Spotify Dataset Clustering Analysis

2022-12-23

# Spotify Dataset Clustering Analysis

Will clustering help us identify patterns between the different songs of the Spotify dataset?

This report includes in these parts:

* K-means clustering
* Hierarchical clustering
* Visual Partition of the n-classes
* Dissimilarity matrix / Gower’s Distance metrics

## Setup

We setup the working directory and load the pre-cleaned Spotify dataset (refer to pre-processing / descriptive statistics):

# Replace with your own working directory if needed  
WD <- "C:/Users/Joseph/Documents/Codes/2022/mvtec-2022/finalproject/spotify-statistics-final/clustering"  
setwd(WD)  
dd <- read.csv("data/cleaneddata.csv", sep=",");  
head(dd)

## popularity duration\_ms explicit danceability energy key loudness mode  
## 1 38 153039 False 0.522 0.854 A -5.274 minor  
## 2 20 212840 False 0.544 0.855 C -3.464 major  
## 3 71 235779 False 0.856 0.617 A#/Bb -5.308 minor  
## 4 12 142759 False 0.478 0.235 D -13.106 major  
## 5 9 292217 False 0.786 0.714 A -10.344 minor  
## 6 38 194506 False 0.650 0.771 D -6.901 major  
## speechiness acousticness instrumentalness liveness valence tempo  
## 1 0.0431 0.049800 0.000253 0.2650 0.699 136.087  
## 2 0.0282 0.003410 0.042800 0.2600 0.609 139.432  
## 3 0.0840 0.171000 0.000000 0.0857 0.759 136.081  
## 4 0.0278 0.917000 0.264000 0.0836 0.382 93.226  
## 5 0.0595 0.000726 0.908000 0.1010 0.269 119.956  
## 6 0.0269 0.001190 0.026100 0.1600 0.743 112.710  
## time\_signature track\_genre multiple\_artists tempo\_cat  
## 1 4 electronic FALSE Allegro  
## 2 4 power-pop FALSE Allegro  
## 3 4 k-pop FALSE Allegro  
## 4 3 honky-tonk FALSE Andante  
## 5 4 detroit-techno FALSE Moderato  
## 6 4 punk-rock FALSE Moderato

Let’s examine all of the variables of the Spotify dataset

names(dd)

## [1] "popularity" "duration\_ms" "explicit" "danceability"   
## [5] "energy" "key" "loudness" "mode"   
## [9] "speechiness" "acousticness" "instrumentalness" "liveness"   
## [13] "valence" "tempo" "time\_signature" "track\_genre"   
## [17] "multiple\_artists" "tempo\_cat"

The dataset has 3000 rows and 18 columns

dim(dd)

## [1] 3000 18

Summary statistics of the 18 variables in the dataset. We can observe that some were categorical and some were numerical.

summary(dd)

## popularity duration\_ms explicit danceability   
## Min. : 0.00 Min. : 28946 Length:3000 Min. :0.0000   
## 1st Qu.:19.00 1st Qu.:167156 Class :character 1st Qu.:0.4270   
## Median :30.00 Median :215889 Mode :character Median :0.5480   
## Mean :32.78 Mean :224103 Mean :0.5410   
## 3rd Qu.:46.00 3rd Qu.:268796 3rd Qu.:0.6653   
## Max. :97.00 Max. :594533 Max. :0.9750   
## energy key loudness mode   
## Min. :0.000242 Length:3000 Min. :-42.631 Length:3000   
## 1st Qu.:0.423000 Class :character 1st Qu.:-11.155 Class :character   
## Median :0.666500 Mode :character Median : -7.498 Mode :character   
## Mean :0.624997 Mean : -8.888   
## 3rd Qu.:0.861250 3rd Qu.: -5.181   
## Max. :0.999000 Max. : 0.377   
## speechiness acousticness instrumentalness liveness   
## Min. :0.00000 Min. :0.000001 Min. :0.0000000 Min. :0.0112   
## 1st Qu.:0.03470 1st Qu.:0.013300 1st Qu.:0.0000000 1st Qu.:0.1010   
## Median :0.04690 Median :0.217000 Median :0.0000889 Median :0.1410   
## Mean :0.07862 Mean :0.348478 Mean :0.1887061 Mean :0.2379   
## 3rd Qu.:0.07490 3rd Qu.:0.678250 3rd Qu.:0.1462500 3rd Qu.:0.3070   
## Max. :0.96200 Max. :0.996000 Max. :1.0000000 Max. :0.9920   
## valence tempo time\_signature track\_genre   
## Min. :0.0000 Min. : 0.0 Min. :0.000 Length:3000   
## 1st Qu.:0.2500 1st Qu.:100.0 1st Qu.:4.000 Class :character   
## Median :0.4730 Median :121.9 Median :4.000 Mode :character   
## Mean :0.4771 Mean :121.9 Mean :3.894   
## 3rd Qu.:0.6933 3rd Qu.:139.7 3rd Qu.:4.000   
## Max. :0.9850 Max. :220.0 Max. :5.000   
## multiple\_artists tempo\_cat   
## Mode :logical Length:3000   
## FALSE:2927 Class :character   
## TRUE :73 Mode :character   
##   
##   
##

## Set a list of numerical variables

Based from the dataset and earlier PCA analysis, we were interested and pre-selected 5 numerical variables to explore:

* loudness
* energy
* acousticness
* instrumentalness
* valence

attach(dd)  
  
numeric\_vars <- c("loudness", "energy", "acousticness", "instrumentalness", "valence")  
dcon <- dd %>% select(all\_of(numeric\_vars))  
head(dcon)

## loudness energy acousticness instrumentalness valence  
## 1 -5.274 0.854 0.049800 0.000253 0.699  
## 2 -3.464 0.855 0.003410 0.042800 0.609  
## 3 -5.308 0.617 0.171000 0.000000 0.759  
## 4 -13.106 0.235 0.917000 0.264000 0.382  
## 5 -10.344 0.714 0.000726 0.908000 0.269  
## 6 -6.901 0.771 0.001190 0.026100 0.743

## K-means Clustering

Using K-means clustering, we ran and test using different number of classes. As of the moment, we do not have knowledge to how many classes to partition our clusters. We can try to test different number of classes until we find the most optimal.

For our first try, we split the dataset into 5 classes and run it through kmeans() function.

k1 <- kmeans(dcon, 5)  
names(dcon)

## [1] "loudness" "energy" "acousticness" "instrumentalness"  
## [5] "valence"

print(k1)

## K-means clustering with 5 clusters of sizes 970, 133, 1000, 268, 629  
##   
## Cluster means:  
## loudness energy acousticness instrumentalness valence  
## 1 -7.514068 0.6587546 0.3123390 0.13592307 0.5194282  
## 2 -25.251895 0.1616488 0.8050250 0.74687103 0.1621206  
## 3 -4.325700 0.8448720 0.1457188 0.07555445 0.5265169  
## 4 -16.653843 0.2919045 0.6755804 0.45942345 0.3105297  
## 5 -11.491707 0.4632712 0.4906543 0.21662819 0.4710480  
##   
## Clustering vector:  
## [1] 3 3 3 5 5 1 3 5 1 1 3 5 1 1 2 5 1 3 1 1 5 5 3 1 1 5 3 5 3 3 3 3 1 2 3 3 1  
## [38] 4 1 3 4 3 3 4 5 4 3 3 3 3 4 5 3 1 3 2 1 1 3 3 5 3 3 3 1 3 1 4 3 5 1 1 3 1  
## [75] 3 1 3 1 1 3 5 5 3 1 2 1 3 1 1 1 3 5 5 3 3 3 1 1 1 5 1 5 3 5 3 3 4 1 1 1 5  
## [112] 5 1 3 3 3 4 4 5 3 1 4 3 5 5 4 1 1 5 5 3 3 1 1 3 5 1 3 3 1 1 1 5 3 4 3 3 2  
## [149] 1 3 3 2 2 5 1 5 4 3 1 3 3 3 4 1 1 3 3 3 5 3 3 1 1 2 1 1 1 5 3 3 1 1 1 4 3  
## [186] 3 1 3 1 4 3 5 1 3 3 3 1 2 3 3 3 2 1 3 1 3 1 5 5 1 1 5 1 3 4 1 1 5 4 1 3 5  
## [223] 4 1 3 5 3 1 1 5 3 1 3 2 1 1 3 3 1 3 1 4 5 4 4 3 1 5 3 5 1 1 1 1 3 1 3 5 3  
## [260] 1 2 3 4 3 4 3 3 1 3 3 1 4 5 4 1 1 1 1 5 5 1 1 5 3 3 1 3 1 1 4 5 3 5 3 3 1  
## [297] 1 1 2 3 1 1 3 1 1 3 2 3 2 5 1 5 1 1 4 3 4 3 1 1 5 1 4 3 1 3 4 3 5 5 5 3 1  
## [334] 3 3 1 5 5 1 5 1 1 5 4 3 3 1 3 1 3 4 5 3 3 1 5 1 3 3 3 5 1 1 1 5 3 1 5 3 4  
## [371] 3 5 3 1 3 3 1 1 1 3 3 1 4 1 5 5 1 3 5 3 1 5 1 1 5 3 5 5 4 3 3 4 5 1 2 1 5  
## [408] 2 5 1 4 5 2 4 1 5 3 4 3 1 3 2 5 3 5 1 1 4 1 1 3 1 3 5 3 3 5 5 3 4 1 1 3 5  
## [445] 5 1 4 1 1 3 5 1 1 2 5 1 5 3 3 2 5 5 5 5 5 1 1 1 3 1 1 3 1 3 3 3 5 5 1 4 4  
## [482] 4 5 5 5 2 3 1 1 3 3 2 4 5 1 1 4 5 5 3 3 5 3 3 3 3 4 3 5 3 1 4 1 1 3 3 1 4  
## [519] 3 1 1 3 3 1 1 3 5 1 3 1 1 5 1 4 3 5 1 3 4 1 5 1 1 4 5 1 3 5 3 3 3 3 2 3 1  
## [556] 1 1 3 3 3 2 3 3 4 5 5 3 5 1 3 3 3 5 1 5 5 1 3 4 3 5 5 3 1 3 1 3 5 3 3 3 1  
## [593] 1 1 2 1 4 5 1 3 5 3 1 1 3 5 4 3 3 5 5 1 3 5 1 5 4 2 1 3 1 3 5 1 1 5 3 1 4  
## [630] 1 5 1 3 3 3 3 5 2 4 3 1 1 3 1 1 3 1 3 1 4 3 5 3 3 1 5 3 4 5 5 5 5 3 3 3 3  
## [667] 3 3 5 3 5 4 1 3 1 4 5 4 5 3 3 2 5 3 3 1 3 3 1 3 3 2 1 1 3 1 1 5 1 1 5 3 1  
## [704] 1 3 3 3 3 3 1 2 5 1 3 4 5 2 1 5 3 5 4 3 3 1 3 3 1 3 5 4 3 3 3 1 5 2 5 3 3  
## [741] 5 3 1 3 3 5 3 2 1 5 3 4 1 5 3 5 5 3 3 3 3 5 3 4 5 4 5 5 4 5 1 1 1 5 5 3 1  
## [778] 3 2 3 3 3 1 1 5 5 1 3 1 4 1 5 2 4 4 3 3 3 5 1 4 1 3 3 5 5 5 3 1 3 3 3 5 3  
## [815] 1 1 3 1 1 5 1 1 4 1 1 3 1 5 3 4 3 5 5 3 3 3 4 5 3 3 4 1 1 5 3 1 3 1 3 1 1  
## [852] 1 3 5 3 5 5 1 3 1 3 1 3 5 5 1 4 4 1 3 5 3 4 5 5 2 5 1 1 5 1 1 1 1 1 3 3 1  
## [889] 3 2 3 1 5 1 1 1 1 3 1 3 4 3 3 5 2 3 3 3 3 3 1 5 1 3 3 1 3 4 5 1 1 4 3 1 5  
## [926] 4 1 5 3 5 3 1 3 1 1 1 1 3 1 5 3 3 1 3 1 3 3 3 4 1 4 3 5 4 3 4 3 5 3 5 1 3  
## [963] 3 3 5 4 1 3 3 3 3 1 5 3 3 1 5 3 3 3 3 5 1 1 3 1 3 4 3 1 4 5 3 3 1 3 1 4 1  
## [1000] 2 5 1 5 1 3 1 2 5 5 3 3 5 1 3 1 3 3 3 5 5 5 1 1 5 1 1 3 1 3 5 5 1 4 3 1 5  
## [1037] 1 5 4 2 3 5 1 3 5 1 4 1 5 3 3 3 5 5 1 1 5 4 1 5 3 5 1 5 5 5 5 5 3 3 1 3 1  
## [1074] 5 1 3 1 1 5 3 1 3 1 2 1 5 5 5 4 4 1 3 3 5 1 3 1 3 4 5 1 1 3 3 1 3 1 3 3 5  
## [1111] 3 1 3 3 3 3 5 1 1 5 3 3 1 1 3 3 5 1 1 5 3 1 5 3 1 3 5 4 1 1 4 5 4 4 3 5 3  
## [1148] 1 5 5 1 1 5 5 3 3 5 4 1 1 3 3 2 3 3 1 3 2 4 5 5 4 2 3 3 3 3 3 3 1 5 3 1 3  
## [1185] 1 4 1 5 5 1 4 1 1 2 3 1 1 1 1 4 1 2 4 3 3 3 1 1 5 2 5 2 3 4 5 1 2 5 5 1 3  
## [1222] 3 2 3 1 3 5 4 1 4 5 1 3 4 5 5 3 5 3 1 1 3 1 3 3 3 3 1 4 3 5 1 1 1 5 1 1 1  
## [1259] 1 3 1 1 1 5 5 3 1 5 4 1 1 3 5 3 3 4 1 3 5 1 3 3 3 3 4 4 1 1 1 1 3 3 1 1 1  
## [1296] 3 3 3 3 3 5 5 2 5 4 5 5 1 3 3 4 3 3 1 1 5 3 3 3 3 2 1 5 1 1 5 3 5 2 3 2 4  
## [1333] 3 5 3 1 1 5 5 1 1 3 1 1 1 3 1 1 3 3 3 5 3 3 4 1 4 1 4 1 5 1 2 5 3 1 5 3 2  
## [1370] 3 5 1 1 1 2 1 1 5 3 3 4 3 1 1 3 5 5 4 1 1 4 3 3 3 3 3 4 5 3 4 3 5 3 3 3 5  
## [1407] 1 5 1 3 1 3 5 3 3 1 3 1 3 1 1 5 1 1 1 4 1 3 5 1 3 3 3 3 3 3 1 3 1 3 3 1 1  
## [1444] 5 3 3 3 4 3 4 1 2 5 5 1 1 4 3 1 1 5 4 1 3 5 5 1 1 1 1 5 3 1 1 5 1 4 3 3 3  
## [1481] 1 1 3 1 1 4 3 5 4 3 3 2 1 5 3 1 5 3 3 3 2 3 4 4 5 1 3 3 1 3 3 3 5 3 4 5 3  
## [1518] 1 3 3 1 1 3 3 4 1 2 5 5 3 1 5 5 1 1 5 1 3 1 5 3 1 1 3 1 3 1 4 5 3 5 5 3 5  
## [1555] 3 1 3 1 3 1 3 1 3 5 1 3 3 1 1 1 1 1 5 1 1 2 3 1 3 3 1 1 3 3 5 3 3 1 1 3 3  
## [1592] 3 3 3 5 3 2 1 3 1 3 3 1 4 3 1 4 1 1 5 3 1 3 1 1 4 5 1 3 1 3 1 1 5 1 4 5 1  
## [1629] 5 2 1 1 1 3 1 1 1 4 3 5 1 1 4 3 3 5 1 3 1 1 5 5 1 1 3 3 5 5 1 5 1 1 3 4 1  
## [1666] 5 3 1 3 1 5 3 3 3 5 1 3 4 3 4 1 3 2 5 1 4 4 5 3 1 5 3 5 3 3 1 3 5 3 1 5 1  
## [1703] 3 1 5 3 5 3 5 3 1 1 3 1 1 5 1 1 1 2 1 3 3 3 4 4 3 1 5 2 5 4 3 4 3 1 5 3 2  
## [1740] 3 5 1 3 1 1 4 5 1 3 3 5 5 3 5 3 3 1 5 2 1 1 1 3 3 3 3 3 1 1 4 5 3 3 3 3 1  
## [1777] 4 3 2 1 5 1 5 3 5 5 1 1 5 5 5 3 4 4 3 5 5 5 3 3 1 3 3 1 1 1 1 3 1 1 3 3 5  
## [1814] 3 3 1 3 5 5 1 3 5 1 2 2 1 1 5 4 3 4 1 5 1 5 1 3 4 1 1 5 1 4 5 1 1 1 1 2 2  
## [1851] 1 5 1 2 3 5 1 1 1 3 3 1 5 3 3 3 1 1 5 3 3 1 1 1 3 1 5 1 3 1 3 3 3 2 4 2 5  
## [1888] 3 1 3 2 1 1 1 3 3 1 5 1 1 2 3 3 3 3 2 3 4 5 3 5 5 1 5 1 3 5 5 1 3 1 3 4 3  
## [1925] 1 1 4 5 3 1 5 4 3 1 3 1 1 4 5 1 3 1 5 5 1 5 4 1 1 4 1 1 1 1 1 5 4 2 1 3 1  
## [1962] 1 2 1 1 1 3 3 5 3 5 2 3 5 2 5 5 5 1 5 3 4 5 3 3 1 5 4 5 2 5 3 3 3 5 5 1 3  
## [1999] 1 3 3 1 2 5 3 3 5 1 5 5 1 3 5 2 5 3 3 5 3 3 5 5 1 3 1 1 3 3 5 1 4 5 5 3 1  
## [2036] 1 4 1 5 1 3 1 5 1 3 5 1 3 3 4 1 4 1 5 1 4 1 3 3 5 1 1 2 3 1 1 3 5 1 3 1 5  
## [2073] 3 1 1 3 3 5 1 3 1 1 3 1 3 3 3 3 3 3 3 3 5 1 4 1 3 1 3 1 3 3 5 2 1 3 3 1 5  
## [2110] 2 5 4 3 4 1 2 3 3 3 1 5 1 1 1 3 3 3 5 1 5 1 3 5 5 1 5 4 1 3 5 3 3 3 1 1 3  
## [2147] 1 3 5 5 3 4 1 4 3 1 1 1 3 3 5 4 1 1 1 3 1 4 1 4 5 1 5 3 5 5 3 3 1 5 1 3 1  
## [2184] 4 1 3 1 4 1 1 3 3 3 3 3 1 5 5 4 3 3 5 1 1 3 1 1 3 3 5 4 5 1 4 1 3 3 3 3 3  
## [2221] 3 3 2 3 1 5 3 1 3 5 1 3 3 5 5 3 3 1 3 3 5 5 3 1 2 5 4 2 3 1 5 1 1 5 1 1 1  
## [2258] 3 4 3 1 1 5 5 2 3 3 3 1 1 1 1 2 2 1 3 1 3 1 1 3 3 1 1 5 4 1 3 3 5 5 1 3 5  
## [2295] 3 3 1 3 3 5 1 1 3 1 3 1 1 5 3 5 1 1 3 1 4 5 2 3 3 1 4 3 3 1 5 5 1 5 1 1 1  
## [2332] 5 5 3 1 5 1 3 4 1 3 3 3 5 5 3 4 1 3 2 3 1 3 3 3 1 5 3 5 3 1 1 3 4 3 5 3 5  
## [2369] 3 1 3 3 2 1 1 5 3 3 1 1 3 5 5 3 3 5 3 3 3 5 3 1 2 5 1 2 4 4 1 1 1 3 3 4 3  
## [2406] 3 5 5 3 4 4 3 1 5 3 1 1 1 3 1 1 4 5 1 5 4 3 3 3 4 3 1 1 5 5 3 5 3 5 1 4 3  
## [2443] 3 4 2 3 5 3 1 1 5 3 3 5 5 1 3 1 1 1 3 3 5 5 3 4 4 3 1 5 5 5 5 5 1 3 3 5 2  
## [2480] 5 1 3 3 1 3 4 5 1 5 5 1 5 5 5 3 3 1 1 1 1 4 2 3 1 1 5 4 1 5 5 4 5 3 1 1 3  
## [2517] 3 5 1 5 1 2 3 1 5 1 1 3 3 2 4 1 5 1 3 1 2 1 3 5 5 1 3 4 5 1 5 2 3 4 3 2 1  
## [2554] 3 1 1 5 5 3 3 1 5 5 3 3 1 1 1 3 5 5 3 1 1 4 3 1 5 3 1 1 3 1 1 3 3 4 5 1 1  
## [2591] 3 3 5 1 3 5 3 3 5 1 1 1 4 2 3 3 2 5 1 1 5 5 1 5 3 1 5 4 1 2 1 3 1 1 1 1 4  
## [2628] 4 5 5 3 1 5 3 3 1 5 2 5 1 3 3 1 4 4 1 1 2 5 1 4 1 3 3 4 4 3 3 1 2 1 4 3 3  
## [2665] 5 5 1 5 1 3 1 5 1 3 5 3 1 1 1 1 4 3 1 2 5 3 1 3 5 1 3 3 3 2 4 3 5 1 1 3 1  
## [2702] 3 4 1 3 5 5 3 5 1 5 1 1 3 3 5 3 5 1 1 1 3 1 3 3 5 4 3 1 3 4 5 3 1 5 5 1 3  
## [2739] 5 1 3 1 3 1 5 3 1 2 1 4 5 3 4 1 1 5 3 3 1 5 4 1 1 3 4 1 5 3 1 1 1 4 1 3 2  
## [2776] 5 5 3 1 1 1 1 3 1 5 1 1 4 5 5 1 2 1 1 3 1 4 1 3 3 1 5 3 1 1 5 1 1 5 5 3 3  
## [2813] 1 1 1 3 3 1 1 1 3 3 1 4 3 1 3 3 1 2 3 3 3 3 4 1 3 5 5 3 1 3 1 1 5 4 3 3 1  
## [2850] 5 1 5 1 5 3 2 3 2 3 5 4 5 1 5 3 3 4 3 3 2 1 5 1 4 1 3 1 1 4 1 1 3 3 4 4 5  
## [2887] 3 1 3 1 1 1 1 3 5 5 3 3 5 3 1 4 3 1 2 2 1 1 1 3 4 1 3 1 3 5 3 5 2 1 1 3 1  
## [2924] 1 3 5 3 1 3 5 3 3 4 5 1 5 5 2 3 1 3 1 5 4 5 3 3 1 5 1 4 5 5 5 5 5 1 5 5 3  
## [2961] 3 1 2 1 3 3 3 3 3 3 3 5 3 1 1 1 3 3 5 1 5 1 1 3 4 3 3 5 1 1 5 2 5 1 1 5 1  
## [2998] 5 1 4  
##   
## Within cluster sum of squares by cluster:  
## [1] 1282.044 1993.708 1308.479 1215.095 1274.792  
## (between\_SS / total\_SS = 91.8 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"

The result kmeans object also provided us with useful attributes

attributes(k1)

## $names  
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"   
##   
## $class  
## [1] "kmeans"

k1$size

## [1] 970 133 1000 268 629

k1$withinss

## [1] 1282.044 1993.708 1308.479 1215.095 1274.792

k1$centers

## loudness energy acousticness instrumentalness valence  
## 1 -7.514068 0.6587546 0.3123390 0.13592307 0.5194282  
## 2 -25.251895 0.1616488 0.8050250 0.74687103 0.1621206  
## 3 -4.325700 0.8448720 0.1457188 0.07555445 0.5265169  
## 4 -16.653843 0.2919045 0.6755804 0.45942345 0.3105297  
## 5 -11.491707 0.4632712 0.4906543 0.21662819 0.4710480

### Decomposition of Inertia

Let’s compute for decomposition of inertia

Bss <- sum(rowSums(k1$centers^2)\*k1$size)  
Bss

## [1] 318346.5

Wss <- sum(k1$withinss)  
Wss

## [1] 7074.118

Tss <- k1$totss  
Tss

## [1] 86098.5

Bss+Wss

## [1] 325420.6

Ib1 <- 100\*Bss/(Bss+Wss)  
Ib1

## [1] 97.82616

Let’s repeat k-means run with k=5

k2 <- kmeans(dcon,5)  
k2$size

## [1] 1000 268 970 629 133

Bss <- sum(rowSums(k2$centers^2)\*k2$size)  
Bss

## [1] 318346.5

Wss <- sum(k2$withinss)  
Wss

## [1] 7074.118

Ib2 <- 100\*Bss/(Bss+Wss)  
Ib2

## [1] 97.82616

Examine the centers of k1 and k2

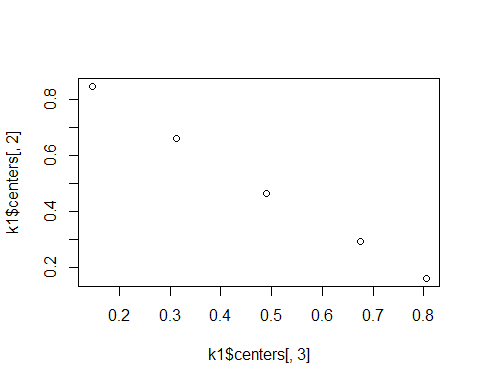
k2$centers

## loudness energy acousticness instrumentalness valence  
## 1 -4.325700 0.8448720 0.1457188 0.07555445 0.5265169  
## 2 -16.653843 0.2919045 0.6755804 0.45942345 0.3105297  
## 3 -7.514068 0.6587546 0.3123390 0.13592307 0.5194282  
## 4 -11.491707 0.4632712 0.4906543 0.21662819 0.4710480  
## 5 -25.251895 0.1616488 0.8050250 0.74687103 0.1621206

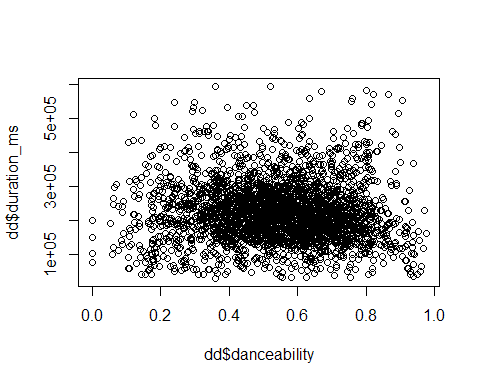
k1$centers

## loudness energy acousticness instrumentalness valence  
## 1 -7.514068 0.6587546 0.3123390 0.13592307 0.5194282  
## 2 -25.251895 0.1616488 0.8050250 0.74687103 0.1621206  
## 3 -4.325700 0.8448720 0.1457188 0.07555445 0.5265169  
## 4 -16.653843 0.2919045 0.6755804 0.45942345 0.3105297  
## 5 -11.491707 0.4632712 0.4906543 0.21662819 0.4710480

plot(k1$centers[,3],k1$centers[,2])



plot(dd$danceability, dd$duration\_ms)



table(k1$cluster, k2$cluster)

##   
## 1 2 3 4 5  
## 1 0 0 970 0 0  
## 2 0 0 0 0 133  
## 3 1000 0 0 0 0  
## 4 0 268 0 0 0  
## 5 0 0 0 629 0

Why did we obtained a different results? Which run is better between different k-means we applied previously?

Let’s try k=8

k3 <- kmeans(dcon,8)  
k3$size

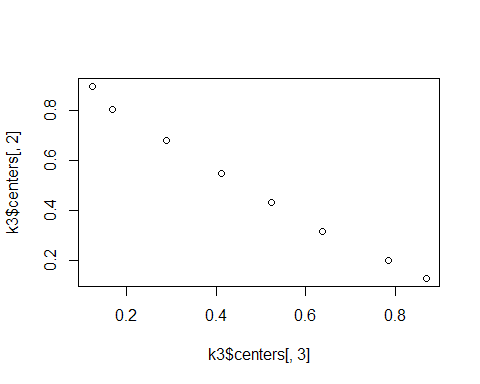
## [1] 508 659 381 145 210 641 51 405

Bss <- sum(rowSums(k3$centers^2)\*k3$size)  
Wss <- sum(k3$withinss)  
  
Ib3 <- 100\*Bss/(Bss+Wss)  
Ib3

## [1] 98.91375

Plotting k3 centers

plot(k3$centers[,3],k3$centers[,2])

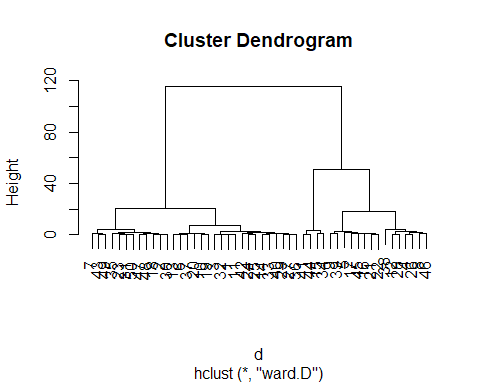


## Hierarchical Clustering

Keep watch of processing hclust on huge datasets. Somehow, running at 10,000 rows would be optimal. In our Spotify dataset case, we narrowed it down to 3,000 rows to minimize run time.

We test first with the first 50 rows to see the basic structure.

d <- dist(dcon[1:50,])  
h1 <- hclust(d,method="ward.D")  
plot(h1)

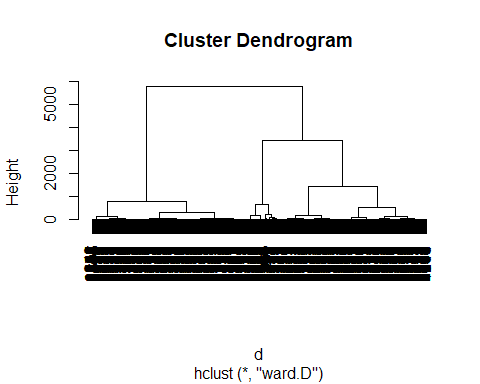


Then, we run the clustering with the purely numerical variable dataset we filtered out earlier.

d <- dist(dcon)  
# The "ward" method has been renamed to "ward.D"; note new "ward.D2"  
h1 <- hclust(d,method="ward.D") # Keep in watch of the memory usage when running huge datasets  
h1

##   
## Call:  
## hclust(d = d, method = "ward.D")  
##   
## Cluster method : ward.D   
## Distance : euclidean   
## Number of objects: 3000

plot(h1)



## Where will you cut the dendrogram and how many classes will we obtain?

Based from the above dendrogram (hierarchical clusters), we can see between 3-4 major group branches that can be identified visually. We have decided to use 4 classes throughout the next steps in further clustering the Spotify dataset.

nc <- 4  
  
c1 <- cutree(h1,nc)  
c1[1:20]

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

Let’s try to cut the tree into 5

nc <- 5  
  
c5 <- cutree(h1,nc)  
c5[1:20]

## [1] 1 2 1 3 3 1 2 3 4 1 1 3 1 1 5 3 4 2 4 1

Let’s tabulate the number of elements for each branch of the cut tree c1

table(c1)

## c1  
## 1 2 3 4   
## 1371 716 602 311

Also, we do it with c5

table(c5)

## c5  
## 1 2 3 4 5   
## 910 461 716 602 311

Cross tabulating c1 and c5 gives us a split between first and second rows of each table

table(c1,c5)

## c5  
## c1 1 2 3 4 5  
## 1 910 461 0 0 0  
## 2 0 0 716 0 0  
## 3 0 0 0 602 0  
## 4 0 0 0 0 311

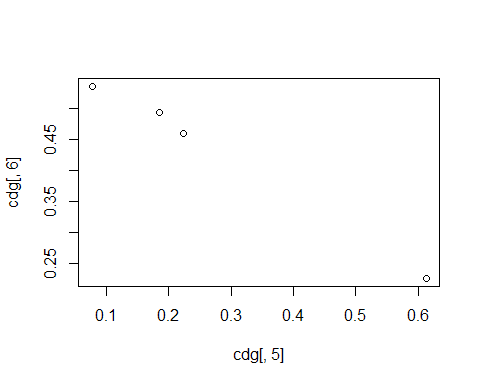
Let’s aggregate the values of every variable for every group based on how we cut it in the tree

cdg <- aggregate(as.data.frame(dcon), list(c1), mean)  
cdg

## Group.1 loudness energy acousticness instrumentalness valence  
## 1 1 -4.905945 0.8101400 0.1832846 0.07652361 0.5362125  
## 2 2 -11.880494 0.4459659 0.5002814 0.22287230 0.4594306  
## 3 3 -8.172080 0.6269236 0.3307687 0.18422472 0.4932940  
## 4 4 -20.939823 0.2172659 0.7614973 0.61326230 0.2262794

Let’s examine the aggregated values between intrumentalness and valence

plot(cdg[,5], cdg[,6])



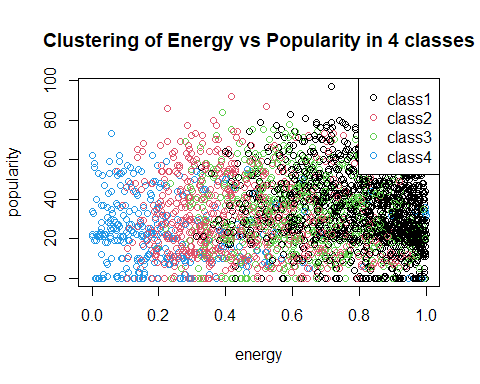
## Visual partition of the Clusters

Cross comparing every numerical variables of the Spotify dataset, we may observe that some combinations create patterns while some does not.

### Are Energetic Songs More Popular? (Energy vs Popularity)

The partitions of the clusters comparing popularity and energy seems to overlap with each other. It seems that songs can still be popular whether it has low or high energy.

# Need to run together  
plot(energy, popularity, col=c1, main="Clustering of Energy vs Popularity in 4 classes")  
legend("topright", c("class1", "class2", "class3", "class4"), pch=1, col=c(1:4))



### Are More Energetic Songs Louder? (Energy vs Loudness)

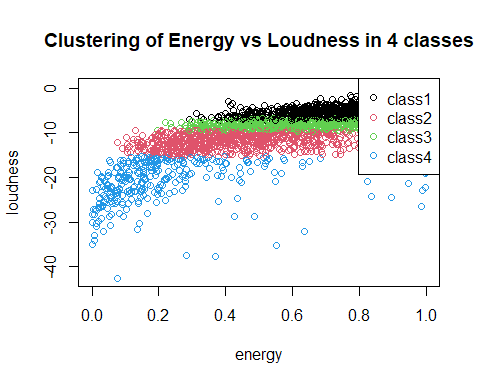
Also by using basic common sense, the higher the energy, the louder it is.

Energy and loudness seems to follow a logarithmic relationship. The higher the songs energy, the loudness of the song reaches a certain limit.

If we review our knowledge in Physics, the decibel formula is calculated as a logarithm of sound intensity. That could explain why this trend shows.

The clusters were grouped into 4 classes in terms of loudness. The class grouping were not as obvious as it seems it follows a logarithmic trend and could work better with regression.

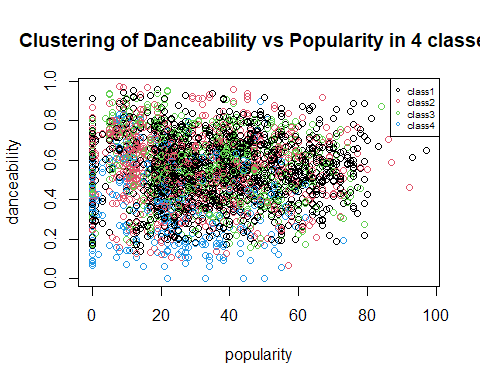
plot(energy, loudness, col=c1, main="Clustering of Energy vs Loudness in 4 classes")  
legend("topright", c("class1", "class2", "class3", "class4"), pch=1, col=c(1:4))



### Are Popular Songs Danceable? (Danceability vs Popularity)

The sweet spot between danceability and popularity is somewhere in the middle. Too popular or too indie might not make a song danceable.

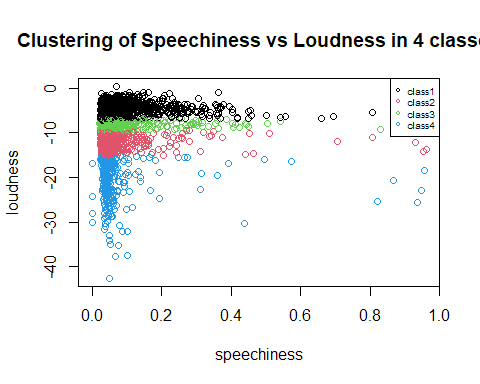
plot(popularity, danceability, col=c1, main="Clustering of Danceability vs Popularity in 4 classes")  
legend("topright",c("class1", "class2", "class3", "class4"),pch=1,col=c(1:4), cex=0.6)



### Does rapping makes a song louder? (Speechiness vs Loudness)

Louder songs seem to use lower words

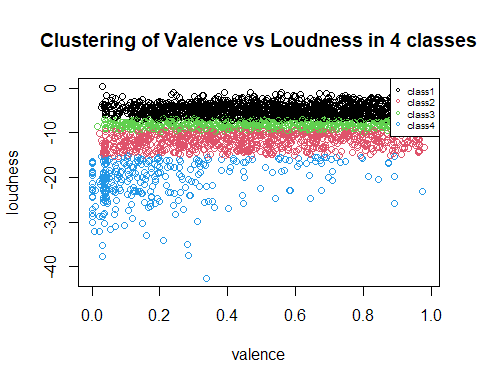
plot(speechiness, loudness, col=c1,main="Clustering of Speechiness vs Loudness in 4 classes")  
legend("topright",c("class1", "class2", "class3", "class4"),pch=1,col=c(1:4), cex=0.6)



### Are Happier Songs Louder Than Sad Songs? (Valence vs Loudness)

Songs whether happy or sad can still be loud based on how the clusters spread out. The classes were also partitioned according to loudness..

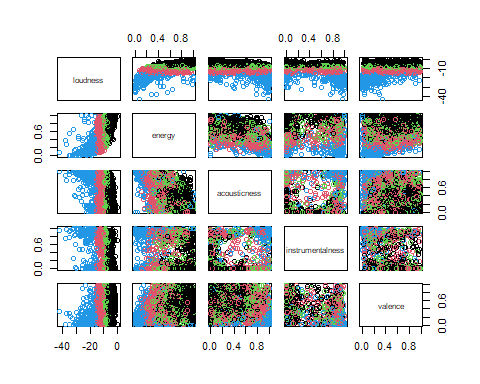
plot(valence, loudness ,col=c1,main="Clustering of Valence vs Loudness in 4 classes")  
legend("topright",c("class1", "class2", "class3", "class4"),pch=1,col=c(1:4), cex=0.6)



### Visualizing all the variables altogether into matrix

The partitioning seems to have been based from loudness by the clustering algorithm.

pairs(dcon, col=c1)  
legend("topright",c("class1", "class2", "class3", "class4"),pch=1,col=c(1:4), cex=0.6)



### Quality of the Hierarchical Partition

Bss <- sum(rowSums(cdg^2)\*as.numeric(table(c1)))  
  
Ib4 <- 100\*Bss/Tss  
Ib4

## [1] 380.8549

### Move to Gower mixed distance to deal

Simultaneously with numerical and qualitative data. Gower’s Distance can be used to measure how different two records are. The distance is always a number between 0 (identical) and 1 (maximally dissimilar).

More info on Gower’s Distance: <https://medium.com/analytics-vidhya/gowers-distance-899f9c4bd553>

## Dissimilarity matrix

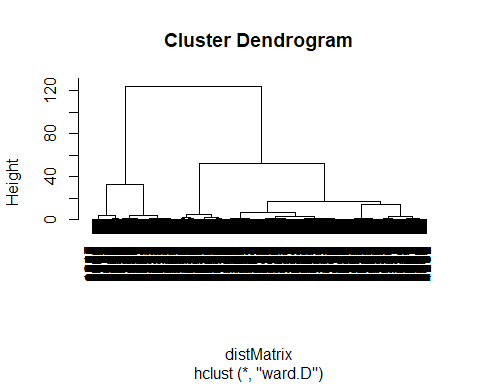
Let’s compute the pairwise dissimilarities between observations in the data set that sets to Gower’s Distance. More info on the daisy() function here: <https://stat.ethz.ch/R-manual/R-devel/library/cluster/html/daisy.html>

dissimMatrix <- daisy(dd[numeric\_vars], metric = "gower", stand=TRUE)  
distMatrix<-dissimMatrix^2

### Running the hierarchical clustering for the distMatrix

Using the distMatrix data, generate the hierachical clustering chart using hclust()

h1 <- hclust(distMatrix, method="ward.D") # Keep watch of the memory  
plot(h1)



### Cutting the tree of the distMatrix

Based on the dendrogram, we can either cut the tree either by 3 or 4. Let’s use 4 as the number of cuts instead.

c2 <- cutree(h1, 4)  
  
# Class sizes   
table(c2)

## c2  
## 1 2 3 4   
## 1811 520 412 257

#comparing with other partitions  
table(c1, c2)

## c2  
## c1 1 2 3 4  
## 1 1172 72 124 3  
## 2 250 276 134 56  
## 3 364 105 116 17  
## 4 25 67 38 181

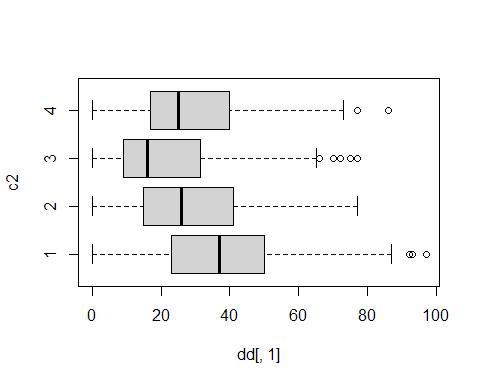
### Visualize distribution for the variable using boxplot

names(dd)

## [1] "popularity" "duration\_ms" "explicit" "danceability"   
## [5] "energy" "key" "loudness" "mode"   
## [9] "speechiness" "acousticness" "instrumentalness" "liveness"   
## [13] "valence" "tempo" "time\_signature" "track\_genre"   
## [17] "multiple\_artists" "tempo\_cat"

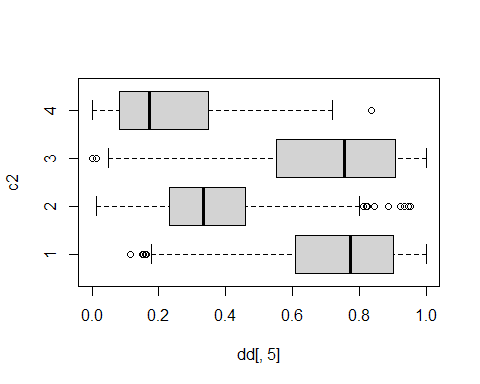
#### Popularity

boxplot(dd[,1]~c2, horizontal=TRUE)



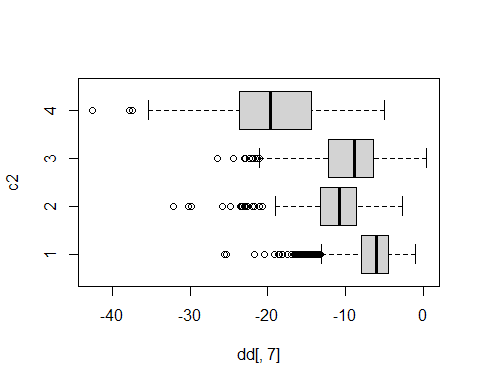
#### Energy

boxplot(dd[,5]~c2, horizontal=TRUE)



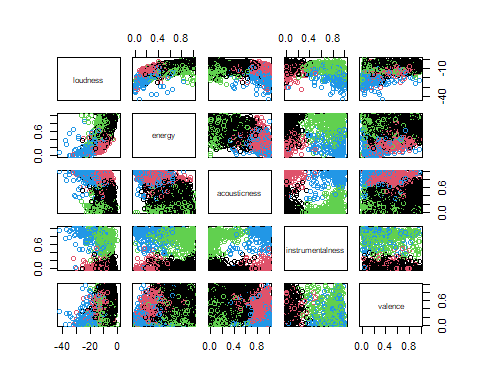
#### Loudness

boxplot(dd[,7]~c2, horizontal=TRUE)

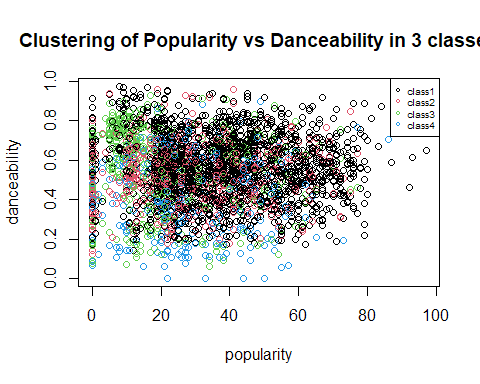


### Visualize distMatrix for every variable in a matrix

pairs(dcon, col=c2)



plot(popularity, danceability, col=c2, main="Clustering of Popularity vs Danceability in 3 classes")  
legend("topright",c("class1", "class2", "class3", "class4"),pch=1,col=c(1:4), cex=0.6)

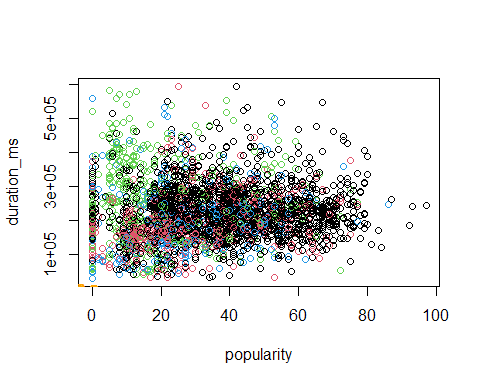


### Aggregate the distMatrix data

cdg <- aggregate(as.data.frame(dcon),list(c2),mean)  
cdg

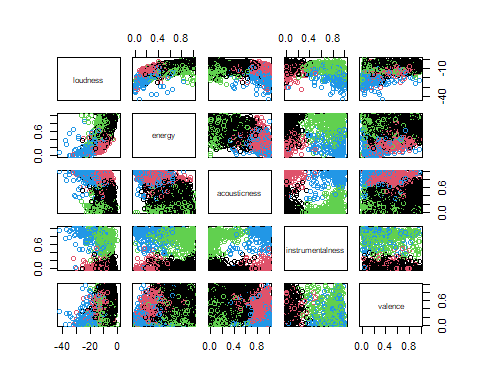
## Group.1 loudness energy acousticness instrumentalness valence  
## 1 1 -6.583336 0.7380751 0.19736549 0.01214573 0.5458278  
## 2 2 -11.235687 0.3561565 0.81888654 0.01971245 0.4513965  
## 3 3 -9.548485 0.7146766 0.09705143 0.77733010 0.3702006  
## 4 4 -19.320650 0.2283625 0.86458366 0.83117471 0.2167026

plot(popularity, duration\_ms, col= c2)  
points(cdg[,4],cdg[,5],pch=16,col="orange")  
text(cdg[,4],cdg[,5], labels=cdg[,1], pos=2, font=2, cex=0.7, col="orange")



### Plotting the same previous cluster but now partitioned visually by the distMatrix

pairs(dcon, col=c2)  
legend("topright",c("class1", "class2", "class3", "class4"),pch=1,col=c(1:4), cex=0.6)



### Steps before moving to Profiling

Add the cluster vector (c2 which was cut into 4) to the dataset (dcon) Save the dcon dataset which was used for the clustering and run it through the profiling script

dcon\_clust <- dcon %>% bind\_cols(tibble(cluster = c2))  
head(dcon\_clust)

## loudness energy acousticness instrumentalness valence cluster  
## 1 -5.274 0.854 0.049800 0.000253 0.699 1  
## 2 -3.464 0.855 0.003410 0.042800 0.609 1  
## 3 -5.308 0.617 0.171000 0.000000 0.759 1  
## 4 -13.106 0.235 0.917000 0.264000 0.382 2  
## 5 -10.344 0.714 0.000726 0.908000 0.269 3  
## 6 -6.901 0.771 0.001190 0.026100 0.743 1

The next phase which is Profiling is in the clustering-profiling.Rmd which will be using the dataset saved below

write.csv(dcon\_clust, file="data/SpotifyClustersData.csv", row.names=F)

## Check song if fits profile

Based from the charts above, we describe the classes with these profiles:

* Class 1 - Mostly loudest
* Class 1 & 3 - Most energetic
* Class 2 & 4 - Most acoustic
* Class 3 & 4 - Most instrumental
* From Happiest to saddest Class 1, 2, 3, 4

We then picked the most popular song for every class and see if the profiling fits that song:

dd\_clust <- read.csv("data/cleaneddata-withtitles.csv") %>%  
 bind\_cols(tibble(cluster = c2))  
head(dd\_clust)

## popularity duration\_ms explicit danceability energy key loudness mode  
## 1 38 153039 False 0.522 0.854 A -5.274 minor  
## 2 20 212840 False 0.544 0.855 C -3.464 major  
## 3 71 235779 False 0.856 0.617 A#/Bb -5.308 minor  
## 4 12 142759 False 0.478 0.235 D -13.106 major  
## 5 9 292217 False 0.786 0.714 A -10.344 minor  
## 6 38 194506 False 0.650 0.771 D -6.901 major  
## speechiness acousticness instrumentalness liveness valence tempo  
## 1 0.0431 0.049800 0.000253 0.2650 0.699 136.087  
## 2 0.0282 0.003410 0.042800 0.2600 0.609 139.432  
## 3 0.0840 0.171000 0.000000 0.0857 0.759 136.081  
## 4 0.0278 0.917000 0.264000 0.0836 0.382 93.226  
## 5 0.0595 0.000726 0.908000 0.1010 0.269 119.956  
## 6 0.0269 0.001190 0.026100 0.1600 0.743 112.710  
## time\_signature track\_genre multiple\_artists tempo\_cat  
## 1 4 electronic FALSE Allegro  
## 2 4 power-pop FALSE Allegro  
## 3 4 k-pop FALSE Allegro  
## 4 3 honky-tonk FALSE Andante  
## 5 4 detroit-techno FALSE Moderato  
## 6 4 punk-rock FALSE Moderato  
## artists album\_name  
## 1 Babasónicos Infame  
## 2 Hoodoo Gurus Mars Needs Guitars!  
## 3 BTS Love Yourself 承 'Her'  
## 4 George Jones Mr. Country and Western  
## 5 Omar S Thank You for Letting Me Be Myself  
## 6 Patricio Rey y sus Redonditos de Ricota Último Bondi a Finisterre  
## track\_name cluster  
## 1 Once 1  
## 2 The Other Side Of Paradise - 2005 Remaster 1  
## 3 Go Go 1  
## 4 Your Tender Years 2  
## 5 Dumpster Graves 3  
## 6 Alien Duce 1

names(dd\_clust)

## [1] "popularity" "duration\_ms" "explicit" "danceability"   
## [5] "energy" "key" "loudness" "mode"   
## [9] "speechiness" "acousticness" "instrumentalness" "liveness"   
## [13] "valence" "tempo" "time\_signature" "track\_genre"   
## [17] "multiple\_artists" "tempo\_cat" "artists" "album\_name"   
## [21] "track\_name" "cluster"

Here’s the top 2 songs for every cluster. Try to listen to each song and check if it fits the profile we created.

pop\_per\_class <- dd\_clust %>%  
 select(popularity, track\_name, artists, track\_genre, cluster) %>%  
 arrange(desc(popularity)) %>%  
 group\_by(cluster) %>%  
 top\_n(2, popularity) %>%  
 arrange(cluster)  
pop\_per\_class

## # A tibble: 9 × 5  
## # Groups: cluster [4]  
## popularity track\_name artists track\_genre cluster  
## <int> <chr> <chr> <chr> <int>  
## 1 97 Tití Me Preguntó Bad Bunny latin 1  
## 2 93 Sweater Weather The Neighbourhood alt-rock 1  
## 3 77 Your Power Billie Eilish electro 2  
## 4 75 I Didn't Change My Number Billie Eilish electro 2  
## 5 75 I Drink Wine Adele british 2  
## 6 77 Afraid The Neighbourhood alt-rock 3  
## 7 75 Sweet Cigarettes After Sex ambient 3  
## 8 86 everything i wanted Billie Eilish electro 4  
## 9 77 NDA Billie Eilish electro 4