UNIVERSITÀ degli STUDI di CATANIA
Master's Degree in
Data Science for Management
A.Y. 2019/2020

Data Analysis and Statistical Learning
"Data Analysis" module
Prof. A. Punzo

REPORT

Ylenia Messina

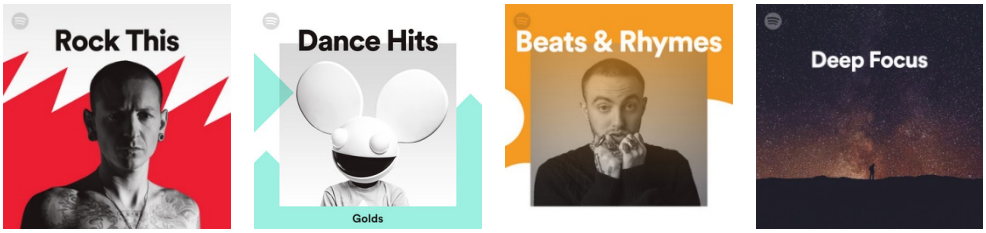
DATASET : Spotify data

The dataset being analyzed was obtained through the Spotify Web API using the 'spotifyr' R package, which allows to easily retrieve artists, songs and playlists information, along with audio features.

spotifyr: https://cran.r-project.org/web/packages/spotifyr/index.html

```
> # COLLECT DATA
> library(spotifyr)
> get_spotify_authorization_code(source("set_token.R"))  # access authorization credentials
>
> playlist_username <- 'Spotify'
> playlist_uris <- c('37i9dQZF1DXcF6B6QPhFDv',           # Rock This
+                    '37i9dQZF1DX0BcQWzuB7ZO',           # Dance Hits
+                    '37i9dQZF1DXcA6dRp8rwj6',           # Beats & Rhymes
+                    '37i9dQZF1DWZeKCadgRdKQ')           # Deep Focus
> playlist_audio_features <- get_playlist_audio_features(playlist_username, playlist_uris)
```

The above code was responsible for the *audio features* data retrieve of the following four playlists of different music genre, selected among the wide range of playlists created by Spotify algorithm. Data were then arranged and reorganized, picking out only a subset of the total several features available.




```
> # ARRANGE DATA
> library(tidyr)
> # unnest "track.artists" variable, getting one row of output for each element of the list
> my_dataset <- unnest(data = playlist_audio_features, cols = track.artists, keep_empty = TRUE)
>
> my_dataset <- data.frame(
+   "song" = my_dataset$track.name,
+   "artist" = my_dataset$name,
+   "playlist" = my_dataset$playlist_name,
+   "acousticness" = my_dataset$acousticness,
+   "danceability" = my_dataset$danceability,
+   "energy" = my_dataset$energy,
+   "loudness" = my_dataset$loudness,
+   "tempo" = my_dataset$tempo,
+   "valence" = my_dataset$valence
+ )
>
> library(tidyverse)
> my_dataset <- my_dataset %>% distinct()                  # remove duplicate rows
> my_dataset <- my_dataset[!duplicated(my_dataset$song),]  # keep only one row per song
>
> # subsetting in order to have 50 tracks from each playlist
> rock <- sample_n(subset(my_dataset, playlist == "Rock This"), 50)
> dance <- sample_n(subset(my_dataset, playlist == "Dance Hits"), 50)
> hiphop <- sample_n(subset(my_dataset, playlist == "Beats & Rhymes"), 50)
> cinematic <- sample_n(subset(my_dataset, playlist == "Deep Focus"), 50)
>
> my_dataset <- rbind(rock, dance, hiphop, cinematic)      # combine dataframes
> my_dataset <- sample_n(my_dataset, size = 200)
>
> write.csv(my_dataset, "spotify_dataset.csv", row.names = FALSE)
```

```
> # MY DATASET
> data <- read.csv("spotify_dataset.csv")
> # convert first column to row name/index
> my_dataset <- data[,-1]
> rownames(my_dataset) <- data[,1]

> head(my_dataset)   # preview
                                artist       playlist acousticness danceability energy loudness   tempo valence
Life Is Good (feat. Drake)       Future Beats & Rhymes      0.07060        0.676  0.609   -5.831 142.037   0.508
Far From Home              Swanky Tunes     Dance Hits      0.01320        0.578  0.939   -3.872 125.071   0.471
Unstoppable                New Politics      Rock This      0.00683        0.545  0.899   -4.276 170.128   0.593
Love Me More               Trippie Redd Beats & Rhymes      0.22000        0.638  0.539   -5.853  83.381   0.247
Renaissance Man                Jay Park Beats & Rhymes      0.05340        0.796  0.726   -6.173 160.011   0.855
Essential Attitudes              Peals     Deep Focus      0.44900        0.444  0.497  -12.601  99.093   0.250

> str(my_dataset)    # internal structure
'data.frame':   200 obs. of  8 variables:
 $ artist      : Factor w/ 184 levels "21 Savage","24kGoldn",..: 55 152 110 165 74 124 121 40 100 168
...
 $ playlist    : Factor w/ 4 levels "Beats & Rhymes",..: 1 2 4 1 1 3 4 4 2 2 ...
 $ acousticness: num  0.0706 0.0132 0.00683 0.22 0.0534 0.449 0.00151 0.00619 0.0404 0.169 ...
 $ danceability: num  0.676 0.578 0.545 0.638 0.796 0.444 0.496 0.547 0.677 0.614 ...
 $ energy      : num  0.609 0.939 0.899 0.539 0.726 0.497 0.88 0.922 0.744 0.922 ...
 $ loudness    : num  -5.83 -3.87 -4.28 -5.85 -6.17 ...
 $ tempo       : num  142 125.1 170.1 83.4 160 ...
 $ valence     : num  0.508 0.471 0.593 0.247 0.855 0.25 0.148 0.844 0.631 0.615 ...

> my_dataset <- my_dataset[,-c(1,2)] # drop non-numeric variables
```

The final '`my_dataset`', that will be used in the analysis, contains a total of 200 observations (each row represents a song) and 6 numeric variables. Here is a brief description of each numeric variable, as provided by the Spotify API documentation[1]:

`acousticness`: A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.

`danceability`: Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.

`energy`: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.

`loudness`: The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.

`tempo`: The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.

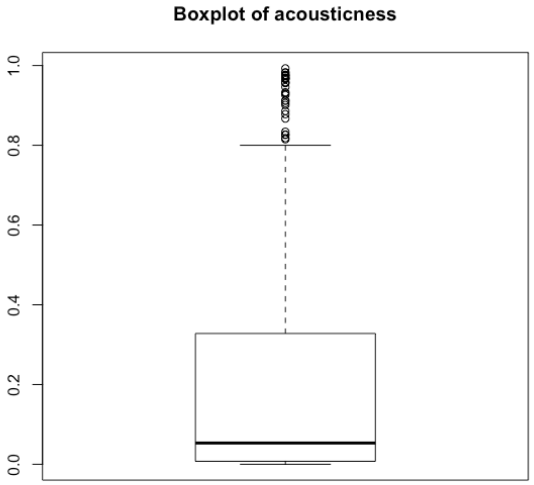
`valence`: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

[1] https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/

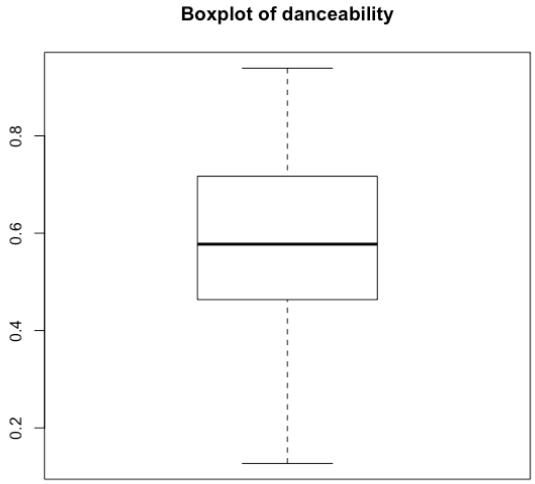
UNIVARIATE ANALYSIS

```
> # Main statistical indices of each variable
> summary(my_dataset)
```

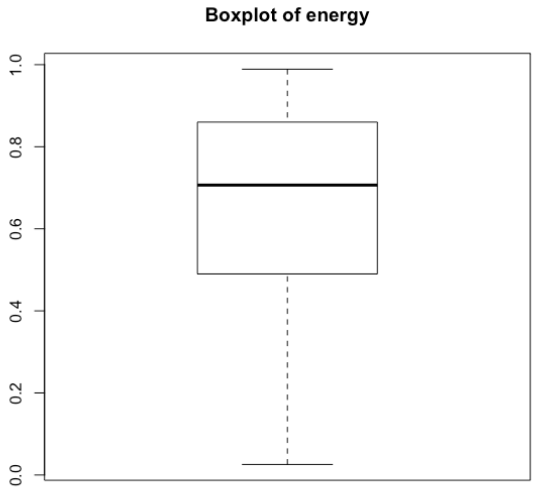
```
  acousticness
Min.   :0.0000051
1st Qu.:0.0076275
Median :0.0532500
Mean   :0.2382166
3rd Qu.:0.3270000
Max.   :0.9930000
```




```
 danceability
Min.   :0.1270
1st Qu.:0.4637
Median :0.5775
Mean   :0.5697
3rd Qu.:0.7170
Max.   :0.9390
```

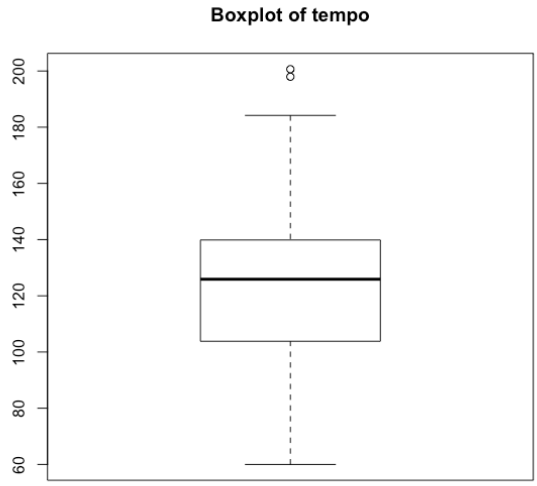



```
    energy
Min.   :0.0257
1st Qu.:0.4935
Median :0.7065
Mean   :0.6307
3rd Qu.:0.8600
Max.   :0.9890
```

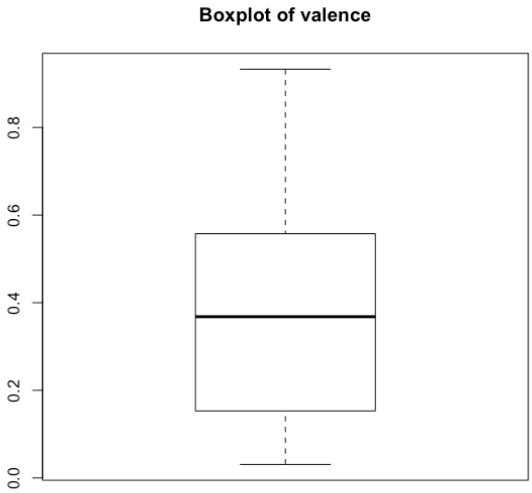


```
    loudness
Min.   :-27.828
1st Qu.: -9.047
Median : -6.175
Mean   : -8.449
3rd Qu.: -4.129
Max.   : -0.818
```


```
     tempo
Min.   : 59.99
1st Qu.:104.24
Median :125.90
Mean   :124.36
3rd Qu.:139.86
Max.   :200.64
```




```
    valence
Min.   :0.0308
1st Qu.:0.1530
Median :0.3680
Mean   :0.3751
3rd Qu.:0.5557
Max.   :0.9330
```



The six features are all continuous quantitative variables, but based on different scale measures: e.g. `loudness` is measured in decibels (dB), `tempo` is measured in beats per minute (BPM) etc. In particular:

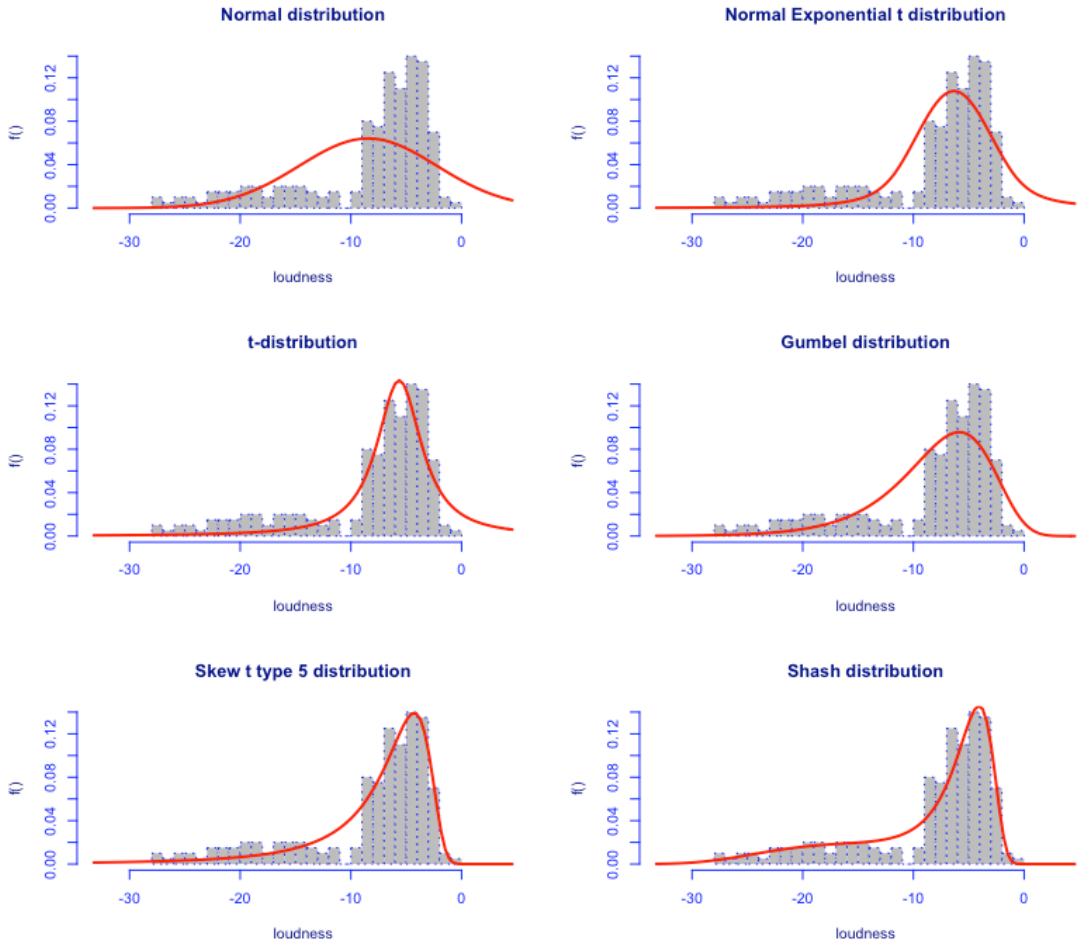
- acousticness, danceability, energy and valence range in [0, 1];
- loudness typically ranges in [-60, 0], so it is supported on the whole real line;
- tempo takes positive values that potentially tend to infinite.

Hereinafter, only the three variables “loudness”, “valence” and “tempo” – which have different supports – have been individually taken as argument in the `histDist()` function (in *gamlss r* package) in order to determine the theoretical distribution that best fits data.

```
> # FIT UNIVARIATE MODELS
> library(gamlss)
```

Loudness

```
> # Fit real-valued continuous distributions models on "loudness"
```




```
> AIC(fit.NO,fit.NET,fit.TF,fit.GU,fit.ST5, fit.SHASH)
          df      AIC
fit.SHASH  4 1136.975
fit.ST5    4 1166.849
fit.GU     2 1212.264
fit.TF     3 1250.520
fit.NET    2 1276.401
fit.NO     2 1302.876

# Defining a function to extract both logLik & sbc from an object of class ‘gamlss’
> fit.info <- function(distribution, parameter) {
+   cbind(logLik(distribution), distribution$sbc)
+ }

> fit.criteria <- rbind(fit.info(fit.NO), fit.info(fit.NET), fit.info(fit.TF),
+                       fit.info(fit.GU), fit.info(fit.ST5), fit.info(fit.SHASH))
> colnames(fit.criteria) <- c("logLik", "BIC")
> row.names(fit.criteria) <- c("fit.NO","fit.NET","fit.TF","fit.GU","fit.ST5", "fit.SHASH")
> fit.criteria
             logLik      BIC
fit.NO    -649.4379 1309.472
fit.NET   -636.2007 1293.595
```

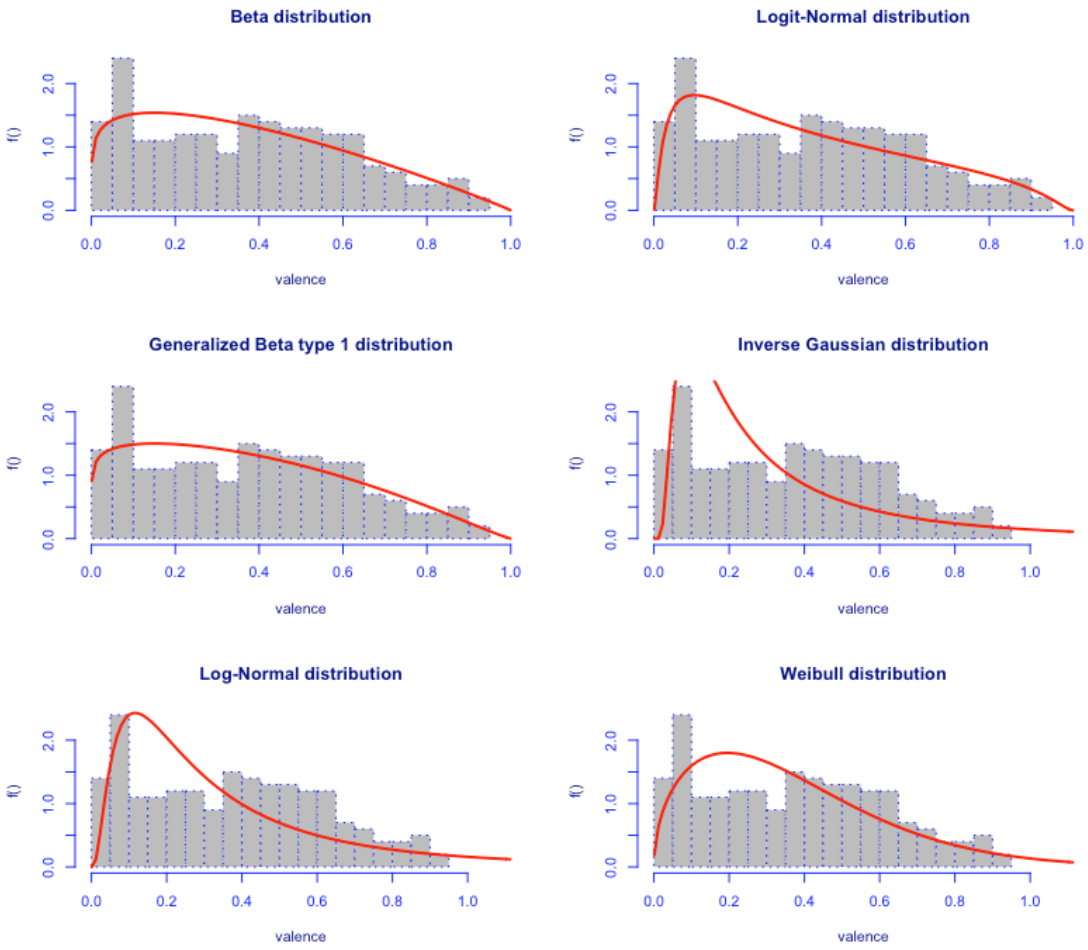
```
fit.TF    -622.2598 1260.415
fit.GU    -604.1319 1218.860
fit.ST5   -579.4244 1180.042
fit.SHASH -564.4874 1150.168
```

Among all the fitted models for real-valued continuous distributions, according to the Akaike's “An Information Criterion” (AIC), as well as to the log-Likelihood and Bayesian Information Criterion (BIC) results, the *Shash distribution model* is the one that better fits ‘loudness’ distribution. It’s then reasonable to assume that ‘loudness’ data were generated by a Shash distribution, characterized by four parameters whose maximum likelihood estimations are the following:

```
> fitted(fit.SHASH, "mu")[1]    # ML estimated 1st parameter
[1] -6.268791
> fitted(fit.SHASH, "sigma")[1] # ML estimated 2nd parameter
[1] 72840.56
> fitted(fit.SHASH, "nu")[1]    # ML estimated 3rd parameter
[1] 5359.652
> fitted(fit.SHASH, "tau")[1]   # ML estimated 4th parameter
[1] 28095.03
```

Valence

```
> # Fit continuous distributions in [0,1] & in the real positive line on "valence"
```




```
> AIC(fit.BE,fit.LOGITNO,fit.GB1,fit.IG,fit.LOGNO,fit.WEI)
            df       AIC
fit.LOGITNO  2 -53.32478
fit.BE       2 -49.99146
fit.GB1      4 -46.24458
fit.WEI      2 -26.96863
fit.LOGNO    2  19.46137
```

```
fit.IG       2  34.63518
```

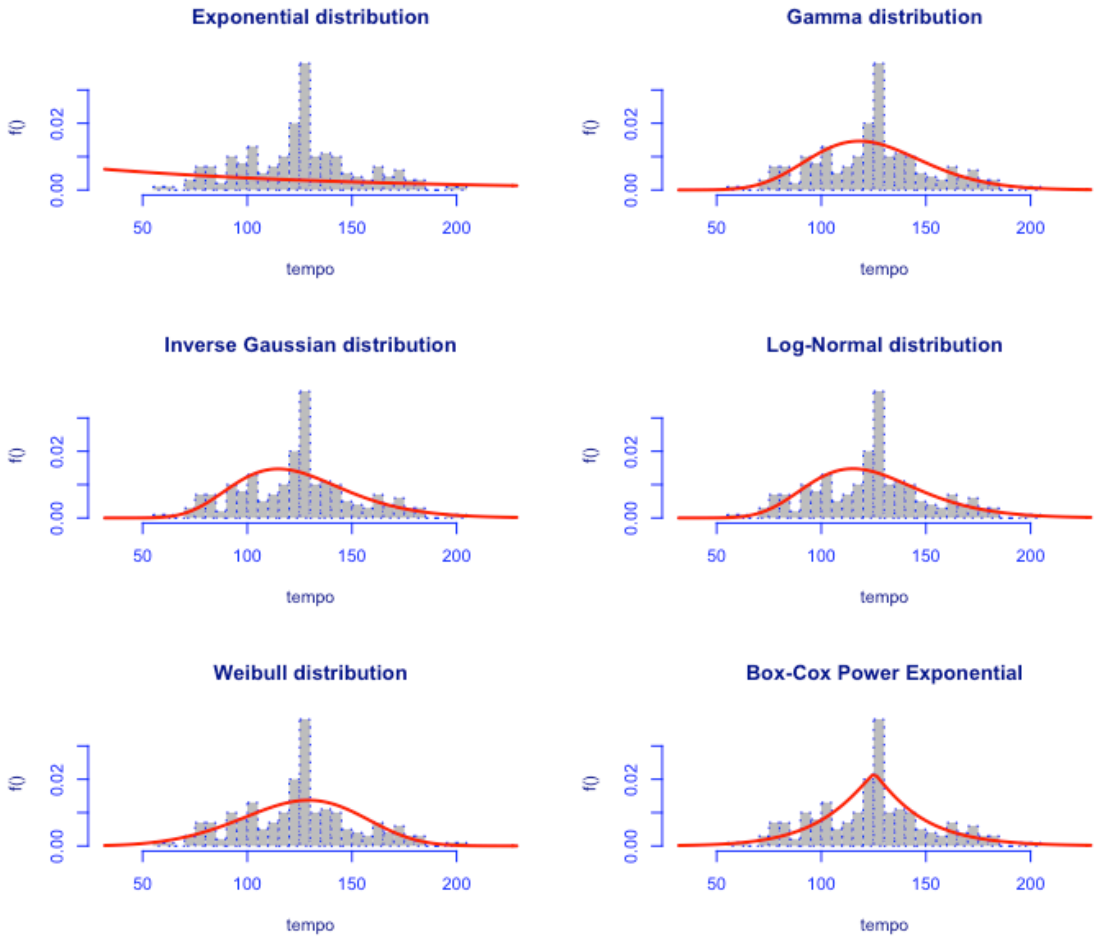
```
> fit.criteria <- rbind(fit.info(fit.BE), fit.info(fit.LOGITNO), fit.info(fit.GB1),
+                       fit.info(fit.IG), fit.info(fit.LOGNO), fit.info(fit.WEI))
> colnames(fit.criteria) <- c("logLik", "BIC")
> row.names(fit.criteria) <- c("fit.BE","fit.LOGITNO","fit.GB1","fit.IG","fit.LOGNO","fit.WEI")
> fit.criteria
               logLik       BIC
fit.BE       26.995732 -43.39483
fit.LOGITNO  28.662391 -46.72815
fit.GB1      27.122290 -33.05131
fit.IG      -15.317592  41.23182
fit.LOGNO    -7.730683  26.05800
fit.WEI      15.484315 -20.37200
```

Among the fitted models, according to the Akaike's “An Information Criterion” (AIC), as well as to the log-Likelihood and Bayesian Information Criterion (BIC) results, the *Logit-Normal distribution model* is the one that better fits ‘valence’ distribution. It’s then reasonable to assume that ‘valence’ data were generated by a Logit-Normal distribution, characterized by two parameters whose maximum likelihood estimations are the following:

```
> fitted(fit.LOGITNO, "mu")[1]    # ML estimated 1st parameter
[1] 0.3261096
> fitted(fit.LOGITNO, "sigma")[1] # ML estimated 2nd parameter
[1] 1.37023
```

Tempo

```
> # Fit continuous distributions in the real positive line on "tempo"
```



```
> AIC(fit.EXP,fit.GA,fit.IG,fit.LOGNO,fit.WEI,fit.BCPE)
          df      AIC
fit.BCPE   4 1892.202
fit.GA     2 1894.863
fit.LOGNO  2 1900.414
fit.WEI    2 1900.625
fit.IG     2 1900.772
fit.EXP    1 2331.272
```

```
> fit.criteria <- rbind(fit.info(fit.EXP), fit.info(fit.GA), fit.info(fit.IG),
+                       fit.info(fit.LOGNO), fit.info(fit.WEI), fit.info(fit.BCPE)
> colnames(fit.criteria) <- c("logLik", "BIC")
> row.names(fit.criteria) <- c("fit.EXP","fit.GA","fit.IG","fit.LOGNO","fit.WEI", “fit.BCPE”)
> fit.criteria
              logLik      BIC
fit.EXP   -1164.6362 2334.571
fit.GA     -945.4316 1901.460
fit.IG     -948.3862 1907.369
fit.LOGNO  -948.2070 1907.011
fit.WEI    -948.3123 1907.221
fit.BCPE   -942.1008 1905.395
```

Among the fitted models belonging to the family of real positive continuous distributions, according to the Akaike's “An Information Criterion” (AIC), as well as to the log-Likelihood results, the *Box-Cox Power Exponential distribution model* is the one that better fits ‘tempo’ distribution.
On the other hand though, BIC is minimized by the fitting with *Gamma distribution*, which is understandable due to the fact that BIC penalizes model complexity more heavily, so that it has a larger chance than AIC – for any given *n* – of choosing too small a model.

```
> LR.test(fit.GA,fit.BCPE)
 Likelihood Ratio Test for nested GAMLSS models.
 (No check whether the models are nested is performed).

       Null model: deviance= 1890.863 with  2 deg. of freedom
 Altenative model: deviance= 1884.202 with  4 deg. of freedom

 LRT = 6.661573 with 2 deg. of freedom and p-value= 0.03576496
```

Performing the likelihood ratio test for the two fitted nested models, the low p-value indicates Null Hypothesis – according to which ‘tempo’ data were generated by a Gamma distribution – is rejectable. It is instead reasonable to assume that, with respect to the two fitted nested models, ‘tempo’ data were more likely generated by a Box-Cox Power Exponential distribution. However it must be taken into account that the p-value is not so far from the level of significance *alpha* = 0.05, for which there is a risk of mistakenly reject Gamma distribution when it could be actually the “true” model.

Anyway we can assume that ‘tempo’ data were generated by a Box-Cox Power Exponential distribution, characterized by four parameters whose maximum likelihood estimations are the following:

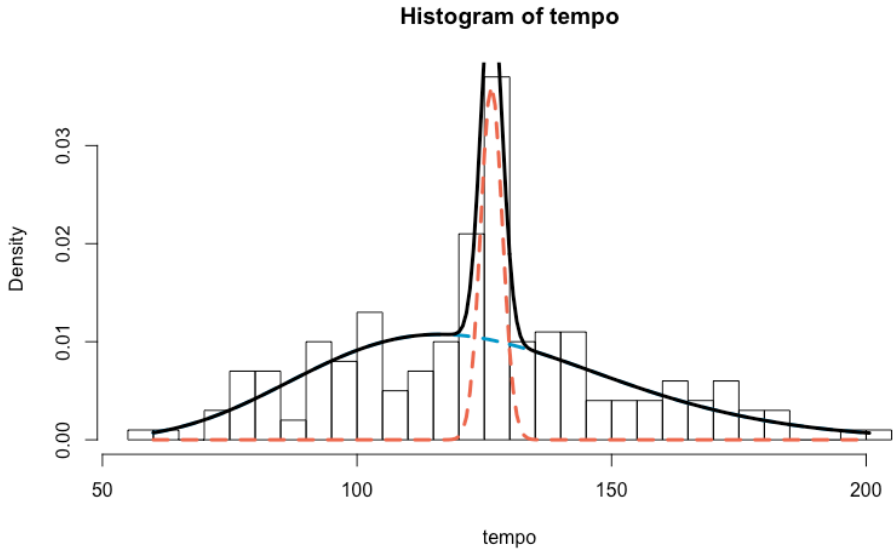
```
> fitted(fit.BCPE, "mu")[1]    # ML estimated 1st parameter
[1] 125.2646
> fitted(fit.BCPE, "sigma")[1] # ML estimated 2nd parameter
[1] 0.227205
> fitted(fit.BCPE, "nu")[1]    # ML estimated 3rd parameter
[1] 0.7067616
> fitted(fit.BCPE, "tau")[1]   # ML estimated 4th parameter
[1] 1.121889
```

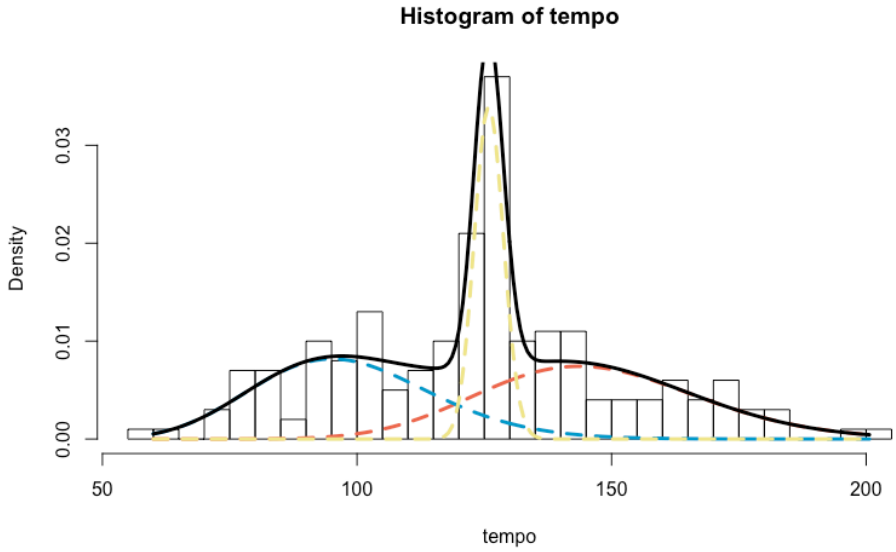
Despite AIC and Likelihood-ratio test results, it's visually noticeable from the histogram above, how it still seems not to fit very well 'tempo' data, losing some of its information. Therefore, it is worth checking whether 'tempo' data could be better defined by a mixture of distribution models.

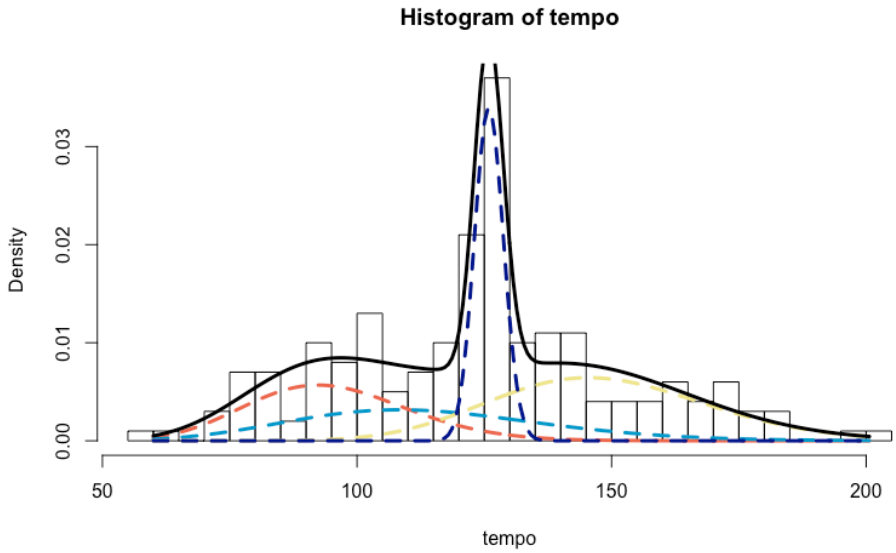
```
> # FIT MIXTURE DISTRIBUTIONS ON 'tempo'
> library(gamlss.mx)
```

Recalling that *Gamma distribution* model had the best results in terms of BIC, we can imagine 'tempo' data to come from a mixture of gamma distributions. Here are the fitting attempts with mixtures of *K* gamma distributions.

```
> XX <- my_dataset$tempo
> fit.BCPE.2 <- gamlssMXfits(n = 5, XX~1, family = BCPE, K = 2) # mixture of 2 distributions
> fit.BCPE.3 <- gamlssMXfits(n = 5, XX~1, family = BCPE, K = 3) # mixture of 3 distributions
> fit.BCPE.4 <- gamlssMXfits(n = 5, XX~1, family = BCPE, K = 4) # mixture of 4 distributions
```








```
> fit.GA.2$sbc
[1] 1873.931
> fit.GA.3$sbc
[1] 1883.676
> fit.GA.4$sbc
[1] 1899.58
```

Comparing the fitting with univariate Gamma distribution model versus the fitting with mixtures of K (= 2,..., 4) Gamma distributions, according to BIC, the best one seems to be the mixture of two Gamma distributions. Among the fitted mixtures in fact, the mixture of two Gamma distributions minimizes BIC value, so it is more likely to be the true model from which 'tempo' data were generated. Also, it is reasonable to suppose the presence of two latent clusters underlying this variable.

PRINCIPAL COMPONENT ANALYSIS

The reference dataset would require spending a not so short amount of time in order to perform a multivariate analysis, since it would imply *d(d-1)/2 = 6(6-1)/2 = 15* scatterplots to be examined.

Therefore, PCA provides a tool to find a low-dimensional representation of '*my_dataset*', allowing to summarize it with a smaller number of representative variables that still collectively explain most of the variability in the original set defined by 6 variables.

Univariate analysis already made us aware of the presence of different scale measures for the variables, resulting in different values of mean and variance which thus are not reliable for comparison purposes, as it can be resumed below:

```
> round(apply(my_dataset, 2, mean), 4)
acousticness      danceability      energy      loudness       tempo      valence
      0.2382            0.5697      0.6307       -8.4493    124.3601       0.3751
> round(apply(my_dataset, 2, var), 4)
acousticness      danceability      energy      loudness       tempo      valence
      0.1136            0.0353      0.0728       38.9202     746.8366       0.0586
```

As said previously, 'loudness' and 'tempo' have different scales with respect to the other variables so that 'tempo' data seems to be the more variable, which may also be true, but it is strongly unreliable since the difference with the variance values of the other features is very big.
Since PCA is sensitive to scaling and it is undesirable for the *principal components* obtained to depend on the arbitrary choice of scaling individual variables, each variable was scaled to have standard deviation equal to 1.

```
> df.scaled <- scale(my_dataset)
> head(round(df.scaled, digits = 3))
                           acousticness danceability energy loudness  tempo valence
Life Is Good (feat. Drake)       -0.497        0.566 -0.080    0.420  0.647   0.549
Far From Home                    -0.668        0.044  1.143    0.734  0.026   0.396
Unstoppable                      -0.687       -0.132  0.995    0.669  1.675   0.900
Love Me More                     -0.054        0.364 -0.340    0.416 -1.500  -0.529
Renaissance Man                  -0.548        1.205  0.353    0.365  1.305   1.982
Essential Attitudes               0.625       -0.669 -0.496   -0.665 -0.925  -0.517
```

Principal components of the standardized variables can then be obtained by doing the eigen decomposition of the sample correlation matrix.

```
> cor_matrix <- cor(df.scaled)            # correlation matrix
> head(round(cor_matrix, digits = 3))
              acousticness danceability energy loudness  tempo valence
acousticness         1.000       -0.488 -0.865   -0.871 -0.289  -0.573
danceability        -0.488        1.000  0.441    0.503  0.130   0.611
energy              -0.865        0.441  1.000    0.921  0.243   0.622
loudness            -0.871        0.503  0.921    1.000  0.269   0.580
tempo               -0.289        0.130  0.243    0.269  1.000   0.207
valence             -0.573        0.611  0.622    0.580  0.207   1.000

> eigen <- eigen(cor_matrix)              # compute eigenvectors & eigenvalues
> phi <- eigen$vectors                    # extract loadings & store them in 'phi' matrix
```

```
> head(round(phi, digits = 2))
       [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
[1,]  0.472  0.043  0.287 -0.160  0.817  0.010
[2,] -0.349  0.338  0.644  0.575  0.069  0.117
[3,] -0.477  0.001 -0.354 -0.083  0.375  0.707
[4,] -0.480 -0.012 -0.310  0.151  0.424 -0.686
[5,] -0.187 -0.919  0.342  0.034  0.043  0.027
[6,] -0.403  0.197  0.407 -0.783 -0.072 -0.122

> phi <- -phi                              # use positive-pointing vector
> row.names(phi) <- colnames(my_dataset)   # rename rows in phi matrix
> colnames(phi) <- c("PC1", "PC2", "PC3",  # rename columns in phi matrix
+                    "PC4", "PC5", "PC6")
> head(round(phi, digits = 2))
                PC1    PC2    PC3    PC4    PC5    PC6
acousticness -0.472 -0.043 -0.287  0.160 -0.817 -0.010
danceability  0.349 -0.338 -0.644 -0.575 -0.069 -0.117
energy        0.477 -0.001  0.354  0.083 -0.375 -0.707
loudness      0.480  0.012  0.310 -0.151 -0.424  0.686
tempo         0.187  0.919 -0.342 -0.034 -0.043 -0.027
valence       0.403 -0.197 -0.407  0.783  0.072  0.122
```

By briefly examining the loadings we note that:

- the first loading vector ϕ_1, considering the absolute value, places equal weight (`0.47`-`0.48`) on loudness, energy and acousticness, and slightly less weight on valence and danceability (respectively `0.40` and `0.35`). Much less weight is instead placed on tempo (`0.19`). This seems to suggest that this component (`PC1`) mostly corresponds to a measure of the first three audio features mentioned, and likely not a measure of tempo.
- the second loading vector ϕ_2 places a really heavy weight (`0.92`) on tempo and very much less weight on all the other variables, indicating how `PC2` definitely is a measure of the speed or pace of a track.
- the third loading vector ϕ_3 places a quite considerable weight on danceability (`0.64`) but also on valence. Less weight on the remaining ones.
- ...

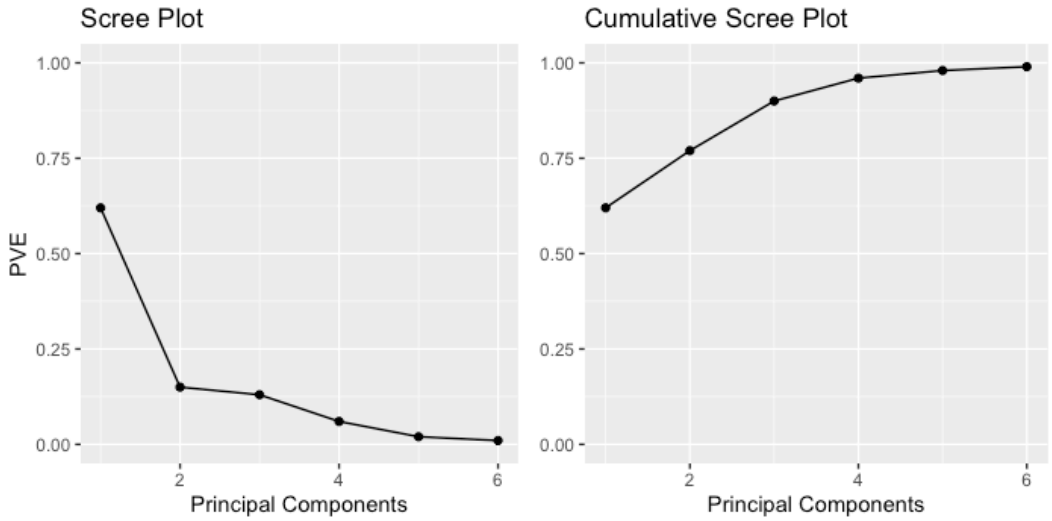
At this point of the PCA process, the appropriate number *q (< 6)* of *principal components* must be selected. The choice can be based on the *cumulative proportion of variance explained* criteria, as well as on the *scree plot* criteria.

```
> PVE <- eigen$values/sum(eigen$values)
> (PVE <- round(PVE, 2))
[1] 0.62 0.15 0.13 0.06 0.02 0.01
> cumsum(PVE)
[1] 0.62 0.77 0.90 0.96 0.98 0.99
```

Computing the proportion of variance explained by each eigenvector, it results that:
- The first principal component explains the **62**% of the variability;
- The second principal component explains the **15**% of the variability.
- The third principal component explains the **13**% of the variability.

It follows that, together, `PC1` and `PC2` are able to explain the **77**% of variability, which is not a poor proportion, but would imply a 23% loss of information. Obviously, the situation in which the **90**% of the variance is explained at the cost of losing only the 10%, would be a better compromise. Therefore, although the choice of only two

principal components could also be reasonable, it is certainly preferable reducing the dimensionality of ‘my_dataset’ selecting the first three principal components.
This conclusion is also reinforced by visualizing the *scree plot.* In fact, looking at the scree plot on the left panel below, it is clearly noticeable the significant jump that occurs in correspondence of the value $m = 2$, which could be easily identified in the first place as the “elbow point”. However, the curve does not immediately tend to flatten itself, which instead seems to happen after $m = 3$.



Therefore, recalling the first three PCs loadings 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:

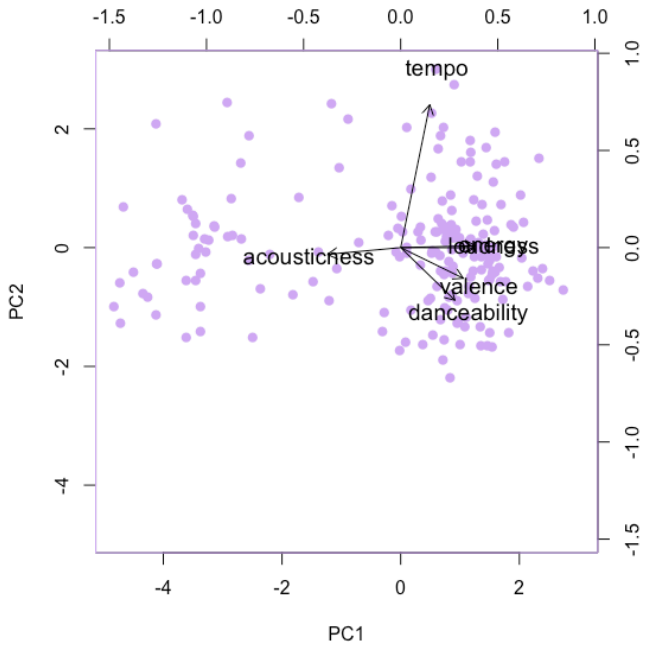
```
> head(round(phi[,1:3], digits = 2))
               PC1   PC2   PC3
acousticness -0.47 -0.04 -0.29
danceability  0.35 -0.34 -0.64
energy        0.48  0.00  0.35
loudness      0.48  0.01  0.31
tempo         0.19  0.92 -0.34
valence       0.40 -0.20 -0.41
```

- PC1 as a measure of the level of energy and loudness of a track, as well as of the presence in it of electronic sounds (as opposite of acustic);
- PC2 as a measure of tempo;
- PC3 as a measure of no-situability of the track for dancing, and of the level of its negative valence, intended in terms of unpleasant emotions – such as sadness, angriness ecc... – conveyed to listeners.

Despite the choice of $q = 3$ principal components and since the first 2 PCs explain anyway almost a 80% proportion of variance, the results achieved can be further and more clearly explored by visualizing the biplot where both the $n = 200$ units and the $d = 6$ variables of ‘my_dataset’ are graphically displayed on a bidimensional Cartesian plane spanned by the first two PCs.

Obviously, taking in only two principal components implies lowering the level of correlation to be considered significant (let say ≥ 0.35), with the consequence of a slightly different interpretation of the *principal components*, mostly regarding PC1.

```
> head(round(phi[,1:2], digits = 2))
               PC1   PC2
acousticness -0.47 -0.04
danceability  0.35 -0.34
energy        0.48  0.00
loudness      0.48  0.01
tempo         0.19  0.92
valence       0.40 -0.20
```



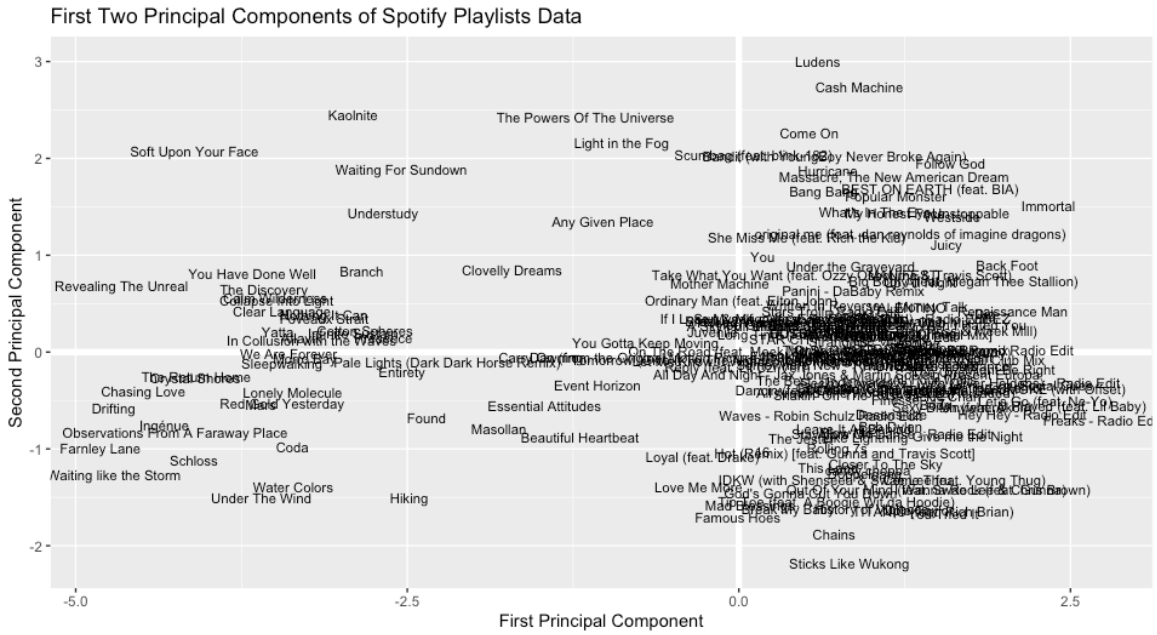
The biplot graphically summarizes the outcome of PCA directly performed on ‘my_dataset’ via `princomp()` r function, which uses the spectral decomposition approach (examining the covariance/correlations between variables) as well as was done so far in separate steps.
First of all, it is clearly noticeable how *energy* and *loudness* are two features strongly and positively correlated since the arrows representing them are basically overlapped, meaning that when energy increases/decreases there is also an increase/decrease of the same entity in loudness (and viceversa). Both of them are positively associated with `PC1`, as we can see that both the (energy and loudness) arrows are parallel to the bottom axis representing `PC1`, and definitely orthogonal with respect to `PC2`. Roughly same situation with regards to *acousticness*, but with the main difference that this feature is negatively associated with `PC1`. The two features *danceability* and *valence* – that in a three-dimensional hyperplane would be mostly associated with `PC3` – in the bidimensional space of biplot are more likely to be positively associated with `PC1`.
It is then observable how acousticness, energy and loudness are pretty much uncorrelated with ‘tempo’, forming with it an angle very close to 90°, while danceability and valence seem to be a little bit more correlated with it, but going in opposite direction. Finally, the *tempo* feature is positively associated with `PC2`.
In conclusion the first principal component can be viewed as a measure of quality of a track’s sound, determined by both objective technical aspects such as the level of acousticness and loudness, and more perceptual aspects such as the level of energy, danceability and musical positiveness conveyed by the track. In particular, first principal component increases both with:
- increasing of energy, loudness, danceability and valence;
- decreasing of acousticness;
suggesting that songs with high values of `PC1` tend to have high values of energy, loudness, danceability and valence but low values of acousticness, and vice versa.
On the other hand, the second principal component primarily measures the speed and pace of a track, so that songs with high values of `PC2` tend to be very fast while songs with low values of `PC2` are slow.

Computing the *principal component scores* – which are, geometrically speaking, the projections of the *200* data points along the directions defined by loading vectors – it is possible to make some considerations about each specific song of the sample dataset, with reference to the original variables.

```
> PC1 <- df.scaled %*% phi[,1]       # 1st principal component scores for each song
> PC2 <- df.scaled %*% phi[,2]       # 2nd principal component scores for each song
> PC3 <- df.scaled %*% phi[,3]       # 3rd principal component scores for each song
> PC <- data.frame(PC1, PC2, PC3)    # data frame with principal component scores
> head(round(PC, digits = 3))
                            PC1    PC2    PC3
Life Is Good (feat. Drake)  0.938  0.321 -0.565
Far From Home               1.392 -0.033  0.624
Unstoppable                 1.749  1.443 -0.098
Love Me More               -0.303 -1.389  0.518
Renaissance Man             2.065  0.428 -1.633
Essential Attitudes        -1.466 -0.556  0.396
```

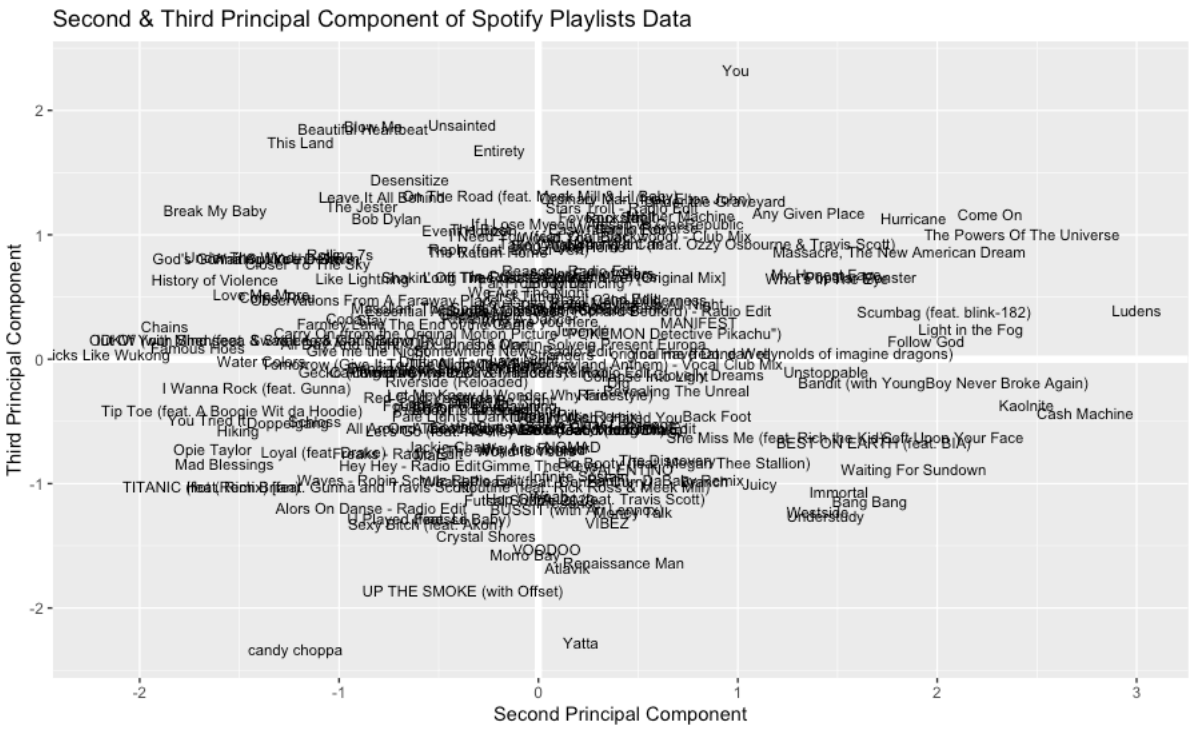
With concern to the signs of the scores and focusing only on the features mainly explained by each principal component (since their corresponding "weights" are the heavier), it can be said for example for "`Life Is Good`" that its:

- `PC1` score roughly indicates a above-average level of both energy and loudness, and a below-average of acousticness;
- `PC2` score roughly indicates a above-average tempo;
- `PC3` score, reminding that `PC3` is negatively influenced by both danceability and valence, roughly indicates a above-average level of these two variables, suggesting that "Life is Good" is actually a quite danceable song which also conveys positive feelings.



Considering the *first principal component* (x-axis), songs such as "Freaks – Radio Edit" and "Immortal" seem to be very dynamic and loud tracks, with a more electronic sound, as opposed to songs such as "Waiting like the Storm" and "Revealing the Unreal" which are very calm and quiet tracks, involving a more acoustic sound.

Considering the *second principal component* (y-axis), songs such as “Ludens” and “Cash Machine” are very fast tracks in terms of beats per minute, as opposed to songs such as “Sticks Like Wukong” and “Chains” which are tracks with a very slow rhythm. Also, songs close to the origin such as “You Gotta Keep Moving” or “Ordinary Man” and a great amount of other songs from the sample dataset, are close to average in both categories.



Considering the *third principal component* (y-axis in this plot), songs such as “You” and “Unsainted” are likely to be not very danceable songs and also associated with negative feelings, as opposed to songs such as “candy choppa” and “Yatta” which seem to be happy and danceable tracks.

CLUSTER ANALYSIS

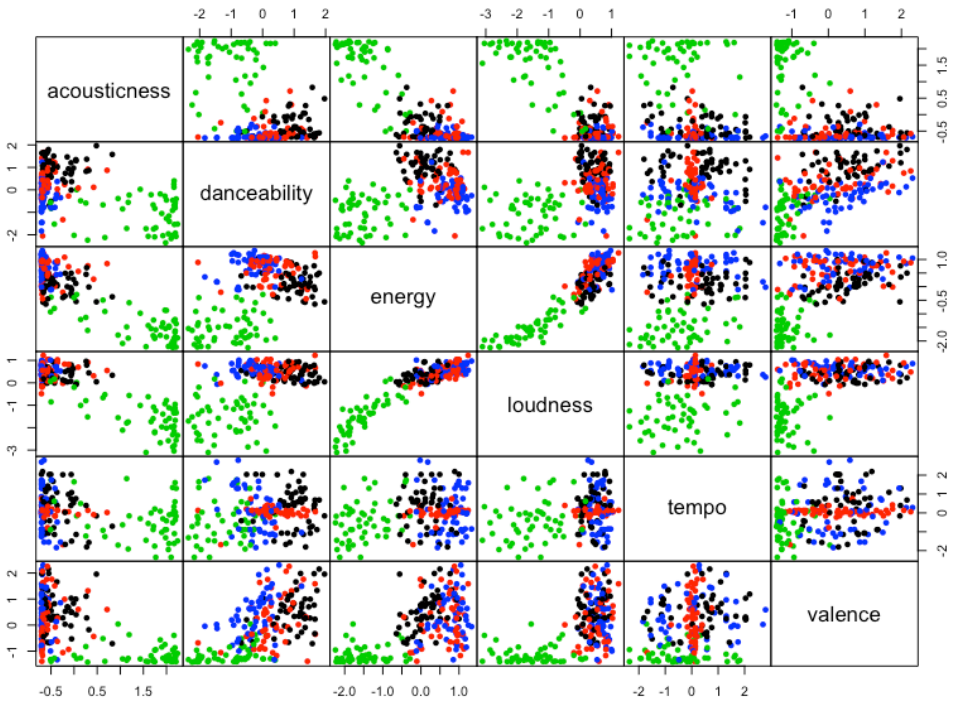
The dataset considered is again the standardized one ‘df.scaled’, computed already within the Principal Component Analysis.

```
> head(round(df.scaled, digits = 3))
                          acousticness danceability energy loudness  tempo valence
Life Is Good (feat. Drake)       -0.497        0.566 -0.080    0.420  0.647   0.549
Far From Home                    -0.668        0.044  1.143    0.734  0.026   0.396
Unstoppable                      -0.687       -0.132  0.995    0.669  1.675   0.900
Love Me More                     -0.054        0.364 -0.340    0.416 -1.500  -0.529
Renaissance Man                  -0.548        1.205  0.353    0.365  1.305   1.982
Essential Attitudes               0.625       -0.669 -0.496   -0.665 -0.925  -0.517
```

Before applying any clustering algorithm, the first step concerns assessing whether it is actually suitable applying cluster analysis to the reference dataset.
Preliminary considerations can be done visualizing the scatterplots matrix of the original variables, taking in as external information the four levels of the factor variable providing the title playlist the song is listed in. Each playlist is genre-based, that is each playlist collects tracks of the same music genre.

```
> levels(data$playlist)
[1] "Beats & Rhymes"    "Dance Hits"    "Deep Focus"    "Rock This"
> # renaming levels
> levels(data$playlist) <- factor(c("Hip-Hop",         # Playlist: Beats & Rhymes
+                                   "Electro-Dance",   # Playlist: Dance Hits
+                                    "Ambient Music",  # Playlist: Deep Focus
+                                    "Rock"))          # Playlist: Rock This
> table(data$playlist)
      Hip-Hop    Electro-Dance    Ambient Music     Rock
           50               50               50       50
> genre <- as.numeric(data$playlist)  # coerces labels of genre vector into objects of numeric type
```

Therefore, we can visually check whether the external information reveals a possible underlying clustering based on the music genre.



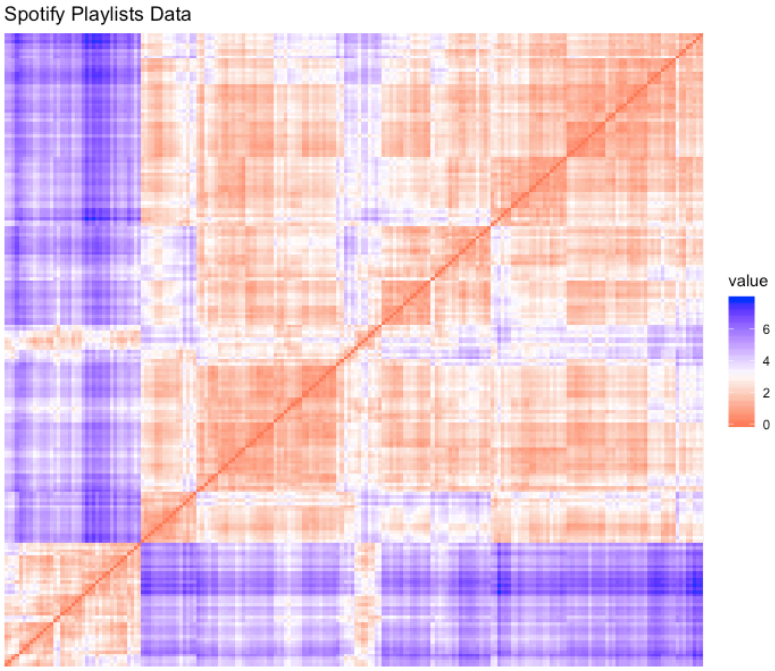
Apparently, the genre more or less seems to create a cluster structure, or at least there seems to be a clear separation between the group represented by green points and all the other points.

In order to properly evaluate clustering tendency/feasibility we can apply the two main methods.

```
> # Hopkins statistic
> library(clustertend)
>
> set.seed(123)
> hopkins(df.scaled,
+         n = nrow(df.scaled)-1, # number of points selected from data (all rows minus 1)
+         header = TRUE)         # 1st column is not deleted in the calculation
$H
[1] 0.2512639
```

The Hopkins statistic computed considering a number of rows $m = n - 1 = 199$, returns a value equal to **0.25**, which is lower than the significance level 0.5. Consequently, we can reject the null hypothesis under which the data is uniformly distributed and state that the dataset is indeed clusterable.

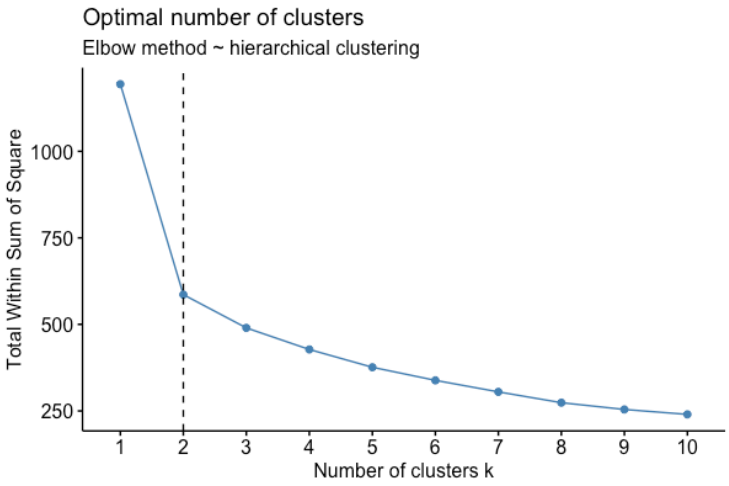
The existence of a potential cluster structure in the standardized dataset, seems to be also suggested by the output of the Visual Assessment of Clustering Tendency algorithm.

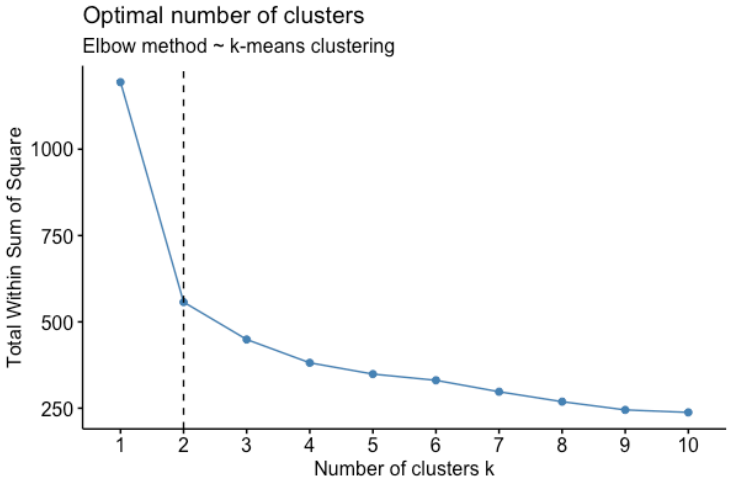


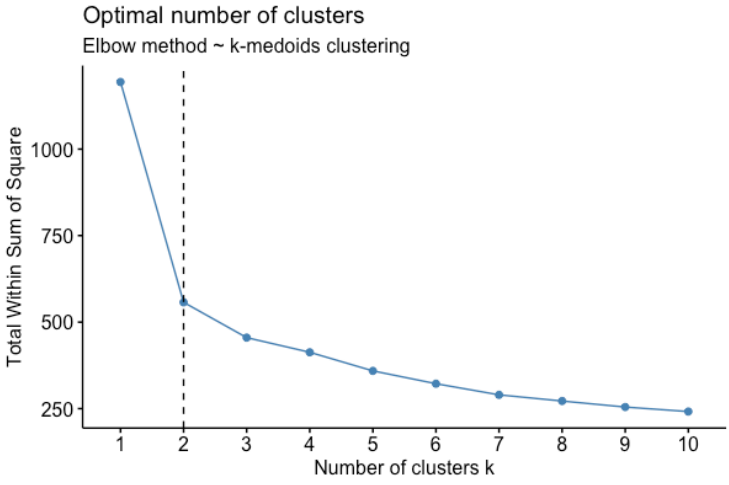
In fact, from the ordered dissimilarity image displayed, at least two square shaped red blocks can be detected along the diagonal – one small block on the left corner in the bottom, and a bigger and less definite one on the right side – denoting high similarity (i.e. low dissimilarity) between the close units located within each block and high dissimilarity between units of the two blocks.

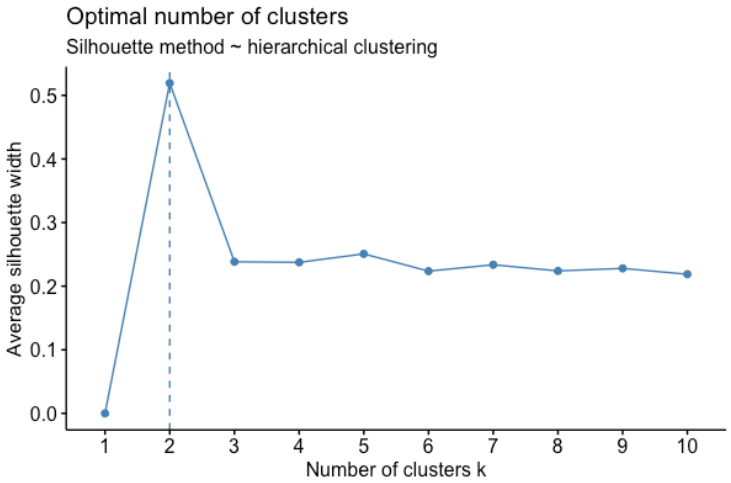
Assessed the feasibility of Cluster Analysis, a subsequent and fundamental issue refers to the determination of the optimal number of clusters K to consider when applying any of the clustering algorithms.

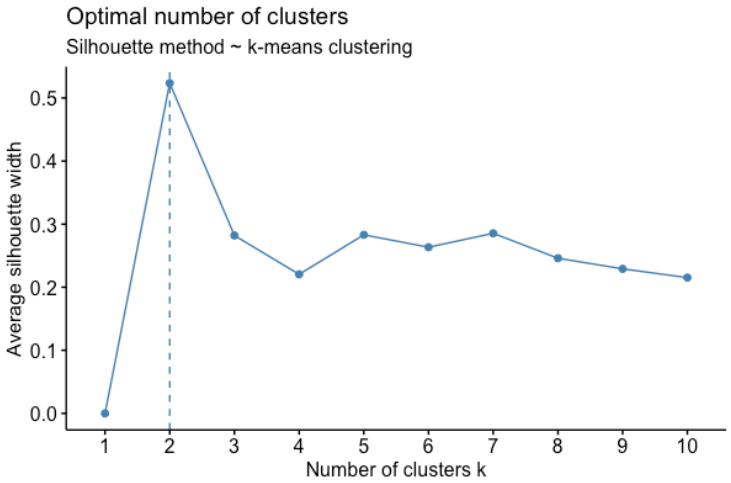
For each clustering procedure (hierarchical clustering, k-means and k-medoids), the "elbow", "silhouette" and "gap statistic" methods suggest the following results:

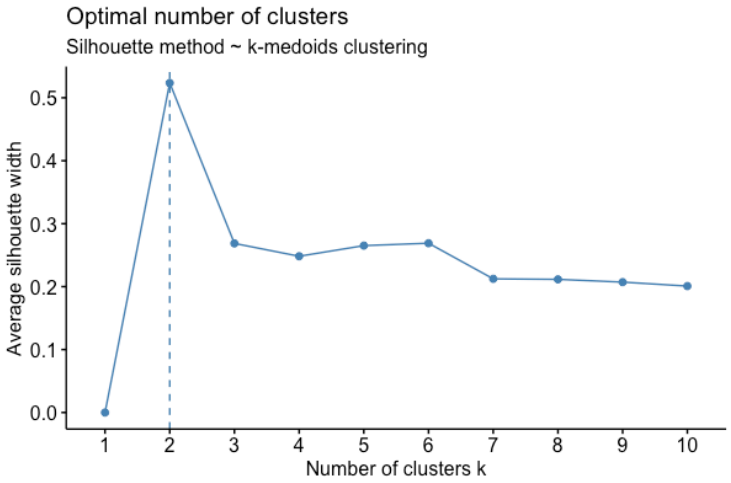


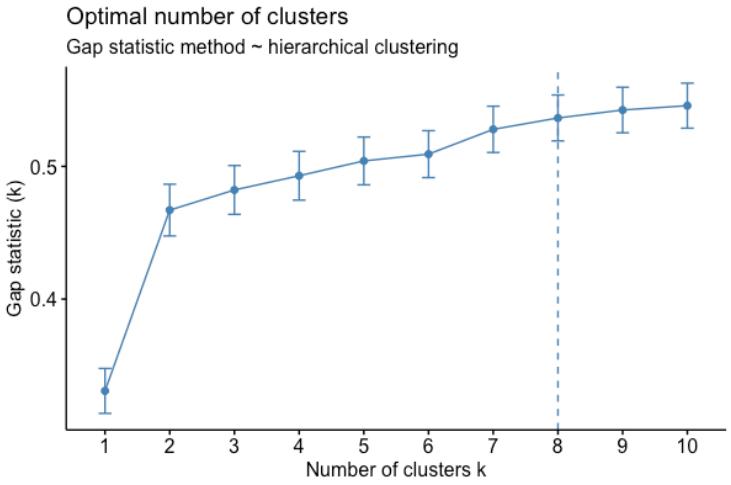


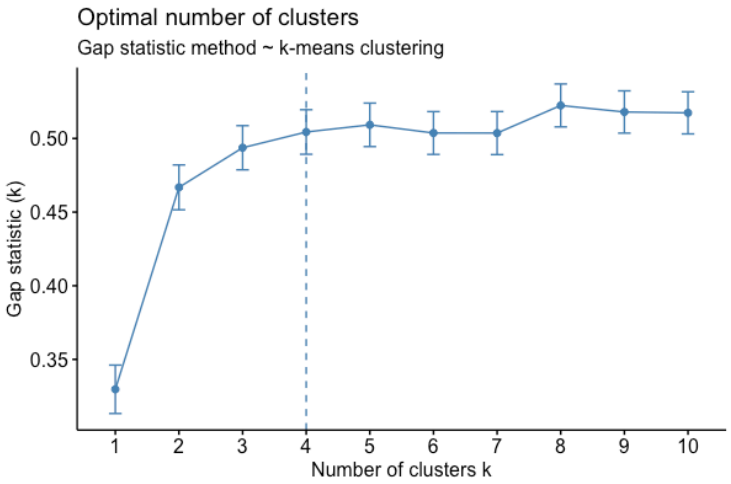


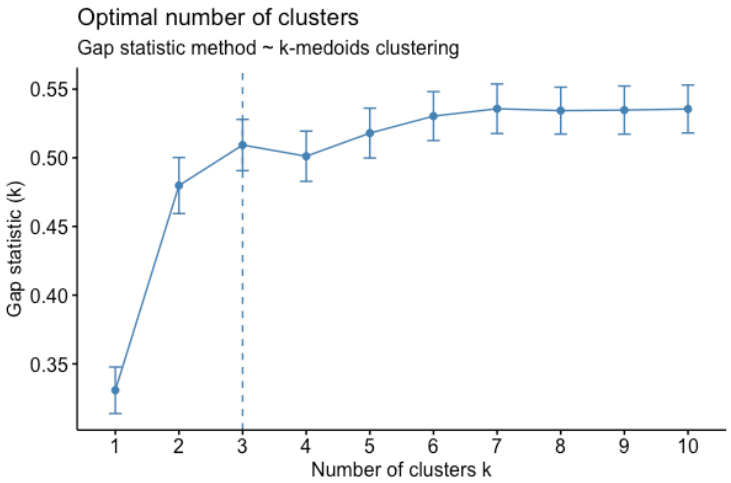

Optimal number of clusters
Silhouette method ~ hierarchical clustering
Average silhouette width
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
4
5
6
7
8
9
10
Number of clusters k


Optimal number of clusters
Silhouette method ~ k-means clustering
Average silhouette width
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
4
5
6
7
8
9
10
Number of clusters k


Optimal number of clusters
Silhouette method ~ k-medoids clustering
Average silhouette width
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
4
5
6
7
8
9
10
Number of clusters k







Both the direct methods “elbow” and “silhouette” firmly indicate $K = 2$ as optimal number of clusters, while “gap statistic” solutions differ according to the clustering method considered, ranging from a number of 3 to 8 clusters.
Therefore, according to the overall results, it’s possible to define $K = 2$ as the optimal number of clusters in the data since it has been chosen the most times. It is further confirmed by the output of the `NbClust()` function (in the `NbClust` package) computed both for hierarchical clustering and K-means.

```
> library("NbClust")
> nb.hclust <- NbClust(df.scaled,
+                      distance = "euclidean",
+                      min.nc = 2, max.nc = 10,
+                      method = "ward.D2"
+                      )
*** : 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
                significant increase of the value of the measure i.e the significant peak in Hubert
                index 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
                second differences plot) that corresponds to a significant increase of the value
of
                the measure.

*******************************************************************
* Among all indices:
* 12 proposed 2 as the best number of clusters
* 7 proposed 3 as the best number of clusters
* 1 proposed 6 as the best number of clusters
* 1 proposed 7 as the best number of clusters
* 1 proposed 8 as the best number of clusters
* 1 proposed 9 as the best number of clusters
* 1 proposed 10 as the best number of clusters

                   ***** Conclusion *****

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


*******************************************************************

> nb.kmeans <- NbClust(df.scaled,
+                      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
                significant increase of the value of the measure i.e the significant peak in Hubert
                index 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
                second differences plot) that corresponds to a significant increase of the value
of
                the measure.

*******************************************************************
* Among all indices:
* 10 proposed 2 as the best number of clusters
* 8 proposed 3 as the best number of clusters
* 2 proposed 4 as the best number of clusters
* 1 proposed 8 as the best number of clusters
* 1 proposed 9 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 addition to determining the optimal number of clusters, another preliminary task for performing clustering is choosing the best algorithm.
For this purpose, exploiting the R `cLValid()` function (in the `clValid` package) allows to compare simultaneously multiple clustering algorithms using both "internal" and "stability" cluster validation measures.

```
> library(clValid)
> clust.methods <- c("hierarchical","kmeans","pam")
>
> # Internal measures
> internal <- clValid(df.scaled,
+                     nClust = 2:6,               # number of clusters to be evaluated
+                     clMethods = clust.methods,  # clustering methods to be compared
+                     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   10.8754  20.1250  23.0540  30.3183  30.7151
             Dunn            0.2317   0.1957   0.1957   0.1631   0.1631
             Silhouette      0.5106   0.3569   0.2255   0.2109   0.1791
kmeans       Connectivity    6.7357  61.3603  67.1627  91.8345 111.1571
             Dunn            0.2409   0.0774   0.0658   0.0751   0.0841
             Silhouette      0.5235   0.2814   0.2876   0.2305   0.2154
pam          Connectivity    6.7357  59.6698  96.0127  90.2433  95.7798
             Dunn            0.2409   0.0592   0.0553   0.1020   0.0658
             Silhouette      0.5235   0.2686   0.2483   0.2651   0.2690

Optimal Scores:

             Score  Method Clusters
Connectivity 6.7357 kmeans 2
Dunn         0.2409 kmeans 2
Silhouette   0.5235 kmeans 2
```

It can be seen that the approaches with $K = 2$ clusters perform the best for all the considered measures (`Connectivity`, `Dunn` and `Silhouette`), in particular for the partitioning methods whose results are the same (for $K = 2$). However, K-means is the recommended method.

```
> # Stability measures
> stability <- clValid(df.scaled,
+                      nClust = 2:6,
+                      clMethods = clust.methods,
+                      validation = "stability")
> summary(stability)

Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6

Validation Measures:
                       2      3      4      5      6
```

```
hierarchical APN  0.0508 0.0469 0.1297 0.2081 0.2261
             AD   2.2960 2.2099 2.1767 2.1476 2.1065
             ADM  0.2663 0.2258 0.3399 0.4810 0.5045
             FOM  0.6843 0.6703 0.6688 0.6678 0.6655
kmeans       APN  0.0211 0.1420 0.2398 0.2229 0.3404
             AD   2.2344 2.0660 2.0147 1.9218 1.9097
             ADM  0.1026 0.3712 0.6514 0.5991 0.7883
             FOM  0.6803 0.6611 0.6623 0.6636 0.6626
pam          APN  0.0114 0.1897 0.3203 0.2390 0.2259
             AD   2.2214 2.0876 2.0487 1.8980 1.7961
             ADM  0.0554 0.4334 0.6674 0.5225 0.4731
             FOM  0.6718 0.6633 0.6651 0.6581 0.6492

Optimal Scores:

    Score  Method Clusters
APN 0.0114 pam    2
AD  1.7961 pam    6
ADM 0.0554 pam    2
FOM 0.6492 pam    6
```

PAM method is the suggested clustering algorithm for all the stability measures but, if for APN and ADM measures $K = 2$ is again the best solution to perform with, for the other measures PAM with $K = 6$ clusters give the best scores.

To summarize, $K = 2$ is established as number of patterns that is more likely to be found within the dataset, either performing K-means or PAM method.
Then, Cluster Analysis will be performed with both the two partitioning methods: K-means method in the first place, and PAM method next.

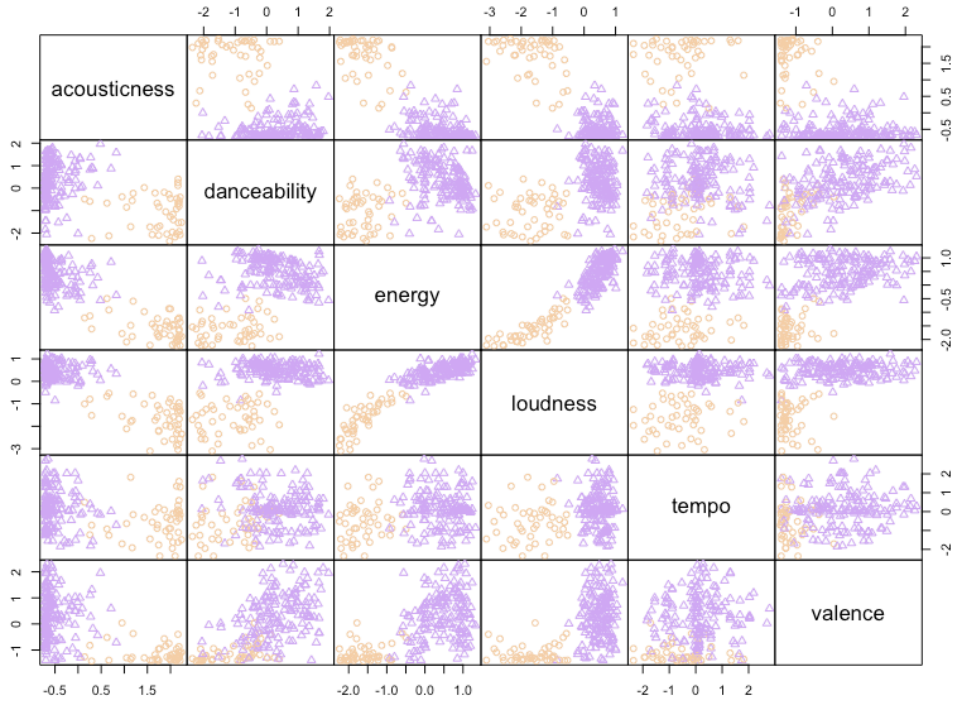
♢ K-means

```
> set.seed(123)
> kmeans.clust <- kmeans(df.scaled,
+                        centers = 2,    # number of clusters or initial (distinct) cluster
centers
+                        iter.max = 10,  # maximum number of iterations allowed
+                        nstart = 50     # number of random starting assignments
+                        )

> # Accessing to kmeans() results
> kmeans.clust$size            # cluster size
[1]  45 155

> kmeans.clust$centers         # cluster means
  acousticness danceability     energy   loudness      tempo    valence
1    1.7100148   -1.0687694 -1.6012851 -1.6597257 -0.5561139 -1.1277047
2   -0.4964559    0.3102879  0.4648892  0.4818558  0.1614524  0.3273981

> # Mean of the original variables for each cluster
> aggregate(my_dataset, by = list(cluster = kmeans.clust$cluster), mean)
  cluster acousticness danceability    energy  loudness    tempo   valence
1       1   0.81455556    0.3689556 0.1987444 -18.80369 109.1624 0.1020689
2       2   0.07089243    0.6279871 0.7561032  -5.44320 128.7723 0.4543897

> # Adding the point classifications (cluster memberships) to the original data
> head(cbind(my_dataset, cluster = kmeans.clust$cluster))
                         acousticness danceability energy loudness   tempo valence cluster
Life Is Good (feat. Drake)      0.07060        0.676  0.609   -5.831 142.037   0.508       2
Far From Home                   0.01320        0.578  0.939   -3.872 125.071   0.471       2
Unstoppable                     0.00683        0.545  0.899   -4.276 170.128   0.593       2
```

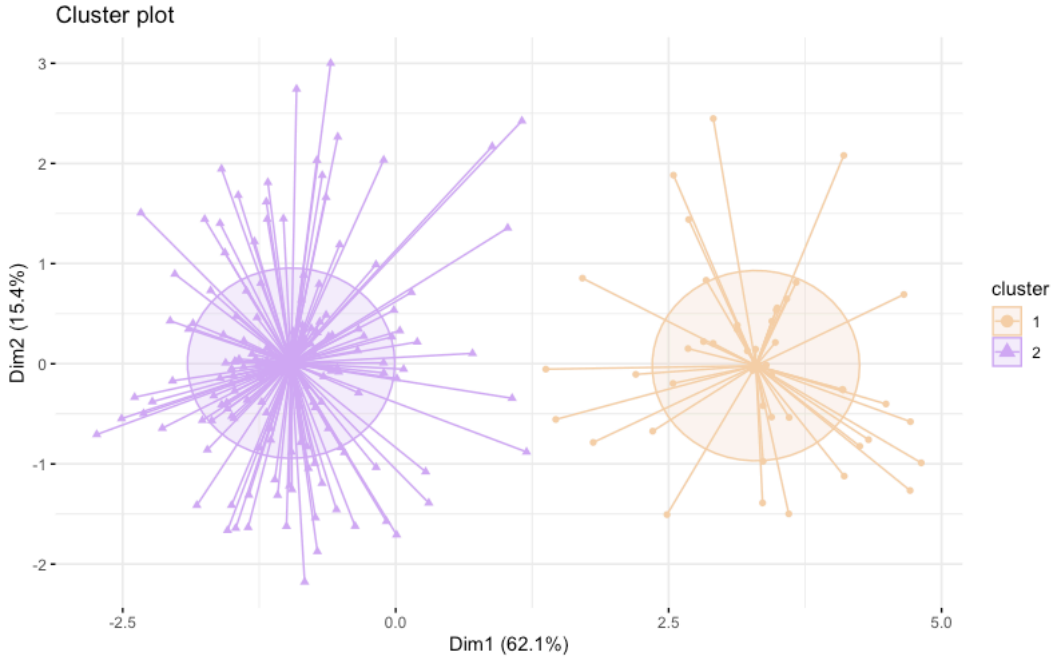
```
Love Me More                0.22000         0.638  0.539   -5.853  83.381   0.247       2
Renaissance Man             0.05340         0.796  0.726   -6.173 160.011   0.855       2
Essential Attitudes         0.44900         0.444  0.497  -12.601  99.093   0.250       1
```

The printed output of K-means clustering algorithm displays two clusters of different size: cluster 2 is in fact four times larger than cluster 1. Also, looking at the cluster centroids as well as to the mean of the original variables for each cluster, it seems like the first cluster is more likely to contain units with small values of all variables but acousticness. Vice versa with regard to the second cluster. In this sense, it is possible to characterize `cluster 1` as a pattern of mainly-acoustic, soft, quiet, slow songs, as opposed to `cluster 2` featuring more energetic and very rhythmic loud songs with electronic sound.

```
> # Visualizing K-means clusters in the original space
```




```
> # Visualizing K-means clusters in the first 2 PCs space
```

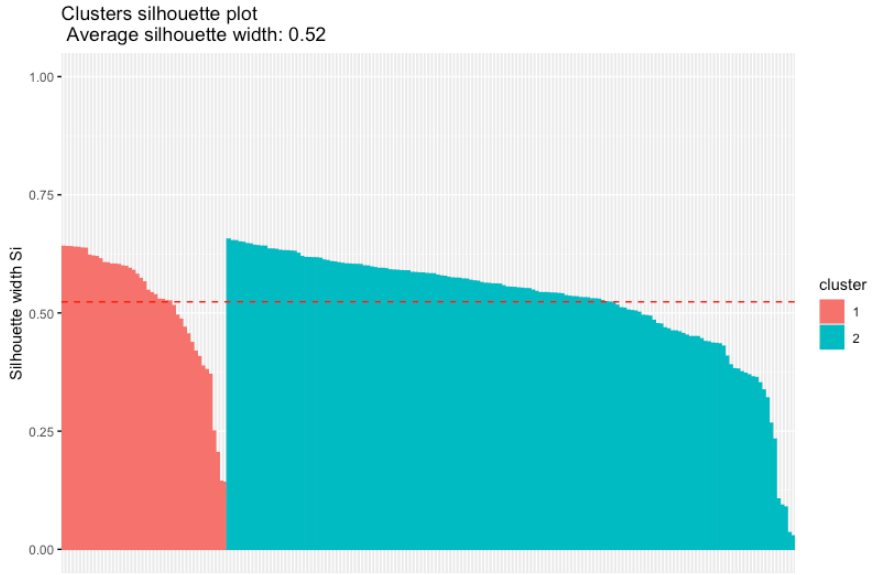


```
> kmeans.clust$totss           # total sum of squares
[1] 1194
> kmeans.clust$withinss        # within-cluster sum of squares for each cluster
[1] 120.5217 436.7310
> kmeans.clust$tot.withinss    # total within-cluster sum of squares
[1] 557.2528
> kmeans.clust$betweenss       # between-cluster sum of squares (totss-tot.withinss)
[1] 636.7472
> (kmeans.clust$withinss/kmeans.clust$totss)*100
[1] 10.09395 36.57714
> (kmeans.clust$betweenss/kmeans.clust$totss)*100
[1] 53.32891
```

53.33% is the proportion of the total variability explained by the separation between the 2 clusters. Thus, the clustering was able to explain the 53.33% of the total variance in the dataset, in the sense that, classifying the *n = 200* units into *K = 2* clusters, the total sum of squares – representing the dissimilarity of all the observations with respect to their overall mean – was half reduced. It is not a very high percentage though: this may suggest that the 2 patterns identified are not very significant clusters after all.

The goodness of such a clustering structure can be better evaluated using the internal information of the clustering process.

```
> # Internal cluster validation
> kmeans <- eclust(df.scaled, "kmeans", k = 2, nstart = 25, graph = FALSE)
> fviz_silhouette(kmeans, palette = "jco")
  cluster size ave.sil.width
1       1   45          0.52
2       2  155          0.52
```




```
> silhouette_info <- kmeans$silinfo          # silhouette information
> head(silhouette_info$widths, 10)           # silhouette widths of each observation
                      cluster neighbor sil_width
Red Gold Yesterday          1        2 0.6408448
Mars                        1        2 0.6401897
Ingénue                     1        2 0.6399049
Lonely Molecule             1        2 0.6386952
Collapse Into Light         1        2 0.6385592
We Are Forever              1        2 0.6369572
Chasing Love                1        2 0.6366146
Schloss                     1        2 0.6219759
The Discovery               1        2 0.6202017
```

```
Revealing The Unreal       1        2 0.6190907
> silhouette_info$clus.avg.widths                    # avg. silhouette width of each cluster
[1] 0.5192586 0.5246910
> silhouette_info$avg.width                          # total avg. of all individual sil. widths
[1] 0.5234687

> sil <- kmeans$silinfo$widths[, 1:3]                # silhouette width of observations

> neg_sil_index <- which(sil[, "sil_width"] < 0)     # units with negative silhouette width
> sil[neg_sil_index, , drop = FALSE]
[1] cluster   neighbor  sil_width
<0 rows> (or 0-length row.names)

> low_sil_index <- which(sil[, "sil_width"] < 0.30)  # units with low silhouette width
> sil[low_sil_index, , drop = FALSE]
                          cluster neighbor  sil_width
Masollan                        1        2 0.25009467
Clovelly Dreams                 1        2 0.20458709
Essential Attitudes             1        2 0.14360891
Dawning                         1        2 0.14138794
Loyal (feat. Drake)             2        1 0.26670175
You Gotta Keep Moving           2        1 0.23300981
Light in the Fog                2        1 0.10637350
Any Given Place                 2        1 0.09331921
Event Horizon                   2        1 0.08937951
Beautiful Heartbeat             2        1 0.03525040
The Powers Of The Universe      2        1 0.02830117
```

The Silhouette information reveals an *average silhouette width* of 0.52 for both clusters – the maximum of individual *silhouette width* reached is around 0.64 – resulting in a *total average silhouette width* of 0.52 as well. The index suggests that each unit in `cluster 1` is similar enough to its own cluster compared to `cluster 2`. Same thing for each unit in `cluster 2`.
All units have been assigned by the K-means algorithm to the as right as possible cluster, as we can see that there are zero units with negative silhouette width and only few units with low silhouette widths. These latter ones, on the border of two clusters, might represent outliers in data. Consequently, the overall clustering configuation appears appropriate.

```
> library(fpc)
> kmeans_stats <- cluster.stats(dist(df.scaled), kmeans$cluster)
> kmeans_stats$dunn             # Dunn index
[1] 0.2408892
> kmeans_stats$min.separation   # inter-cluster separation
[1] 1.312778
> kmeans_stats$max.diameter     # intra-cluster compactness
[1] 5.449719
```

At the same time though, *Dunn Index* value is low, around 0.24. It is due to a maximum diameter of the clusters of 5.45, which is greater than the minimum distance between the clusters of 1.32. Therefore, the data might not contain very compact and well-separated clusters.

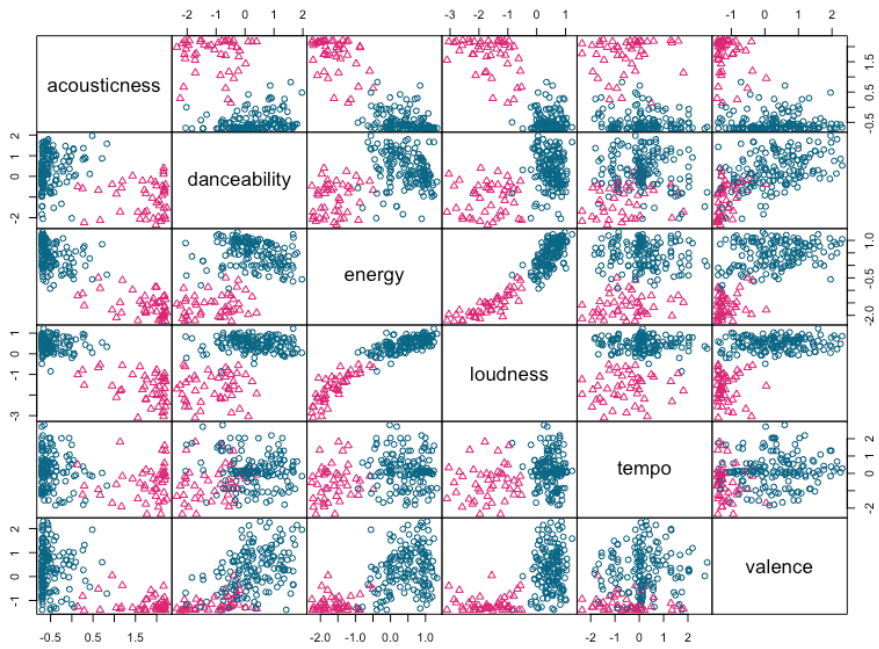
◇ K-medoids

Another partitioning clustering process is reported, as perfomed with K-medoids algorithm using R `pam()` function. The K-medoids clustering method, despite being a robust alternative to K-means due to its less sensitiveness to noise and outliers, shows almost identical results to the previous ones.

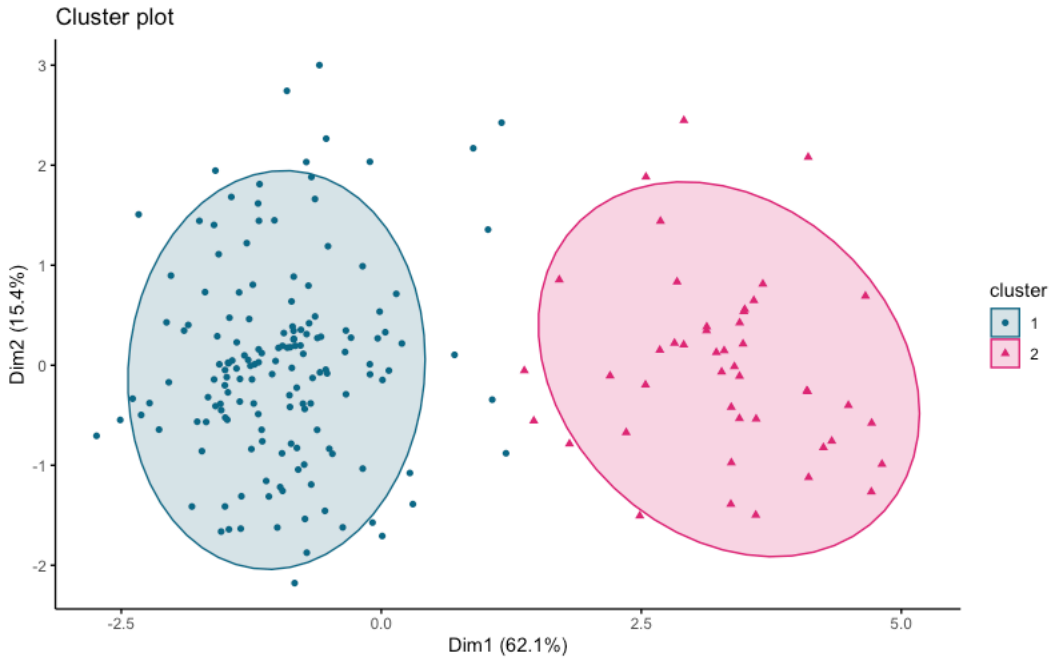
```
> library(cluster)
> pam.clust <- pam(df.scaled, k = 2)

> # Accessing pam() results
> pam.clust$id.med                    # id cluster medoids
[1] 194  21

> pam.clust$medoids                   # cluster medoids matrix
               acousticness danceability     energy   loudness      tempo    valence
Underwater       -0.6504223    0.2198503  0.6609819  0.2916211 -0.2326179  0.2184074
Lonely Molecule   1.7113335   -0.6692405 -1.8005201 -1.5954984 -0.8870308 -1.3212756
```

Tracks “Underwater” and “Lonely Molecule” are identified as the representative units (medoids) which minimize the sum of the dissimilarities of the observations with respect to their closest representative unit.

```
> pam.clust$clusinfo                  # cluster info
     size max_diss  av_diss diameter separation
[1,]  155 3.503527 1.641557 5.449719   1.312778
[2,]   45 2.980106 1.639063 4.407878   1.312778

> # Adding the point classifications to the original data
> head(cbind(my_dataset, cluster = pam.clust$cluster))
                          acousticness danceability energy loudness   tempo valence cluster
Life Is Good (feat. Drake)      0.07060        0.676  0.609   -5.831 142.037   0.508       1
Far From Home                   0.01320        0.578  0.939   -3.872 125.071   0.471       1
Unstoppable                     0.00683        0.545  0.899   -4.276 170.128   0.593       1
Love Me More                    0.22000        0.638  0.539   -5.853  83.381   0.247       1
Renaissance Man                 0.05340        0.796  0.726   -6.173 160.011   0.855       1
Essential Attitudes             0.44900        0.444  0.497  -12.601  99.093   0.250       2

> # Visualizing PAM clusters in the original space
```



```
> # Visualizing PAM clusters in the first 2 PCs space
```




```
> # Internal cluster validation
> pam_silhouette <- pam.clust$silinfo # silhouette info
> pam_silhouette$clus.avg.widths      # average silhouette widths of each cluster
[1] 0.5246910 0.5192586

> pam_silhouette$avg.width            # total average of all individual silhouette widths
[1] 0.5234687

> head(pam_silhouette$widths, 10)     # silhouette widths of 10 observations
                                  cluster neighbor sil_width
Rhythm Is A Dancer                      1        2 0.6561430
We Are The Night                        1        2 0.6527374
Anywhere                                1        2 0.6526286
Oh Yeah!                                1        2 0.6498917
Far From Home                           1        2 0.6492376
Body                                    1        2 0.6464948
The Disease                             1        2 0.6456921
why are you here                        1        2 0.6429669
The Spark - Tiësto vs twoloud Remix     1        2 0.6421646
The Violence                            1        2 0.6410786

> pam.sil <- pam.clust$silinfo$widths[, 1:3]                # silhouette width of observations

> neg_pam_sil_index <- which(pam.sil[, "sil_width"] < 0)    # units with negative sil. width
> pam.sil[neg_pam_sil_index, , drop = FALSE]
     cluster neighbor sil_width
> low_pam_sil_index <- which(pam.sil[, "sil_width"] < 0.30) # units with low silhouette width
> pam.sil[low_pam_sil_index, , drop = FALSE]
                          cluster neighbor  sil_width
Loyal (feat. Drake)             1        2 0.26670175
You Gotta Keep Moving           1        2 0.23300981
Light in the Fog                1        2 0.10637350
Any Given Place                 1        2 0.09331921
Event Horizon                   1        2 0.08937951
Beautiful Heartbeat             1        2 0.03525040
The Powers Of The Universe      1        2 0.02830117
Masollan                        2        1 0.25009467
Clovelly Dreams                 2        1 0.20458709
Essential Attitudes             2        1 0.14360891
Dawning                         2        1 0.14138794
```

```
> library(fpc)
> pam_stats <- cluster.stats(dist(df.scaled), pam.clust$cluster)
> pam_stats$dunn             # Dunn index
[1] 0.2408892
> pam_stats$min.separation   # inter-cluster separation
[1] 1.312778
> pam_stats$max.diameter     # intra-cluster compactness
[1] 5.449719
```

Since k-means and pam algorithms provided same clustering results, external cluster validation statistics have been computed only for *k-means* clustering outcome.

```
> # External Cluster Validation
> levels(data$playlist) <- factor(c("Hip-Hop", "Electro-Dance", "Ambient Music", "Rock"))
> genre <- as.numeric(data$playlist)

> table(data$playlist, kmeans.clust$cluster)

               1  2
  Hip-Hop      0 50
  Electro-Dance  0 50
  Ambient Music 45  5
  Rock         0 50
```

Starting by computing the cross-tabulation (confusion matrix) between *K-means* clusters and the reference *Genre* labels, it can be seen that:

- All *Hip-Hop* songs (*n = 50*) have been classified into `cluster 2`.
- All *Electro-Dance* songs (*n = 50*) have been classified into `cluster 2`.
- A large number of *Ambient* songs (*n = 45*) has been classified into `cluster 1`. Only a very fewer number (*n = 5*) of them has been classified into `cluster 2`.
- All *Rock* songs (*n = 50*) have been classified into `cluster 2`.

```
> km.external.stats <- cluster.stats(dist(df.scaled), genre, kmeans$cluster)
> km.external.stats$corrected.rand          # Rand index
[1] 0.2604663
> km.external.stats$vi                      # Meila's vi index
[1] 1.015672
```

The agreement between *Genre* labels and the partitions defined by *K-means*, according to the corrected Rand index is very poor. Similarly, the Variation of Information index indicates that the clustering configuration obtained is very far from the “true” original partitions. Such index values were predictable since the number of *k-means* clusters doesn’t match the number of external known labels (2 clusters vs. 4 labels).

CLUSTER ANALYSIS ON PCA OUTPUT

Optimal number of clusters & better algorithm

```
> library(clValid)
> internal <- clValid(PC, nClust = 2:6, clMethods = clust.methods, validation = "internal")
> optimalScores(internal)
                Score       Method Clusters
Connectivity 5.1527778 hierarchical        2
Dunn         0.1572976 hierarchical        4
Silhouette   0.5738483       kmeans        2
```

Internal cluster validation measures mostly suggest a number of $K = 2$ clusters, to obtain performing the hierarchical clustering method.

```
> stability <- clValid(PC, nClust = 2:6, clMethods = clust.methods, validation = "stability")
> optimalScores(stability)
         Score       Method Clusters
APN 0.02287582 hierarchical        2
AD  1.91310961          pam        6
ADM 0.58098952          pam        2
FOM 1.23951707          pam        4
```

The majority of stability measures still suggest a number of $K = 2$ clusters, but the clustering method choosen more times is the K-medoids one.

Hierarchical clustering

For the sake of completeness, having already used `pam()` algorithm, this time clustering process will be performed with the hierarchical method.

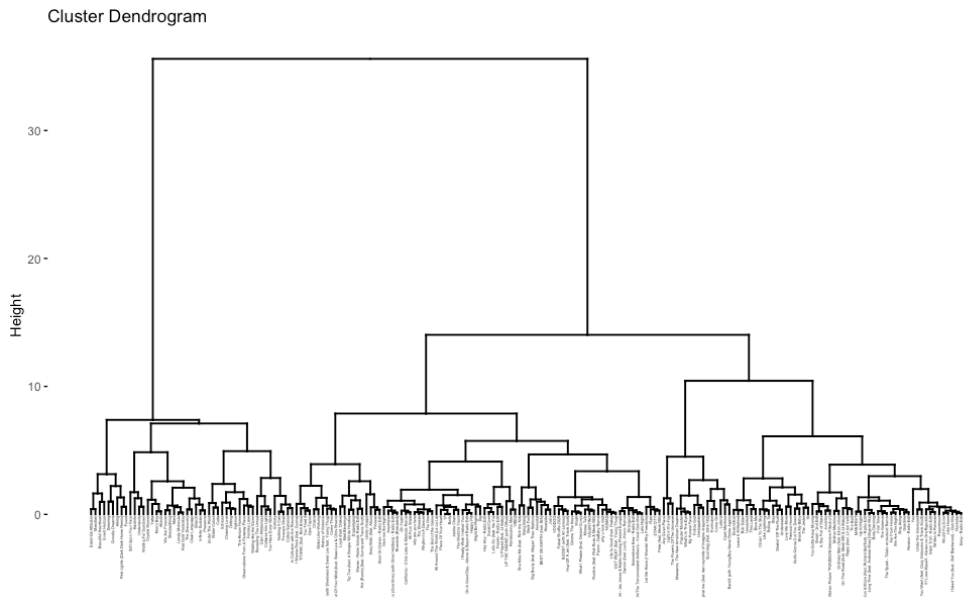
```
> # computing dissimilarities using distance measures for continuous quantitative variables
> eucl.dist <- stats::dist(PC, method = "euclidean")

> # reformatting the distance vector into distance matrix
> eucl.dist.matrix <- round(as.matrix(eucl.dist)[1:5, 1:5], 2)
> head(eucl.dist.matrix, 5)
                         Life Is Good (feat. Drake) Far From Home Unstoppable Love Me More Renaissance
Man
Life Is Good (feat. Drake)                       0.00         1.32       1.46        2.37           1.56
Far From Home                                    1.32         0.00       1.68        2.17           2.40
Unstoppable                                      1.46         1.68       0.00        3.55           1.87
Love Me More                                     2.37         2.17       3.55        0.00           3.68
Renaissance Man                                  1.56         2.40       1.87        3.68           0.00
```

The distance information displayed in the matrix above, makes us aware that among those 5 songs, "Love Me More" and "Renaissence Man" are the units more distant from each other (3.68), in terms of the features explained by three `PCs`.

Hierarchical clustering is then performed, using the "ward.D2" method as linkage criterion, according to which each merge leads to an increase of WSS.

```
> hclust <- hclust(d = eucl.dist, method = "ward.D2")   # perform hierarchical clustering
```

The corresponding dendrogram produced – using `fviz_dend()` function – by the output of `hclust()` is the following:



The hierarchical cluster tree generated can be analyzed computig the cophenetic distances/dissimiliarities between the units, to quantify how similar the units have to be in order to be grouped into the same cluster, or in other words, at what height of the dendrogram the fusions of the "branches" occur.

```
> coph.dist <- cophenetic(hclust)  # cophenetic distances
> round(as.matrix(coph.dist)[1:5, 1:5], 2)
                          Life Is Good (feat. Drake) Far From Home Unstoppable Love Me More Renaissance
Man
Life Is Good (feat. Drake)                       0.00        14.03       14.03        14.03           5.74
Far From Home                                   14.03         0.00       10.44         6.10          14.03
Unstoppable                                     14.03        10.44        0.00        10.44          14.03
Love Me More                                    14.03         6.10       10.44         0.00          14.03
Renaissance Man                                  5.74        14.03       14.03        14.03           0.00
```

According to this cophenetic dissimilarity matrix displaying some of the distances, the first two units that would merge together as more similar (5.74) are "Life Is Good" and "Renaissance Man". Subsequently, "Far From Home" and "Love Me More" would merge. At this point, the branch including "Unstoppable" and the branch including "Far From Home" and "Love Me More" would be fused into one single branch.

```
> # verifying cluster tree
> cor(eucl.dist, coph.dist)        # correlation between cophenetic and original distances
[1] 0.8351091

> # executing again with the average linkage method
> hclust2 <- hclust(eucl.dist, method = "average")
> cor(eucl.dist, cophenetic(hclust2))
[1] 0.8593963
```

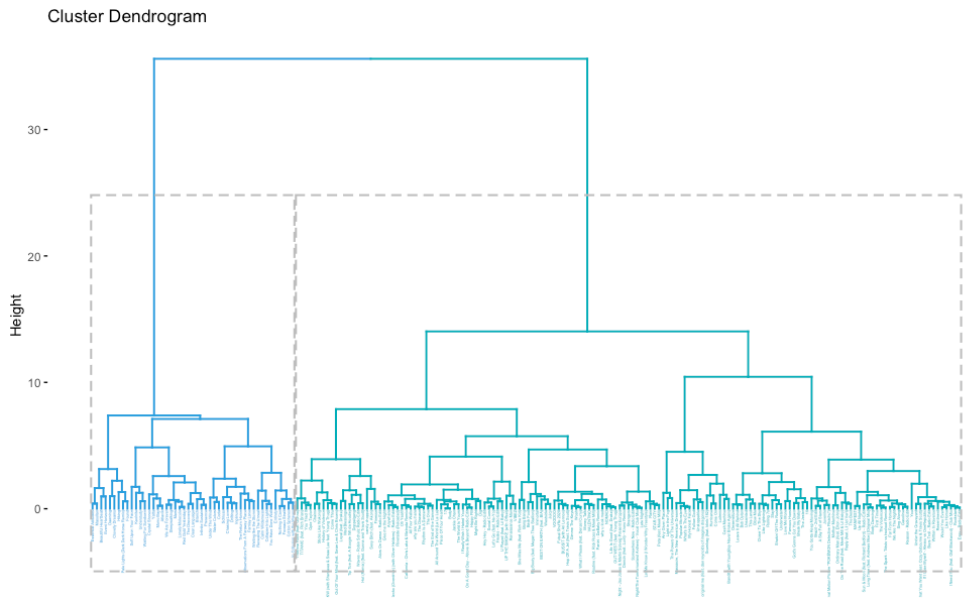
The cluster tree seems to reflect accurately the data, encouraging the validity of the clustering solution. In fact, the cophenetic distances have a strong correlation (0.84) with the distances between units in the original distance matrix. The correlation coefficient shows an even higher value (0.86) when executing hierarchical clustering using the "average linkage method".

As suggested before by internal and stability measures scores, the hierarchical cluster tree is therefore "cut" at given height in order to partition the data into $K = 2$ clusters.

```
> groups <- cutree(hclust, k = 2)  # cutting dendrogram into 2 groups
```

```
> table(groups)                    # number of members in each cluster
groups
  1   2
153  47

> # mean of the PCs for each cluster
> aggregate(PC, by = list(cluster = groups), mean)
  cluster        PC1         PC2         PC3
1       1  0.9851823  0.01370219  0.04976083
2       2 -3.2070827 -0.04460499 -0.16198739

> # adding point classifications (cluster memberships) to PC data
> head(cbind(PC, cluster = groups))
                                  PC1         PC2         PC3 cluster
Life Is Good (feat. Drake)  0.9375455  0.32143907 -0.56465205       1
Far From Home               1.3923147 -0.03261675  0.62428018       1
Unstoppable                 1.7493158  1.44292301 -0.09811344       1
Love Me More               -0.3032540 -1.38909899  0.51780861       1
Renaissance Man             2.0654199  0.42822740 -1.63318379       1
Essential Attitudes        -1.4659321 -0.55588884  0.39630546       2
```

The cut procedure resulted into two groups whose sizes differ very little from the ones obtained when performing partitioning clustering on the original variables: only two units are in fact located differently.




```
# Internal cluster validation
> pca.hc <- eclust(PC, "hclust", k = 2, hc_metric = "euclidean",
+                  hc_method = "ward.D2", graph = FALSE)
> hc_silhouette <- pca.hc$silinfo          # silhouette info
> hc_silhouette$clus.avg.widths            # average silhouette widths of each cluster
[1] 0.5788362 0.5324668

> head(hc_silhouette$widths, 10)           # silhouette widths of each observation
                                                    cluster neighbor sil_width
The One                                                   1        2 0.7079978
All Four Walls                                            1        2 0.7071181
Oh Yeah!                                                  1        2 0.7061365
why are you here                                          1        2 0.7042004
California - Chris Lake & Matroda Remix                   1        2 0.7025503
Rhythm Is A Dancer                                        1        2 0.7023256
The Disease                                               1        2 0.7021825
Gecko (Overdrive) (with Oliver Heldens) - Radio Edit      1        2 0.7019582
Riverside (Reloaded)                                      1        2 0.6991306
The End of the Game                                       1        2 0.6983039
```

```
> hc_silhouette$avg.width                                  # total avg. silhouette width
[1] 0.5679394

> hc.sil <- pca.hc$silinfo$widths[, 1:3]                   # silhouette width of observations
> low_hc_sil_index <- which(hc.sil[, "sil_width"] < 0.30) # units with low silhouette widths
> hc.sil[low_hc_sil_index, , drop = FALSE]
                         cluster neighbor   sil_width
You Gotta Keep Moving          1        2  0.23811218
Light in the Fog               1        2  0.10318439
Any Given Place                1        2  0.07945587
The Powers Of The Universe     1        2  0.02341026
Clovelly Dreams                2        1  0.27835966
Essential Attitudes            2        1  0.17858445
Dawning                        2        1  0.15727133
Beautiful Heartbeat            2        1 -0.01623910
Event Horizon                  2        1 -0.06683065

> pca_hc_stats <- cluster.stats(dist(PC), pca.hc$cluster)
> pca_hc_stats$dunn             # Dunn index
[1] 0.1519048
> pca_hc_stats$min.separation
[1] 0.8173066
> pca_hc_stats$max.diameter
[1] 5.380385
```

To summarize, Silhouette information reveals a clustering configuration of the same quality of the one assessed with partitioning clustering on the original variables, with slightly better values: the maximum value of *silhouette width* reached by an observation is around 0.71, and there are fewer units with low silhouette widths. At same time, two units ("Beautiful Heartbeat" and "Event Horizon") with negative silhouette widths are detected in `cluster 2`. They are presumably the two songs that, when perfoming partitioning clustering on original variables, were assigned to the larger cluster.
On the other hand, Dunn index instead shows a lower value.

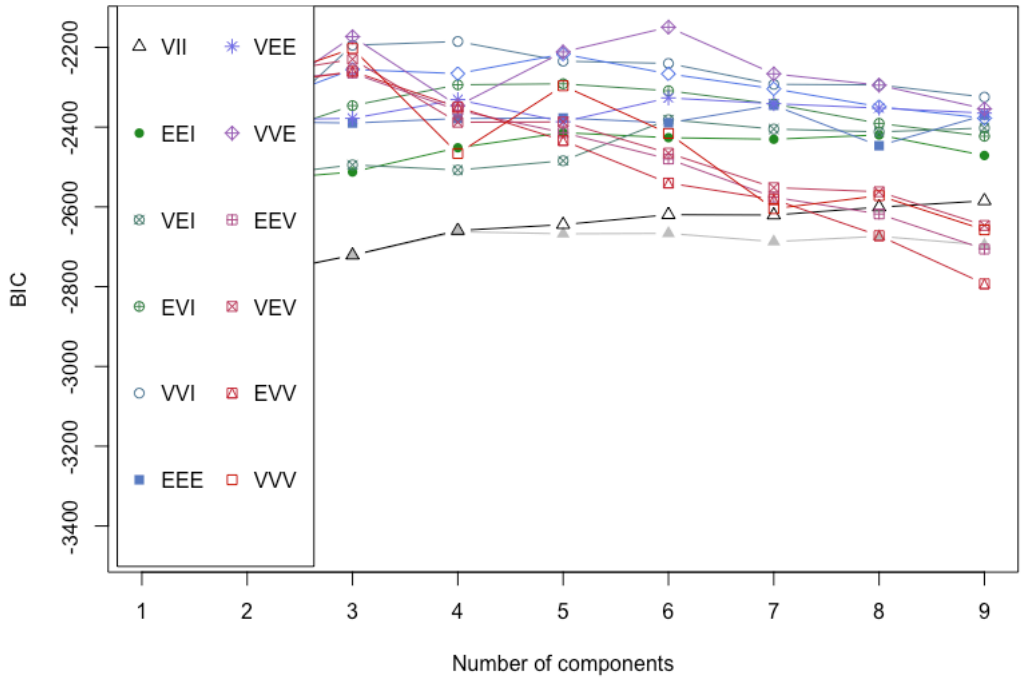
Consequently, it is possible to assess that performing clustering after dimensionality reduction through PCA, didn't really improve cluster quality on the reference dataset, since the outcomes respectively of clustering on original variables and on `PCs` are almost the same.

MODEL-BASED CLUSTERING

```
> library(mclust)
> df.scaled <- scale(my_dataset)
```

Turning to the alternative *model-based clustering* approach, `Mclust()` function is run in order 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.

```
> # Fit the parsimonious models
> model <- Mclust(df.scaled, G = 1:9, modelNames = NULL)

> # Plot of the BIC values for all the fitted models
```

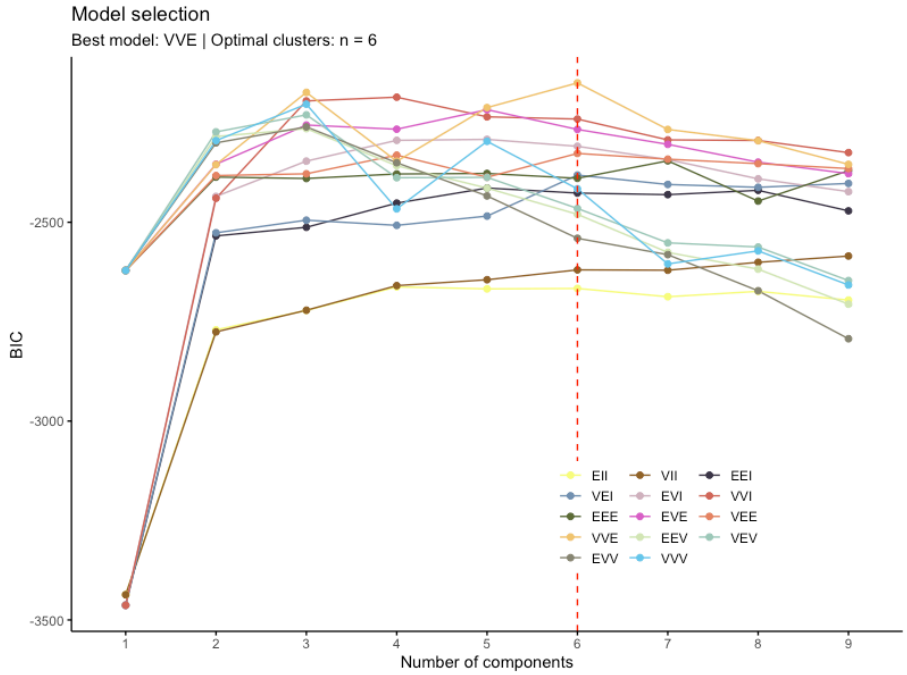



```
> # Visualize the top three BIC models
> summary(model$BIC)
Best BIC values:
            VVE,6       VVE,3       VVI,4
BIC      -2149.53 -2173.39549 -2185.41935
BIC diff     0.00   -23.86542   -35.88929
```

According to the Bayesian Information Criterion used by `Mclust()` function, top three models the data were more likely generated by, are: the VVE model with six or three components, and the VVI model with four components. As a matter of fact, the symbols representing the mentioned models in the plot above, are the ones located at higher position.

```
> # Visualize the main output
> summary(model)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VVE (ellipsoidal, equal orientation) model with 6 components:
```

```
 log-likelihood   n df      BIC       ICL
      -831.0424 200 92 -2149.53 -2174.989

Clustering table:
 1  2  3  4  5  6
44 45 28 16 38 29
```



In particular, it can be seen that model-based clustering selected a model with six components (i.e. clusters). The optimal selected model name is VVE model. That is the six components are ellipsoidal with variable volume and shape but equal orientation. The summary contains also the clustering table specifying cluster sizes, ranging from a number of 16 to 45 observations "soft-assigned" to clusters. The larger ones, in terms of number of observations classified into them, are the first and second cluster.

```
> # Useful quantities
> probabilities <- as.matrix(round(model$z, 6))              # Probability to belong to a given cluster
> colnames(probabilities) <- c("pi1", "pi2", "pi3", "pi4", "pi5", "pi6")
> classification <- as.matrix(model$classification)          # Cluster assignment of each observation
> colnames(classification) <- "Classification"
> uncertainty <- as.matrix(round(model$uncertainty, 6))      # Classification Uncertainty
> colnames(uncertainty) <- "Uncertainty"

> # Combining the overall information in one single matrix for a more orderly display of data
> model_output <- cbind(probabilities, classification, uncertainty)
> head(model_output, 30)
                                pi1      pi2      pi3      pi4      pi5      pi6 Classification Uncertainty
Life Is Good (feat. Drake)  0.878489 0.000000 0.119696 0.000004 0.000000 0.001811              1    0.121511
Far From Home               0.002482 0.856325 0.000012 0.000000 0.000000 0.141181              2    0.143675
Unstoppable                 0.000391 0.990376 0.000005 0.000000 0.000000 0.009228              2    0.009624
Love Me More                0.106009 0.000000 0.000000 0.893991 0.000000 0.000000              4    0.106009
Renaissance Man             0.025000 0.000000 0.000008 0.000000 0.000000 0.974991              6    0.025009
Essential Attitudes         0.000001 0.000000 0.000000 0.999995 0.000003 0.000000              4    0.000005
Under the Graveyard         0.004662 0.964828 0.029708 0.000295 0.000000 0.000507              2    0.035172
Back Foot                   0.000114 0.956707 0.000000 0.000000 0.000000 0.043179              2    0.043293
Piece Of Your Heart         0.043441 0.000000 0.000886 0.000000 0.000000 0.955672              6    0.044328
The One                     1.000000 0.000000 0.000000 0.000000 0.000000 0.000000              1    0.000000
Yatta                       0.000000 0.000000 0.000000 0.000213 0.999787 0.000000              5    0.000213
Tomorrow                    0.053275 0.000000 0.936555 0.010150 0.000000 0.000020              3    0.063445
Morro Bay                   0.000000 0.000000 0.000000 0.000074 0.999926 0.000000              5    0.000074
Gecko (Overdrive)           0.000504 0.905293 0.000000 0.000000 0.000000 0.094203              2    0.094707
Branch                      0.000000 0.000000 0.000000 0.000521 0.999479 0.000000              5    0.000521
Loyal (feat. Drake)         0.034142 0.000000 0.000000 0.965851 0.000007 0.000000              4    0.034149
Masollan                    0.000000 0.000000 0.000000 1.000000 0.000000 0.000000              4    0.000000
Drifting                    0.000000 0.000000 0.000000 0.000115 0.999885 0.000000              5    0.000115
STAR CITY                   0.998733 0.000000 0.000000 0.001267 0.000000 0.000000              1    0.001267
```

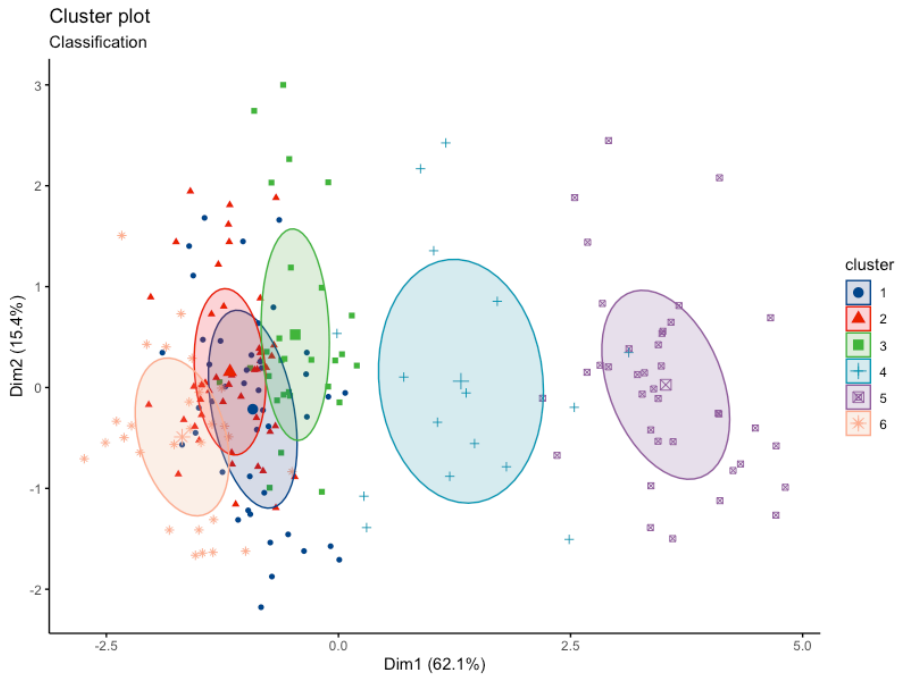
```
On A Good Day          0.008272 0.000000 0.000022 0.000000 0.000000 0.991706  6  0.008294
Lonely Molecule        0.000000 0.000000 0.000000 0.000442 0.999558 0.000000  5  0.000442
original me            0.034685 0.899486 0.030338 0.000000 0.000000 0.035490  2  0.100514
The Discovery          0.000000 0.000000 0.000000 0.000138 0.999862 0.000000  5  0.000138
Follow God             0.000850 0.997264 0.000117 0.000000 0.000000 0.001769  2  0.002736
Sexy Bitch (feat. Akon) 0.986108 0.000000 0.000491 0.000000 0.000000 0.013401  1  0.013892
Social Cues            0.000251 0.007958 0.000000 0.000000 0.000000 0.991791  6  0.008209
VIBEZ                  0.999999 0.000000 0.000000 0.000000 0.000000 0.000000  1  0.000001
Dawning                0.000000 0.000000 0.000000 0.999106 0.000893 0.000000  4  0.000894
I Need You             0.000032 0.999840 0.000001 0.000000 0.000000 0.000126  2  0.000160
You Gotta Keep Moving  0.012274 0.000000 0.000000 0.987726 0.000000 0.000000  4  0.012274
```

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

- "Life Is Good" has been soft-assigned:
  - to the first component (i.e. cluster) of the mixture distribution with a fitted posterior probability of 87.85% to belong to it;
  - to the third component with a probability of 11.97% to belong to it;
  - to the remaining components with very close to zero probabilities.

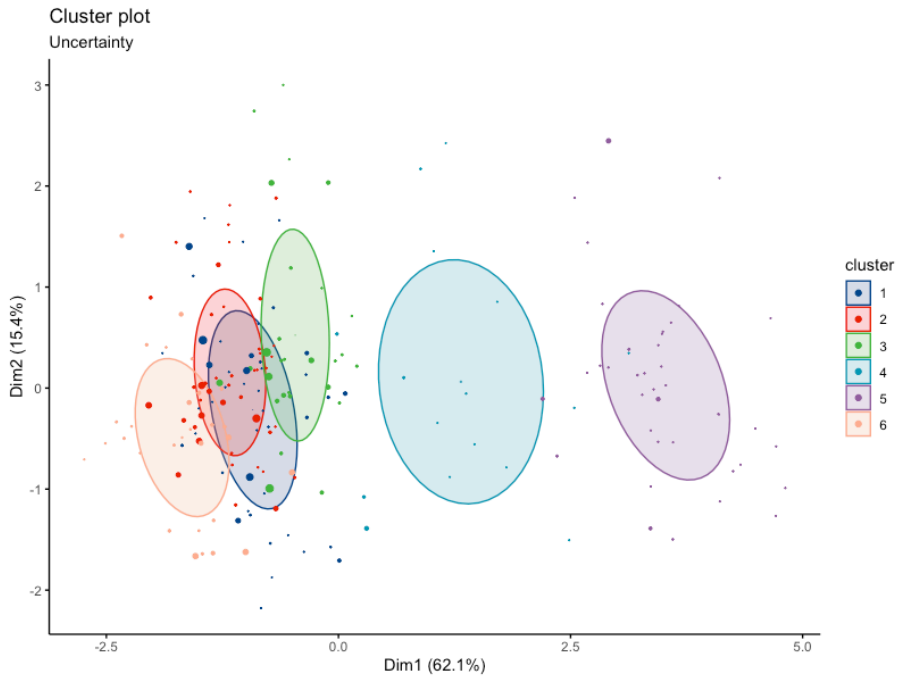
  Therefore, the observation has been classified into `cluster 1`, with an uncertainty of 12.15%.
- "Yatta" has been soft-assigned to the first component (i.e. cluster) of the mixture distribution with a fitted posterior probability of 100%. That is, the observation was confidently – with zero uncertainty – classified into `cluster 1`.

```
> # Visualizing Classification (i.e. clustering) in PC space
```



The plot showing classification results in the first two PCs space, makes noticeable how `clusters 1`, `2`, `3` and `6` – corresponding to the first three components of the mixture plus the sixth one – are partially overlapped, especially `cluster 1` and `2`. It suggests that observations classified in these clusters share similar values in terms of the first two principal components which collectively are able to explain the 76.5% of the variability in the dataset.

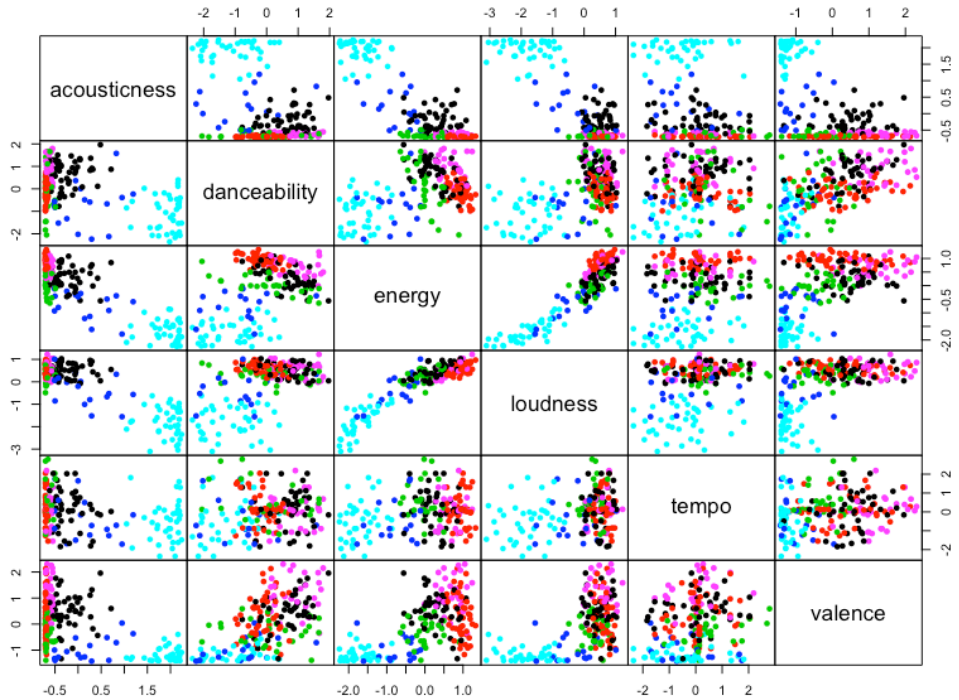
```
> # Visualizing Classification Uncertainty in first 2 PCs space
```




```
# Detecting the most uncertain units in terms of classification with their PCs scores
> PC <- prcomp(my_dataset, scale. = TRUE)
> PC <- PC$x
> cbind(PC[model_output[,"Uncertainty"] > 0.50, 1:3],
+       Uncertainty = model_output[model_output[,"Uncertainty"] > 0.50, 8],
+       Classification = model_output[model_output[,"Uncertainty"] > 0.50, 7])
                                                 PC1        PC2        PC3   Uncertainty Classification
Rolling 7s                                -0.7416045 -0.9937239  -0.8414045    0.546773              3
Money Talk                                -1.4603499  0.4744816   1.2221533    0.552449              1
Sun & Moon (feat. Richard Bedford)        -0.7732429  0.3539118  -0.3930145    0.565408              3

> model_output[model_output[,"Uncertainty"] > 0.50, 1:6]
                                                       pi1      pi2      pi3      pi4 pi5      pi6
Rolling 7s                                        0.368266 0.145899 0.453227 0.000041   0 0.032567
Money Talk                                        0.447551 0.000004 0.204603 0.000000   0 0.347842
Sun & Moon (feat. Richard Bedford) - Radio Edit 0.419455 0.000000 0.434592 0.000228   0 0.145725
```

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 three data points in particular, that have an uncertainty level associated to their classification greater than 50%. Two observations have been assigned to `cluster 3` – the green ellipse – and most likely correspond to the two larger green points: one of them – supposedly Sun & Moon – is located at the overlap between ellipses of `cluster 1` and `3`; the other one – supposedly Rolling 7s – is located a little lower in the plot, embedded inside the blue ellipse despite its different membership color. It might suggest that `cluster 1` could have been a good assignment destination for them as well. Sun & Moon has been in fact soft-assigned to `cluster 3` with a probability of 43% and to `cluster 1` with a probability of 42%. Rolling 7s has been soft-assigned to `cluster 3` with a probability of 45%, to `cluster 1` with a probability of 37%.
The other observation was instead classified into `cluster 1`. It probably corresponds to the largest blue point, embedded inside the red ellipse and close to `cluster 6` ellipse. Money Talk has been in fact soft-assigned to `cluster 1` with a probability of 45%, to `cluster 6` with a probability of 35%, likely indicating it is similar enough to `cluster 6` units to potentially belong as much to it.

```
> # Pairs plot with Classification
```




```
> # Mean of the original (scaled) variables for each cluster
> aggregate(df.scaled, by = list(cluster = model$classification), mean)
  cluster acousticness danceability      energy   loudness       tempo    valence
1       1   -0.1879298   0.67903647  0.28378634  0.4928278  0.10990149  0.5079991
2       2   -0.6928932  -0.12277203  0.95252320  0.6071687  0.14378944  0.2773387
3       3   -0.6523838   0.07648539  0.02124555  0.3030477  0.50067054 -0.2937337
4       4    0.1727987  -0.73811842 -0.72407638 -0.3559792 -0.39918967 -0.9659621
5       5    1.9164489  -1.08856746 -1.69267330 -1.7909072 -0.51376456 -1.1844405
6       6   -0.6163444   0.92003461  0.68833756  0.5606135  0.02017663  1.1674647

> # External Cluster Validation
> table(data$playlist, model$classification)  # confusion matrix

                1  2  3  4  5  6
  Hip-Hop      27  1  9  2  0 11
  Electro-Dance 11 14 12  1  0 12
  Ambient Music  0  0  0 12 38  0
  Rock           6 30  7  1  0  6
```

From the confusion matrix between clusters brought out by model-based clustering and the reference *Genre* labels, it can be seen that:

- The majority of *Hip-Hop* songs (*n = 27*) has been classified into `cluster 1`. A less large number (*n = 11*) has been classified into `cluster 6`, and the remaining ones between `cluster 2`, `3` and `4`. None in `cluster 5`.
- *Electro-Dance* songs have been more or less equally distributed among `cluster 1`, `2`, `3` and `6`. One song has been classified into `cluster 4`. None in `cluster 5`.
- A large number of *Ambient* songs (*n = 38*) has been classified into `cluster 5`. The remaining ones (*n = 12*) have been classified into `cluster 4`.
- The majority of *Rock* songs (*n = 30*) has been classified into `cluster 2`. The remaining ones have been more or less equally distributed between `cluster 1`, `3` and `6`. Only one song has been classified into `cluster 4`. None in `cluster 5`.

```
> # Adjusted (or Corrected) Rand Index
> adjustedRandIndex(data$playlist, model$classification)
[1] 0.329522
```

According to the adjusted Rand index, the agreement between *Genre* labels and the clusters obtained via model-based clustering, is poor (even if a little bit better than *k-means* clustering). Also in this case such a Rand index value was predictable, since the number of components (i.e. clusters) in the parsimonious model – that better fits data – doesn't match the number of external known labels.