Spotify Song Recommendation based on Clustering

Keya Satpathy & Parth Doshi

4/3/2020

Introduction

Brief Overview

Spotify is one of the biggest digital music, podcast, and video streaming service in the world that gives access to millions of songs and other content from artists all over the world. Not only does Spotify gives us access to good songs everywhere (work, home, in the car), it has also introduced us to artists that we would never have listened to before and in genres that we had never experienced, Spotify uses very advanced technology to track and identify each song uploaded to its platform.

The Spotify database provides an interesting look into their listening data. Not just the popularity of tracks, but also features of the tracks they have in their library is recorded in their database. In this project, we are trying to analyze a track's popularity based on several audio features provided in the dataset to find answer to 'Can we predict a track's popularity from key features about the song?' We are

also trying to do a custom analysis based on user's listening profile which shall enable Spotify to stock up similar hit tracks more on their platform and let go off songs that are not much popular among the listeners.

I am considering you as a Spotify user. So as a Spotify user, won't you be impressed if you can get a list of the most popular songs tailored to your taste without having to manually search for them extensively? Also, won't you be a happy and recurring customer if you keep on getting a list of top latest music, divided by genre and have easy access to recently released music? We will be solving for providing easy access to popular songs and that is the reason, you, as a Spotify user should be interested.

Methodology Used

We will study if there is a relation between popularity and different audio features and genres. We will do clustering analysis using K-means method to provide song recommendation based on recent user listening on Spotify and will try to reduce dimensionality using Principle Component Analysis (PCA). We are trying to check if we can extract a pattern to do clustering from the available variables in the dataset and how does each cluster differ to predict the most and least popular clusters. We also plan on using association rules for track correlation analysis.

Proposed Analytical Approach

K-means clustering and Association rules will provide information about customer listening behavior which shall help Spotify upsell, cross-sell or combine both to increase profit. Using k-means clustering, we are trying to study if there is correlation between clusters and our popularity rate.

Also, with PCA, we are trying to reduce the dimensionality of the dataset to prevent our model from overfitting. This will enable our future model to be better suited for generalization beyond the training set. We will also analyze how difference in features like acousticness, liveness, loudness, energy, instrumentalness, loudness, valence effect a track's popularity and can remove features which are not affecting our analysis. Models also become more efficient as a reduced feature set boosts learning rates and diminishes computation costs by removing redundant features.

Analysis

Consumer of our analysis would be Spotify's programming team. Our analysis shall enable the team to upsell, cross-sell or combine both to increase profit. Having a better understanding of different clusters and association rules shall enable Spotify to make a better targeted content distribution, leading to reduced churn rate.

For example, if the team knows that 30% of customers who listens to track A also listens to track B, Spotify can market track B to customers shortly after they listen to track A to speed up that process and capture those who might not have otherwise considered listening to track B. Also, for those customers who do not know of track B, getting suggestions will make them happy and impressed. This is how our analysis would help Spotify in providing better services to their consumers and keep them ahead of the curve.

Packages Required

```
library(tidyverse)
library(dplyr)
library(corrplot)
```

Information on Packages used

The packages that we will be using for our analysis are:

tidyverse – The packages under the tidyverse umbrella help us in performing and interacting with the data such as subsetting, transforming, visualizing, etc.

dplyr – It is the most useful package in R for data manipulation. We have used this package to use the pipe function "%>%" to combine different functions. We have also used this for performing several other manipulations on the dataset like selecting columns from the (select()) dataset, grouping different observations together such that the original dataset does not change (group_by()) and creating new columns by preserving the existing variables (mutate()).

corrplot – This package is essentially used to visualize the correlation between different variables.

factoextra – We will use this library for for PCA and k-means analysis (to be included in detail in the final project).

arules – We will use this library to implement Association rules (to be included in detail in the final project).

Data Preparation

Data Source

The dataset used for this project is the Spotify song list prepared by Zaheen Hamidani which we got from kaggle. (https://www.kaggle.com/zaheenhamidani/ultimate-spotify-tracks-db (https://www.kaggle.com/zaheenhamidani/ultimate-spotify-tracks-db))

Data Information

Originally, the dataset was created by Zaheen Hamidani and uploaded to Kaggle in July 2019. Alternately, it is also available in a R package version 2.1.1 Spotify R package (https://www.rcharlie.com/spotifyr/). Charlie Thompson, Josia Parry,Donal Phipps, and Tom Wolff authored this package to make it easier to get data or general metadata arounds songs from Spotify's API. It allows to enter an artist's name and retrieve their entire audio history (collection of all songs) in seconds, along with Spotify's audio features and track/album popularity metrics.

The primary purpose of the data was to analyze the behaviour between valence and all the measures that Spotify API gives for every track. Approximately 10,000 songs were selected per genre and there are 26 genres. But, the same data can also be used to analyze different statistics and obtain other useful information.

There is not much peculiarity in the data. It is moderately clean with only 15 missing values. Since every track made is unique is some sense, we have not done any missing value imputation and have just removed them.

Data Importing

First, we will load the Spotify songs dataset into R to kickstart with the analysis.The dataset has been imported using the read.csv function and saved as 'spotify'.

```
spotify <- read.csv("C:/Users/khann/Desktop/COLLEGE/R/spotify_songs.csv")
```

```
glimpse(spotify)
```

```
## Observations: 32,833
## Variables: 23
## $ track_id                <fct> 6f807x0ima9a1j3VPbc7VN, 0r7CVbZTWZgbT...
## $ track_name              <fct> I Don't Care (with Justin Bieber) - L...
## $ track_artist            <fct> Ed Sheeran, Maroon 5, Zara Larsson, T...
## $ track_popularity        <int> 66, 67, 70, 60, 69, 67, 62, 69, 68, 6...
## $ track_album_id          <fct> 2oCs0DGTsRO98Gh5ZSl2Cx, 63rPSO264uRjW...
## $ track_album_name        <fct> I Don't Care (with Justin Bieber) [Lo...
## $ track_album_release_date <fct> 2019-06-14, 2019-12-13, 2019-07-05, 2...
## $ playlist_name           <fct> Pop Remix, Pop Remix, Pop Remix, Pop ...
## $ playlist_id             <fct> 37i9dQZF1DXcZDD7cfEKhW, 37i9dQZF1DXcZ...
## $ playlist_genre          <fct> pop, pop, pop, pop, pop, pop, pop, po...
## $ playlist_subgenre       <fct> dance pop, dance pop, dance pop, danc...
## $ danceability            <dbl> 0.748, 0.726, 0.675, 0.718, 0.650, 0....
## $ energy                  <dbl> 0.916, 0.815, 0.931, 0.930, 0.833, 0....
## $ key                     <int> 6, 11, 1, 7, 1, 8, 5, 4, 8, 2, 6, 8, ...
## $ loudness                <dbl> -2.634, -4.969, -3.432, -3.778, -4.67...
## $ mode                    <int> 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1...
## $ speechiness             <dbl> 0.0583, 0.0373, 0.0742, 0.1020, 0.035...
## $ acousticness            <dbl> 0.10200, 0.07240, 0.07940, 0.02870, 0...
## $ instrumentalness        <dbl> 0.00e+00, 4.21e-03, 2.33e-05, 9.43e-0...
## $ liveness                <dbl> 0.0653, 0.3570, 0.1100, 0.2040, 0.083...
## $ valence                 <dbl> 0.518, 0.693, 0.613, 0.277, 0.725, 0....
## $ tempo                   <dbl> 122.036, 99.972, 124.008, 121.956, 12...
## $ duration_ms             <int> 194754, 162600, 176616, 169093, 18905...
```

Our dataset has 32,833 observations and 23 variables.

Data Cleaning

1. Removed NA's – Using the colSums function, we observe that there are 5 NA's in track_name, track_artist and track_album_name. We have removed the respective observations using the na.omit function.

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

```
##                 track_id               track_name             track_artist
##                        0                        5                        5
##         track_popularity           track_album_id         track_album_name
##                        0                        0                        5
## 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
```

```
spotify <- na.omit(spotify) #Remove NA's
```

2. Filtered for unique tracks – We have removed all the duplicated tracks using the duplicated function.

```
spotify <- spotify[!duplicated(spotify$track_id),]
```

3. We have converted key and mode to factors since that seemed logical observing the type of data these columns contain.

```
spotify <- spotify  %>%
  mutate(mode = as.factor(mode),
         key = as.factor(key))
```

4. We have also converted duration_ms to duration in mins (duration_min) since it is more sensible for the analysis. We have not removed duration_ms as of now from the dataset and have kept both duration_ms and duration_min. We will remove duration_ms in the final project and will do the analysis only with duration_min.

```
spotify <- spotify %>% mutate(duration_min = duration_ms/60000)
```

5. For exploring the distribution on popularity, we have made new variables that divide popularity into 4 groups for further cluster analysis.

```
spotify <- spotify %>%
  mutate(popularity_group = as.factor(case_when(
    ((track_popularity > 0) & (track_popularity < 20)) ~ "1",
    ((track_popularity >= 20) & (track_popularity < 40))~ "2",
    ((track_popularity >= 40) & (track_popularity < 60)) ~ "3",
    TRUE ~ "4"))
    )


table(spotify$popularity_group)
```

```
##
##    1    2    3    4
## 4182 6162 8975 9033
```

6. We have also removed track_id, track_album_id and playlist_id from the dataset since it is not useful for our analysis. These unique id's have been maintained in the Spotify dataset only to uniquely identify a track in the database.

```
spotify <- spotify %>% select(-c(track_id, track_album_id, playlist_id))
summary(spotify)
```

```
##     track_name                     track_artist   track_popularity
##  Breathe :   18   Queen                  :  130   Min.   :  0.00
##  Paradise:   17   Martin Garrix          :   87   1st Qu.: 21.00
##  Poison  :   16   Don Omar               :   84   Median : 42.00
##  Alive   :   15   David Guetta           :   81   Mean   : 39.34
##  Forever :   14   Dimitri Vegas & Like Mike:   68   3rd Qu.: 58.00
##  Stay    :   14   Drake                  :   68   Max.   :100.00
##  (Other) :28258   (Other)                :27834
##                     track_album_name track_album_release_date
##  Greatest Hits              :  135   2020-01-10:  201
##  Ultimate Freestyle Mega Mix:   42   2013-01-01:  189
##  Gold                       :   34   2019-11-22:  185
##  Rock & Rios (Remastered)   :   29   2019-12-06:  184
##  Asian Dreamer              :   20   2019-11-15:  183
##  Trip Stories               :   20   2008-01-01:  176
##  (Other)                    :28072   (Other)   :27234
##             playlist_name   playlist_genre
##  Indie Poptimism   :  294   edm  :4877
##  Permanent Wave    :  223   latin:4136
##  Hard Rock Workout :  211   pop  :5132
##  Southern Hip Hop  :  174   r&b  :4504
##  post teen pop     :  159   rap  :5398
##  Urban Contemporary:  157   rock :4305
##  (Other)           :27134
##                  playlist_subgenre  danceability        energy
##  southern hip hop         : 1582   Min.   :0.0000   Min.   :0.000175
```

```
##  indie poptimism          : 1547   1st Qu.:0.5610   1st Qu.:0.579000
##  neo soul                 : 1478   Median :0.6700   Median :0.722000
##  progressive electro house: 1460   Mean   :0.6534   Mean   :0.698372
##  electro house            : 1416   3rd Qu.:0.7600   3rd Qu.:0.843000
##  gangster rap             : 1314   Max.   :0.9830   Max.   :1.000000
##  (Other)                  :19555
##       key           loudness       mode        speechiness
##  1      : 3436   Min.   :-46.448   0:12318   Min.   :0.0000
##  0      : 3001   1st Qu.: -8.310   1:16034   1st Qu.:0.0410
##  7      : 2907   Median : -6.261             Median :0.0626
##  9      : 2631   Mean   : -6.818             Mean   :0.1079
##  11     : 2577   3rd Qu.: -4.709             3rd Qu.:0.1330
##  2      : 2478   Max.   :  1.275             Max.   :0.9180
##  (Other):11322
##   acousticness    instrumentalness       liveness         valence
##  Min.   :0.0000   Min.   :0.0000000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0143   1st Qu.:0.0000000   1st Qu.:0.0926   1st Qu.:0.3290
##  Median :0.0797   Median :0.0000206   Median :0.1270   Median :0.5120
##  Mean   :0.1772   Mean   :0.0911294   Mean   :0.1910   Mean   :0.5104
##  3rd Qu.:0.2600   3rd Qu.:0.0065725   3rd Qu.:0.2490   3rd Qu.:0.6950
##  Max.   :0.9940   Max.   :0.9940000   Max.   :0.9960   Max.   :0.9910
##
##      tempo         duration_ms      duration_min     popularity_group
##  Min.   :  0.00   Min.   :  4000   Min.   :0.06667   1:4182
##  1st Qu.: 99.97   1st Qu.:187741   1st Qu.:3.12902   2:6162
##  Median :121.99   Median :216933   Median :3.61555   3:8975
##  Mean   :120.96   Mean   :226575   Mean   :3.77624   4:9033
##  3rd Qu.:134.00   3rd Qu.:254975   3rd Qu.:4.24959
##  Max.   :239.44   Max.   :517810   Max.   :8.63017
##
```

Variables in the dataset

Each row indicates 1 song and column contain attributes for each song.

- *track_id*: Track id on spotify
- *track_name*: Title of the song
- *track_artist*: Name of the artist
- *track_popularity*: Measure the popularity from 0 to 100 based on play number of the track
- *track_album_id*: Album id on Spotify
- *track_album_name*: Album name on Spotify
- *track_album_release_date*: Date when album was released
- *playlist_genre*: Genre of the playlist
- *playlist_subgenre*: Subgenre of the playlist
- *danceability*: Describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat

strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable

- *energy*: Measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy
- *key*: Estimated overall key of the track. Integers map to pitches using standard Pitch Class notation . E.g. 0 = C, 1 = C♯/D???, 2 = D, and so on. If no key was detected, the value is -1.
- *loudness*: Overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks.
- *mode*: Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
- *speechiness*: Detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
- *acousticness*: Measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic
- *instrumentalness*: Measure whether a track contains no vocals. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0
- *liveness*: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
- *valence*: Measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry). The distribution of values for this feature look like this: Valence distribution.

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

Variables in the cleaned dataset

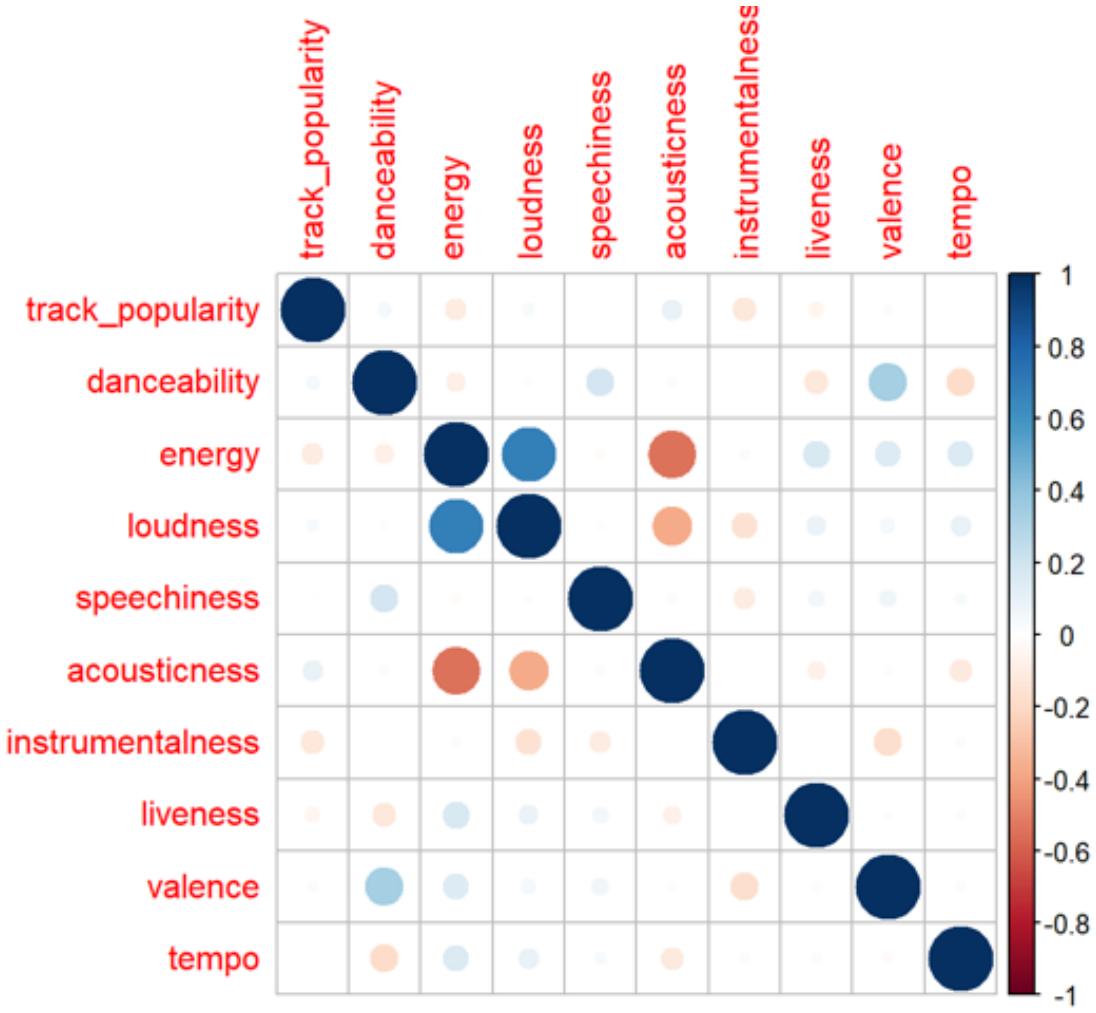
We have removed track_id, track_album_id and playlist_id in the final dataset. We have also included duration_min for duration analysis.Our final dataset has 28352 observations and 22 variables.

Exploratory Data Analysis

The best way to uncover useful information from data that is not self-evident is by performing EDA efficiently. EDA helps us to make sense of our data. Before performing a formal analysis, it is essential to explore a data set. No models can be done without a proper EDA. This will help us to better understand the patterns within the data, detect outliers or anomalous events and find interesting relations among the variables. We have used histograms, boxplots and correlation plot to find such answers.

1. **Correlation plot**

```
df1 <- select(spotify, track_popularity, danceability, energy, loudness, speechine
ss, acousticness, instrumentalness, liveness, valence, tempo)
corrplot(cor(df1))
```

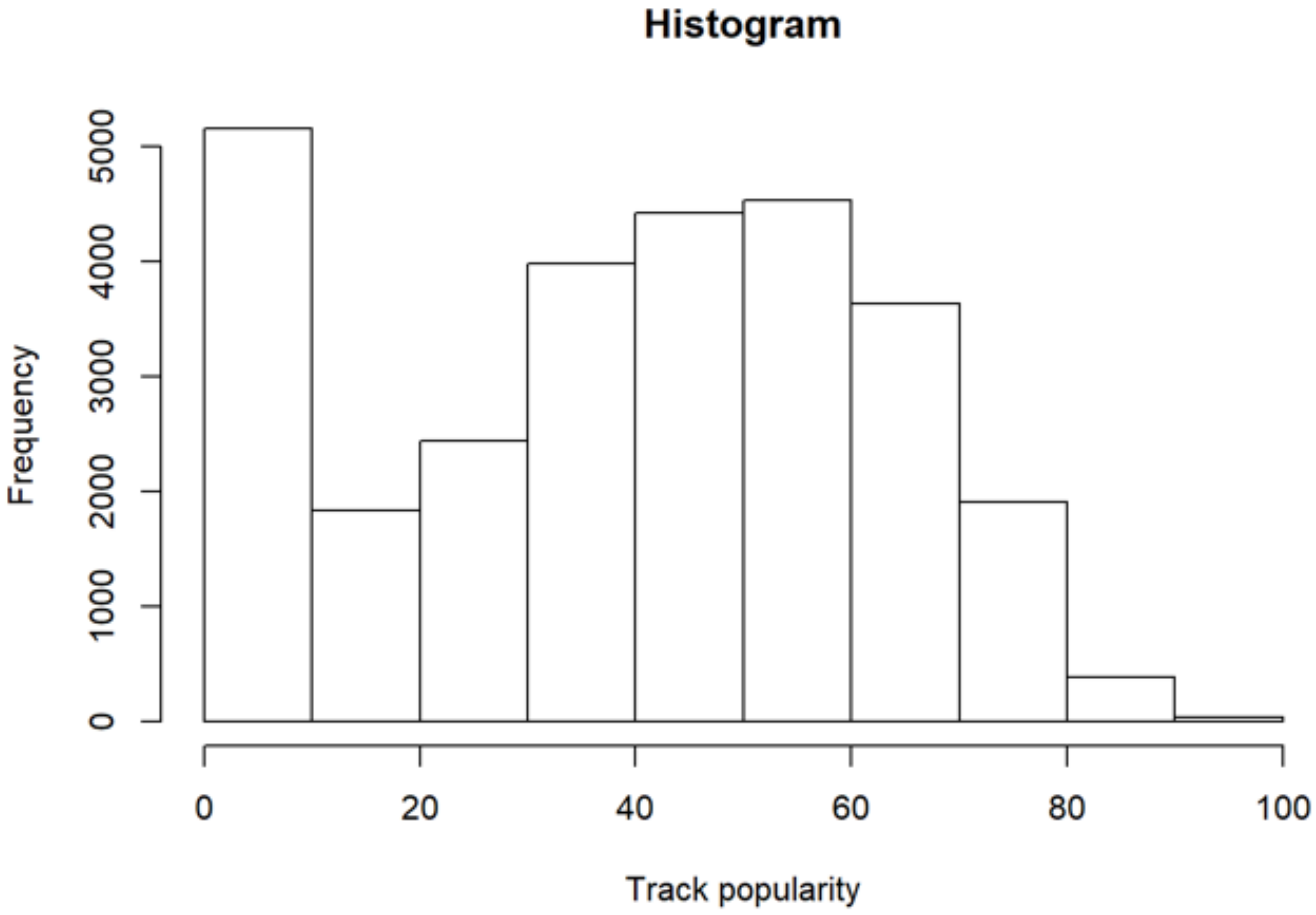


The plot shows popularity does not have strong correlation with other track features. But we found some variables have strong correlation with each other, indicating that this dataset has multicollinearity and might not be suitable for various classification algorithms.

2. **Histogram**

Histogram for track popularity

```
hist(spotify$track_popularity, main = 'Histogram', xlab = 'Track popularity', brea
ks = 10)
```



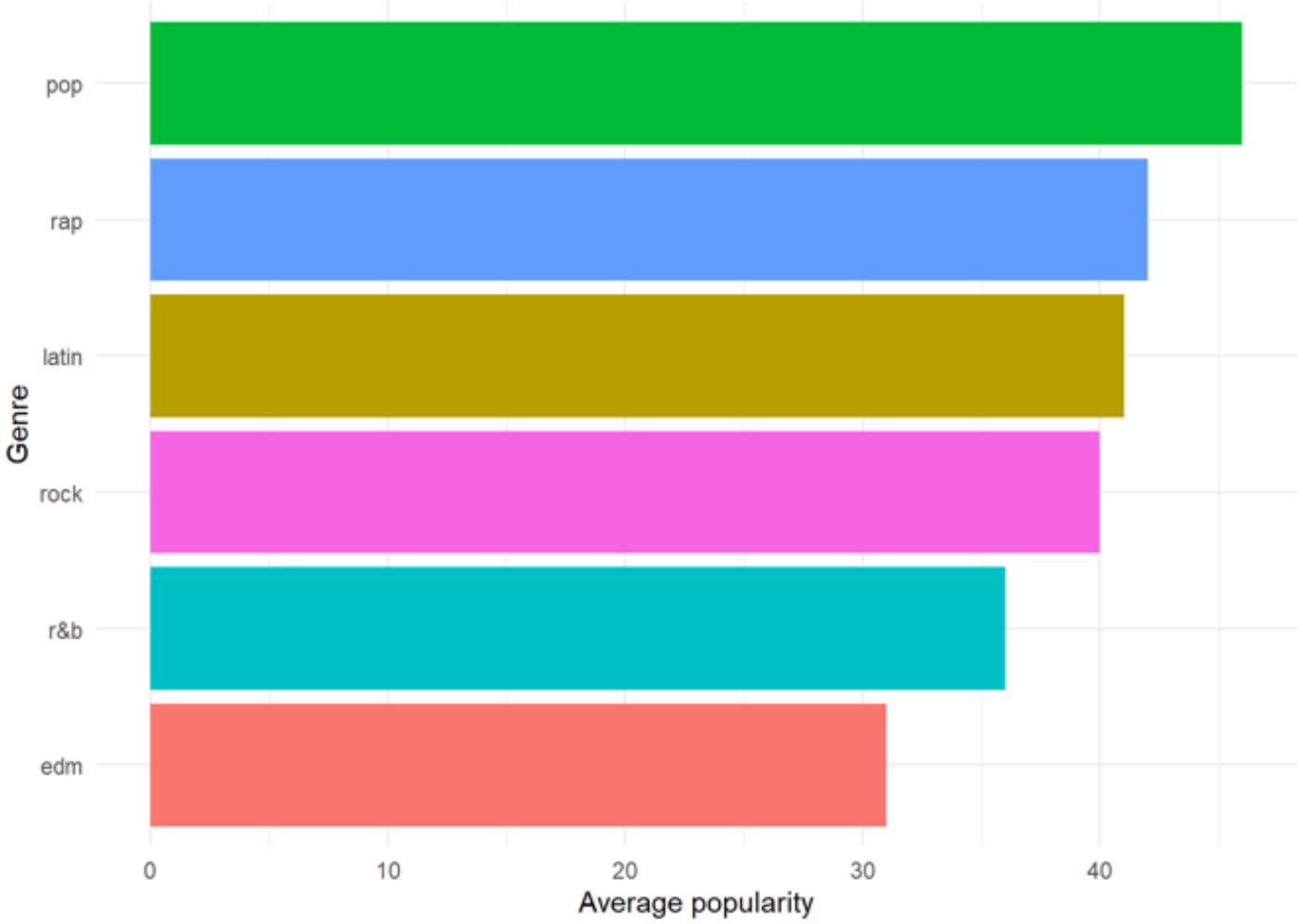
From the histogram above we can check distribution for popularity of tracks. We observe distribution of popularity have spike in the middle, in the range of 50-55.

Histogram for Genre Popularity

```
genre_popularity <- spotify %>% select(track_popularity, playlist_genre) %>% group
_by(playlist_genre) %>% summarise("average_popularity" = round(mean(track_populari
ty)))
```

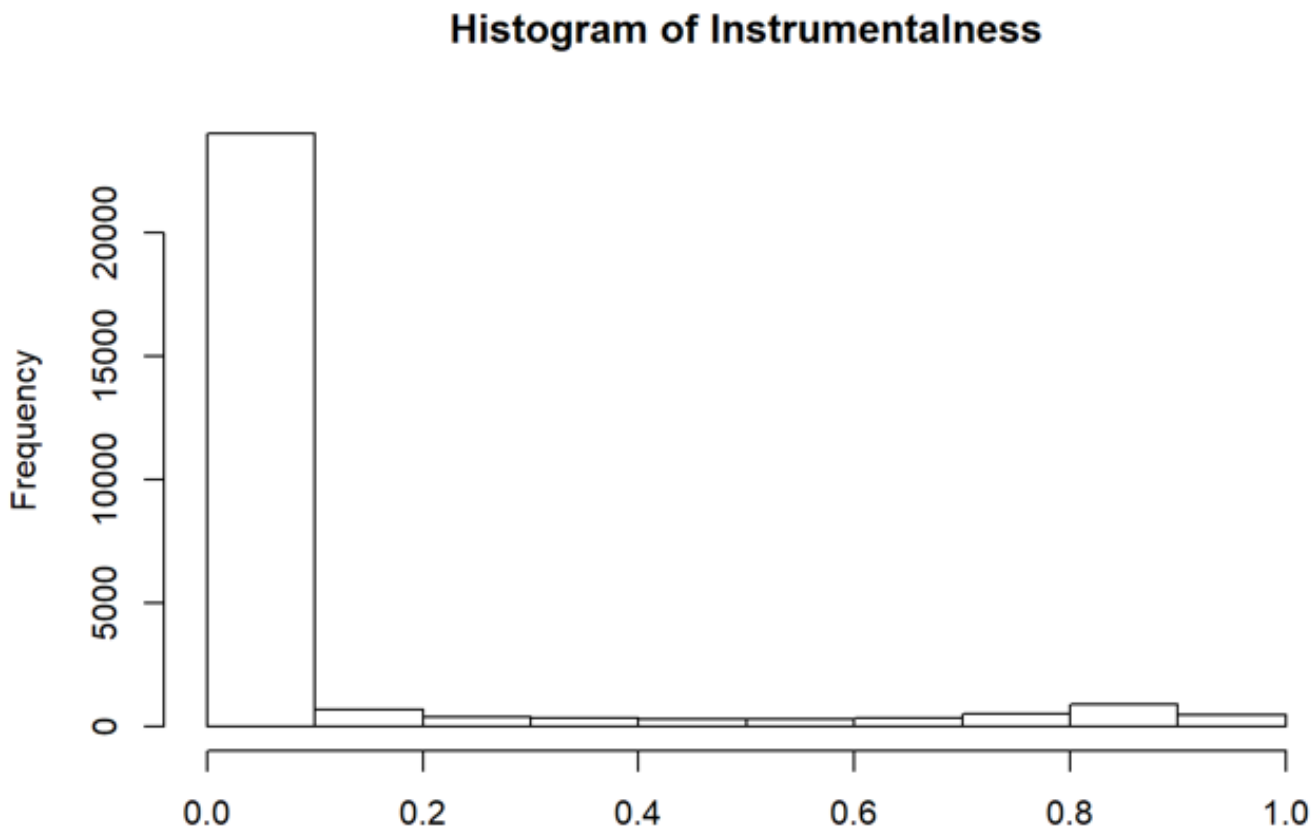
```
ggplot(data=genre_popularity, mapping = aes(x = reorder(playlist_genre,average_pop
ularity), y = average_popularity, fill = playlist_genre)) +
  geom_col() +
  coord_flip() +
  theme_minimal() +
  theme(
    legend.position = "none",

  ) +
  labs(
    y = "Average popularity",
    x = "Genre"
  )
```



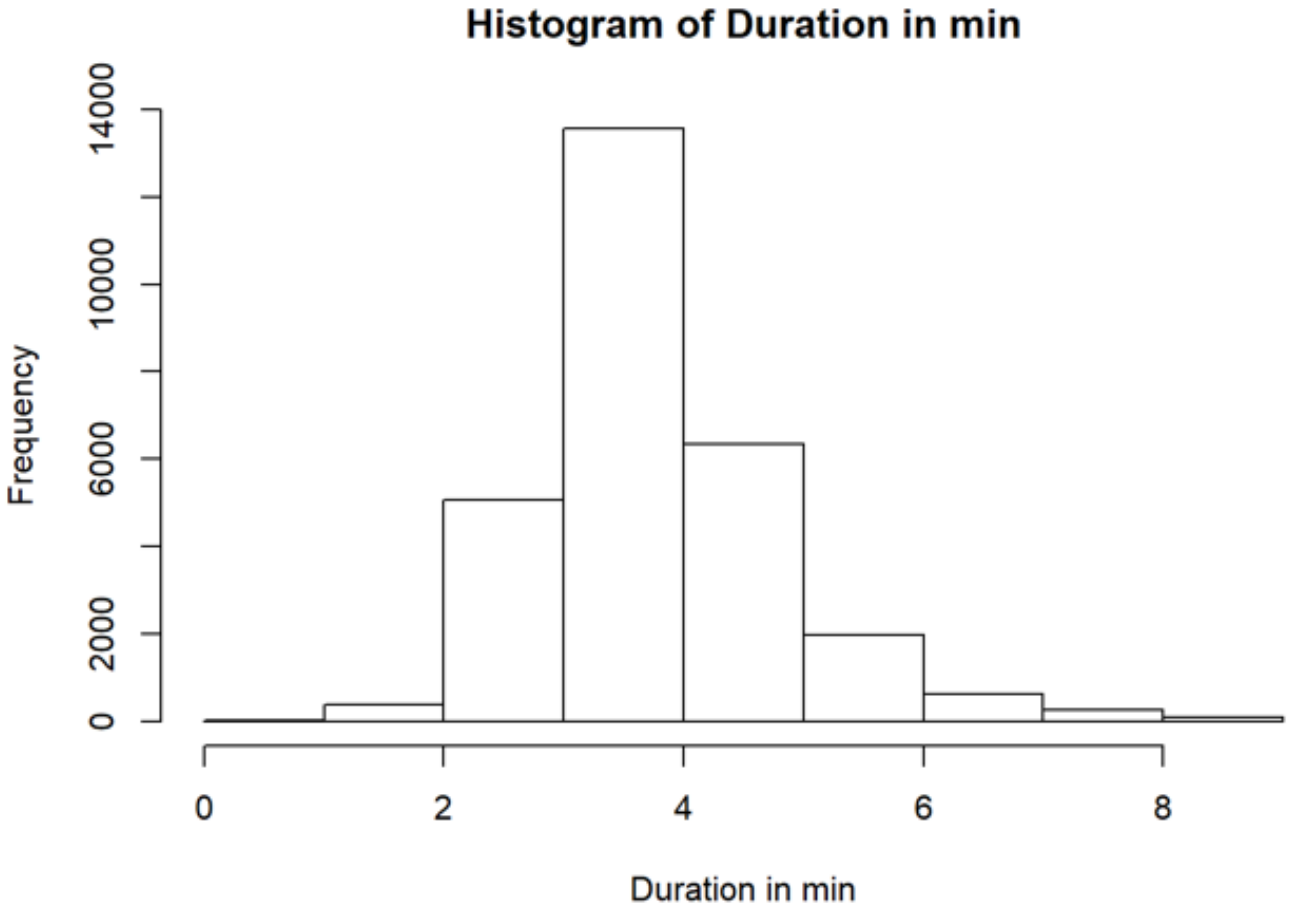
Interpreting genre based on popularity, we observe that pop, rap and latin are the genres having highest popularity respectively.

```
hist(spotify$instrumentalness, main = 'Histogram of Instrumentalness', xlab = '',
breaks = 10)
```



We see that majority (85.4323394%) observations have a value no larger than 0.1 in instrumentalness, and this is the reason why the difference between mean and median of instrumentalness is quite large.

```
hist(spotify$duration_min, main = 'Histogram of Duration in min', xlab = 'Duration
in min', breaks = 10)
```

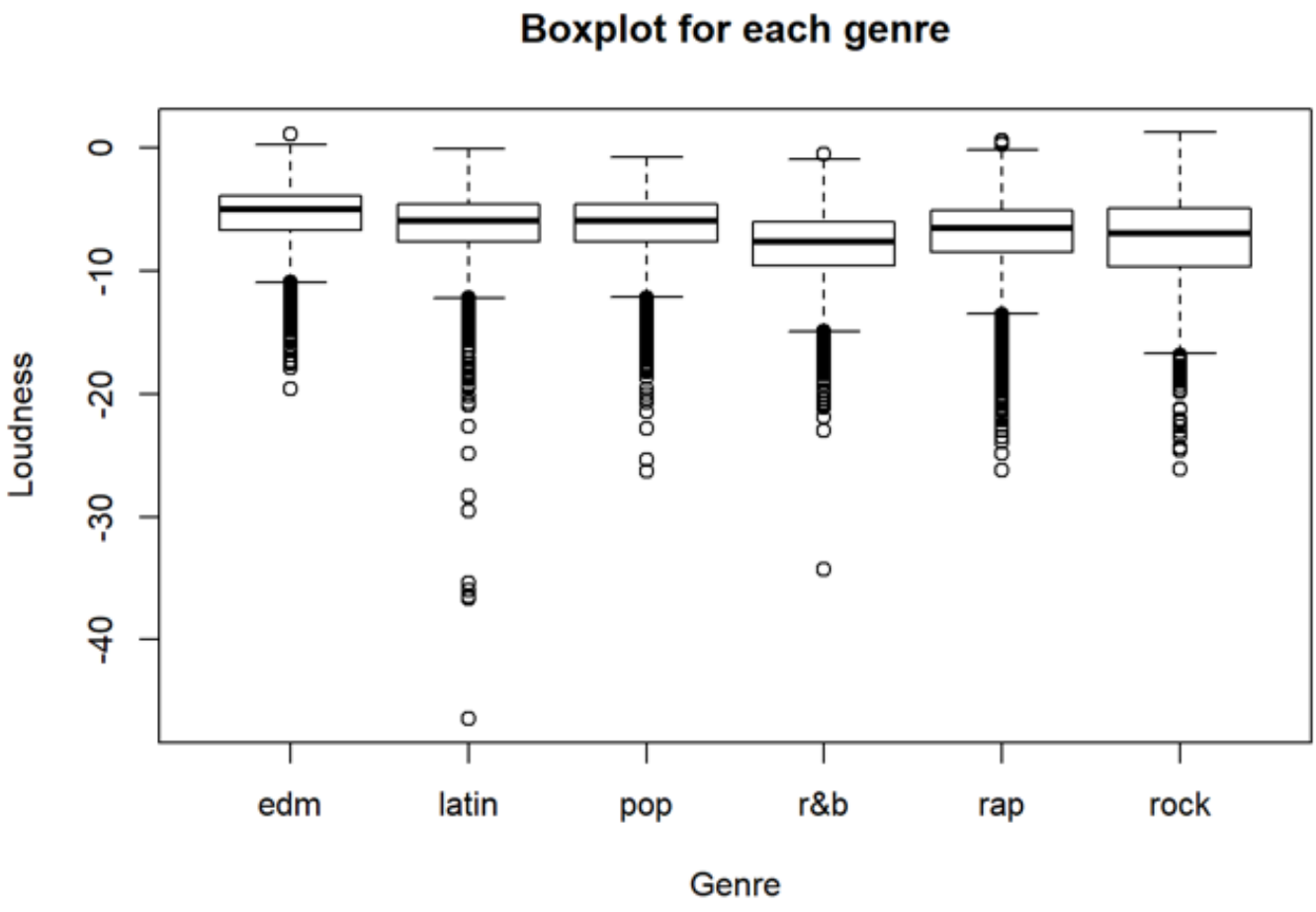


We see that majority of songs listened to have a duration of about 3-4 mins with songs longer than that duration having lower frequency of listeners.

3. **Boxplot**

Let's check the loudness distribution corresponding to each genre.

```
boxplot(spotify$loudness~spotify$playlist_genre, main = 'Boxplot for each genre',
xlab = 'Genre', ylab = 'Loudness')
```



loudness: Minimum of it is -46.448 while the first quarter is -8.171. Genre latin relatively has more outliers than other genres.

EDA to illustrate findings to our questions

We will use correlation plot to analyze how popularity is correlated with the various track features. From this graph, we can also check if there is presence of multicollinearity among the variables and determine whether the dataset is suitable for various classification algorithms.

We will use histogram for the track features to analyze the data distribution. This shall help us in determining trends in the dataset like skewness or spikes. With this information, we can determine which factor influences track popularity more than the others.

Before we use k-mean for clustering, we will visualize boxplot popularity with the different factor variables like key and genre to analyze if they have any significant relation to popularity. This shall assist the Spotify programming team in adding more popular tracks in the database. For instance, if we find that tracks of Pop genre and key A# are more popular compared to the others, adding such tracks to lead to increased customer satisfaction.

Scope of further learning

There are various powerful functions, tools, visualization plots and packages in R which can help us further deep dive in our analysis and help build models using ML techniques.

We still want to explore the use of gglot2 function. For the final project, we also intend to use R Shiny for better presentation.

Application of Machine Learning techniques

Yes, we plan on incorporating the following machine learning techniques to answer our questions:

1. We will use **linear regression** to determine which predictor variables are significant for popularity. Adding, different interaction terms between certain variables, like energy and loudness or acousticness and instrumentalness, we shall observe if the model performance improves.
2. We will study if there is a relation between popularity and different audio features and genres. We will do this through clustering analysis using **K-means clustering** method to provide song recommendation based on recent user listening on Spotify.
3. We will try to reduce dimensionality using **Principle Component Analysis (PCA)**. We will check if we can extract a pattern to do clustering from the available variables in the dataset and study how does each cluster differ to predict the most and least popular clusters.
4. We will also try to apply **Association Rules** to help Spotify cross-sell and upsell.