Spotify Data Analysis

Kriti Singh

2022-07-08

Problem Statement

Determining what categories (clusters) of songs have been released during the pandemic and how they performed during this particular time period.

Objective:

Points taken into consideration while designing your solution in R and Python: - What measures would you report to describe this dataset? - Would you engineer any particular features ahead of your analysis? - Would you pre-process the data in any particular way? - How would you cluster this dataset and why? - K -means model to identify the most popular songs - which genres are popular - audio features/attributes performance across clusters - Create one (max two) key visualisations that you believe would contain the most insights for the client

Approach:

1. Preprocessing of Data and Feature Engineering
2. Exploratory Data Analysis
    - Visualisation techniques for audio features and behaviour analysis
    - Feature Correlation with Dependent variable
    - Bivariate Analysis
    - Statistical Analysis on variable behavioural :
        - Pearson’s Chi square testing for categorical variables
        - Correlation plot for numeric variables
3. K - Means Clustering to understand popularity of songs
4. Result Analysis

Importing Packages

```
#Importing all the libraries
library(knitr)
#install.packages("kableExtra", dependencies = TRUE)
library(kableExtra)
```

```
## Warning in !is.null(rmarkdown::metadata$output) && rmarkdown::metadata$output
## %in% : 'length(x) = 2 > 1' in coercion to 'logical(1)'
```

```
#install.packages("DT", dependencies = TRUE)
library(magrittr)
#install.packages("magrittr", dependencies = TRUE)
library(DT)
#install.packages(summarytools)
library(summarytools)
```

```
## Warning in fun(libname, pkgname): couldn't connect to display ":0"
```

```
## system might not have X11 capabilities; in case of errors when using dfSummary(), set st_options(use.x11 = FAL
SE)
```

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✔ ggplot2 3.3.6     ✔ purrr   0.3.4
## ✔ tibble  3.1.7     ✔ dplyr   1.0.9
## ✔ tidyr   1.2.0     ✔ stringr 1.4.0
## ✔ readr   2.1.2     ✔ forcats 0.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ tidyr::extract()    masks magrittr::extract()
## ✖ dplyr::filter()     masks stats::filter()
## ✖ dplyr::group_rows() masks kableExtra::group_rows()
## ✖ dplyr::lag()        masks stats::lag()
## ✖ purrr::set_names()  masks magrittr::set_names()
## ✖ tibble::view()      masks summarytools::view()
```

```
library(dplyr)
library(tidyr)

library(ggplot2)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(RColorBrewer)
library(viridis)
```

```
## Loading required package: viridisLite
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(fpc)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
import numpy as np
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
from yellowbrick.target import FeatureCorrelation

import warnings
warnings.filterwarnings("ignore")
color = sns.color_palette()
color_pal = [x['color'] for x in plt.rcParams['axes.prop_cycle']]
```

Importing data

```
# Loading the data
spotify_all <- read.csv("spotify_songs copy.csv", sep=",",
                comment.char = "",check.names = FALSE, quote="\"",
                na.strings=c("NA","NaN", " ") )
spotify <- subset(spotify_all, track_album_release_date >= "2019-12-01")
spotifyAllCol <- spotify
# About the data
dim(spotify)
```

```
## [1] 1963   23
```

Data Statistics Description

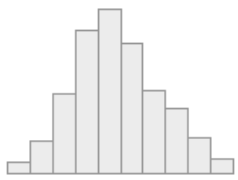
```
print(dfSummary(spotify), method = 'render')
```

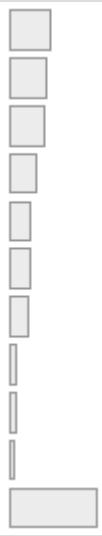
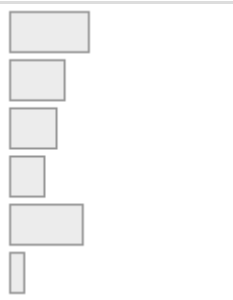
Data Frame Summary

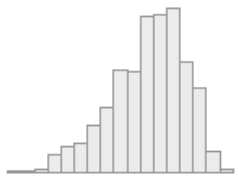
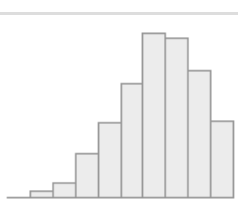
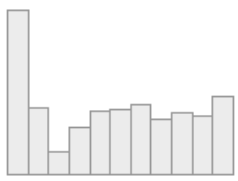
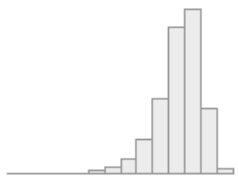
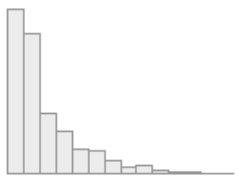
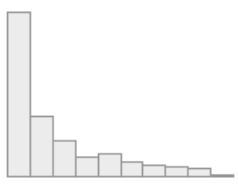
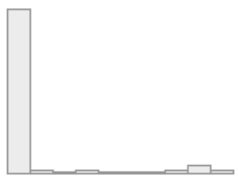
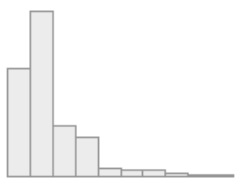
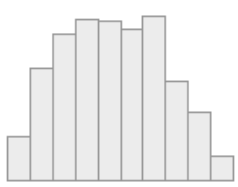
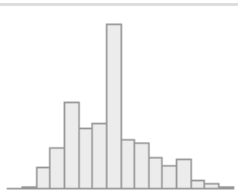
spotify

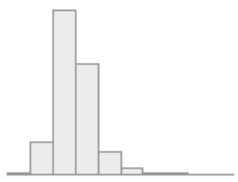
Dimensions: 1963 x 23
Duplicates: 0

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing
1	track_id [character]	1. 0nbXyq5TXYPCO7pr3N8S4I 2. 7h0d2h0fUmzbs7zeFigJPn 3. 3yOlyBJuViE2YSGn3nVE1K 4. 41L3O37CECZt3N7ziG2z7l 5. 5yY9lUy8nbvjM1Uyo1Uqoc 6. 7HMmFQsKsljwTw8bS7lu19 7. 0bMbDctzMmTyK2j74j3nF3 8. 2Ec33AVlkTTq8BHFgBTdQs 9. 2usxQITOSDqvkYiI0oIwao 10. 1ToprX3cpBiXoAe5eNSk74 [1592 others]	8 (0.4%) 8 (0.4%) 7 (0.4%) 7 (0.4%) 7 (0.4%) 7 (0.4%) 6 (0.3%) 6 (0.3%) 6 (0.3%) 5 (0.3%) 1896 (96.6%)		1963 (100.0%)	0 (0.0%)
2	track_name [character]	1. Adore You 2. Rare 3. HIGHEST IN THE ROOM (feat 4. The Box 5. Life Is Good (feat. Drake 6. My Oh My (feat. DaBaby) 7. Sigues Con El 8. Yummy 9. Alone, Pt. II 10. Futsal Shuffle 2020 [1560 others]	9 (0.5%) 9 (0.5%) 8 (0.4%) 8 (0.4%) 7 (0.4%) 7 (0.4%) 7 (0.4%) 6 (0.3%) 6 (0.3%) 1889 (96.2%)		1963 (100.0%)	0 (0.0%)
3	track_artist [character]	1. Selena Gomez 2. Roddy Ricch 3. Arcangel 4. Camila Cabello 5. Harry Styles 6. Coopex 7. Halsey 8. Mac Miller 9. Martin Garrix 10. Travis Scott [1389 others]	15 (0.8%) 13 (0.7%) 12 (0.6%) 12 (0.6%) 12 (0.6%) 11 (0.6%) 10 (0.5%) 9 (0.5%) 9 (0.5%) 8 (0.4%) 1852 (94.3%)		1963 (100.0%)	0 (0.0%)
4	track_popularity [integer]	Mean (sd) : 49.1 (19.3) min ≤ med ≤ max: 0 ≤ 48 ≤ 98 IQR (CV) : 26 (0.4)	96 distinct values		1963 (100.0%)	0 (0.0%)
5	track_album_id [character]	1. 0CPLMVp7rMi3BkzAMve96K 2. 1Sf8GsXG32t0jNrX11xqWx 3. 3YPFaTR7WMi1Hd4NVKdCJx 4. 52u4anZbHd6UlnnmHRFzba 5. 3Vsbl0diFGw8HNSjG8ue9m 6. 7xV2TzoaVc0ycW7fwBwAml 7. 1jJdkoOAj3Uk6Tbv3S4fsa 8. 1SN6N3fNkZk5oXQ9X46QZ3 9. 5uCEoLCj3ZZZ1EtzQdQWVl 10. 01GR4NL5O5CZM51k0aejKD [1520 others]	15 (0.8%) 15 (0.8%) 13 (0.7%) 13 (0.7%) 12 (0.6%) 8 (0.4%) 7 (0.4%) 7 (0.4%) 7 (0.4%) 6 (0.3%) 1860 (94.8%)		1963 (100.0%)	0 (0.0%)

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing
6	track_album_name [character]	1. Historias de un Capricorn 2. JACKBOYS 3. Rare 4. Please Excuse Me For Bein 5. Romance 6. Fine Line 7. Life Is Good (feat. Drake 8. MONTANA 9. Yummy 10. Alone, Pt. II [1493 others]	15 (0.8%) 15 (0.8%) 15 (0.8%) 13 (0.7%) 12 (0.6%) 8 (0.4%) 7 (0.4%) 7 (0.4%) 7 (0.4%) 6 (0.3%) 1858 (94.7%)		1963 (100.0%)	0 (0.0%)
7	track_album_release_date [character]	1. 2020-01-10 2. 2019-12-06 3. 2019-12-13 4. 2019-12-20 5. 2020-01-03 6. 2020-01-17 7. 2019-12-27 8. 2019-12-12 9. 2020-01-09 10. 2019-12-18 [40 others]	270 (13.8%) 235 (12.0%) 220 (11.2%) 180 (9.2%) 133 (6.8%) 131 (6.7%) 115 (5.9%) 42 (2.1%) 37 (1.9%) 33 (1.7%) 567 (28.9%)		1963 (100.0%)	0 (0.0%)
8	playlist_name [character]	1. The Pulse of Indie Poptim 2. Charts 2020 🔥 Top 2020🔥 Hit 3. The Edge of Indie Poptimi 4. Jazz Vibes 5. Brand New EDM 6. 2020 Hits & 2019 Hits – 7. Trap Nation 8. Latin Pop Rising 9. Dance Pop: Japan 10. Waves Pop and EDM [161 others]	77 (3.9%) 72 (3.7%) 68 (3.5%) 53 (2.7%) 51 (2.6%) 46 (2.3%) 42 (2.1%) 39 (2.0%) 37 (1.9%) 37 (1.9%) 1441 (73.4%)		1963 (100.0%)	0 (0.0%)
9	playlist_id [character]	1. 5qFXOOxrQVyS4UCq3UilZN 2. 3xMQTDLOIGvj3lWH5e5x6F 3. 5vIdb67lQcdFwoVkE4UMep 4. 37i9dQZF1DX0SM0LYsmbMT 5. 2dNitDEHM9FpUGEHWc7zyW 6. 4JkkvMpVl4lSioqQjeAL0q 7. 0NCspsyf0OS4BsPgGhkQXM 8. 37i9dQZF1DX8womvTyUjrN 9. 37i9dQZF1DXahYFr91pFvG 10. 64k01l4j6QtnZ8jMaI84AA [163 others]	77 (3.9%) 72 (3.7%) 68 (3.5%) 53 (2.7%) 51 (2.6%) 46 (2.3%) 42 (2.1%) 39 (2.0%) 37 (1.9%) 37 (1.9%) 1441 (73.4%)		1963 (100.0%)	0 (0.0%)
10	playlist_genre [character]	1. edm 2. latin 3. pop 4. r&b 5. rap 6. rock	512 (26.1%) 358 (18.2%) 304 (15.5%) 224 (11.4%) 470 (23.9%) 95 (4.8%)		1963 (100.0%)	0 (0.0%)

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing
11	playlist_subgenre [character]	1. hip hop 2. trap 3. pop edm 4. latin pop 5. hip pop 6. indie poptimism 7. electro house 8. dance pop 9. big room 10. reggaeton [12 others]	255 (13.0%) 179 (9.1%) 173 (8.8%) 158 (8.0%) 156 (7.9%) 153 (7.8%) 145 (7.4%) 116 (5.9%) 109 (5.6%) 87 (4.4%) 432 (22.0%)		1963 (100.0%)	0 (0.0%)
12	danceability [numeric]	Mean (sd) : 0.7 (0.1) min ≤ med ≤ max: 0.2 ≤ 0.7 ≤ 1 IQR (CV) : 0.2 (0.2)	531 distinct values		1963 (100.0%)	0 (0.0%)
13	energy [numeric]	Mean (sd) : 0.7 (0.2) min ≤ med ≤ max: 0.1 ≤ 0.7 ≤ 1 IQR (CV) : 0.3 (0.3)	626 distinct values		1963 (100.0%)	0 (0.0%)
14	key [integer]	Mean (sd) : 5.5 (3.6) min ≤ med ≤ max: 0 ≤ 6 ≤ 11 IQR (CV) : 7 (0.7)	12 distinct values		1963 (100.0%)	0 (0.0%)
15	loudness [numeric]	Mean (sd) : -6.7 (2.9) min ≤ med ≤ max: -26.2 ≤ -6.3 ≤ -0.2 IQR (CV) : 3.3 (-0.4)	1471 distinct values		1963 (100.0%)	0 (0.0%)
16	mode [integer]	Min : 0 Mean : 0.5 Max : 1	0 : 958 (48.8%) 1 : 1005 (51.2%)		1963 (100.0%)	0 (0.0%)
17	speechiness [numeric]	Mean (sd) : 0.1 (0.1) min ≤ med ≤ max: 0 ≤ 0.1 ≤ 0.7 IQR (CV) : 0.1 (0.9)	772 distinct values		1963 (100.0%)	0 (0.0%)
18	acousticness [numeric]	Mean (sd) : 0.2 (0.2) min ≤ med ≤ max: 0 ≤ 0.1 ≤ 1 IQR (CV) : 0.3 (1.1)	1085 distinct values		1963 (100.0%)	0 (0.0%)
19	instrumentalness [numeric]	Mean (sd) : 0.1 (0.2) min ≤ med ≤ max: 0 ≤ 0 ≤ 1 IQR (CV) : 0 (2.5)	845 distinct values		1963 (100.0%)	0 (0.0%)
20	liveness [numeric]	Mean (sd) : 0.2 (0.1) min ≤ med ≤ max: 0 ≤ 0.1 ≤ 1 IQR (CV) : 0.1 (0.8)	644 distinct values		1963 (100.0%)	0 (0.0%)
21	valence [numeric]	Mean (sd) : 0.5 (0.2) min ≤ med ≤ max: 0 ≤ 0.5 ≤ 1 IQR (CV) : 0.4 (0.5)	766 distinct values		1963 (100.0%)	0 (0.0%)
22	tempo [numeric]	Mean (sd) : 122.5 (26.8) min ≤ med ≤ max: 57.5 ≤ 123 ≤ 204.1 IQR (CV) : 37 (0.2)	1490 distinct values		1963 (100.0%)	0 (0.0%)

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing
23	duration_ms [numeric]	Mean (sd) : 196964.9 (45270.6) min ≤ med ≤ max: 65000 ≤ 192560 ≤ 515703 IQR (CV) : 48605.5 (0.2)	1523 distinct values		1963 (100.0%)	0 (0.0%)

Generated by summarytools (https://github.com/dcomtois/summarytools) 1.0.1 (R (https://www.r-project.org/) version 4.2.1)
2022-07-09

Data Wrangling

Missing values

```
colSums(is.na(spotify))
```

```
##                  track_id                track_name               track_artist
##                         0                         0                          0
##          track_popularity              track_album_id            track_album_name
##                         0                         0                          0
##  track_album_release_date             playlist_name                playlist_id
##                         0                         0                          0
##            playlist_genre          playlist_subgenre               danceability
##                         0                         0                          0
##                    energy                       key                   loudness
##                         0                         0                          0
##                      mode               speechiness               acousticness
##                         0                         0                          0
##          instrumentalness                  liveness                    valence
##                         0                         0                          0
##                     tempo               duration_ms
##                         0                         0
```

Column Encoding using label encoding

From the data summary, we have playlist_genre with 6 distinct levels, playlist_subgenre with 24 distinct level, 12 keys, and two modes

We would transform the above attributes to factors and also fix some other attributes

```
#Changing Data Types
spotify <- spotify %>%
  mutate(
  track_name =  as.factor(spotify$track_name),
  track_artist = as.factor(spotify$track_artist),
  playlist_genre = as.factor(spotify$playlist_genre),
  playlist_subgenre = as.factor(spotify$playlist_subgenre),
  key = as.factor(spotify$key),
  mode = as.factor(spotify$mode),
  track_popularity = as.numeric(spotify$track_popularity),
  duration_ms = as.numeric(spotify$duration_ms)
  )
```

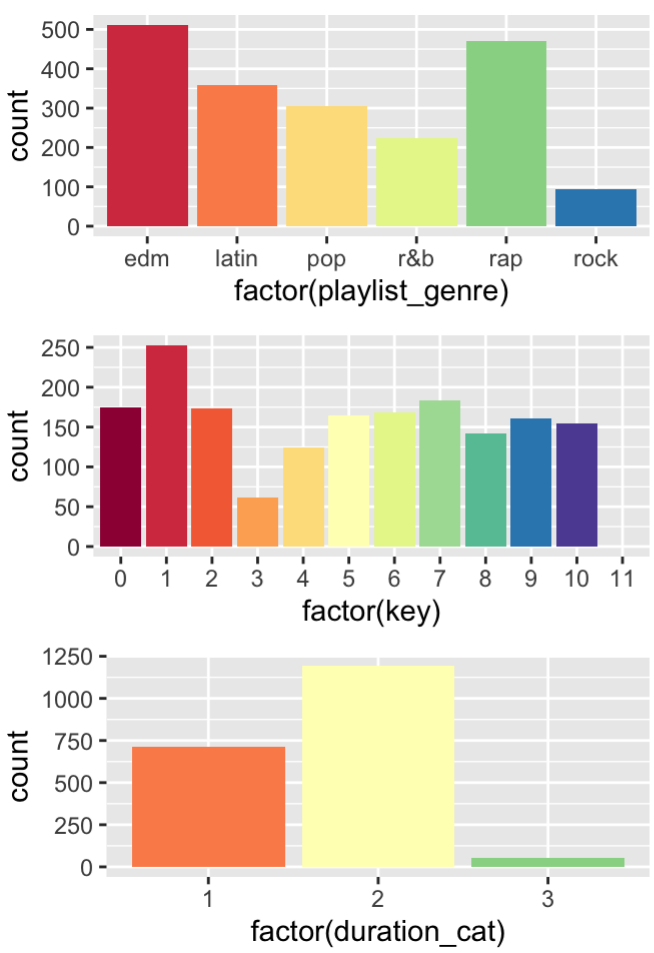
Feature Engineering

The duration of tracks is in milliseconds, therefore we would convert them into suitable categories based on the quantiles.

```
# hrs = spotify$duration_ms/(60 * 60 * 1000)
# mins = (duration_ms/(60 * 60 * 1000) %% 1) * 60
# c(2.807, 3.209, 3.618)
# spotify = spotify %>% mutate(duration_cat = (duration_ms/(60 * 60 * 1000) %% 1) * 60 )
spotify[["duration_hrs"]] <- spotify[["duration_ms"]]/(60 * 60 * 1000)
spotify[["duration_mins"]] <- (spotify[["duration_hrs"]]%% 1)*60
spotify[["duration_cat"]] <- findInterval(spotify[["duration_mins"]], c(1,3,5))
spotify[["duration_hrs"]]<-NULL
spotify[["duration_mins"]]<-NULL
spotify <- spotify %>%
  mutate(duration_cat = as.factor(spotify$duration_cat)
  )
```

```
p1 <- ggplot(spotify, aes(x=factor(playlist_genre),fill=playlist_genre)) +
  geom_bar() + scale_fill_brewer(palette="Spectral") +
  theme(legend.position="none")
p2 <- ggplot(spotify, aes(y=factor(playlist_subgenre),fill=playlist_genre)) +
  geom_bar() + scale_fill_brewer(palette="Spectral") +
  theme(legend.position="none")
p3 <- ggplot(spotify, aes(x=factor(key),fill=key)) + geom_bar() +
  scale_fill_brewer(palette="Spectral") + theme(legend.position="none")
p4 <- ggplot(spotify, aes(x=factor(mode),fill=mode)) + geom_bar() +
  scale_fill_brewer(palette="Spectral") + theme(legend.position="none")
p5 <- ggplot(spotify, aes(x=factor(duration_cat),fill=duration_cat)) +
  geom_bar() + scale_fill_brewer(palette="Spectral") +
  theme(legend.position="none")
grid.arrange(p1,p2,p3,p4,p5,nrow=3,ncol=2)
```

```
## Warning in RColorBrewer::brewer.pal(n, pal): n too large, allowed maximum for palette Spectral is 11
## Returning the palette you asked for with that many colors
```



Now, the variables Track_id, track_album_id, track_album_name, sub genre are not important to the analysis and hence they are dropped.

```
colnames(spotify)
```

```
##  [1] "track_id"                "track_name"
##  [3] "track_artist"            "track_popularity"
##  [5] "track_album_id"          "track_album_name"
##  [7] "track_album_release_date" "playlist_name"
##  [9] "playlist_id"             "playlist_genre"
## [11] "playlist_subgenre"       "danceability"
## [13] "energy"                  "key"
## [15] "loudness"                "mode"
## [17] "speechiness"             "acousticness"
## [19] "instrumentalness"        "liveness"
## [21] "valence"                 "tempo"
## [23] "duration_ms"             "duration_cat"
```

```
spotify <- spotify %>% select(2,3,4,10,12:24)
```

Data Wrangling Result

The cleaned dataset has 1963 observations of 16 variables

```
str(spotify)
```

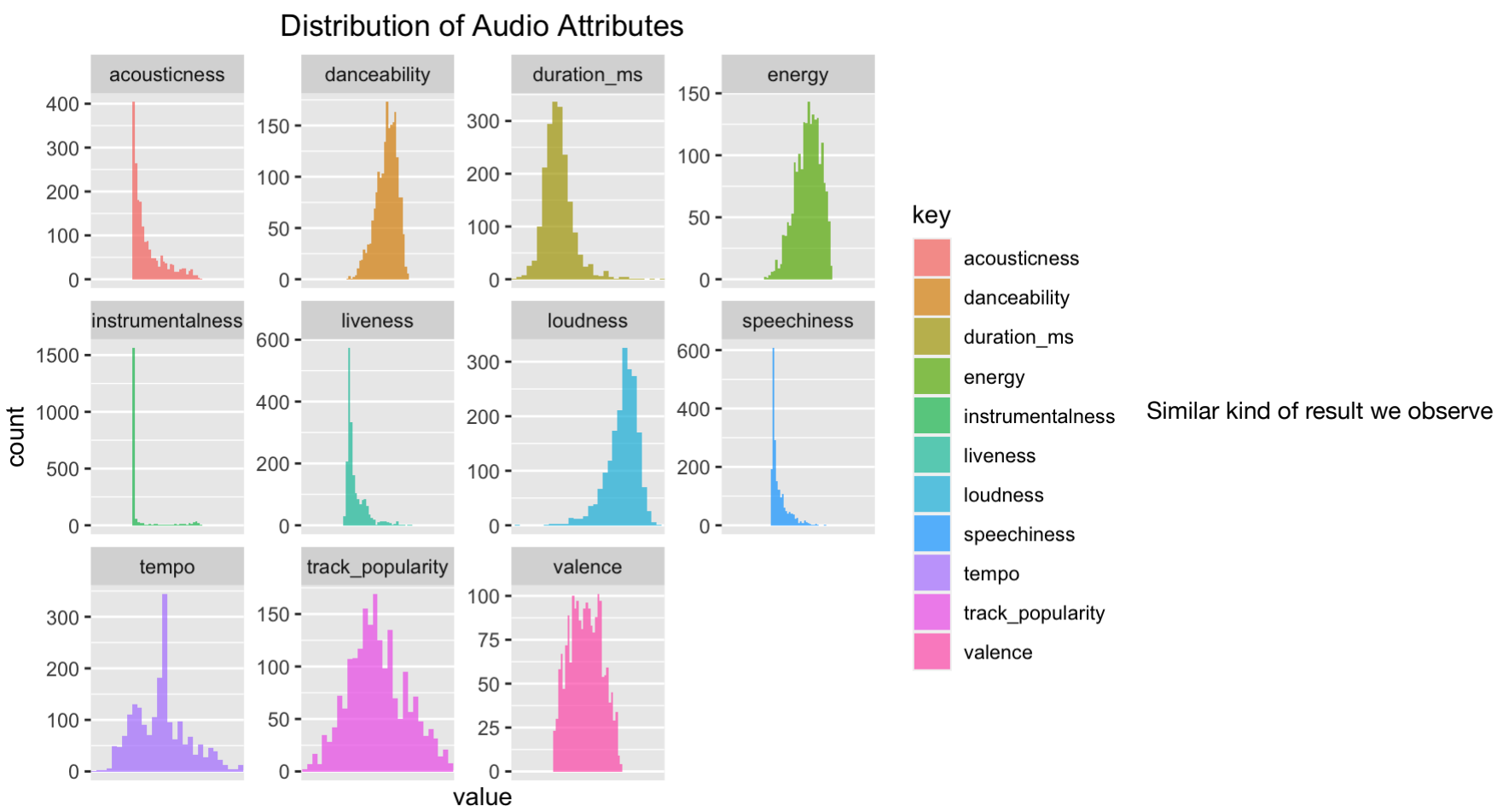
```
## 'data.frame':    1963 obs. of  17 variables:
##  $ track_name      : Factor w/ 1570 levels "@ ME","#NAKAMA",..: 849 1250 469 1390 655 351 145 66 796 784 ...
##  $ track_artist    : Factor w/ 1399 levels ".Sinh",".SØN",..: 815 368 1041 1141 1085 522 114 894 459 149 ...
##  $ track_popularity: num  67 66 55 45 74 68 68 57 58 48 ...
##  $ playlist_genre  : Factor w/ 6 levels "edm","latin",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ danceability    : num  0.726 0.805 0.355 0.607 0.57 0.654 0.682 0.762 0.632 0.791 ...
##  $ energy          : num  0.815 0.835 0.789 0.955 0.672 0.787 0.855 0.736 0.816 0.541 ...
##  $ key             : Factor w/ 12 levels "0","1","2","3",..: 12 1 6 8 9 5 8 8 9 2 ...
##  $ loudness        : num  -4.97 -4.6 -4.36 -3.61 -5.89 ...
##  $ mode            : Factor w/ 2 levels "0","1": 2 2 1 1 1 2 2 2 2 2 ...
##  $ speechiness     : num  0.0373 0.0896 0.165 0.102 0.0723 0.0502 0.0401 0.0778 0.0349 0.054 ...
##  $ acousticness    : num  0.0724 0.13 0.0148 0.0163 0.177 0.00515 0.0566 0.135 0.00205 0.123 ...
##  $ instrumentalness: num  4.21e-03 5.03e-06 0.00 1.21e-01 0.00 0.00 1.00e-06 5.38e-02 1.06e-01 4.74e-05 ...
##  $ liveness        : num  0.357 0.365 0.0355 0.0725 0.0768 0.144 0.101 0.113 0.119 0.121 ...
##  $ valence         : num  0.693 0.722 0.169 0.501 0.448 0.409 0.304 0.248 0.134 0.433 ...
##  $ tempo           : num  100 125 80.4 126 120 ...
##  $ duration_ms     : num  162600 188230 237686 230476 208222 ...
##  $ duration_cat    : Factor w/ 3 levels "1","2","3": 1 2 2 2 2 1 2 1 2 1 ...
```

Exploratory Data Analysis

Data Distribution

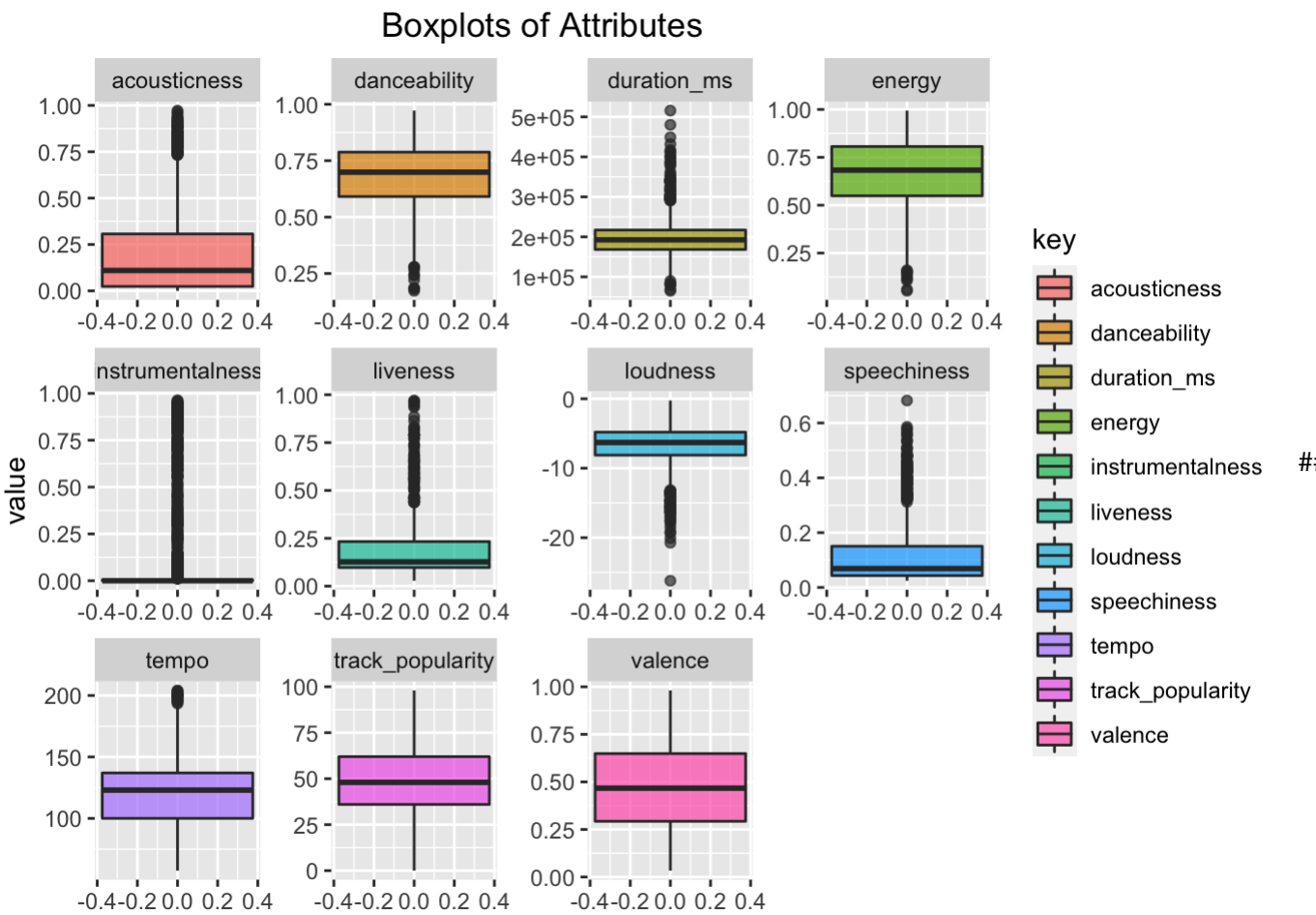
Danceability, energy, loudness, tempo, track_popularity, valence are normally distributed and rest attributes have skewness in some form.

```
#Plotting numeric values
spotify %>%
  keep(is.numeric) %>% #hist only for numeric
  gather() %>% #converts to key value
  ggplot(aes(value, fill = key)) +
  facet_wrap(~ key, scales = "free") +
  geom_histogram(alpha = 0.7, bins = 30) +
  ggtitle("Distribution of Audio Attributes") +
  scale_x_discrete(guide = guide_axis(check.overlap = TRUE)) +
  theme(plot.title = element_text(hjust = 0.5))
```



Similar kind of result we observe from the box-plot

```
#Boxplot for numeric values
spotify %>%
  keep(is.numeric) %>% #hist only for numeric
  gather() %>% #converts to key value
  ggplot(aes(value, fill = key)) +
  facet_wrap(~ key, scales = "free") +
  geom_boxplot(alpha = 0.7) +
  ggtitle("Boxplots of Attributes") +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_flip()
```



Feature Correlation Feature

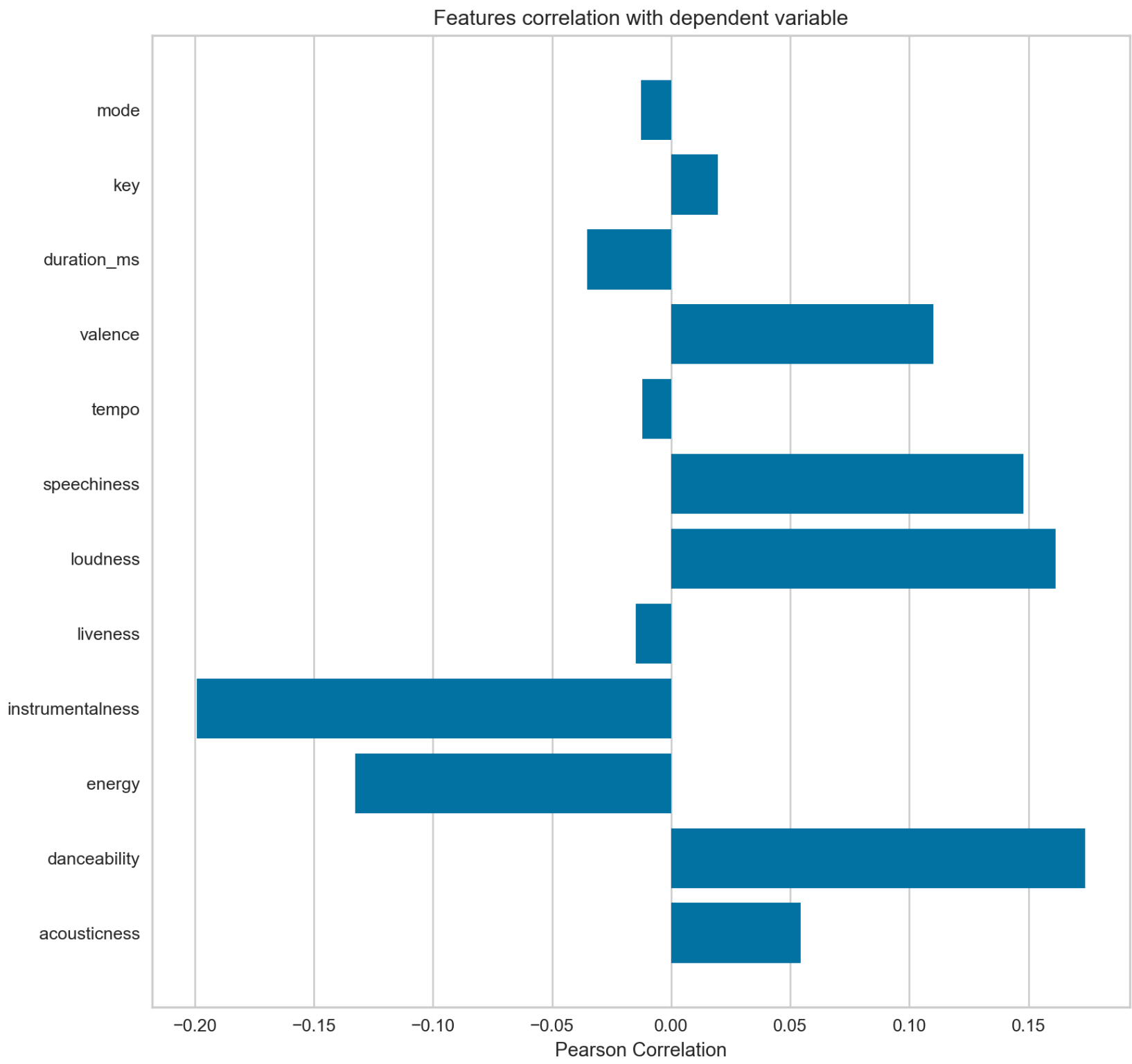
Correlation of different attributes w.r.t track_popularity

```
feature_names = ['acousticness', 'danceability', 'energy', 'instrumentalness',
       'liveness', 'loudness', 'speechiness', 'tempo', 'valence','duration_ms','key','mode']

X, y = r.spotifyAllCol[feature_names], r.spotifyAllCol['track_popularity']

# Create a list of the feature names

features = np.array(feature_names)

# Instantiate the visualizer
visualizer = FeatureCorrelation(labels=features)

plt.rcParams['figure.figsize']=(10,10)
visualizer.fit(X, y)        # Fit the data to the visualizer
```

▾ **FeatureCorrelation**

```
FeatureCorrelation(ax=<AxesSubplot:>,
                   labels=array(['acousticness', 'danceability', 'energy', 'instrumentalness',
       'liveness', 'loudness', 'speechiness', 'tempo', 'valence',
       'duration_ms', 'key', 'mode'], dtype='<U16'))
```

```
visualizer.show()
```



We have 6 features with +ve correlation and 6 -ve w.r.t track_popularity.

```
total <- dim(spotifyAllCol)[1]

popularity_score_more_than_50 <- dim(subset(spotifyAllCol, track_popularity > 50))[1]

probability <- (popularity_score_more_than_50/total)*100
sprintf("Probability of song getting more than 50 in popularity :%f", probability)
```

```
## [1] "Probability of song getting more than 50 in popularity :43.352012"
```

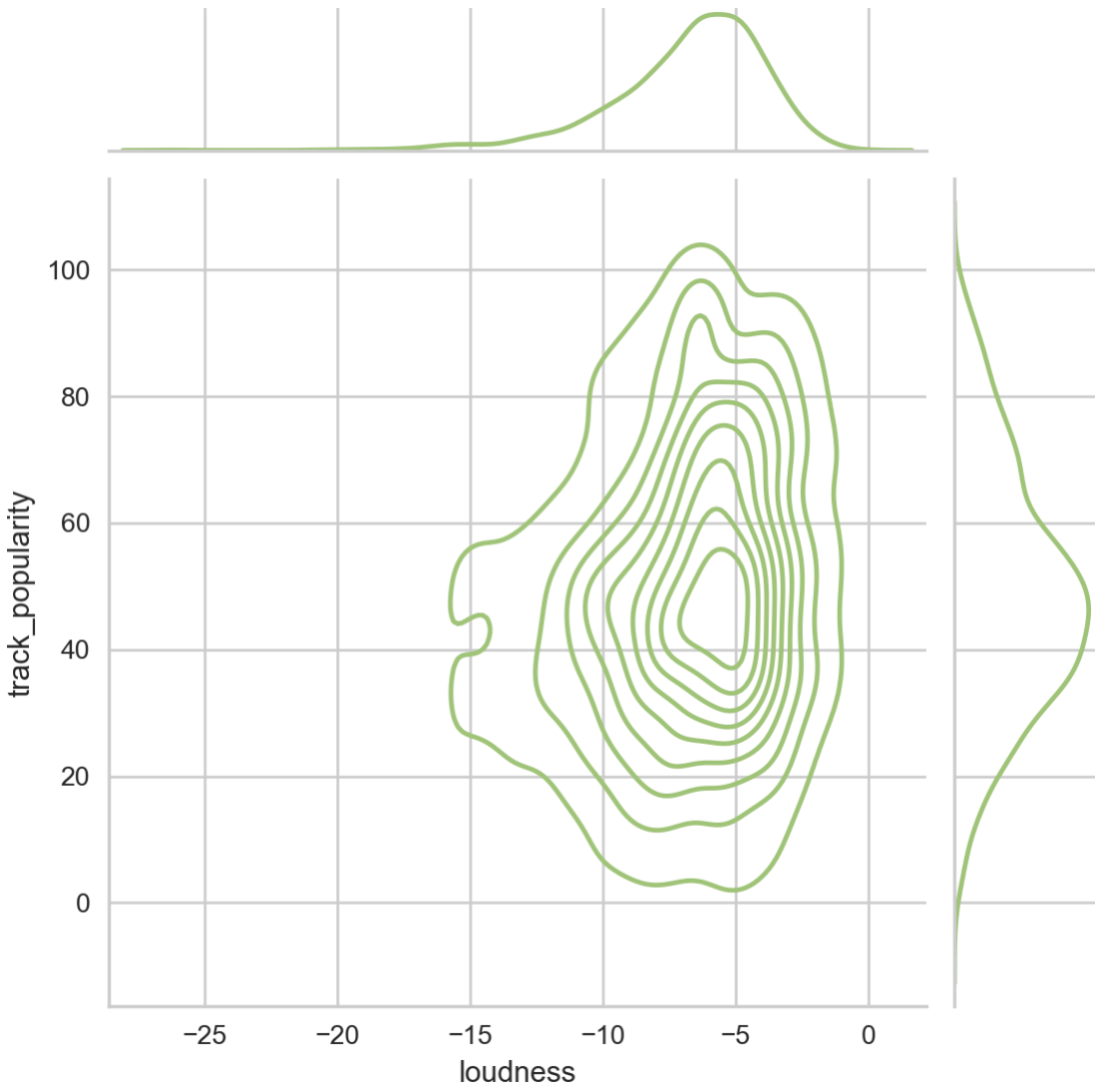
so popularity ranges from 0 to 100 … so Probability of song getting more than 50 is just 48.35.

Bivariate analysis

“Loudness” of the song VS “popularity” with size “key” and for a binary attributes “mode” of the song.

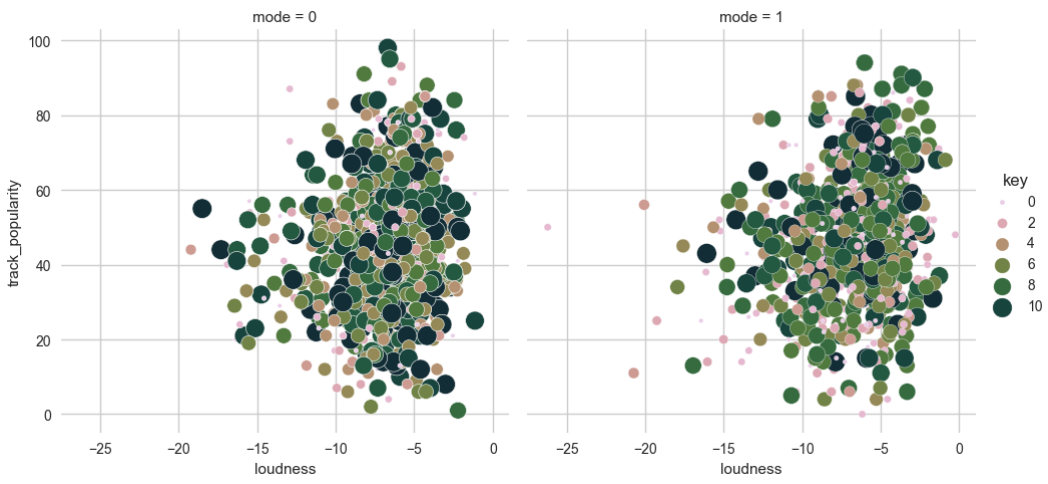
Loudness VS Popularity

```
fig1 = sns.jointplot(x="loudness", y="track_popularity", data=r.spotifyAllCol,
                  kind="kde", truncate=False,
                  color=color[1])
plt.show()
```



Good this Loudness have good relation with popularity .. well loud songs get more popularity !!

```
cmap_ = sns.cubehelix_palette(rot=-1, as_cmap=True)
g = sns.relplot(
    data=r.spotifyAllCol,
    x="loudness", y="track_popularity",
    hue="key",size="key",col="mode",
    palette=cmap_, sizes=(10, 200),
)

g.despine(left=True, bottom=True)
```



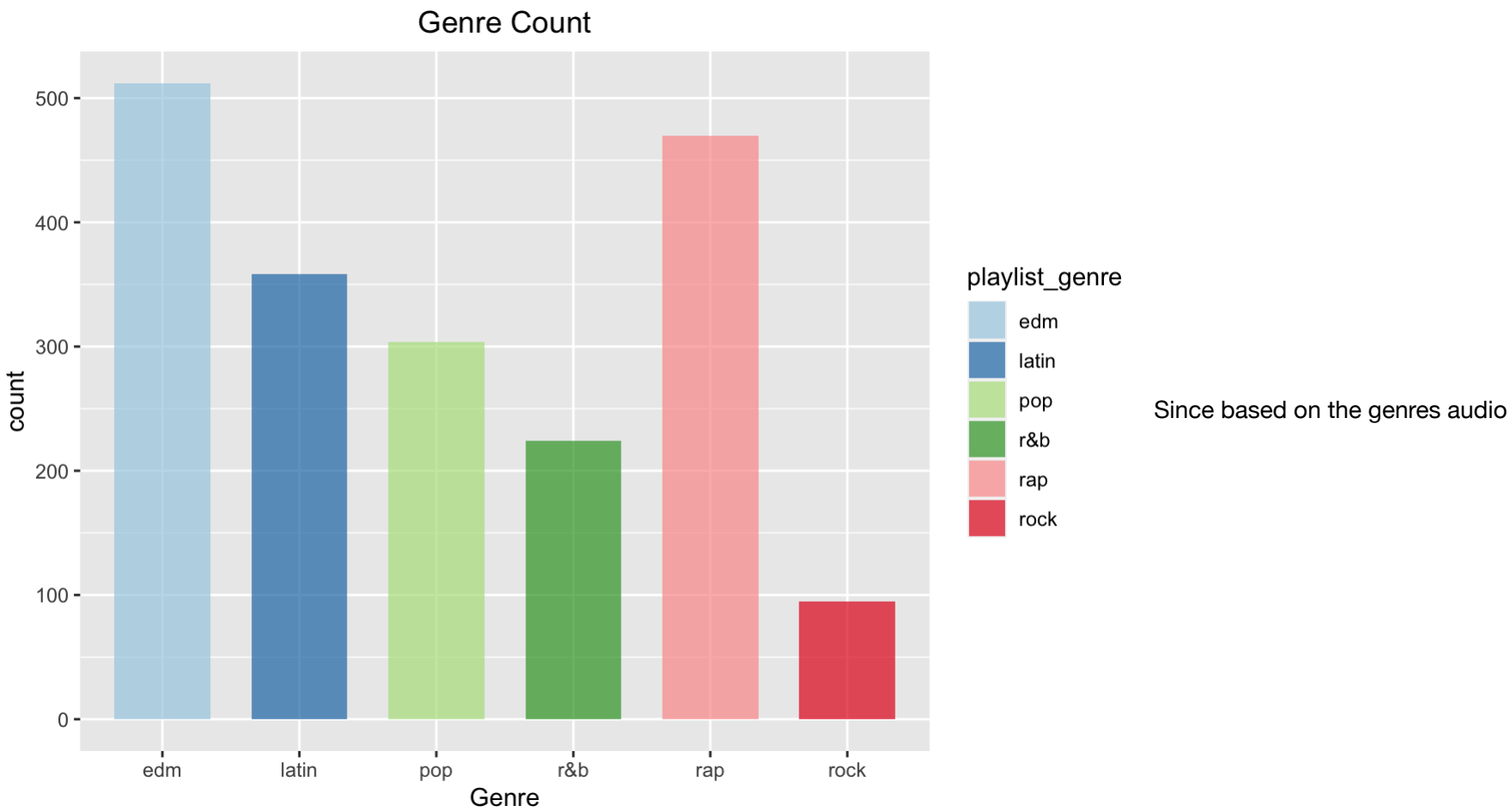
For both modes, at higher key notes, the loudness and track_popularity find a good cluster.

Genres Analysis

From below plot, it can be seen that the maximum number of songs belong to:

- EDM
- Rap
- Latin
- Pop

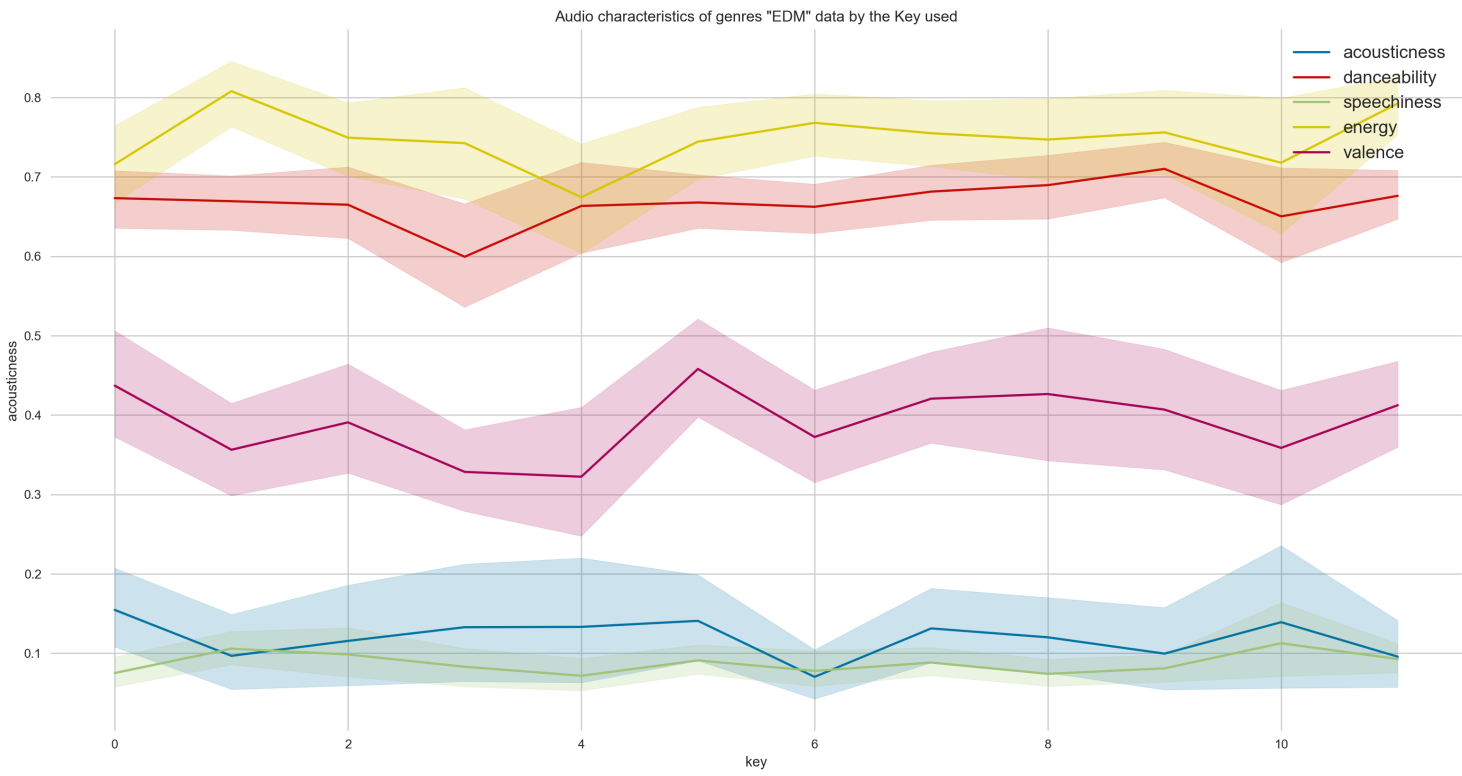
```
p1 <- ggplot(spotify, aes(x=factor(playlist_genre))) +
      geom_bar(width=0.7,
           aes(fill=playlist_genre),
           alpha=0.7) +
      scale_fill_brewer(palette = "Paired") +
      ggtitle("Genre Count") +
      theme(plot.title = element_text(hjust = 0.5)) +
      xlab("Genre")
grid.arrange(p1, nrow=1, ncol=1)
```



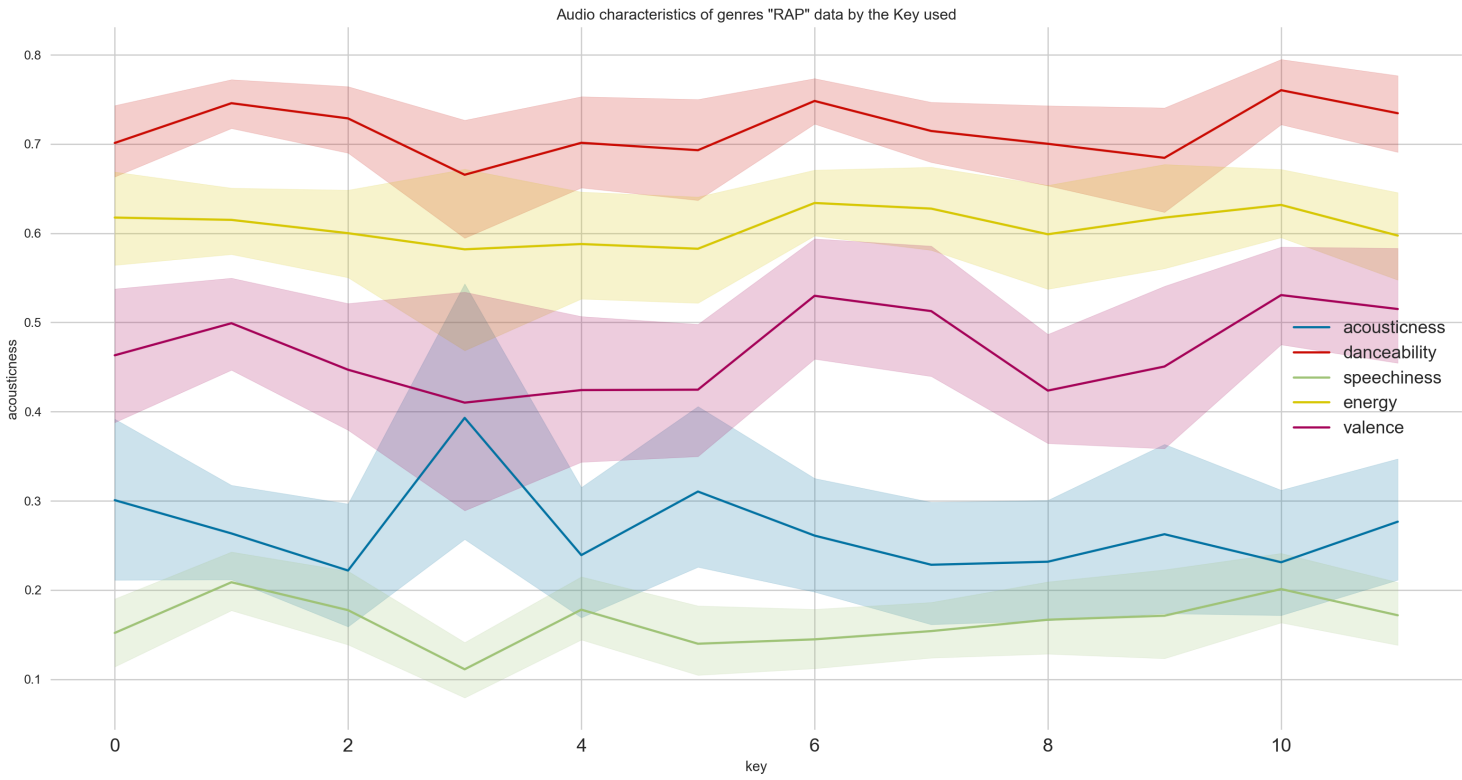
Since based on the genres audio characteristics will modify by key used where all keys on octave encoded as values ranging from 0 to 11, starting on C as 0, C# as 1 and so on. Thus, analysing these audio characteristics like “acousticness”, “danceability”, “speechiness”, “energy” and “valence” whose values ranges between 0 and 1.

```
df_edm = r.spotifyAllCol[r.spotifyAllCol['playlist_genre'] == "edm"]
df_rap = r.spotifyAllCol[r.spotifyAllCol['playlist_genre'] == "rap"]
df_latin = r.spotifyAllCol[r.spotifyAllCol['playlist_genre'] == "latin"]
df_pop = r.spotifyAllCol[r.spotifyAllCol['playlist_genre'] == "pop"]
```

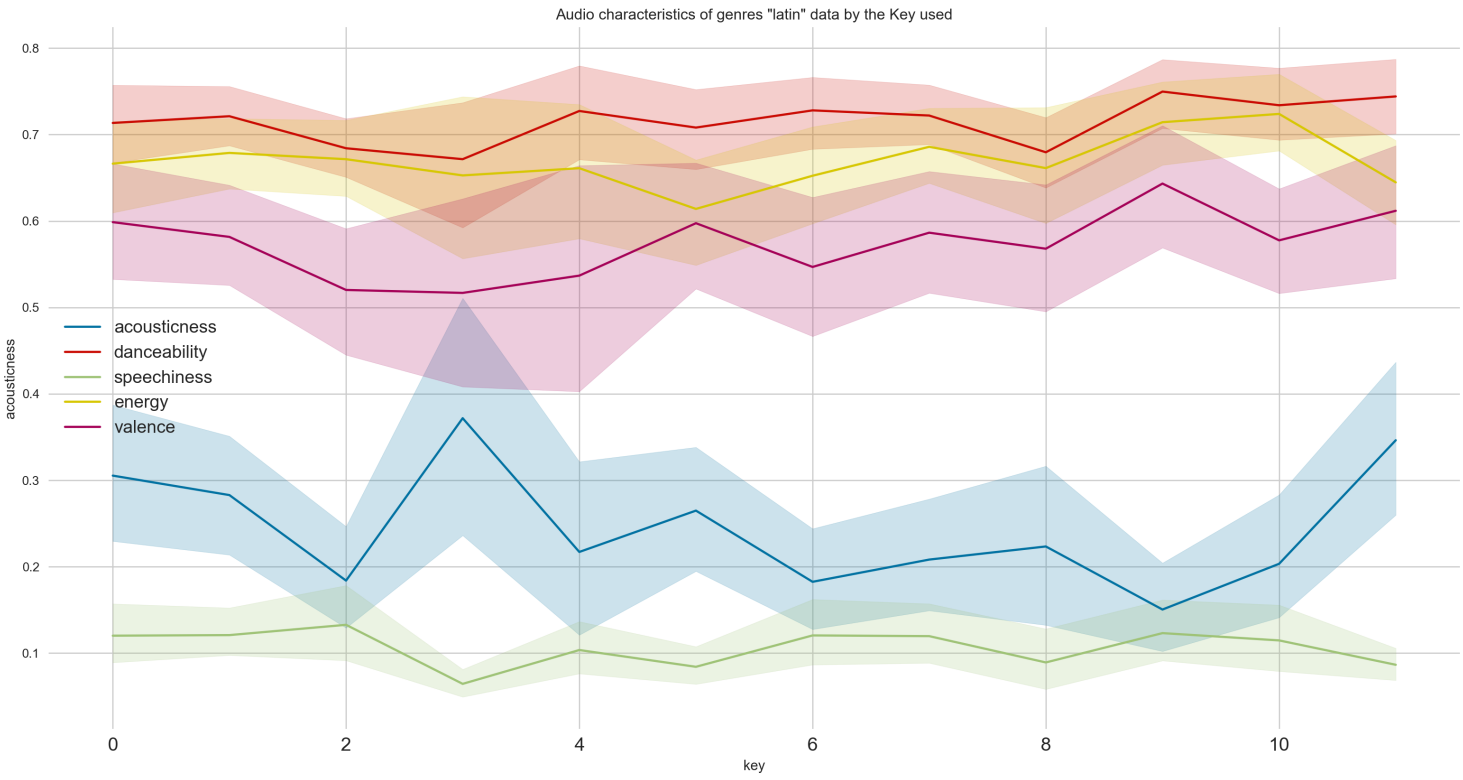
```
#so lets check all the audio characteristics with top 4 geners
# first lets start with "EDM" which is on TOP 1
fig,ax = plt.subplots(figsize=(20, 10))
sns.despine(fig, left=True, bottom=True)
# sns.set_context("notebook", font_scale=2, rc={"lines.linewidth": 3})

sns.lineplot(x="key", y="acousticness", data=df_edm, color="b",label = 'acousticness')
sns.lineplot(x="key", y="danceability", data=df_edm, color="r",label = 'danceability')
sns.lineplot(x="key", y="speechiness", data=df_edm, color="g",label = 'speechiness')
sns.lineplot(x="key", y="energy", data=df_edm, color="y",label = 'energy')
sns.lineplot(x="key", y="valence", data=df_edm, color="m",label = 'valence')

plt.rcParams["xtick.labelsize"] = 15

ax.set_title('Audio characteristics of genres "EDM" data by the Key used')
ax.legend(fontsize = 14)
plt.show()
# now lets check for top2 genre that is RAP
```



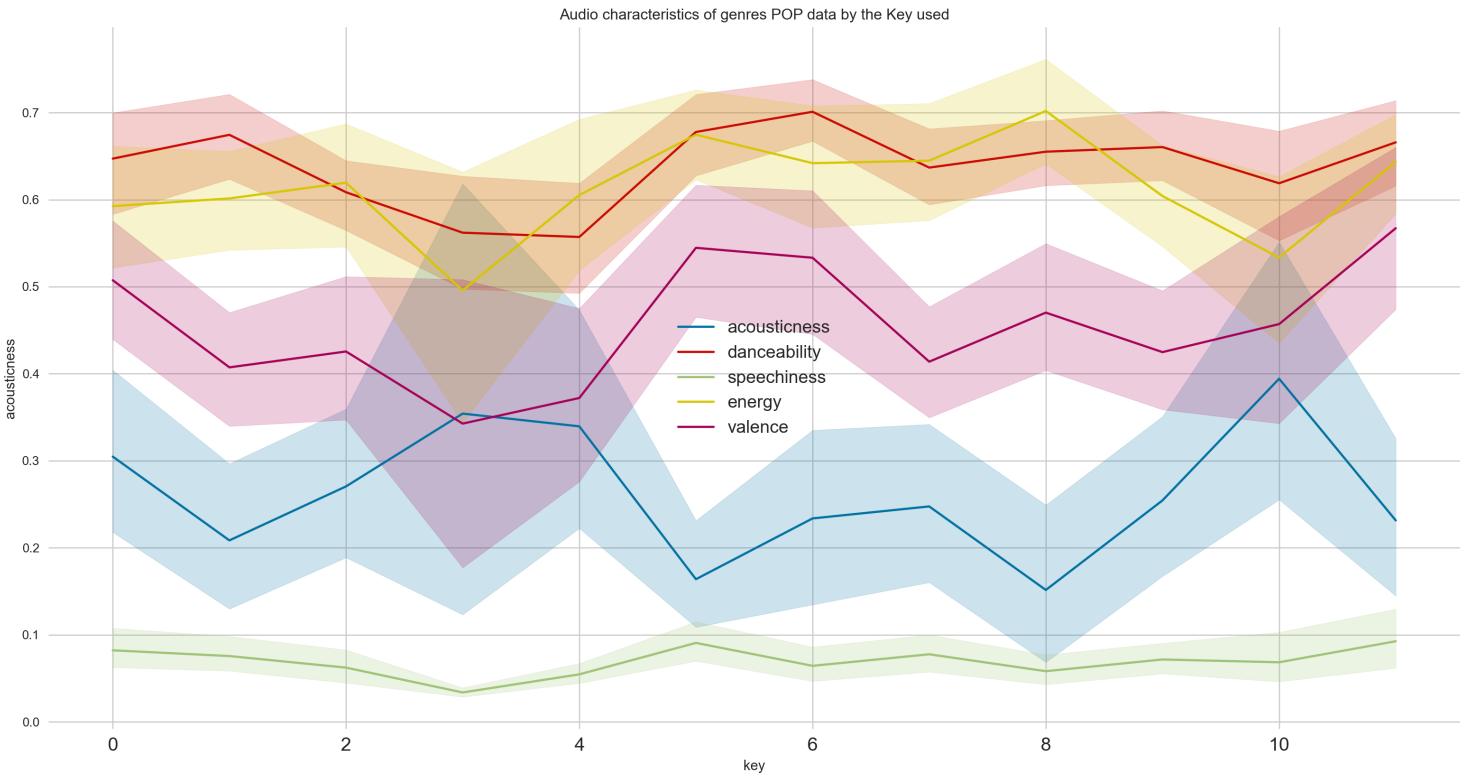

```
fig,ax = plt.subplots(figsize=(20, 10))
sns.despine(fig, left=True, bottom=True)
# sns.set_context("notebook", font_scale=2, rc={"lines.linewidth": 3})

sns.lineplot(x="key", y="acousticness", data=df_rap, color="b",label = 'acousticness')
sns.lineplot(x="key", y="danceability", data=df_rap, color="r",label = 'danceability')
sns.lineplot(x="key", y="speechiness", data=df_rap, color="g",label = 'speechiness')
sns.lineplot(x="key", y="energy", data=df_rap, color="y",label = 'energy')
sns.lineplot(x="key", y="valence", data=df_rap, color="m",label = 'valence')

plt.rcParams["xtick.labelsize"] = 15

ax.set_title('Audio characteristics of genres "RAP" data by the Key used')
ax.legend(fontsize = 14)
plt.show()
# now lets check for top3 genre that is latin
```



```
fig,ax = plt.subplots(figsize=(20, 10))
sns.despine(fig, left=True, bottom=True)
# sns.set_context("notebook", font_scale=2, rc={"lines.linewidth":3})

sns.lineplot(x="key", y="acousticness", data=df_latin, color="b", label = 'acousticness')
sns.lineplot(x="key", y="danceability", data=df_latin, color="r", label = 'danceability')
sns.lineplot(x="key", y="speechiness", data=df_latin, color="g", label = 'speechiness')
sns.lineplot(x="key", y="energy", data=df_latin, color="y", label = 'energy')
sns.lineplot(x="key", y="valence", data=df_latin, color="m", label = 'valence')

plt.rcParams["xtick.labelsize"] = 15

ax.set_title('Audio characteristics of genres "latin" data by the Key used')
ax.legend(fontsize=14)
plt.show()
# now lets check for top3 gener that is POP
```




```
fig,ax = plt.subplots(figsize=(20, 10))
sns.despine(fig, left=True, bottom=True)
# sns.set_context("notebook", font_scale=2, rc={"lines.linewidth":3})

sns.lineplot(x="key", y="acousticness", data=df_pop, color="b", label = 'acousticness')
sns.lineplot(x="key", y="danceability", data=df_pop, color="r", label = 'danceability')
sns.lineplot(x="key", y="speechiness", data=df_pop, color="g", label = 'speechiness')
sns.lineplot(x="key", y="energy", data=df_pop, color="y", label = 'energy')
sns.lineplot(x="key", y="valence", data=df_pop, color="m", label = 'valence')

plt.rcParams["xtick.labelsize"] = 15


ax.set_title('Audio characteristics of genres POP data by the Key used')
ax.legend(fontsize=14)
plt.show()
```

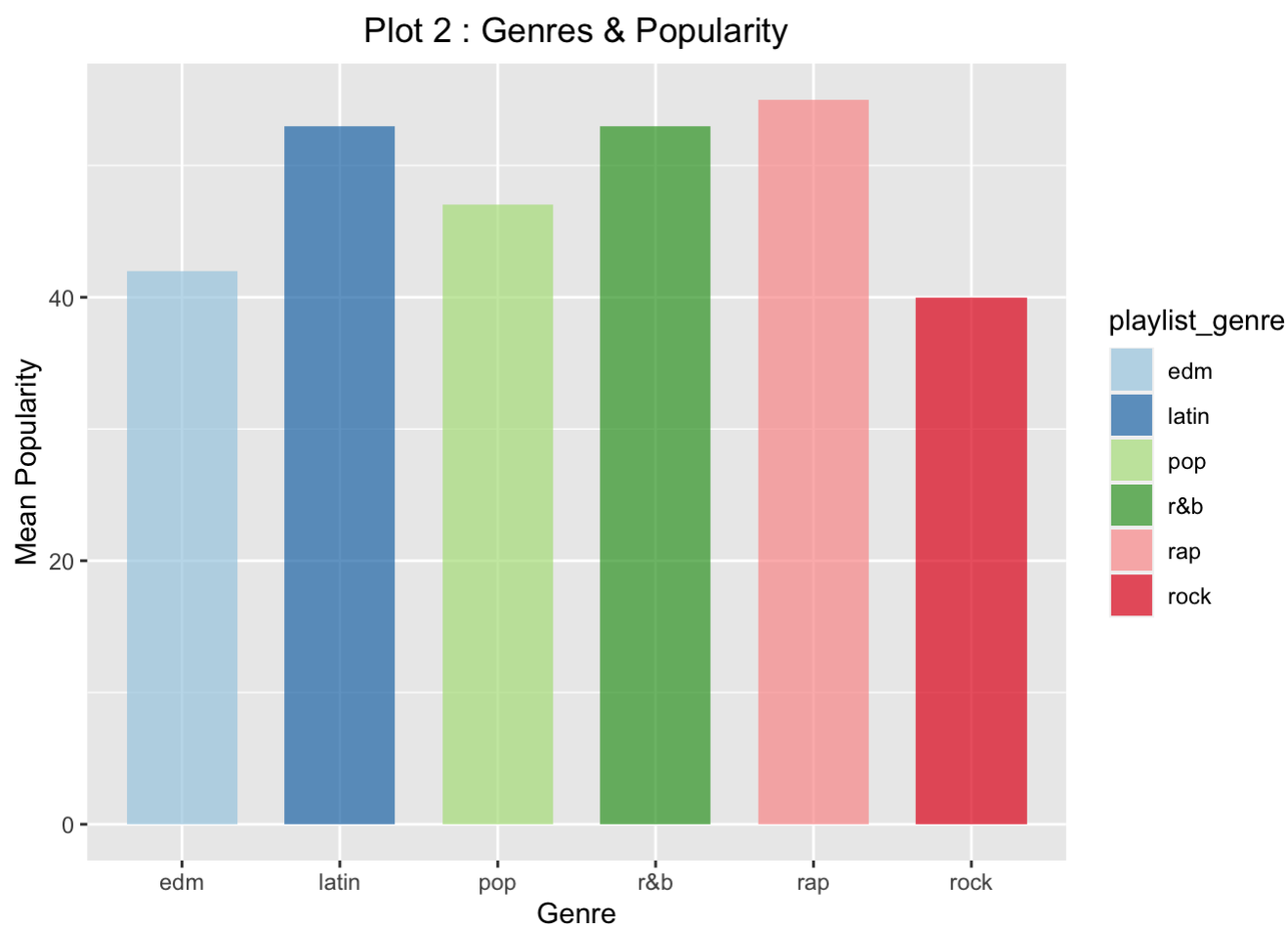


To understand genres better, genres are plotted by their average popularity. From below plot, it can be seen that the maximum number of popular songs belong to :

- Pop
- Latin
- Rap

Hence in cluster analysis, the focus can be seen on these genres.

```
avg_popularity <- spotify %>%
                  select(track_popularity, playlist_genre) %>%
                  group_by(playlist_genre) %>%
                  summarise("average_popularity" = round(mean(track_popularity)))

p2 <- ggplot(data=avg_popularity,
             mapping = aes(x = (playlist_genre),
                           y = average_popularity,
                           fill = playlist_genre)) +
      geom_col(width = 0.7,alpha=0.7) +
      scale_fill_brewer(palette = "Paired") +
      ggtitle("Plot 2 : Genres & Popularity") +
      xlab("Genre") + ylab("Mean Popularity") +
      theme(plot.title = element_text(hjust = 0.5))

grid.arrange(p2, nrow=1, ncol=1)
```



Keys & Mode

In music “key” is short for “key signature” and refers to an ascending series of notes that will be used in a melody, and to the number of sharps or flats in the scale.

But will a mode enhance the key?

To understand keys & mode better,a chi square test on them reveal information as they are categorical variables

Ho : Key is independent of mode Ha : Key is not independent of mode On applying the chisq.test function on the two variables, the p-value is found to be lesser than 2.2e-16, which is too significant for an α of 0.05.

Hence, the null hypothesis is rejected.

Therefore, the key is dependent on mode, and the mode will sharpen the keys.

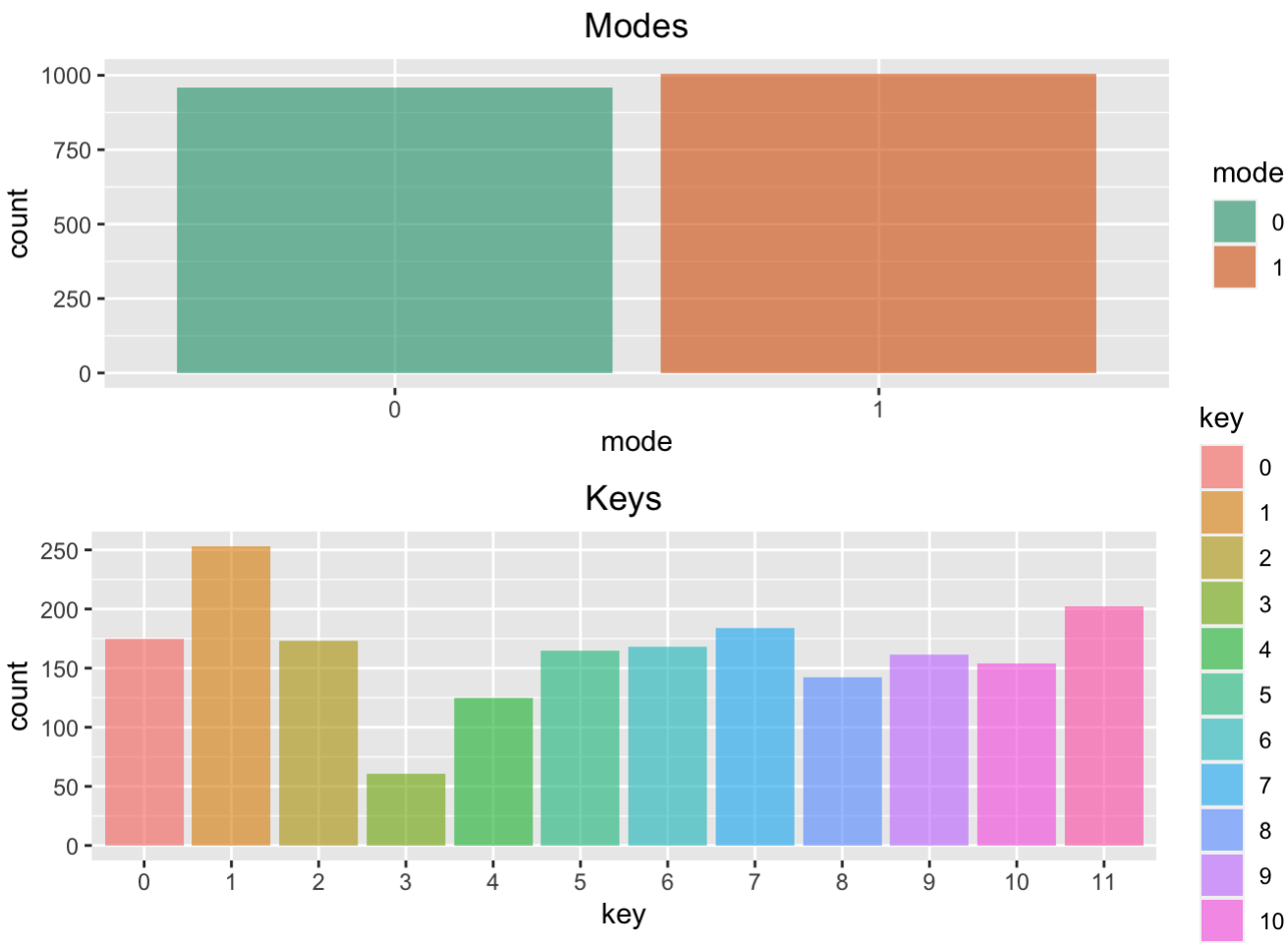
```
chisq.test(spotify$key, spotify$mode)
```

```
##
##  Pearson's Chi-squared test
##
## data:  spotify$key and spotify$mode
## X-squared = 204.24, df = 11, p-value < 2.2e-16
```

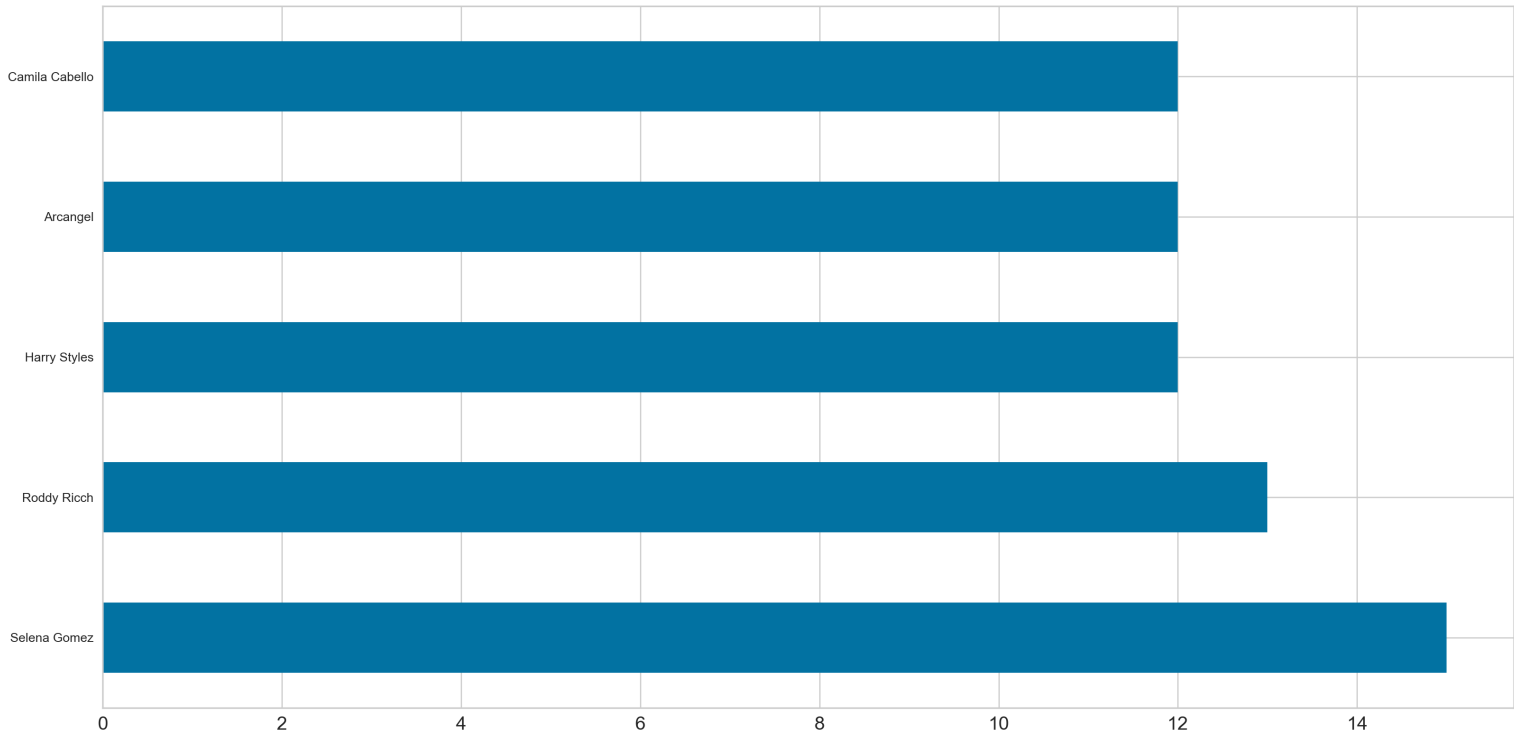
From the chart below it is seen that songs have mode 1 (major track) more often than mode 2(minor track).

Pitch 1 is the most frequenly key occuring in songs

```
#Plotting Mode & Keys

g1 <- ggplot(spotify,aes(mode)) +
      geom_bar(aes(fill=mode),alpha = 0.6) +
      ggtitle("Modes") +
      theme(plot.title = element_text(hjust = 0.5)) +
      scale_fill_brewer(palette = "Dark2")
g2 <- ggplot(spotify,aes(key)) +
      geom_bar(aes(fill=key), alpha = 0.6) +
      ggtitle("Keys") +
      theme(plot.title = element_text(hjust = 0.5))

grid.arrange(g1,g2,ncol=1)
```




```
r.spotifyAllCol['track_artist'].value_counts()[0:5].plot.barh()
```



Approach for Clustering:

- 70% of the data is split as train set and rest 30% as test set.
- The data is scaled to make the numerical attributes comparable
- Understand behaviour of numerical attributes from the correlation plot
- Find the optimal number of centers using elbow method to implement K-Means Clustering
- Fit the K Means Clustering Model
- Group the clusters and the attributes by their mean
- Understand the accuracy of the model
- In-depth analysis cluster wise
- Interpretation of Model and Results

```
spotify <- spotify %>% select(1:15,17)
#Splitting Data into Train & Test
index <- sample(nrow(spotify), 0.7*nrow(spotify))
train_kmeans <- spotify[index,]
test_kmeans <- spotify[-index,]
```

Standardization is an important step in data preprocessing, as it controls the variability of the dataset. It is used to limit the values between -1 and 1 for numeric columns. Therefore, I have scaled the data before implementing K-Means Clustering.

```
#Scaling Data
train_scale <- scale(train_kmeans[,-c(1,2,4,7,9,16)])
test_scale <- scale(test_kmeans[,-c(1,2,4,7,9,16)])
```

```
colnames(train_kmeans)
```

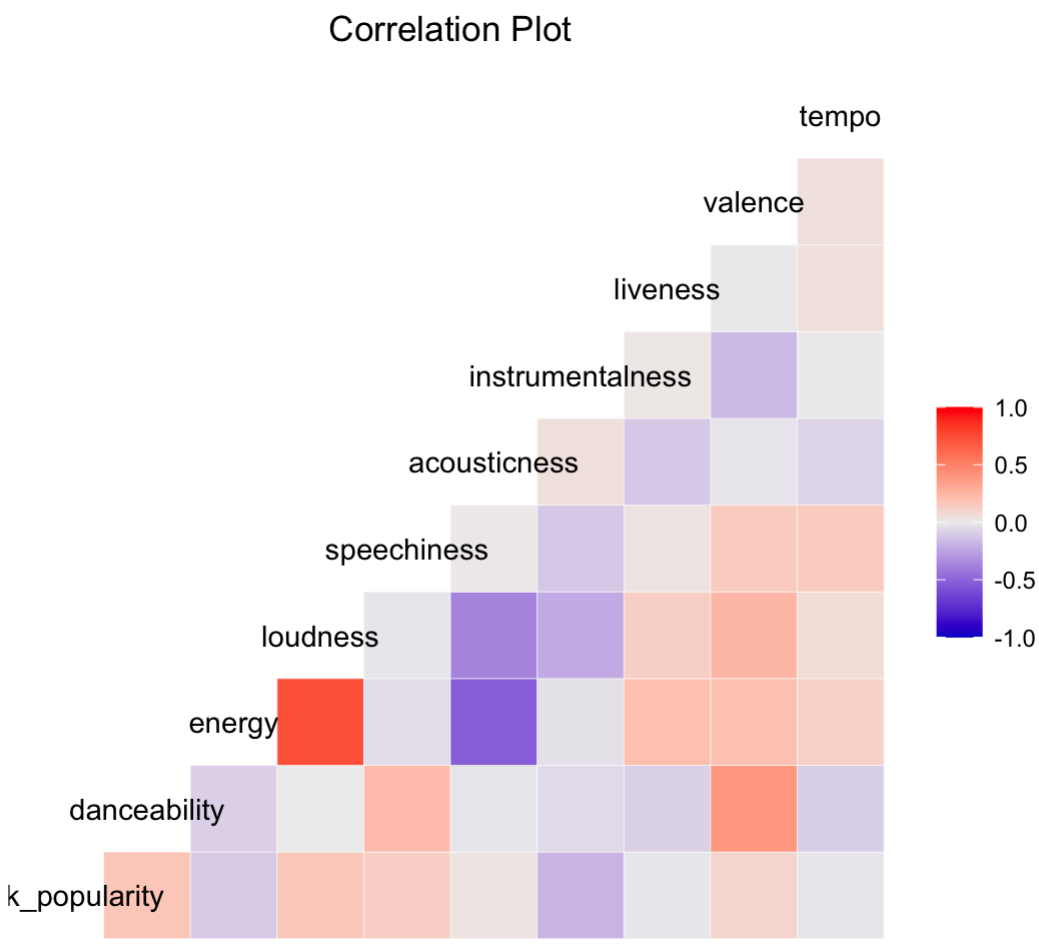
```
##  [1] "track_name"       "track_artist"     "track_popularity" "playlist_genre"
##  [5] "danceability"     "energy"           "key"              "loudness"
##  [9] "mode"             "speechiness"      "acousticness"     "instrumentalness"
## [13] "liveness"         "valence"          "tempo"            "duration_cat"
```

Correlation Plot Insights:

The plot below gives the following top few insights:

Energy has a high positive correlation with loudness and a negative correlation with acousticness. It is also positively related to liveness Like energy, loudness and tempo are negatively related with acoustiness, i.e as acousticness increases, loudness and tempo decrease. Therefore, as expected popularity is negatively correlated with energy, liveness, instrumentalness and positively associated with danceability, loudness and acousticness Valence and Danceability have a positive relation

```
# Correlation Plot
ggcorr(train_scale,
       low = "blue3",
       high = "red") +
      ggtitle("Correlation Plot") +
      theme(plot.title = element_text(hjust = 0.5))
```



K-Means clustering is a simple and quick algorithm which deals with large data sets easily.

The idea behind K-Means is in grouping the data into clusters such that the variation inside the clusters (also known as total within-cluster sum of square or WSS) is minimum, and the variation within the clusters is maximum.

This helps in understanding which songs tend to be popular in which groups

General K-Means Process

Identify the number of clusters (K) to be created, in this analysis Elbow Method has been used for the same Select optimally identified k objects from the data set as the cluster centers and fit the kmeans model Plot the clusters Measure the accuracy Elbow method

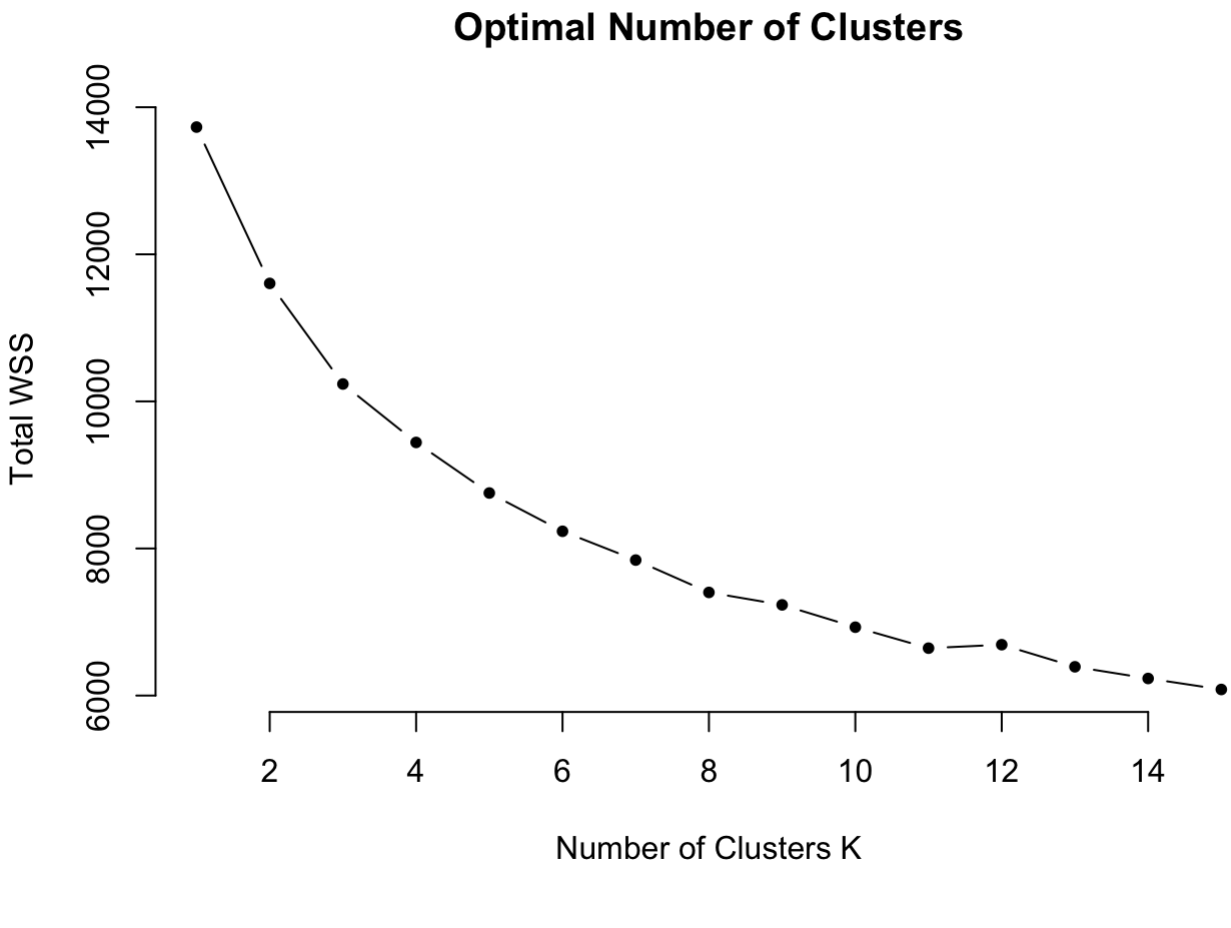
One reason for using this method is that it chooses the correct number of clusters over random assignment of samples to clusters.

In this method, a wss curve is plotted according to the number of clusters k. The location of a bend (knee) in the plot is considered as an indicator of the appropriate number of clusters.

With the elbow method, the ideal number of clusters are identified as 3. Therefore kmeans is implemented with 3 centers.

The total within-cluster sum of square (wss) measures the compactness of the clustering and we want it to be as small as possible.

```
#Elbow Method

wss <- (nrow(train_scale)-1)*sum(apply(train_scale,2,var))

for (i in 2:15) wss[i] <- sum(kmeans(train_scale,centers=i)$withinss)
plot(1:15, wss, type="b", pch=20, frame = FALSE, xlab="Number of Clusters K",ylab="Total WSS",main="Optimal Numbe
r of Clusters")
```



Fitting K Means Model

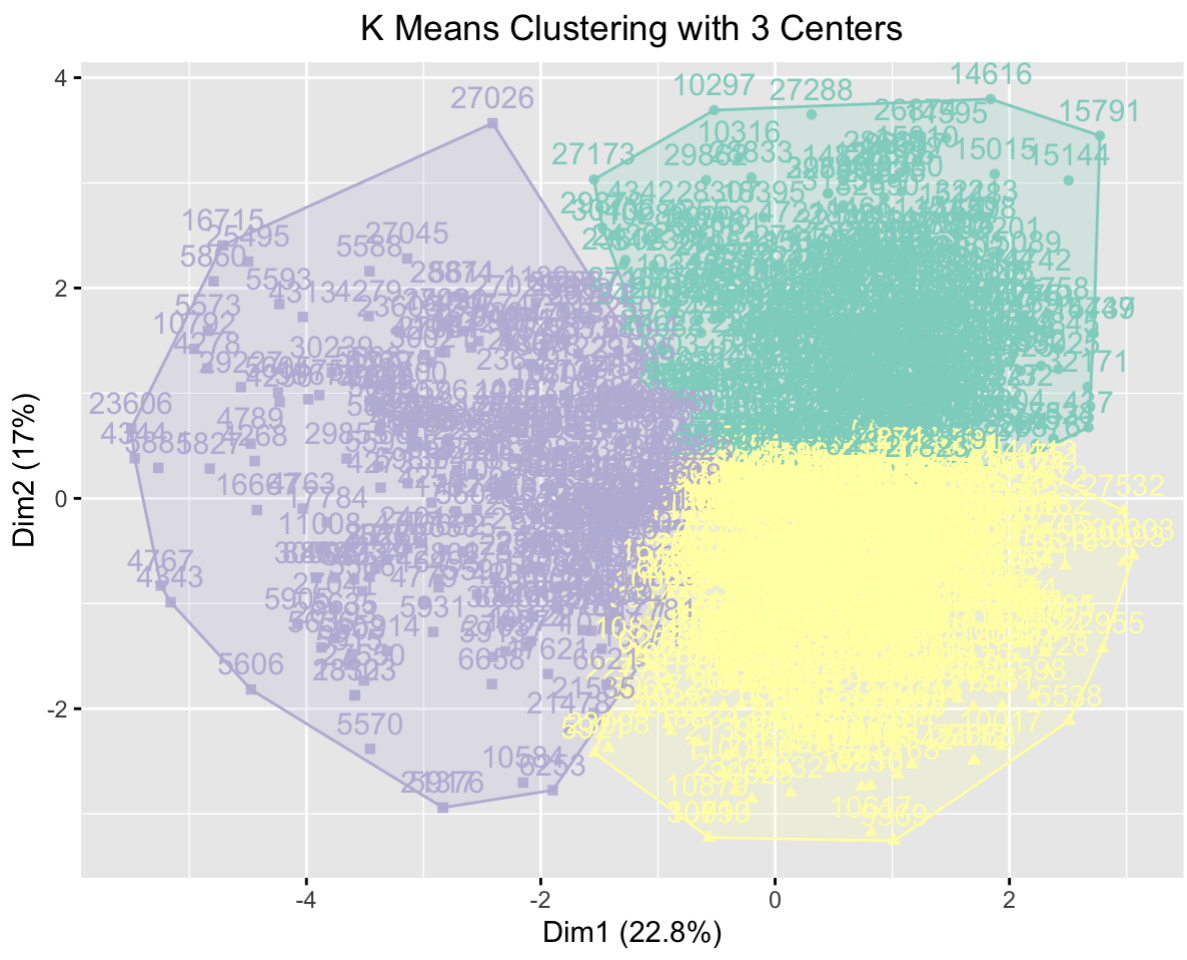
The k means model is fit with 3 centers, while nstart = 25 generates 25 initial configurations and gives out the best one.

As seen from the output this model results in 3 clusters of sizes 7750, 10932, 4297

```
#Fit kmeans
set.seed(13437885)
fit <- kmeans(train_scale, centers = 3, nstart = 25)
fit$size
```

```
## [1] 418 659 297
```

```
#Plotting Kmeans
fviz_cluster(fit,
             geom = c("point", "text"),
             data = train_scale,
             palette = "Set3",
             main = "K Means Clustering with 3 Centers",
             alpha = 0.9) + theme(plot.title = element_text(hjust = 0.5))
```



The clusters are extracted and added to the data to do some descriptive statistics at the cluster level. The datatable below is a result of clustering.

The right most column depicts the cluster the songs belong to, and this will help in further analysis to understand the features of the clusters.

```
#Assiging cluster to df
train_kmeans$cluster <- as.factor(fit$cluster)
datatable(head(train_kmeans,5),options = list(dom = 't',scrollX = T,autoWidth = TRUE))
```

	track_name	track_artist	track_popularity	playlist_genre	danceability	energy	key	loudness
19584	Te Encontré	Mg Poet	36	latin	0.704	0.718	10	-6.791
23102	Life Is Good (feat. Drake)	Future	93	r&b	0.676	0.609	2	-5.831
6330	PAID MY DUES	NF	78	rap	0.812	0.784	2	-4.208
6788	Invierno	Faruz Feet	31	rap	0.557	0.891	1	-6.409
28809	RITMO (Bad Boys For Life) - Steve Aoki Remix	The Black Eyed Peas	63	edm	0.84	0.719	1	-3.333

Interpreting the Quality of Clusters

The BSS is 51208.72.

Between Sum of Squares gives the sum of the squared distance between various cluster centers. The higher it is, the better it is as we want the different cluster centers far apart from each other. A large BSS implies that the characteristics of the clusters are unique and very obviously identifiable.

```
round(fit$betweenss,2)
```

```
## [1] 3493.57
```

The idea is to maximize the `bss/tss%` . To get a high value, we need to increase the number of clusters. But in this case, we found the number of clusters to be ideal at 3, hence we'll stay at it.

```
round((fit$betweenss / fit$totss * 100),2)
```

```
## [1] 25.44
```

Prediction Strength

The prediction strength is defined according to Tibshirani and Walther (2005), who recommend to choose as optimal number of cluster the largest number of clusters that leads to a prediction strength above 0.8 or 0.9.

This function computes the prediction strength of a clustering of a dataset into different numbers of components. The largest cutoff for clusters is 3, hence though there's a low bss/tss% we continue with 3 clusters. The prediction strength for the clusters is decent as it is above 0.5 for all clusters

```
#Prediction Strength
prediction.strength(train_scale, Gmin=2, Gmax=5, M=10,cutoff=0.6)
```

```
## Prediction strength
## Clustering method:  kmeans
## Maximum number of clusters:  5
## Resampled data sets:  10
## Mean pred.str. for numbers of clusters:  1 0.8138715 0.6923703 0.5048142 0.4404646
## Cutoff value:  0.6
## Largest number of clusters better than cutoff:  3
```

Cluster Behaviour Analysis

The behaviours of the clusters can be outlined as below:

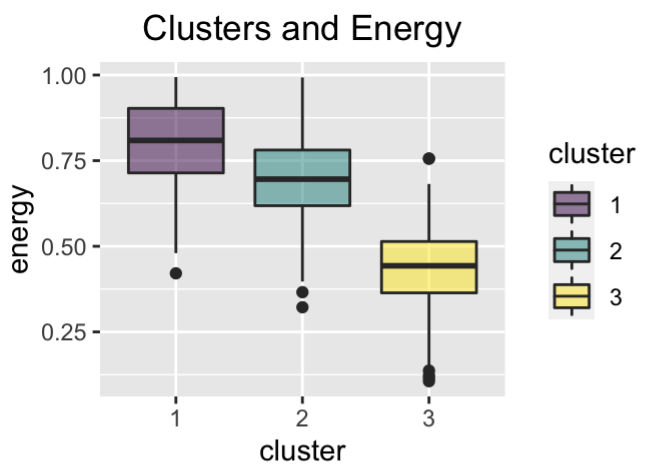
- Cluster 1: Liveness, Energy
- Cluster 1 is second largest
- Cluster 2: Track Popularity, Danceability, Energy, Valence
- Cluster 2 is the largest
- Cluster 3: Acousticness, danceability
- Cluster 3 is the smallest

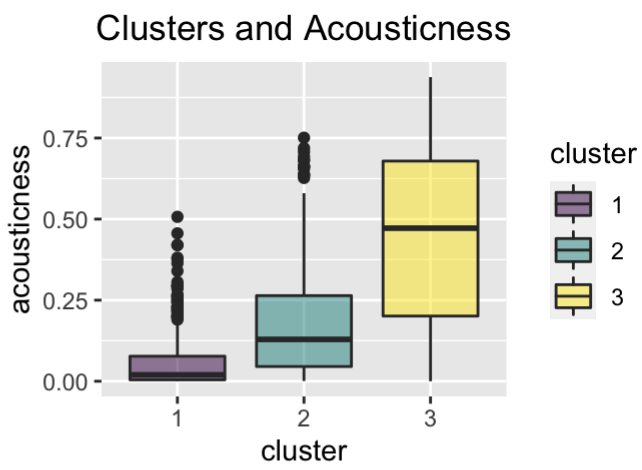
Accousticness and energy vary drastically across the clusters. Hence it will be used in final analysis Popularity of cluster 2 is the highest, followed by cluster 3 and finally cluster 1, but popularity doesn't really distinguish the clusters Similarly danceability is not too distinct amongst the clusters Cluster 1 songs are ranked high on energy Valence is an important virtue for cluster 2 Accousticness is the highest and only significant for cluster 3

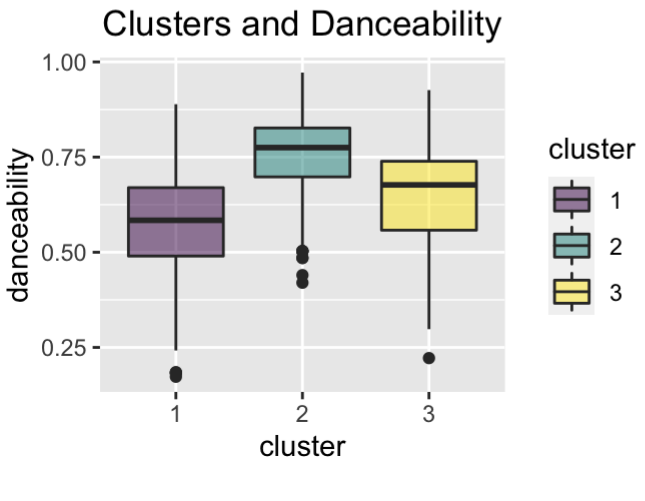
```
#Grouping the Clusters by Mean
cluster_mean <- train_kmeans %>%
                group_by(cluster) %>%
                summarise_if(is.numeric, "mean") %>%
                mutate_if(is.numeric, .funs = "round", digits = 2)

datatable(cluster_mean, options = list(dom = 't',scrollX = T,autoWidth = TRUE))
```

	cluster	track_popularity	danceability	energy	loudness	speechiness	acousticness	instrumentalr
1	1	39.53	0.57	0.8	-5.56	0.08	0.06	
2	2	57.34	0.76	0.7	-5.78	0.14	0.18	
3	3	45.39	0.65	0.43	-10.44	0.1	0.45	

```
#Bar Plots for Clusters
b1 <- train_kmeans %>%
      ggplot(aes(x = cluster,
      y = energy,
      fill = cluster)) +
      geom_boxplot() +
      scale_fill_viridis(option = "D",discrete = TRUE, alpha=0.5) +
      ggtitle("Clusters and Energy") +
      theme(plot.title = element_text(hjust = 0.5))

b2 <- train_kmeans %>%
      ggplot(aes(x = cluster,
      y = acousticness,
      fill = cluster)) +
      geom_boxplot() +
      scale_fill_viridis(option = "D",discrete = TRUE, alpha=0.5) +
      ggtitle("Clusters and Acousticness") +
      theme(plot.title = element_text(hjust = 0.5))

b3 <- train_kmeans %>%
      ggplot(aes(x = cluster,
      y = danceability,
      fill = cluster)) +
      geom_boxplot() +
      scale_fill_viridis(option = "D",discrete = TRUE, alpha=0.5) +
      ggtitle("Clusters and Danceability") +
      theme(plot.title = element_text(hjust = 0.5))

b4 <- train_kmeans %>%
      ggplot(aes(x = cluster,
      y = valence,
      fill = cluster)) +
      geom_boxplot() +
      scale_fill_viridis(option = "D",discrete = TRUE, alpha=0.5) +
      ggtitle("Clusters and Valence") +
      theme(plot.title = element_text(hjust = 0.5))

grid.arrange(b1, b2, b3, b4, nrow=2, ncol=2)
```







Individual Cluster Analysis

For the cluster analysis a baseline for popularity is kept at 90 and above. The popular songs in this cluster are depicted in the table below

Cluster 1 Insights:

Cluster 1 the second largest of the 3 clusters is known for its Liveness, Energy. Since, Pop, rock and rap are the most popular ones, the table for cluster 1 will be based on those.

As expected, most popular songs in cluster 1 are high on energy and low on accousticness

```
#Analysis on Cluster 1
c1 <- train_kmeans[which(train_kmeans$cluster==1), ]

#Grouping cluster by popularity
avg_pop <- c1 %>%
          select(track_popularity, playlist_genre) %>%
          group_by(playlist_genre) %>%
          summarise("average_popularity" = round(mean(track_popularity)))

#Plotting genres across popularity
x1 <- ggplot(data=avg_pop,
             mapping = aes(x = (playlist_genre),
                           y = average_popularity,
                           fill = playlist_genre)) +
      geom_col(width = 0.7,alpha=0.7) +
      scale_fill_brewer(palette = "Spectral") +
      ggtitle("Cluster 1 - Genres & Popularity") +
      xlab("Genre") + ylab("Mean Popularity") +
      theme(plot.title = element_text(hjust = 0.5))
x1
```


```
n <- c1 %>%
  select(track_name,track_artist,playlist_genre,acousticness,energy,track_popularity) %>%
  subset(track_popularity >= 70 & playlist_genre %in% c("rap","rock","pop")) %>%
  distinct(track_name,.keep_all = TRUE)

datatable(n, caption = 'Cluster 1: Top Songs', options = list(scrollX = T, autoWidth = TRUE, order = list((list(6
, 'desc')))))
```

Show 10 **entries** **Search:**

Cluster 1: Top Songs

	track_name	track_artist	playlist_genre	acousticness	energy	track_popularity
2	HIGHEST IN THE ROOM (feat. ROSALÍA & Lil Baby) - REMIX	Travis Scott	rap	0.0567	0.491	89
6	Finally // beautiful stranger	Halsey	pop	0.051	0.55	79
1	Shameless	Camila Cabello	pop	0.0197	0.651	73
5	Answer	ATEEZ	pop	0.0746	0.903	72
3	No Regrets (feat. Don Toliver)	Eminem	rap	0.0048	0.747	71
4	Ordinary Man (feat. Elton John)	Ozzy Osbourne	rock	0.0665	0.599	71

Showing 1 to 6 of 6 entries Previous 1 Next

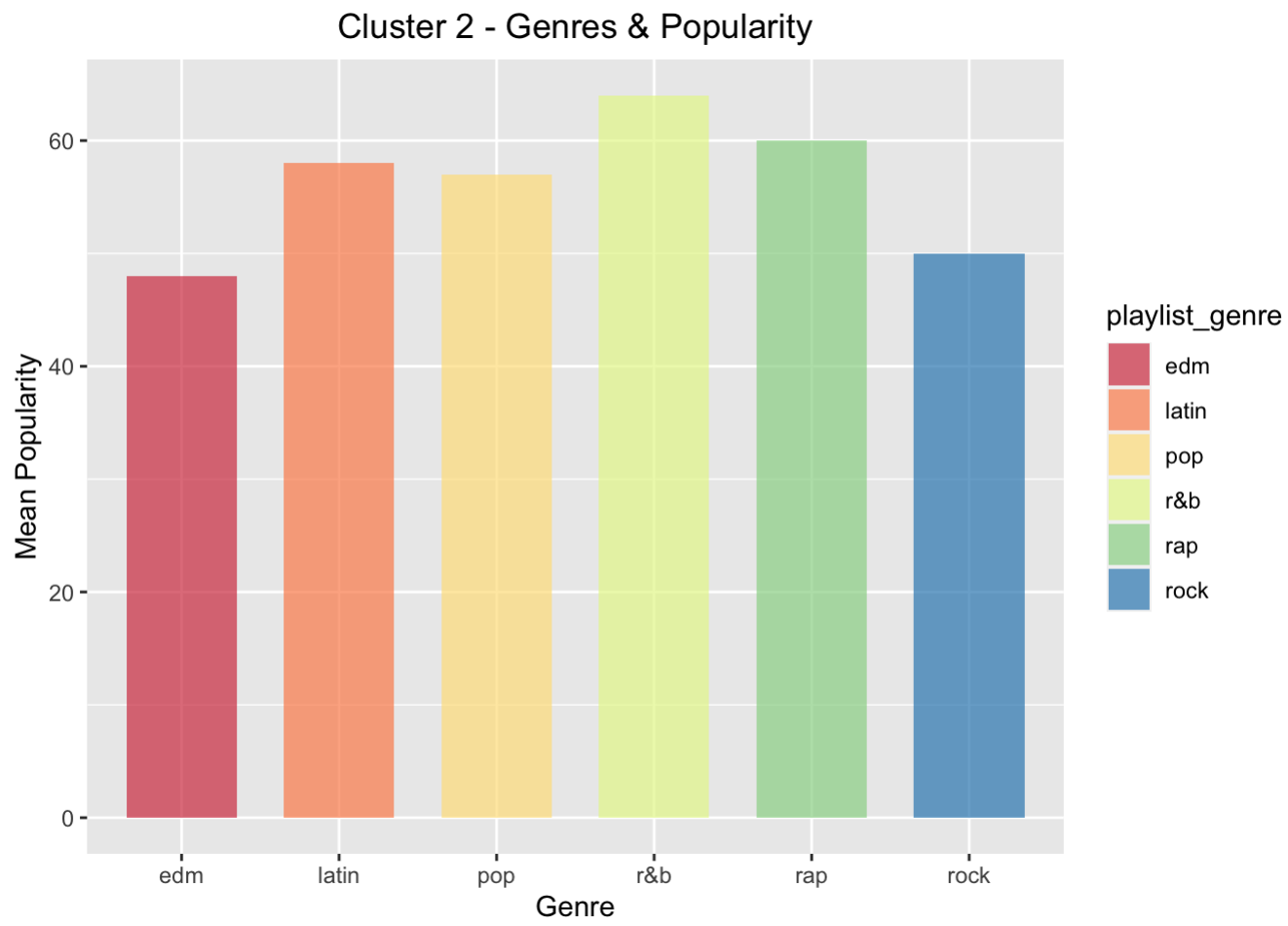
Cluster 2 Insights:

Cluster 2 has the most popular tracks purely coz of the size, and its tracks also have the highest Danceability, Energy, Valence

Therfore the most popular genres are - pop,latin,rock

The cluster two songs are high on energy and low on acousticness.

```
#Analysis on Cluster 2
c2 <- train_kmeans[which(train_kmeans$cluster==2), ]

#Grouping cluster by popularity
avg_pop <- c2 %>%
          select(track_popularity, playlist_genre) %>%
          group_by(playlist_genre) %>%
          summarise("average_popularity" = round(mean(track_popularity)))

#Plotting genres across popularity
x2 <- ggplot(data=avg_pop,
             mapping = aes(x = (playlist_genre),
                           y = average_popularity,
                           fill = playlist_genre)) +
      geom_col(width = 0.7,alpha=0.7) +
      scale_fill_brewer(palette = "Spectral") +
      ggtitle("Cluster 2 - Genres & Popularity") +
      xlab("Genre") + ylab("Mean Popularity") +
      theme(plot.title = element_text(hjust = 0.5))
x2
```




```
n <- c2 %>%
  select(track_name,track_artist,playlist_genre,acousticness,energy,track_popularity) %>%
  subset(track_popularity >= 70 & playlist_genre %in% c("latin","rock","pop","rap")) %>%
  distinct(track_name,.keep_all = TRUE)

datatable(n, caption = 'Cluster 2: Top Songs', options = list(scrollX = T, autoWidth = TRUE, order = list((list(6
, 'desc')))))
```

Show 10 **entries** **Search:**

Cluster 2: Top Songs

	track_name	track_artist	playlist_genre	acousticness	energy	track_popularity
31	The Box	Roddy Ricch	rap	0.104	0.586	98
58	Yummy	Justin Bieber	pop	0.404	0.519	95
23	My Oh My (feat. DaBaby)	Camila Cabello	pop	0.018	0.491	94
3	Life Is Good (feat. Drake)	Future	latin	0.0706	0.609	93
64	Futsal Shuffle 2020	Lil Uzi Vert	rap	0.0327	0.457	91
8	Adore You	Harry Styles	pop	0.0237	0.771	88
27	Own It (feat. Ed Sheeran & Burna Boy)	Stormzy	latin	0.00838	0.781	88
40	Rare	Selena Gomez	latin	0.208	0.545	88

2	Psycho	Red Velvet	latin	0.193	0.666	87
57	OUT WEST (feat. Young Thug)	JACKBOYS	rap	0.0104	0.591	87

Showing 1 to 10 of 86 entries

Previous 1 2 3 4 5 … 9 Next

Cluster 3 Insights

Cluster 3 is the smallest and its tracks have the attributes of high acousticness, danceability and mid level energy compared to other clusters.

Therfore the most popular genres are - pop,latin,rap

The popular songs are high on acousticness with average energy.

```
#Analysis on Cluster 3
c3 <- train_kmeans[which(train_kmeans$cluster==3), ]

#Grouping cluster by popularity
avg_pop <- c3 %>%
          select(track_popularity, playlist_genre) %>%
          group_by(playlist_genre) %>%
          summarise("average_popularity" = round(mean(track_popularity)))

#Plotting genres across popularity
x3 <- ggplot(data=avg_pop,
             mapping = aes(x = (playlist_genre),
                           y = average_popularity,
                           fill = playlist_genre)) +
      geom_col(width = 0.7,alpha=0.7) +
      scale_fill_brewer(palette = "Spectral") +
      ggtitle("Cluster 3 - Genres & Popularity") +
      xlab("Genre") + ylab("Mean Popularity") +
      theme(plot.title = element_text(hjust = 0.5))
x3
```


```
n <- c3 %>%
  select(track_name,track_artist,playlist_genre,acousticness,energy,track_popularity) %>%
  subset(track_popularity >= 70 & playlist_genre %in% c("latin","rap","pop","r&b")) %>%
  distinct(track_name,.keep_all = TRUE)

datatable(n, caption = 'Cluster 3: Top Songs', options = list(scrollX = T, autoWidth = TRUE, order = list((list(6
, 'desc')))))
```

Show 10 **entries** **Search:**

Cluster 3: Top Songs

	track_name	track_artist	playlist_genre	acousticness	energy	track_popularity

15	Falling	Harry Styles	r&b	0.839	0.267	88
8	Lose You To Love Me	Selena Gomez	r&b	0.556	0.343	85
16	Takeaway	The Chainsmokers	pop	0.126	0.511	85
13	Eleven	Khalid	r&b	0.112	0.396	83
4	Put A Little Love On Me	Niall Horan	r&b	0.678	0.465	80
6	20/20	Lil Tjay	rap	0.557	0.336	79
10	Changes	Lauv	r&b	0.452	0.455	79
11	Nur zur Info	Ufo361	rap	0.172	0.365	79
17	Wrong Direction	Hailee Steinfeld	pop	0.472	0.388	79
3	La Difícil	Camilo	latin	0.67	0.36	78

Showing 1 to 10 of 17 entries

Previous 1 2 Next

Summary:

This analysis was aimed to understand what makes the clusters different from each other, which also lead us to top songs in each category The analysis was achieved through Visual Exploration, Statistical testing and K means clustering to arrive at the below takeways To a consumer, this analysis will give an overview on the kind of music he should be followinf on spotify based on his tastes.

Key Takeways:

The three clusters do not vary too much on popularity, but instead vary highly on energy and acousticness. The most popular genres turn out to be - Pop, Latin and Rock Cluster two with low acousticness, mid level energy has the the most number of popular songs. One reason for it can be the high danceability associated with cluster 2.

Limitations:

The K clusters were chosen only on elbow method due to its reputation. But an attempt at Gap static and Silhoutte method, would enhance the quality of the analysis. This analysis does not cover predicting popularity of a song, which would be a good project in its own.