Brief Intro To Clustering in R (And Also R Markdown)

Khoi Trinh

2023-02-21

Before working with R markdown, we need a few packages

```
# The rmarkdown package
#install.packages('rmarkdown')
```

If you want to generate pdf files, you will need to install LaTeX

If you don't plan on using LaTeX anywhere outside of R markdown, I suggest TinyTex

```
#install.packages('tinytex')
#tinytex::install_tinytex()
```

Note that you can and should run these above commands in the RStudio console

Let's Explore Clustering

```
library(cluster)
library(NbClust)
library(factoextra)
library(dplyr)
library(kmed)
```

First, we need some data

Data description

The data I chose is a dataset of tracks from Spotify. There are 586,672 observations; each of with has 20 variables. These include 12 numeric variables, and 7 factor variables.

Data source

The data I chose came from this kaggle URL: Spotify Dataset

Misc data processing:

The data included some variables that are binary, but were recorded as numeric. These included "explicit" which is a flag if a song has explicit lyrics, and "mode" which is a flag for major mode (1) or minor mode (0); both of these were converted to factor.

Additionally, there is the "key" factor which has values from 0 to 11; coinciding to the 12 keys in Western music: A, A#, B, C, C#, D, D#, E, F, G, G#. While this is not a binary feature, it should not be considered numeric, and will also be converted to factor.

While most of numeric data ranges from 0 to 1; tempo and length were not, so scaling was done to ensure the data is normalized.

Fortunately, the data does not have any missing values, so no imputations were needed. Clustering and the subsequent analysis can be performed.

```
spotify = read.csv("tracks.csv")

spotify$explicit <-as.factor(spotify$explicit)
spotify$mode <-as.factor(spotify$mode)
spotify$key <-as.factor(spotify$key)
```

The data had almost 600,000 observations, out of those, the top 500 observations will be chosen for the purpose of this experiment.

```
numericData = spotify %>% #Add data
  dplyr::select(where(is.numeric)) #finds where is.numeric is true
numericData <- head(numericData, 500) # gets only 500 observations
```

Time To Create Some Clusters

One last data processing step

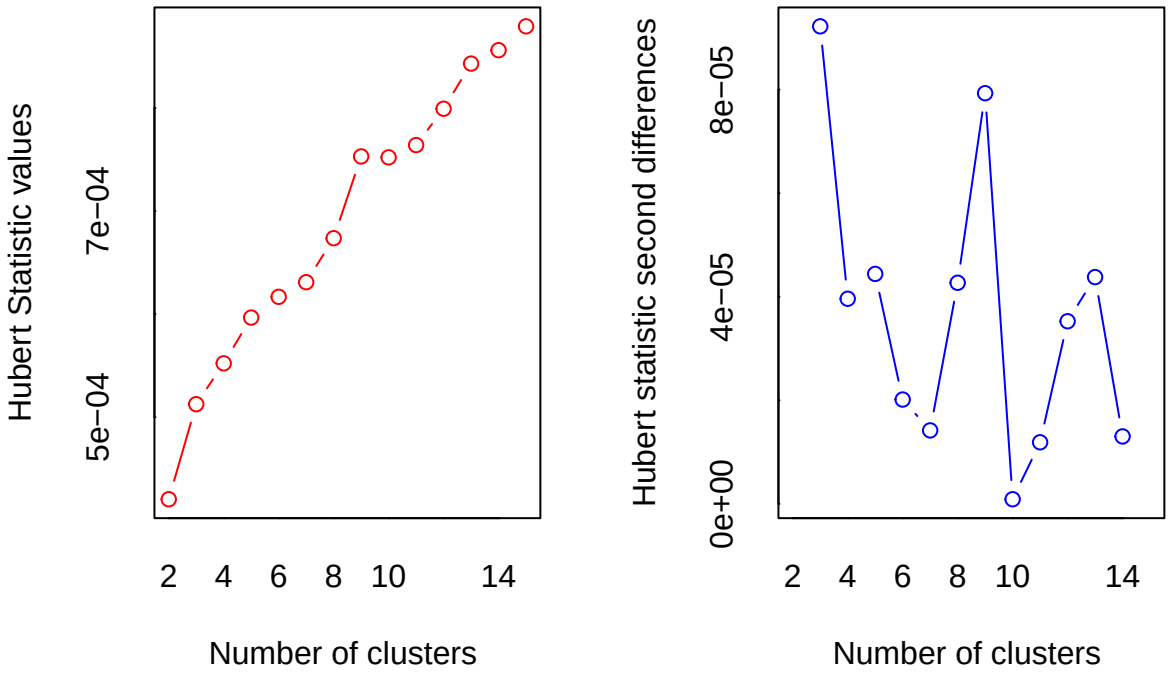
Scale the data, and use that to create our clusters.

```
clusterData <- scale(numericData)
```

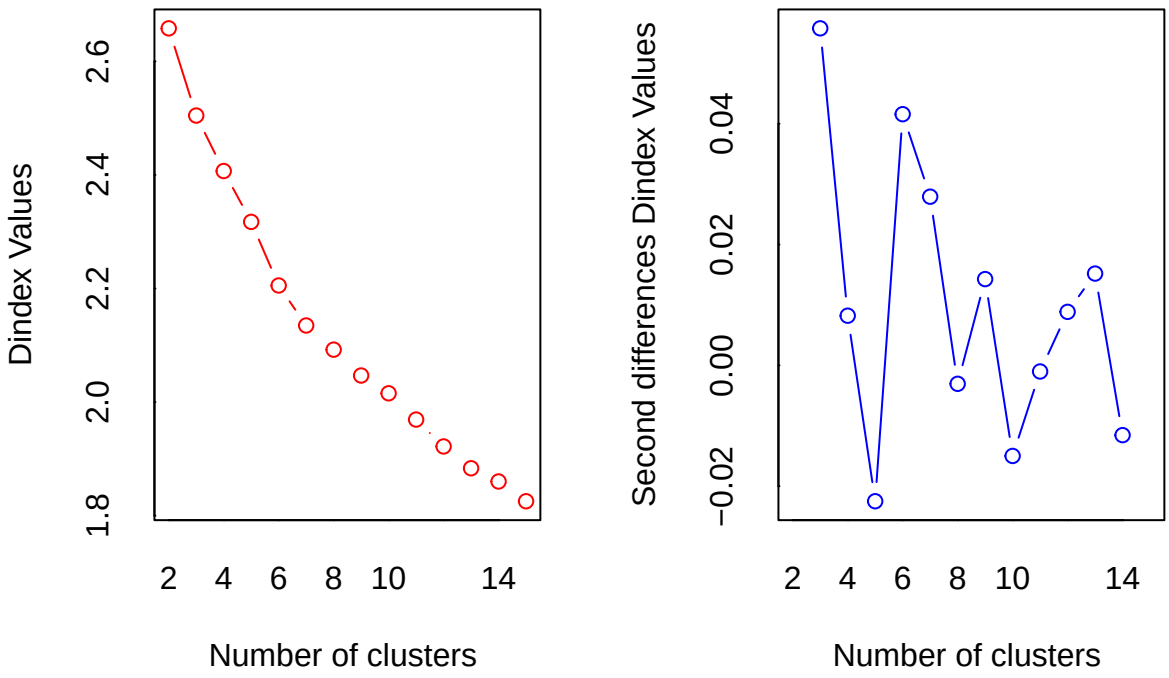
K means

K means clustering are done here with k = 2. This value of k was suggested using the NbClust() method from the `NbClust` package. One of the lines in the results states "According to the majority rule, the best number of clusters is 2"

```
NbClust(clusterData,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:
## * 7 proposed 2 as the best number of clusters
## * 3 proposed 3 as the best number of clusters
## * 3 proposed 4 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 4 proposed 9 as the best number of clusters
## * 1 proposed 11 as the best number of clusters
## * 1 proposed 12 as the best number of clusters
## * 3 proposed 15 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************

## $All.index
##        KL       CH Hartigan     CCC    Scott      Marriot    TrCovW   TraceW
```

```
## 2  3.8104 182.4061  67.0264  4.7093 1549.630 2.793913e+29 181541.71 4382.712
## 3  1.2307 136.7163  51.3458  5.8391 2318.047 1.351938e+29 149773.13 3862.812
## 4  1.1028 117.4389  44.0929  7.3885 2753.053 1.006915e+29 116318.25 3501.108
## 5  1.0030 106.7168  41.5637  9.9305 3328.649 4.975725e+28  94354.86 3215.279
## 6  2.3539 100.6510  24.1989 13.5439 3593.235 4.220871e+28  77666.86 2966.215
## 7  1.3946  91.8314  19.3160 14.8902 3811.920 3.709781e+28  70962.24 2827.698
## 8  0.2894  84.3846  39.7137 15.8603 3935.759 3.782395e+28  66491.73 2721.084
## 9  9.0062  84.5884   5.4070 21.3018 4389.084 1.933387e+28  55326.79 2517.847
## 10 0.1595  76.4624  29.2851 19.0060 4631.703 1.469257e+28  55748.34 2490.421
## 11 1.1438  75.7027  26.5530 21.8257 4632.719 1.774194e+28  47017.00 2349.974
## 12 1.6440  74.8183  18.6097 24.2274 5096.033 8.358887e+27  44550.02 2228.941
## 13 2.6073  72.6005  10.7057 25.2425 5224.251 7.591089e+27  40717.55 2147.064
## 14 0.2860  69.1703  22.3983 24.9176 5318.689 7.288631e+27  39148.16 2100.880
## 15 1.7664  68.6480  14.9034 26.8735 5705.480 3.860201e+27  35338.15 2008.323
##    Friedman  Rubin Cindex     DB Silhouette   Duda  Pseudot2   Beale Ratkowsky
## 2   24.9462 1.3663 0.2602 1.6345     0.3004 1.5083 -117.6152 -2.7417    0.3093
## 3   38.2779 1.5502 0.2301 1.8851     0.3124 1.9829 -192.8228 -4.0326    0.3056
## 4   44.3957 1.7103 0.2630 1.7434     0.3185 1.3785  -88.6888 -2.2241    0.3012
## 5   54.2429 1.8624 0.2515 1.7281     0.3094 1.7052 -117.0410 -3.3179    0.2899
## 6   54.0021 2.0187 0.2369 1.7986     0.2168 0.9134    7.3939  0.7620    0.2840
## 7   54.8662 2.1176 0.2315 1.7375     0.2007 1.3445  -30.4884 -2.0666    0.2699
## 8   47.1672 2.2006 0.2234 1.7947     0.2023 0.5970   25.6568  5.4453    0.2572
## 9   52.6991 2.3782 0.2175 1.6836     0.1979 2.0891  -46.9185 -3.7248    0.2497
## 10  65.7861 2.4044 0.1895 1.7354     0.1662 1.4663  -28.6195 -2.3082    0.2378
## 11  49.5095 2.5481 0.2155 1.6874     0.1761 2.8246  -31.0064 -4.5213    0.2330
## 12  55.1176 2.6865 0.1663 1.6362     0.1801 1.5750  -36.1435 -2.9360    0.2265
## 13  54.2432 2.7889 0.1985 1.5842     0.1813 1.2319  -28.0445 -1.5153    0.2203
## 14  54.5567 2.8502 0.1962 1.5920     0.1797 1.0118   -0.6075 -0.0930    0.2138
## 15  68.8585 2.9816 0.1920 1.5624     0.1714 1.9635  -35.8217 -3.8856    0.2092
##         Ball Ptbiserial    Frey McClain   Dunn Hubert SDindex Dindex   SDbw
## 2  2191.3560     0.4884  0.0431  0.6786 0.1051  4e-04  1.8943 2.6581 0.8981
## 3  1287.6040     0.5846  0.0332  0.9079 0.0952  5e-04  1.8779 2.5046 0.8905
## 4   875.2769     0.6248 -0.0545  0.9945 0.0992  6e-04  1.7736 2.4068 0.8689
## 5   643.0558     0.6612  1.7617  1.0360 0.1003  6e-04  1.7795 2.3173 0.8383
## 6   494.3691     0.5609  1.0792  1.7571 0.0462  6e-04  1.8224 2.2053 0.8604
## 7   403.9568     0.5370  0.8551  2.0264 0.0462  6e-04  1.8763 2.1349 0.6487
## 8   340.1355     0.5018  0.6900  2.5301 0.0720  7e-04  1.8202 2.0924 0.6380
## 9   279.7607     0.4864 -0.6515  2.8002 0.0513  8e-04  2.3394 2.0468 0.8147
## 10  249.0421     0.4077 -0.1671  3.6900 0.0424  8e-04  2.5148 2.0155 0.7809
## 11  213.6340     0.4449 -0.0159  3.4349 0.0513  8e-04  2.3777 1.9692 0.7179
## 12  185.7451     0.4618  0.9491  3.4531 0.0424  8e-04  2.2879 1.9219 0.7228
## 13  165.1588     0.4363  1.3965  3.9828 0.0543  8e-04  2.3081 1.8835 0.6748
## 14  150.0629     0.4102  0.6294  4.5958 0.0543  9e-04  2.2682 1.8603 0.6339
## 15  133.8882     0.3892  0.0697  5.2456 0.0513  9e-04  2.4267 1.8255 0.6287
##
## $All.CriticalValues
##    CritValue_Duda CritValue_PseudoT2 Fvalue_Beale
## 2          0.8703            52.0030       1.0000
## 3          0.8700            58.1093       1.0000
## 4          0.8341            64.2362       1.0000
## 5          0.7813            79.2144       1.0000
## 6          0.7905            20.6744       0.6902
## 7          0.8085            28.1897       1.0000
## 8          0.8076             9.0507       0.0000
```

```
## 9          0.5264            80.9571       1.0000
## 10         0.5480            74.2256       1.0000
## 11         0.5009            47.8198       1.0000
## 12         0.7928            25.8689       1.0000
## 13         0.7983            37.6515       1.0000
## 14         0.7475            17.5664       1.0000
## 15         0.7306            26.9175       1.0000
##
## $Best.nc
##                       KL       CH Hartigan     CCC    Scott      Marriot   TrCovW
## Number_clusters 9.0000   2.0000   9.0000 15.0000   3.0000 3.000000e+00     4.00
## Value_Index     9.0062 182.4061  34.3067 26.8735 768.4169 1.096951e+29 33454.87
##                   TraceW Friedman   Rubin  Cindex      DB Silhouette   Duda
## Number_clusters   9.0000  11.0000  9.0000 12.0000 15.0000     4.0000 2.0000
## Value_Index     175.8128  16.2767 -0.1514  0.1663  1.5624     0.3185 1.5083
##                    PseudoT2   Beale Ratkowsky     Ball PtBiserial Frey McClain
## Number_clusters    2.0000  2.0000    2.0000   3.0000     5.0000    1  2.0000
## Value_Index     -117.6152 -2.7417    0.3093 903.7521     0.6612   NA  0.6786
##                    Dunn Hubert SDindex Dindex    SDbw
## Number_clusters 2.0000      0  4.0000      0 15.0000
## Value_Index     0.1051      0  1.7736      0  0.6287
##
## $Best.partition
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   2   1   2   2   2   2   2   2   2   2   1   1   2   2   2   1   1   1   2   2
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   2   1   2   1   1   2   2   2   2   1   2   2   2   2   2   1   2   2   2   2
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   1   2   2   2   1   2   2   2   1   1   1   2   2   2   1   2   2   2   1   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   2   1   1   2   1   2   1   1   1   1   1   1   1   2   1   1   1   2   2   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   2   2   1   1   2   2   1   1   1   2   2   2   1   2   1   2   2   1   2   2
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   2   2   2   1   2   2   1   2   2   1   1   2   1   1   1   2   2   2   1   1
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   1   2   2   2   1   2   2   2   2   2   2   2   1   2   2   2   2   1   1   1
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
##   1   1   1   2   2   1   2   1   2   2   2   2   2   2   2   2   2   2   2   2
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1   1   1
## 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260
##   1   1   1   1   1   1   2   1   1   1   1   2   2   1   1   1   1   1   1   1
## 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280
##   1   1   2   1   1   1   1   1   1   1   1   2   1   1   2   1   1   1   1   2
## 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
##   2   1   1   1   1   1   2   2   1   1   1   1   1   1   2   1   1   1   1   1
## 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
```

```
##   2   1   1   1   1   2   1   1   1   1   1   1   2   2   1   1   1   1   1   1
## 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
##   1   1   1   2   1   1   1   2   1   1   2   2   1   1   1   2   1   1   1   1
## 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
##   1   2   1   1   1   1   1   2   1   1   1   1   1   1   2   2   1   1   2   1
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380
##   1   1   1   1   2   2   1   2   1   1   1   2   1   1   1   1   1   1   1   1
## 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400
##   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   2   2   1   2   2
## 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420
##   1   1   1   1   1   1   1   1   2   1   2   1   1   1   1   1   2   2   1   1
## 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440
##   1   1   1   1   2   2   1   1   1   1   1   1   1   1   1   1   1   1   2   1
## 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460
##   1   1   1   1   1   1   1   1   1   1   2   1   2   1   1   2   1   1   1   1
## 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480
##   2   2   1   1   1   1   1   1   1   2   1   1   2   1   1   1   1   1   1   2
## 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500
##   1   1   1   1   1   1   1   1   1   2   1   2   1   1   1   1   1   1   1   1
```

For this clustering method, the kmeans() function from the stats package is used.

```
kmean <- kmeans(clusterData,2, nstart=10)
```

Clusters Analysis

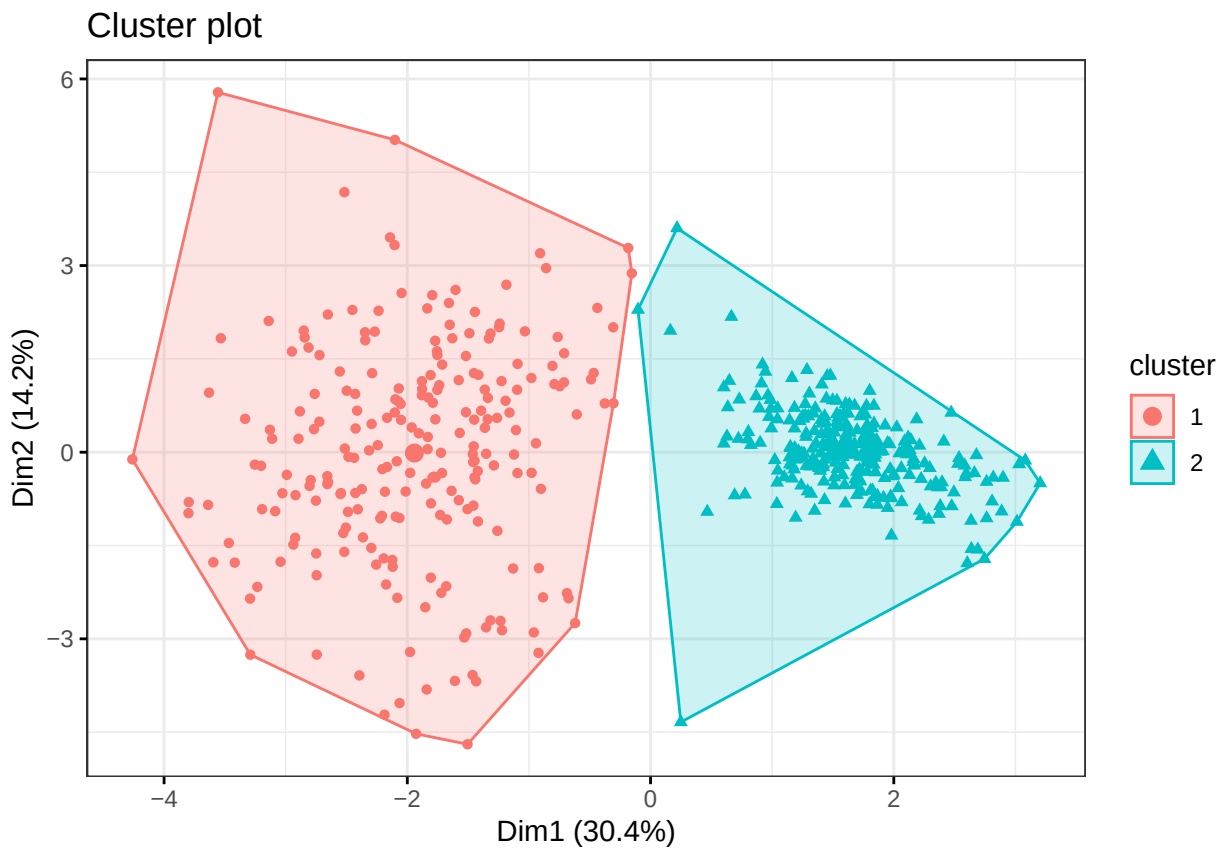
First, let's see the size of the clusters

```
kmeansize <- kmean$size
```

We have 2 clusters, with size 229 and 271.

Visually, the clusters look like so

```
fviz_cluster(kmean, data = clusterData, geom = "point", ellipse.type = "convex",
             ggtheme = theme_bw())
```



Cluster Intepretation

```
kmeaninfo <- data.frame(kmean$centers, kmean$size)
kmeaninfo
```

```
##   popularity duration_ms danceability      energy   loudness speechiness
## 1  0.4445902   0.7527152   -0.6728745  0.09532898  0.5946238  -1.0524128
## 2 -0.3756869  -0.6360583    0.5685913 -0.08055475 -0.5024681   0.8893083
##   acousticness instrumentalness    liveness    valence       tempo
## 1    0.7661786        0.5725982  0.04803213 -0.3470702  0.10613458
## 2   -0.6474351       -0.4838561 -0.04058804  0.2932807 -0.08968568
##   time_signature kmean.size
## 1      0.2514930        229
## 2     -0.2125162        271
```

From the above data frame, cluster 1 represent songs that have high loudness, so they were mixed louder than average, as well as both high acousticness and instrumentalness; and low in valence. This group can be considered as “Loud Angry Songs”.

Cluster 2 have songs that are high in speechiness, danceability, and valence. This group can be considered as “Happy Songs To Dance To”.