115   16.87958
Clus 2             2394.541  342760703 125.8793       69.23546 65.14407   18.51582
```

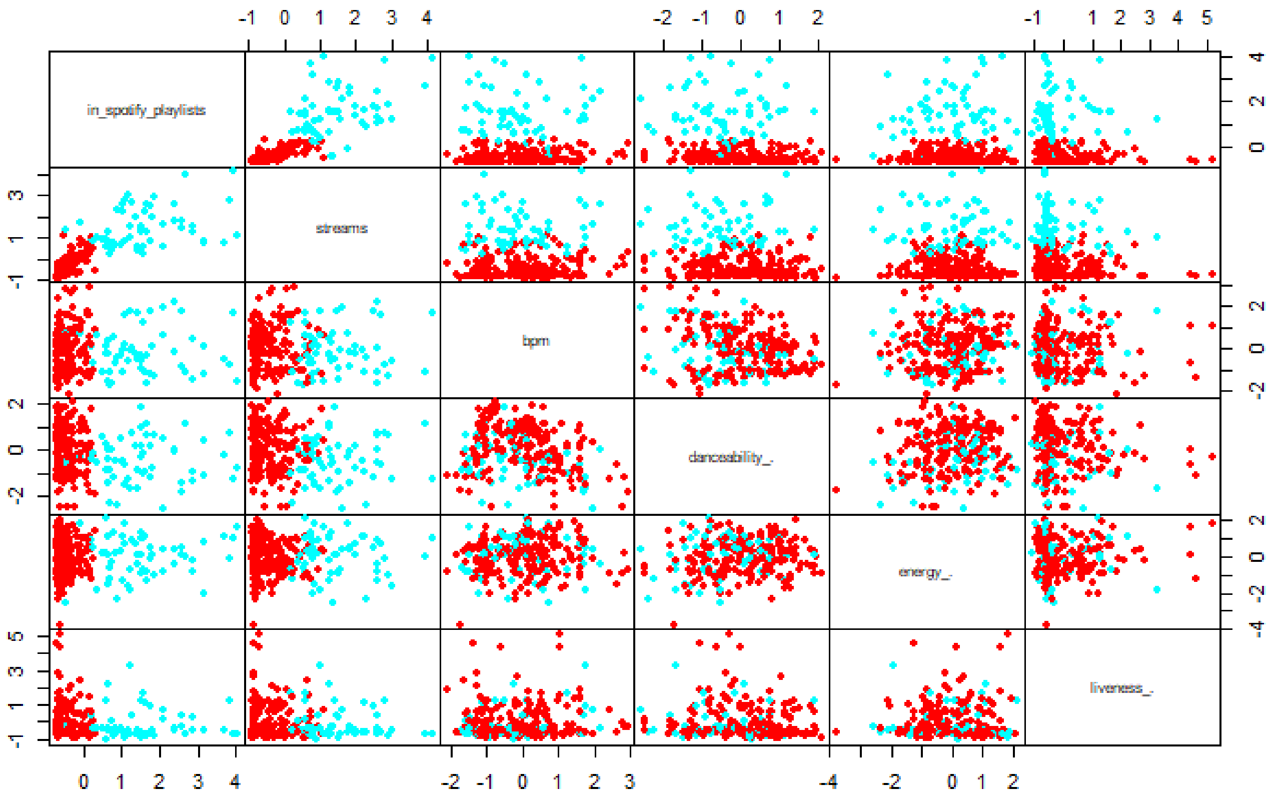
From summary(), we obtain the most important information that we need. I've chosen to use two cluster as suggested by the Fuzzy Silhouette (SIL F), where in k=2 have the biggest value (0.50). the size of the cluster 1 is 58 and of cluster 2 is 192.

It is possible to see the observations like we are in a hard-clustering situation and what are the observations that compose the two clusters. As we said, observations can belong to more than one cluster, shown as the degree of belonging to each cluster, so it is possible to see some unclear assignment, but in this case all observation are clearly assigned.

Then, we see the Membership degree matrix where is expressed, for each observation, the degree (from 0 to 1) to belong to each cluster. For space reasons, I have only shown the first ten observations. The Euclidean distance between the two clusters is 2.76, that is a good distance.

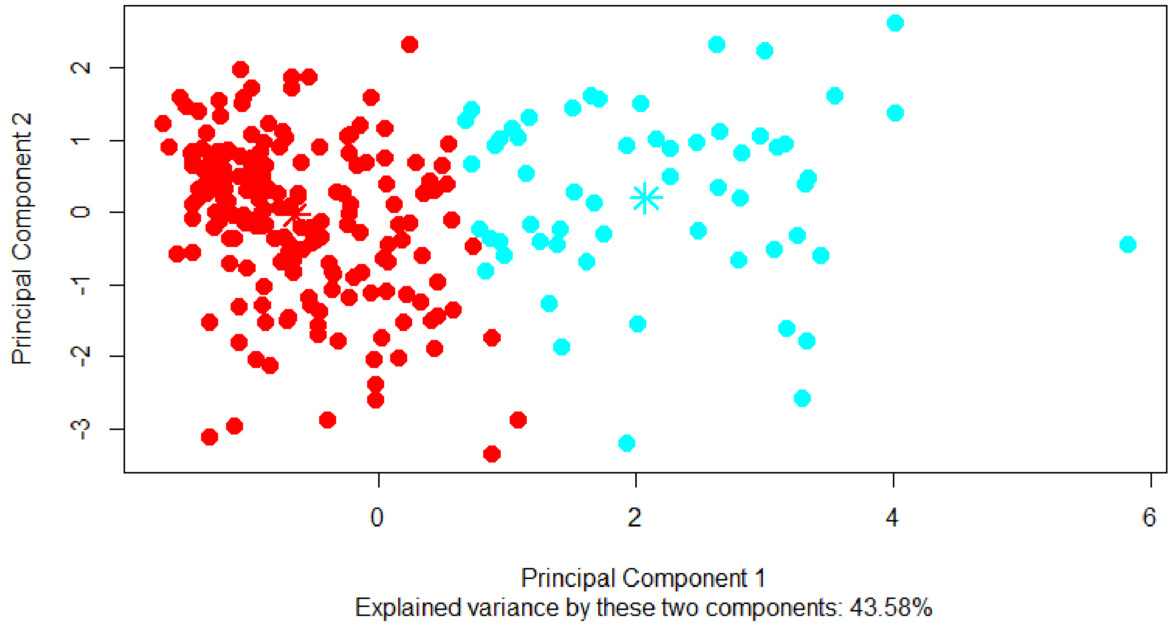
As always, a graphical representation can help us:

```
>pairs(scaled_df, col = fkm$clus[, 1], pch = 16, cex = 0.8, gap = 0)
```



like the result of Hard-Clustering, “streams” and “in_spotify_playlists” seem to explain better the cluster and their form.

```
> plot.fclust(x = fkm, pca = TRUE)
```



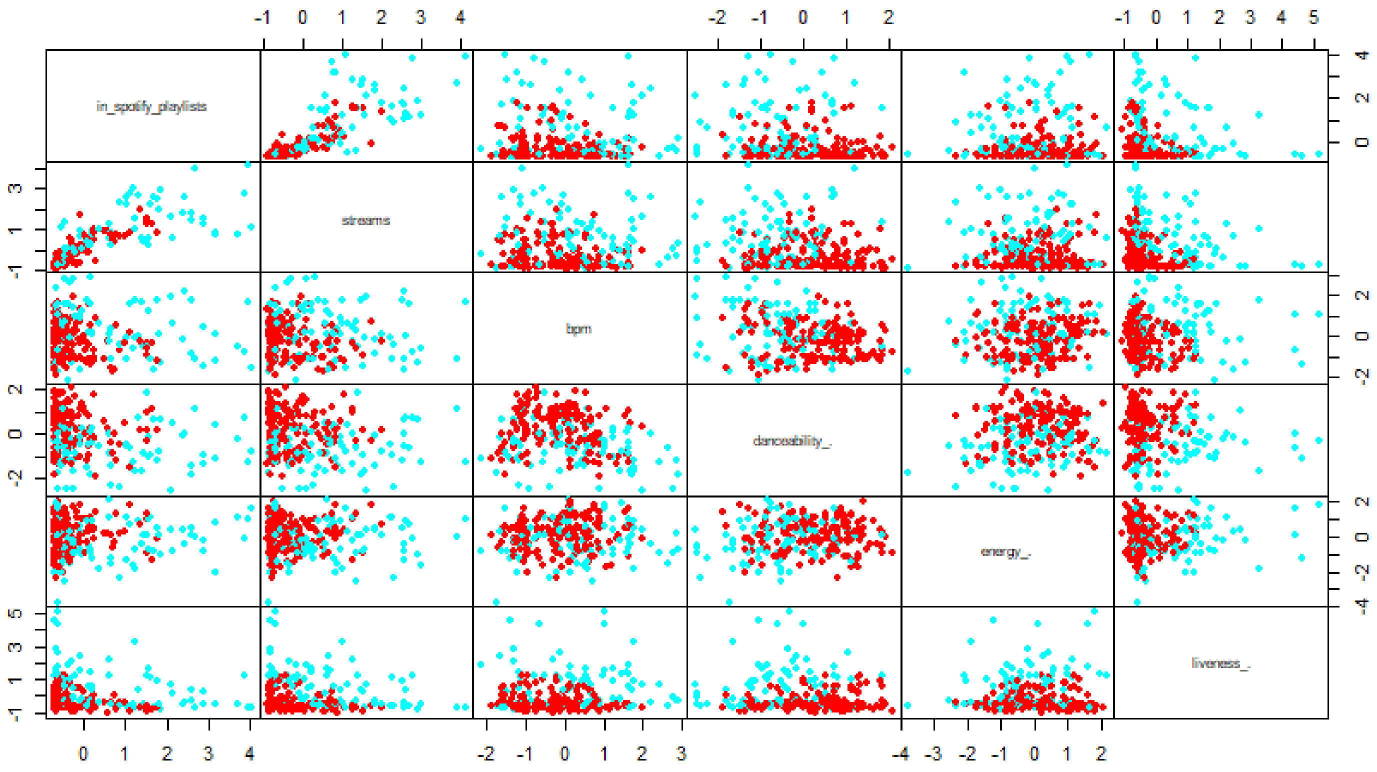
We can see the two cluster in the PC space, where the variance explained by the first PC is equal to 43.58%.

2.FKMed

```
> fkmed <- FKM.med(X = scaled_df, m = 1.2, RS = 50, stand = 1, index = "SIL.F")
The default value k=2:6 has been set
The default value alpha=1 has been set for computing SIL.F
> summary(fkmed)

 Fuzzy clustering object of class 'fclust'

 Number of objects:
 250

 Number of clusters:
 2

 Cluster sizes:
Clus 1 Clus 2
   163     87

 Clustering index values:
SIL.F k=2 SIL.F k=3 SIL.F k=4 SIL.F k=5 SIL.F k=6
0.3944258 0.3186629 0.2142052 0.2466798 0.2781161

 Closest hard clustering partition (first 24 object):
  Obj 1   Obj 2   Obj 3   Obj 4   Obj 5   Obj 6   Obj 7   Obj 8   Obj 9  Obj 10  O
bj 11  Obj 12
      1       1       2       1       1       1       1       1       1       1
1       1
 Obj 13  Obj 14  Obj 15  Obj 16  Obj 17  Obj 18  Obj 19  Obj 20  Obj 21  Obj 22  O
bj 23  Obj 24
      1       1       2       1       2       2       1       2       2       1
2       1

 Cluster memberships:
  Clus 1 (First 25 objects)
 [1] "Obj 1"  "Obj 2"  "Obj 4"  "Obj 5"  "Obj 6"  "Obj 7"  "Obj 8"  "Obj 9"  "Obj
10" "Obj 11"
[11] "Obj 12" "Obj 13" "Obj 14" "Obj 16" "Obj 19" "Obj 22" "Obj 24" "Obj 25" "Obj
26" "Obj 27"
[21] "Obj 30" "Obj 31" "Obj 32" "Obj 33" "Obj 34"
 Clus 2 (First 25 objects)
 [1] "Obj 3"   "Obj 15"  "Obj 17"  "Obj 18"  "Obj 20"  "Obj 21"  "Obj 23"  "Obj 28
"  "Obj 29"
[10] "Obj 42"  "Obj 43"  "Obj 48"  "Obj 49"  "Obj 50"  "Obj 55"  "Obj 56"  "Obj 60
"  "Obj 66"
[19] "Obj 67"  "Obj 68"  "Obj 69"  "Obj 71"  "Obj 72"  "Obj 73"  "Obj 74"

 Number of objects with unclear assignment (maximal membership degree <0.5):
 0

 Membership degree matrix (rounded, first 10 object):
       Clus 1 Clus 2
Obj 1    0.99   0.01
Obj 2    1.00   0.00
Obj 3    0.02   0.98
Obj 4    0.51   0.49
Obj 5    0.99   0.01
Obj 6    0.96   0.04
Obj 7    0.99   0.01
Obj 8    1.00   0.00
Obj 9     0.70   0.30
Obj 10    0.80   0.20

 Cluster summary:
       Cl.size Min.memb.deg. Max.memb.deg. Av.memb.deg. N.uncl.assignm.
Clus 1     163          0.50             1         0.86               0
Clus 2      87          0.52             1         0.81               0

 Euclidean distance matrix for the prototypes (rounded):
       Clus 1
Clus 2    1.5
```

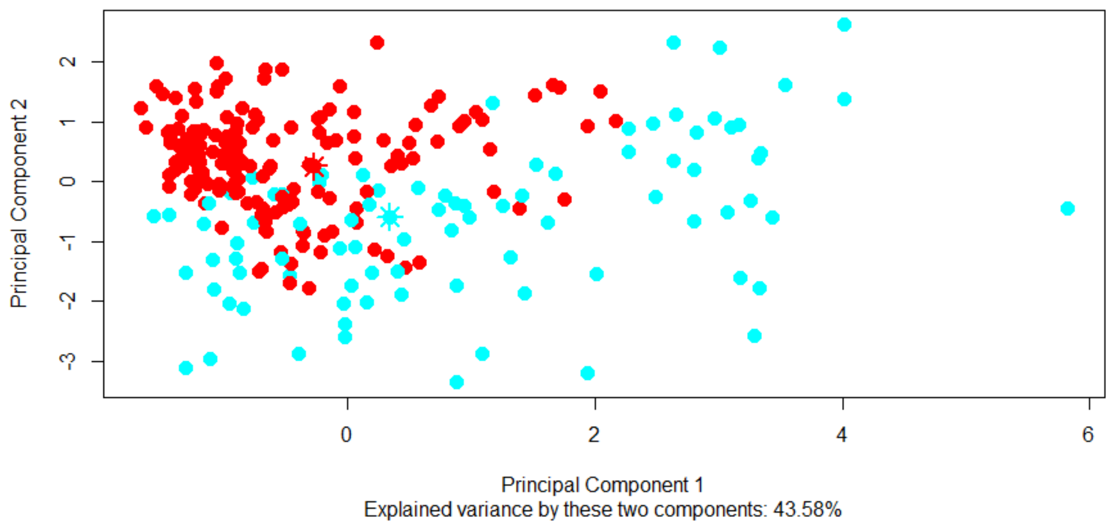
Like first, I've chosen to use two cluster as suggested by the Fuzzy Silhouette (SIL F), where in k=2 have the biggest value (0.39). the size of the cluster 1 is 163 and of cluster 2 is 87.

Also, in this case there are not unclear observations. Now, watching at the Membership degree matrix, seems that some observation has a low degree to belong to a cluster, so it means that the clusters are more united than the clusters of FKM. In fact, the Euclidean distance (1.5) tell us that the separation between the two clusters is not so high. Let's plot these results:



As predicted, the distance between the two clusters is not so much, so the visualization of the clusters it's not so good as in the case of Fuzzy K-means.

```
> plot.fclust(x = fkmed, pca = TRUE)
```



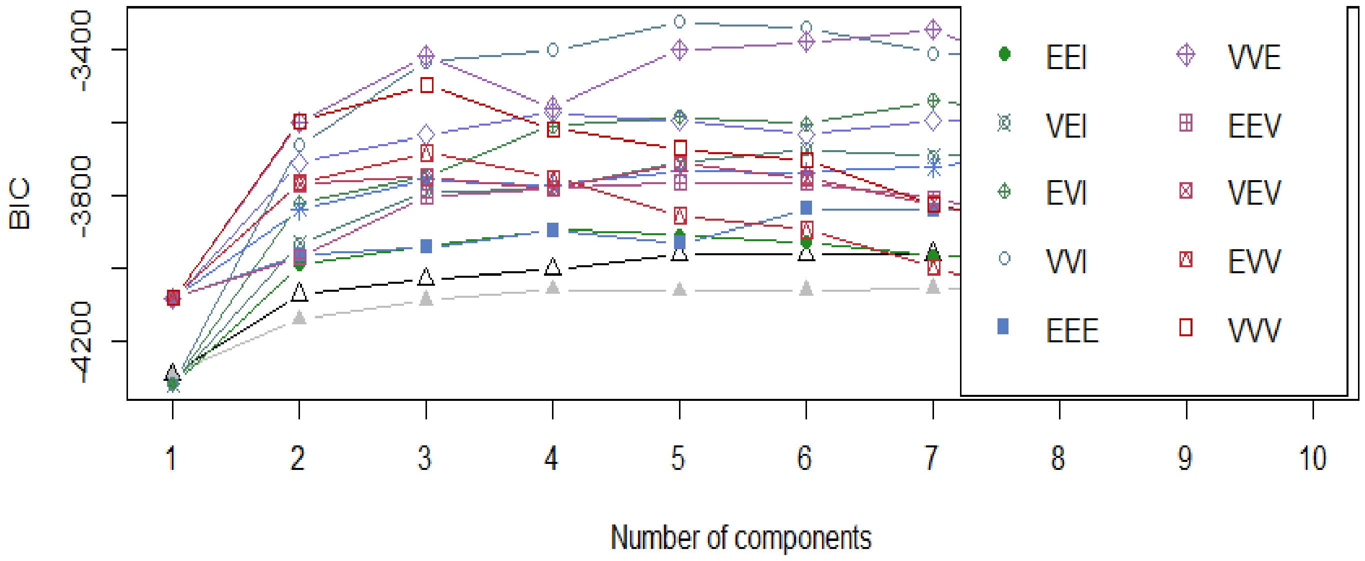
Obviously, the Explained variance is the same, but also in this case we see that the separation is not so obvious.

Model-Based Clustering

Model-based clustering is a statistical technique used in data analysis to group observations or data points into clusters based on a probabilistic model. Unlike traditional clustering algorithms that assign each data point to a single cluster, model-based clustering allows for uncertainty in cluster assignments. It assumes that the data are generated from a mixture of underlying probability distributions, each associated with a specific cluster.

Mclust() function is run to find the best parsimonious configuration of the Gaussian mixture distribution and the most appropriate number of K components. That is assumed as the model from which data were generated.

```
> model <- Mclust(scaled_df, G = 1:10, modelNames = NULL)
fitting ...
  |=================================================================================
=======| 100%
> plot(model$BIC)
```



```
> summary(model$BIC)
Best BIC values:
             VVI,5        VVI,6        VVE,7
BIC      -3323.643 -3338.95667 -3345.89006
BIC diff     0.000   -15.31334   -22.24672
```

According to the BIC, top three models are:

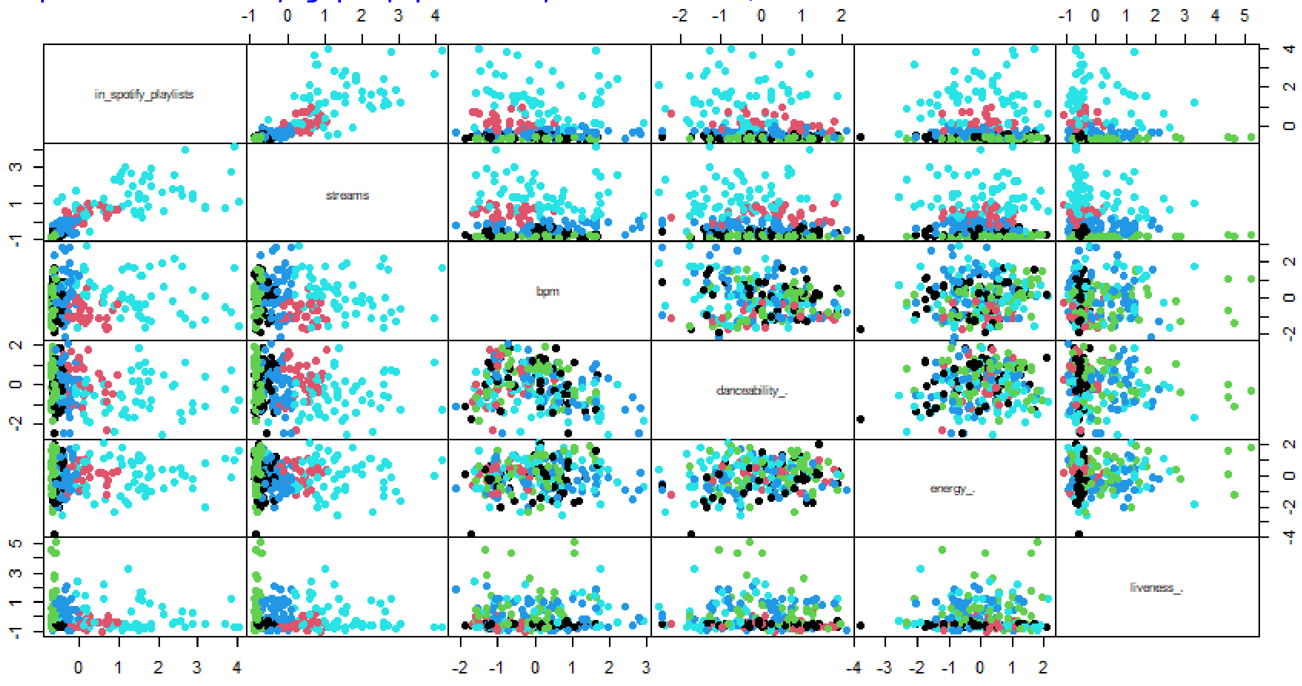
1. VVI model with five components
2. VVI model with six components
3. VVE model with seven components

```
> summary(model)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------
Mclust VVI (diagonal, varying volume and shape) model with 5 components:

 log-likelihood   n df       BIC       ICL
      -1485.135 250 64 -3323.643 -3372.752
```

```
Clustering table:
 1  2  3  4  5 
48 29 57 56 60 
```

The chosen model so is VVI with five component, where the clusters have, in order, 49,29,57,56 and 60 observations

```
> head(round(model$z, 6), 15)    # Probability to belong to a given cluster
          [,1]     [,2]     [,3]     [,4]     [,5]
 [1,] 0.637979 0.000015 0.357763 0.004202 0.000042
 [2,] 0.992578 0.000115 0.000969 0.006309 0.000029
 [3,] 0.000000 0.000000 0.047077 0.947654 0.005269
 [4,] 0.000000 0.002176 0.000000 0.044509 0.953315
 [5,] 0.466063 0.000197 0.000000 0.531955 0.001785
 [6,] 0.917090 0.000019 0.000000 0.082831 0.000060
 [7,] 0.000006 0.012551 0.000000 0.966229 0.021214
 [8,] 0.450720 0.000022 0.547866 0.001374 0.000018
 [9,] 0.000000 0.000000 0.957338 0.042573 0.000090
[10,] 0.001303 0.000002 0.000000 0.996585 0.002110
[11,] 0.000000 0.000028 0.000000 0.996576 0.003396
[12,] 0.002161 0.000000 0.996678 0.001142 0.000018
[13,] 0.000000 0.801365 0.000000 0.000000 0.198635
[14,] 0.039901 0.000436 0.000000 0.954573 0.005090
[15,] 0.000000 0.000000 0.000000 0.000000 1.000000
> head(model$classification, 15) # Cluster assignment of each observation
 [1] 1 1 4 5 4 1 4 3 3 4 4 3 2 4 5
> pairs(scaled_df, gap=0, pch = 16, col = model$classification)
```



We can create a dataframe that contain the most important information for each observations, in terms of probability, classification and uncertainty.

```
> probabilities <- as.matrix(round(model$z, 5)) # Probability to belong to a given
cluster
> colnames(probabilities) <- c("pi1", "pi2", "pi3", "pi4", "pi5")
> classification <- as.matrix(model$classification) # Cluster assignment of each o
bservation
> colnames(classification) <- "Classification"
> uncertainty <- as.matrix(round(model$uncertainty, 5)) # Classification Uncertain
ty
> colnames(uncertainty) <- "Uncertainty"
> # Combining the overall information in one single matrix for a more orderly disp
lay of data
> model_output <- cbind(probabilities, classification, uncertainty)
> head(model_output, 30)
          pi1     pi2     pi3     pi4     pi5 Classification Uncertainty
 [1,] 0.63798 0.00001 0.35776 0.00420 0.00004              1     0.36202
 [2,] 0.99258 0.00011 0.00097 0.00631 0.00003              1     0.00742
 [3,] 0.00000 0.00000 0.04708 0.94765 0.00527              4     0.05235
 [4,] 0.00000 0.00218 0.00000 0.04451 0.95332              5     0.04668
 [5,] 0.46606 0.00020 0.00000 0.53195 0.00178              4     0.46805
 [6,] 0.91709 0.00002 0.00000 0.08283 0.00006              1     0.08291
 [7,] 0.00001 0.01255 0.00000 0.96623 0.02121              4     0.03377
 [8,] 0.45072 0.00002 0.54787 0.00137 0.00002              3     0.45213
 [9,] 0.00000 0.00000 0.95734 0.04257 0.00009              3     0.04266
[10,] 0.00130 0.00000 0.00000 0.99659 0.00211              4     0.00341
[11,] 0.00000 0.00003 0.00000 0.99658 0.00340              4     0.00342
[12,] 0.00216 0.00000 0.99668 0.00114 0.00002              3     0.00332
[13,] 0.00000 0.80136 0.00000 0.00000 0.19864              2     0.19864
[14,] 0.03990 0.00044 0.00000 0.95457 0.00509              4     0.04543
[15,] 0.00000 0.00000 0.00000 0.00000 1.00000              5     0.00000
[16,] 0.00000 0.97789 0.00000 0.00000 0.02210              2     0.02211
[17,] 0.00000 0.00000 0.00000 0.99878 0.00122              4     0.00122
[18,] 0.99421 0.00000 0.00560 0.00010 0.00009              1     0.00579
[19,] 0.56822 0.02285 0.00000 0.40865 0.00028              1     0.43178
[20,] 0.00000 0.00000 0.00000 0.99441 0.00559              4     0.00559
[21,] 0.00000 0.00000 0.99929 0.00069 0.00001              3     0.00071
[22,] 0.00286 0.00000 0.99627 0.00085 0.00002              3     0.00373
[23,] 0.00000 0.01647 0.00000 0.00000 0.98353              5     0.01647
[24,] 0.98694 0.00002 0.00002 0.01300 0.00002              1     0.01306
[25,] 0.00000 0.00000 0.00152 0.99708 0.00140              4     0.00292
[26,] 0.25188 0.00000 0.74770 0.00041 0.00001              3     0.25230
[27,] 0.00000 0.99400 0.00000 0.00058 0.00543              2     0.00600
[28,] 0.00000 0.00000 0.99988 0.00012 0.00000              3     0.00012
[29,] 0.00012 0.00000 0.00000 0.99739 0.00249              4     0.00261
[30,] 0.00000 0.00000 0.00000 0.99392 0.00608              4     0.00608
```

In order, we can see the probability to belong to each cluster, the classification (i.e. in which cluster is located the observation, and the uncertainty (as the summation of probabilities of clusters where the observation is not classified.

If each mixture component corresponds to a cluster, we can say for instance that:

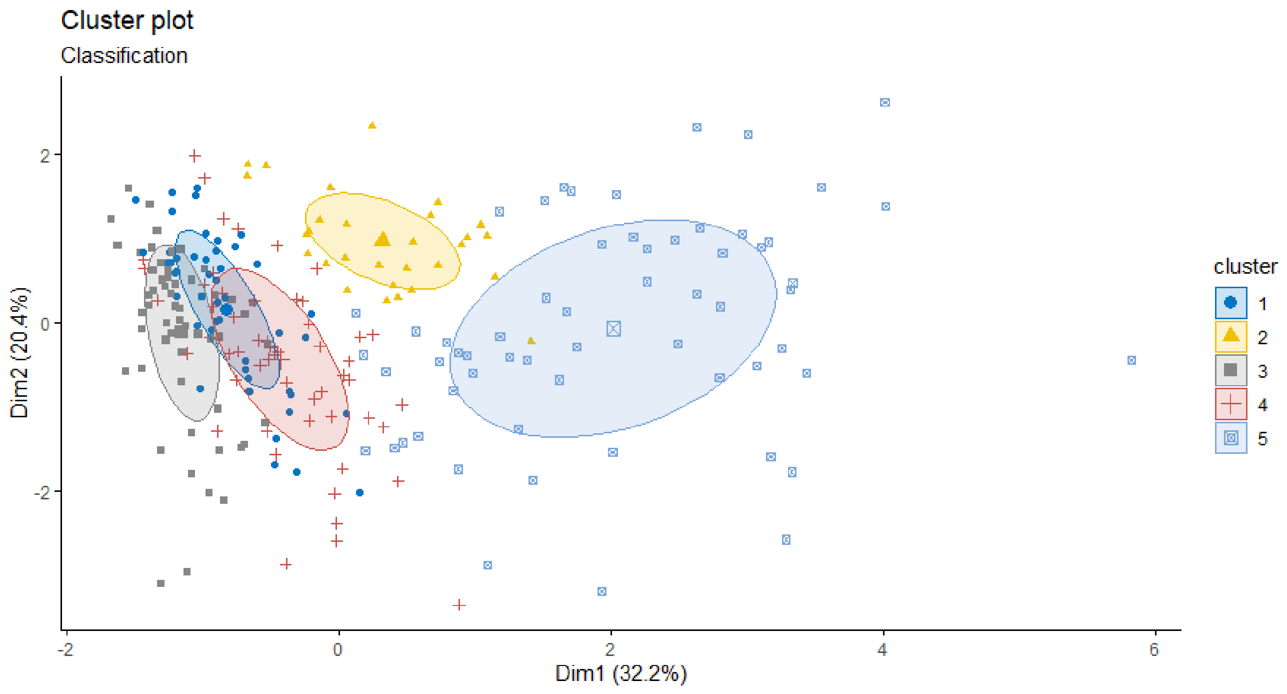
The first song (“Seven”) has been soft-assigned to:

- the first component (i.e. cluster) of the mixture distribution with a fitted posterior probability of 63.79% to belong to it;
- to the third component with a probability of 35.77% to belong to it;
- to the remaining components with very close to zero probabilities.
- Therefore, the observation has been classified into cluster 1, with an uncertainty of 36%.

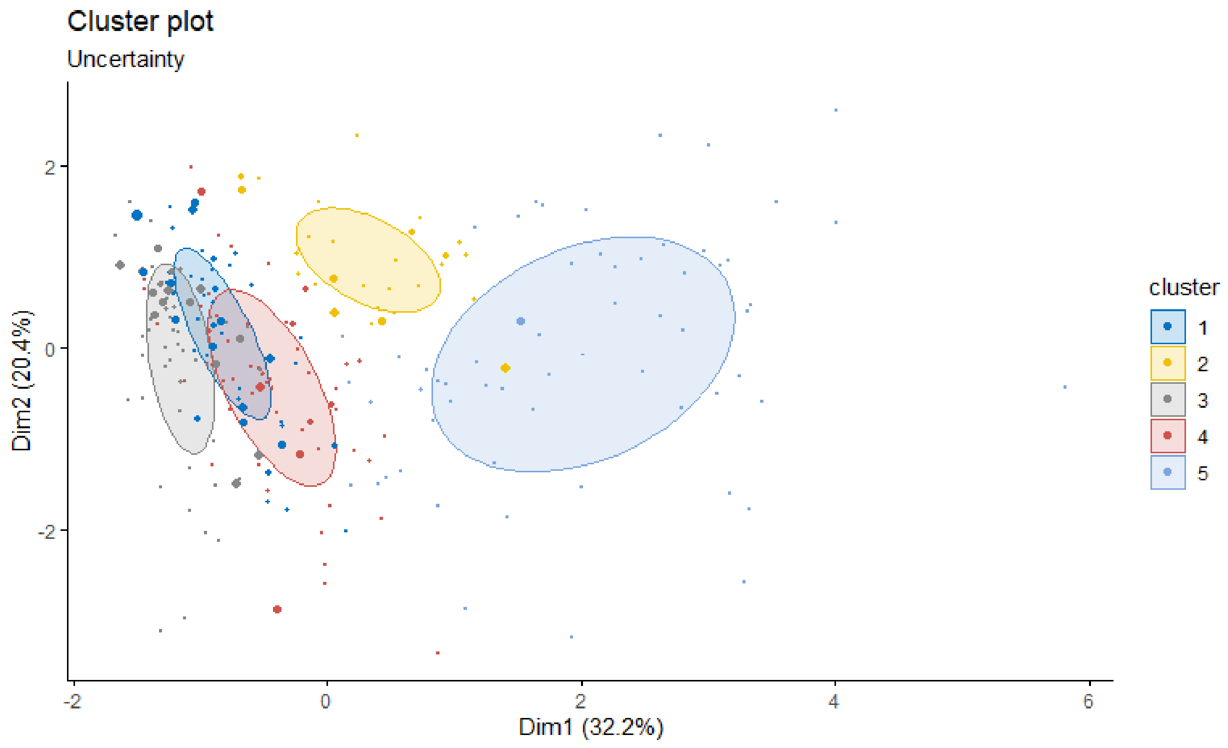
The 15th song, “As It was – Harry Styles” has been soft-assigned to the first component (i.e. cluster) of the mixture distribution with a fitted posterior probability of 100%. That is, the observation was confidently – with zero uncertainty – classified into cluster 5.

Now, we can show the data in the principal component Space, in function of the classification and of the uncertainty.

```
>fviz_mclust(model, "classification", geom = "point", pointsize = 1.5, palette = "
jco")
```



```
> fviz_mclust(model, "uncertainty", palette = "jco")
```



The first plot shows classification results in the first two PCs space, and makes noticeable how clusters 1, 3 and 4 are partially overlapped. It suggests that observations classified in these clusters share similar values in terms of the first two principal components which collectively can explain the 52.6% of the variability in the dataset.

Visualizing the uncertainty results in the first two PCs space, makes us more aware of the presence in data of some points which may have been misclassified, graphically denoted by larger symbols in the plot. By inspecting more thoroughly the data, it turns out that there are two data points, that have an uncertainty level associated to their classification greater than 50%.

```
> cbind(PC[model_output[,"Uncertainty"] > 0.50, 1:2], Uncertainty = model_output[m
odel_output[,"Uncertainty"] > 0.50, 6], Classification = model_output[model_output
[,"Uncertainty"] > 0.50, 7])
             PC1       PC2 Uncertainty Classification
[1,] -1.49629240 1.4637675           1        0.63196
[2,]  0.06233839 0.3858833           2        0.50407
> model_output[model_output[,"Uncertainty"] > 0.50, 1:7]
         pi1     pi2     pi3     pi4     pi5 Classification Uncertainty
[1,] 0.36804 0.00306 0.28402 0.34443 0.00045              1     0.63196
[2,] 0.00000 0.49593 0.00000 0.49245 0.01162              2     0.50407
```

One observation (Amargura – karol G) has been assigned to cluster 1 and most likely corresponds to larger blue point: it is located at the top-left part of the plot. In fact, it has 36% of probability to belong to the first cluster and 34% to belong to the fourth cluster.

In the plot is not so clear where the other one is located, because there's some points with the same dimension but maybe it's the orange point inside the light-blue ellipse. This observation is related to "Escapism – RAYE, 070 Shake". In fact, we see a probability of 49.59% to belong to cluster 2 and a probability of 49.24% of belonging to cluster 4.

These are the most misclassified observation.

Libraries

```
> library("moments")
> library("tidyverse")
> library("gamlss")
> library("gamlss.mx")
> library("DataExplorer")
> library("e1071")
> library("cluster")
> library("factoextra")
> library("stats")
> library("hopkins")
> library("NbClust")
> library("fpc")
> library("gridExtra")
> library("grid")
> library("mclust")
> library("ggplot2")
> library("plotrix")
> library("psych")
> library("corrplot")
> library("clustertend")
> library("stats")
> library("clValid")
> library("dplyr")
> library("fclust")
```