Spotify Music Analysis Project

SAHASRI GUNDAPUNEEDI
SE23UCAM019

Contents

1. Spotify Music Analysis Report

1.1. Data Collection and Loading

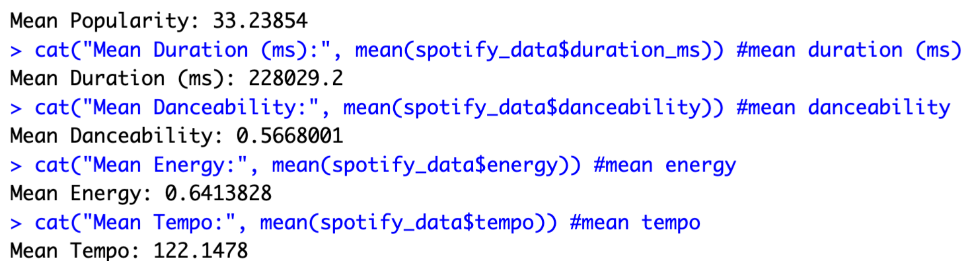
First, the following R libraries were installed and loaded :

```
library(readr)
library(dplyr)
library(tidytext)
library(stringr)
library(tm)
library(ggplot2)
library(wordcloud)
library(RColorBrewer)
```

The dataset used for this analysis was downloaded from Kaggle and saved as `Spotify-Dataset.csv`.

1.2. Data Preprocessing

- **Loading Data:** The dataset `Spotify-Dataset.csv` was loaded into R.
- **Data Cleaning:** Checked for duplicate rows with `any(duplicated(spotify_data))`. Missing values were evaluated by examining the popularity column and its computed mean. If there were any missing values, running `mean(spotify_data$popularity)` would return NA.
- **Descriptive Statistics:** Computed the mean values for key variables : Popularity, duration in milliseconds, danceability, energy, and tempo, which all returned a positive number meaning no values were missed.



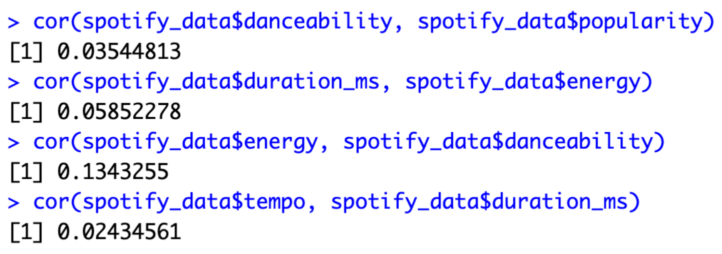
```
Mean Popularity: 33.23854
> cat("Mean Duration (ms):", mean(spotify_data$duration_ms)) #mean duration (ms)
Mean Duration (ms): 228029.2
> cat("Mean Danceability:", mean(spotify_data$danceability)) #mean danceability
Mean Danceability: 0.5668001
> cat("Mean Energy:", mean(spotify_data$energy)) #mean energy
Mean Energy: 0.6413828
> cat("Mean Tempo:", mean(spotify_data$tempo)) #mean tempo
Mean Tempo: 122.1478
```

Also, `cor(spotify_data$numerical_feature_1, spotify_data$numerical_feature_2)` will give us the correlation coefficient between the two specified numerical variables. It ranges from −1 to 1:

+1: Perfect positive correlation. As one variable increases, the other also increases proportionally.

−1: Perfect negative correlation. As one variable increases, the other decreases proportionally.

0: No linear correlation. Changes in one variable do not predict changes in the other.

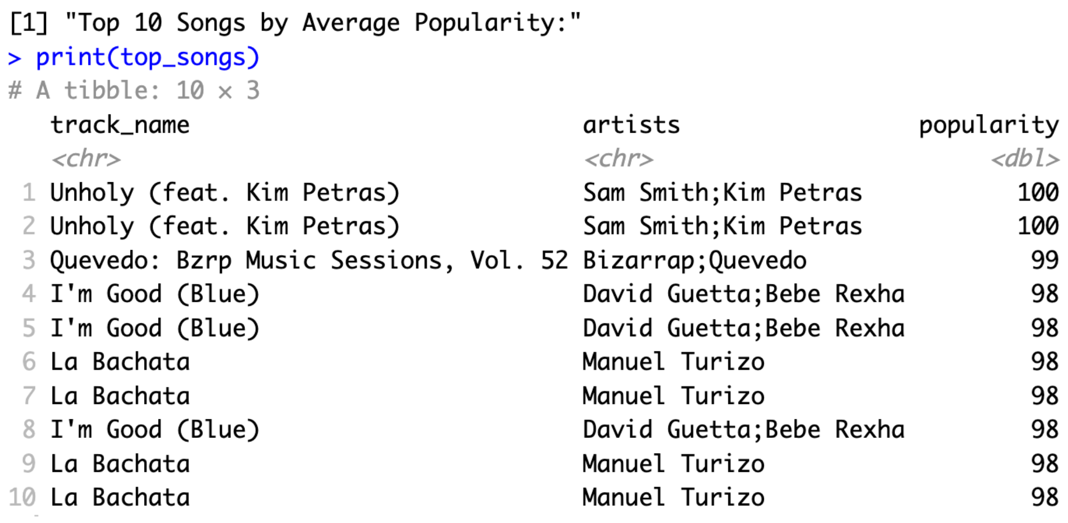


```
> cor(spotify_data$danceability, spotify_data$popularity)
[1] 0.03544813
> cor(spotify_data$duration_ms, spotify_data$energy)
[1] 0.05852278
> cor(spotify_data$energy, spotify_data$danceability)
[1] 0.1343255
> cor(spotify_data$tempo, spotify_data$duration_ms)
[1] 0.02434561
```

1.3. Data Analysis

1.3.1. Top 10 Popular Songs

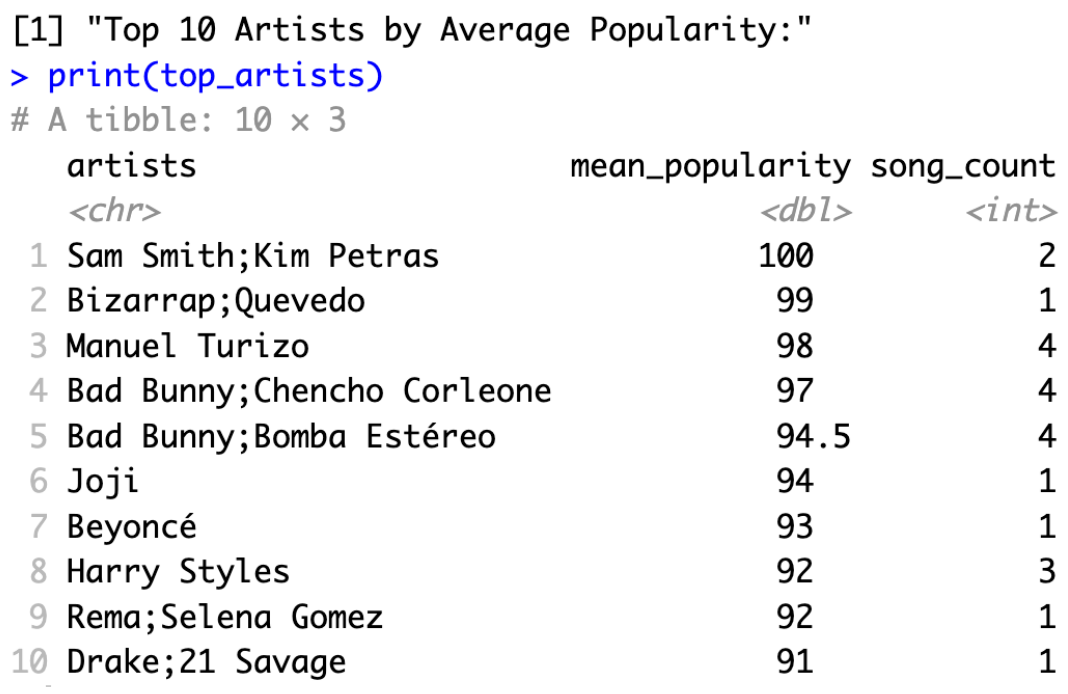
The entire dataset is arranged in descending order by the popularity column. The most popular songs will come first. We select only the relevant columns: `track_name`, `artists`, and `popularity` as they are the only the necessary information for analysis.



```
[1] "Top 10 Songs by Average Popularity:"
> print(top_songs)
# A tibble: 10 × 3
   track_name                             artists                popularity
   <chr>                                  <chr>                       <dbl>
 1 Unholy (feat. Kim Petras)              Sam Smith;Kim Petras          100
 2 Unholy (feat. Kim Petras)              Sam Smith;Kim Petras          100
 3 Quevedo: Bzrp Music Sessions, Vol. 52 Bizarrap;Quevedo               99
 4 I'm Good (Blue)                        David Guetta;Bebe Rexha        98
 5 I'm Good (Blue)                        David Guetta;Bebe Rexha        98
 6 La Bachata                             Manuel Turizo                  98
 7 La Bachata                             Manuel Turizo                  98
 8 I'm Good (Blue)                        David Guetta;Bebe Rexha        98
 9 La Bachata                             Manuel Turizo                  98
10 La Bachata                             Manuel Turizo                  98
```

1.3.2. Top 10 Artists

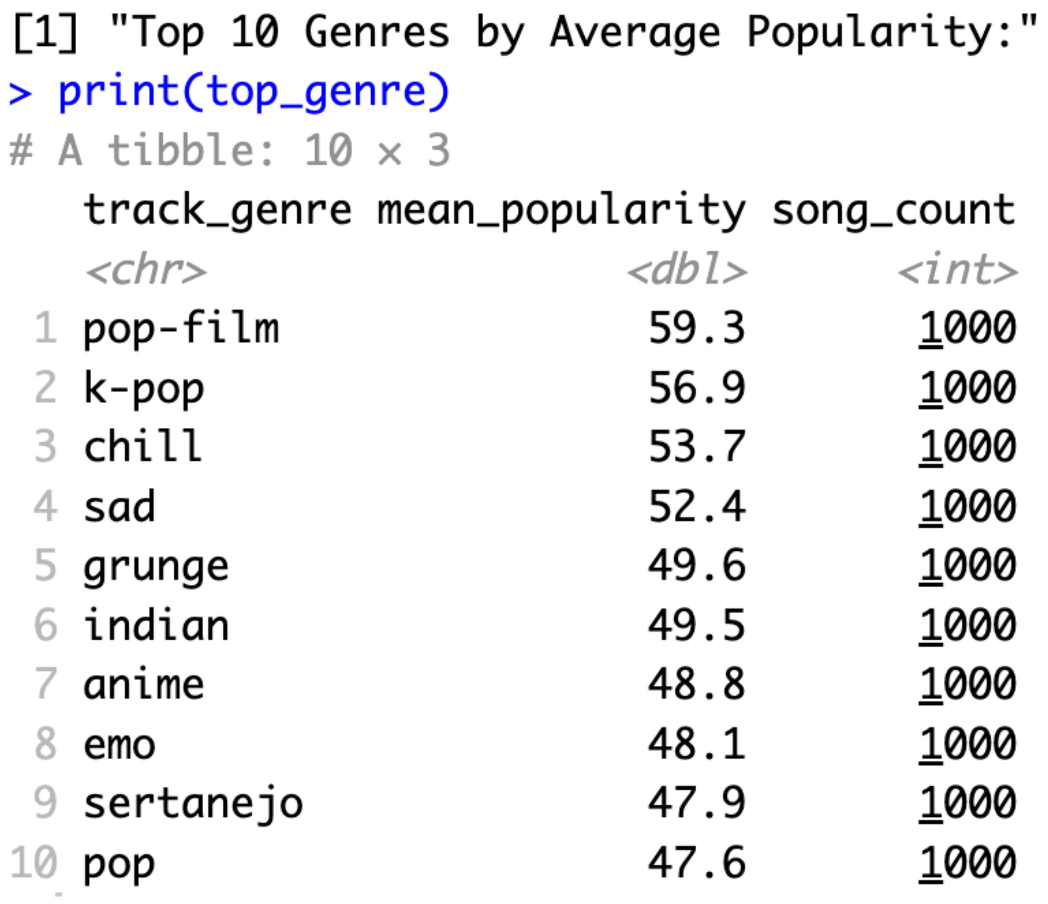
Groups the data by `artists`. We compute the average popularity score for all songs by each artist and sort the summarized data by average popularity in descending order, so the most popular artists are at the top.



```
[1] "Top 10 Artists by Average Popularity:"
> print(top_artists)
# A tibble: 10 × 3
   artists                     mean_popularity song_count
   <chr>                                 <dbl>      <int>
 1 Sam Smith;Kim Petras                  100            2
 2 Bizarrap;Quevedo                       99            1
 3 Manuel Turizo                          98            4
 4 Bad Bunny;Chencho Corleone             97            4
 5 Bad Bunny;Bomba Estéreo                94.5          4
 6 Joji                                   94            1
 7 Beyoncé                                93            1
 8 Harry Styles                           92            3
 9 Rema;Selena Gomez                      92            1
10 Drake;21 Savage                        91            1
```

1.3.3. Top 10 Genres

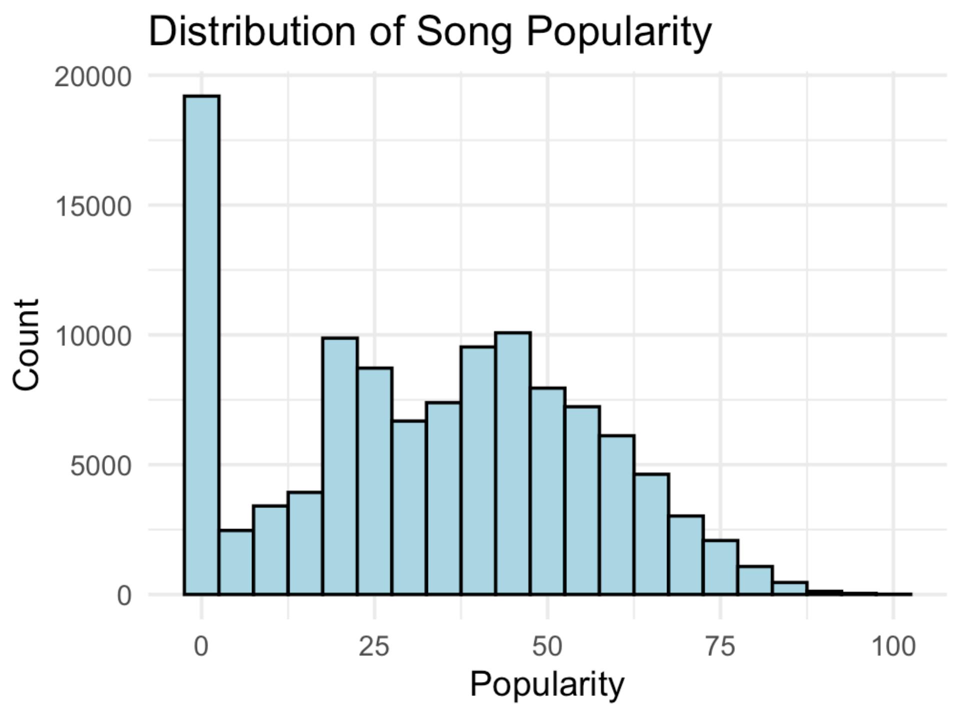
Same as we did for Top 10 Songs, we group by track_genre and compute the average popularity of songs within each genre, arranged them in descending order and select the first i.e top 10 genres.



```
[1] "Top 10 Genres by Average Popularity:"
> print(top_genre)
# A tibble: 10 × 3
   track_genre mean_popularity song_count
   <chr>                 <dbl>      <int>
 1 pop-film               59.3       1000
 2 k-pop                  56.9       1000
 3 chill                  53.7       1000
 4 sad                    52.4       1000
 5 grunge                 49.6       1000
 6 indian                 49.5       1000
 7 anime                  48.8       1000
 8 emo                    48.1       1000
 9 sertanejo              47.9       1000
10 pop                    47.6       1000
```

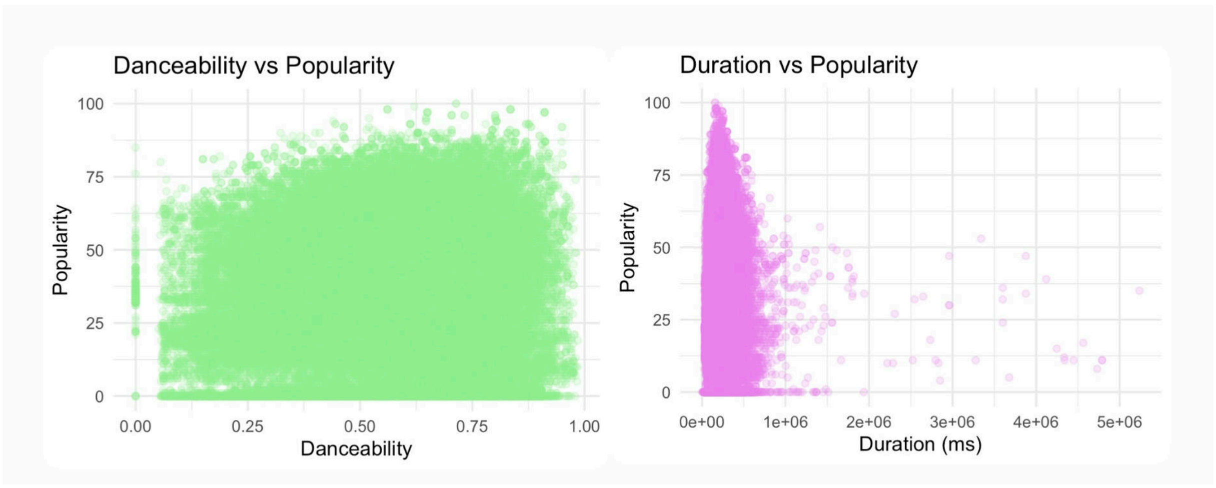
1.3.4. Song Popularity Bar - Plot

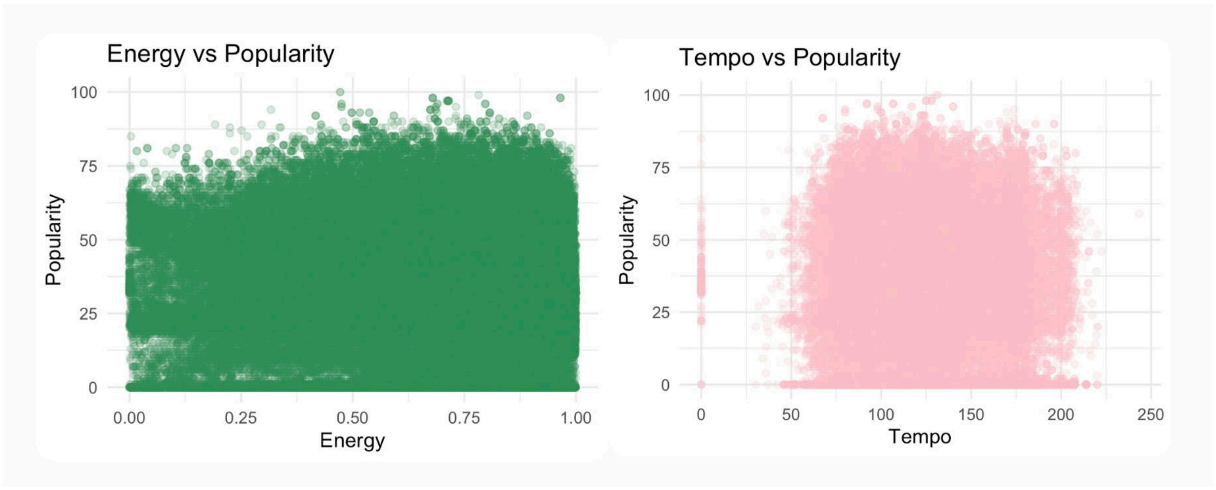
```
ggplot(spotify_data, aes(x = popularity)) +
geom_histogram(binwidth = 5, fill = "lightblue", color = "black") +
labs(title = "Distribution of Song Popularity", x = "Popularity", y = "Count") +
theme_minimal()
```

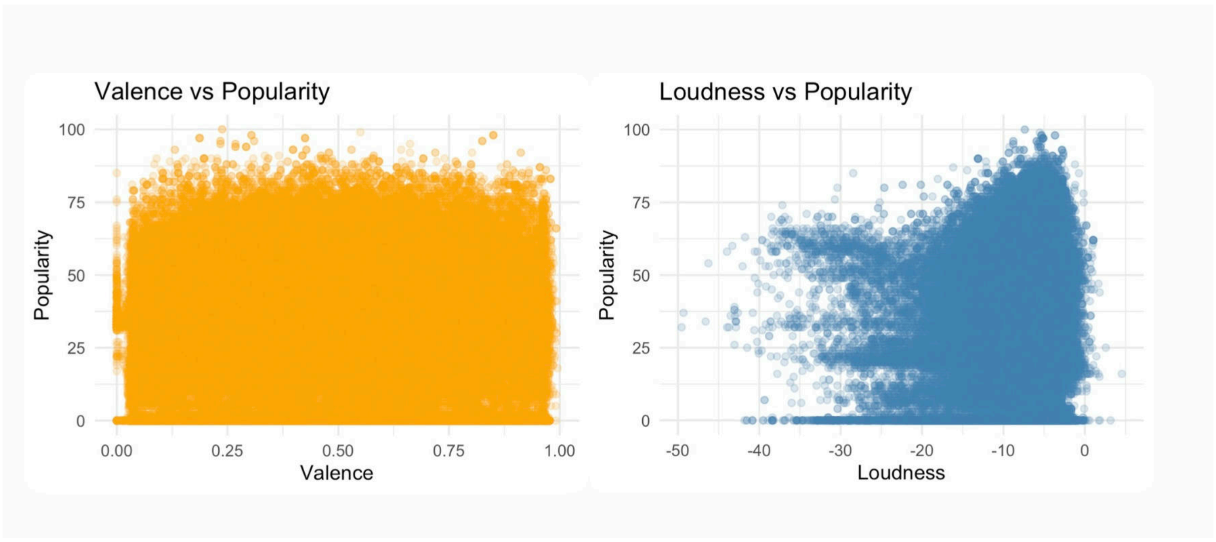


Let us analyze the relationship between song features and popularity using :

```
ggplot(spotify_data, aes(x = FEATURE, y = FEATURE)) +
geom_point(color = "lightgreen", alpha = 0.2) +
labs(title = "FEATURE vs Popularity", x = "FEATURE", y = "Popularity") +
theme_minimal()
```



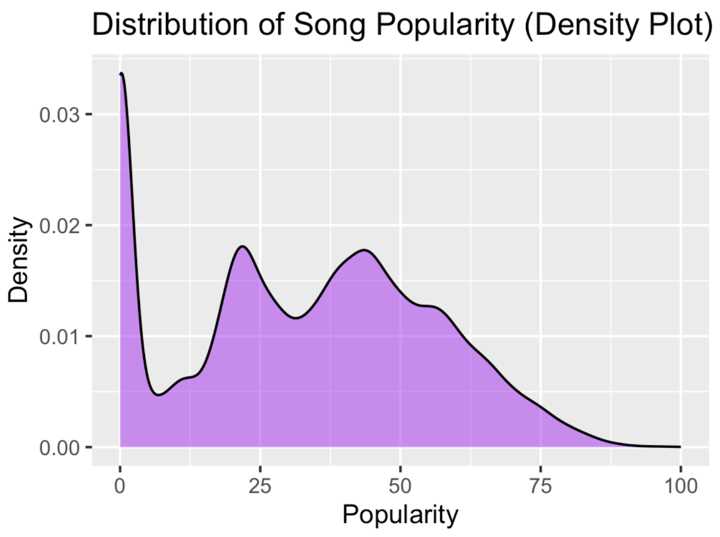




Similarly we can find the feature vs popularity scatter plot for other numerical features too (acousticness, speechiness,instrumentalness,time_signature, etc…)

1.4. Visualization

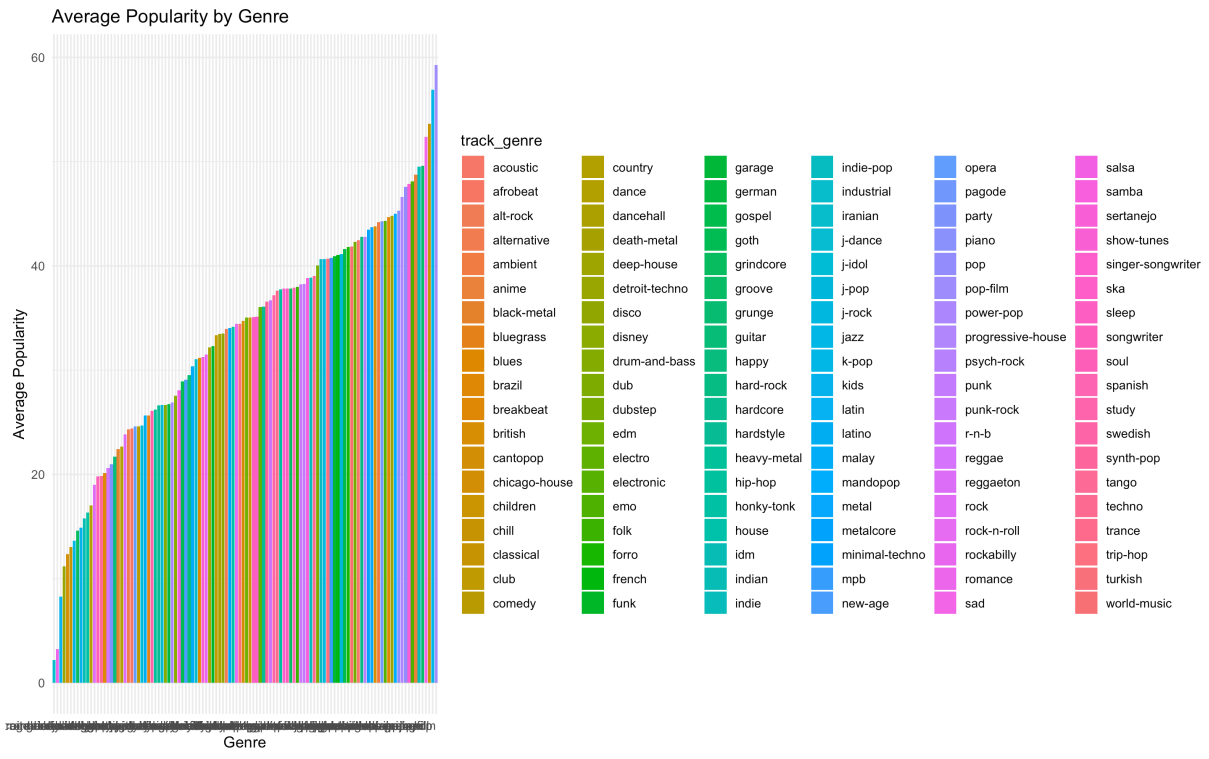
1.4.1. Density Plot of Distribution of Song Popularity (pdf)



1. Most songs have popularity scores clustered between 20 and 60.
2. There is a small number of songs with very high popularity scores near 90 or 100.
3. A sharp drop-off in the density curve might suggest that very few songs have very low popularity scores.

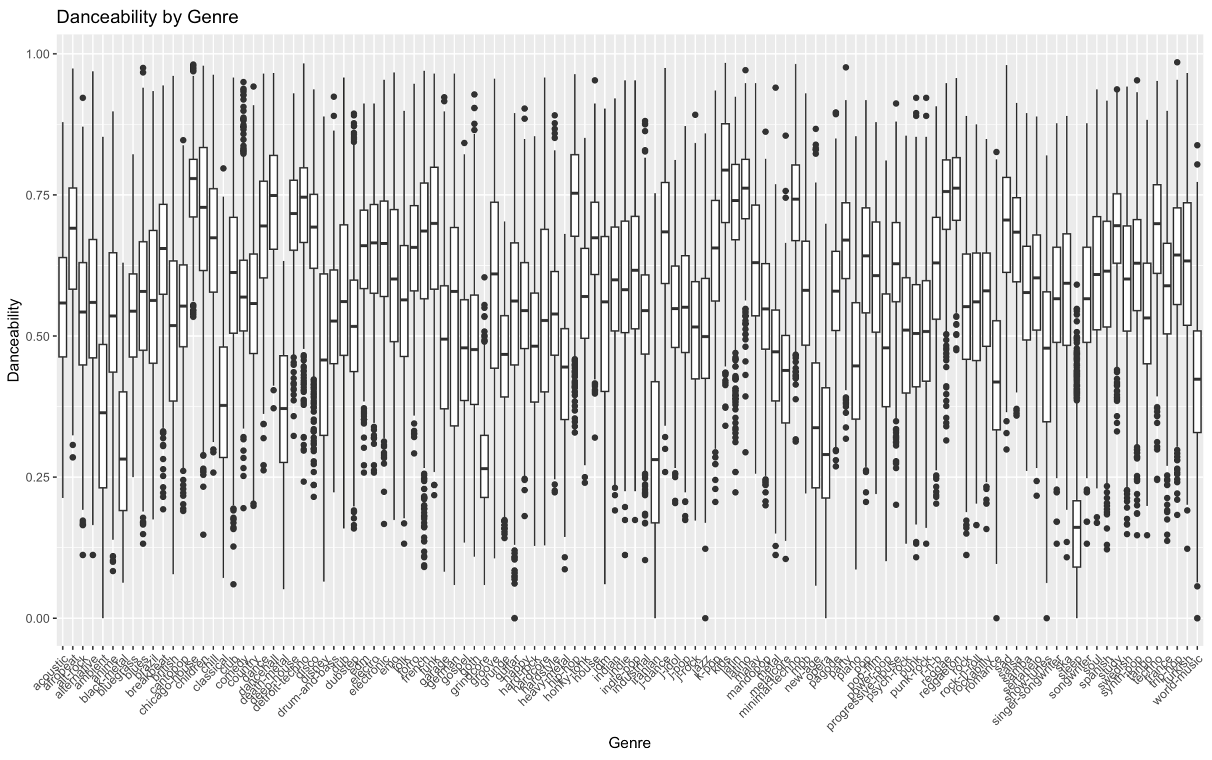
1.4.2. Genre popularity

`genre_avg_popularity` contains precomputed average popularity scores grouped by `track_genre` so we reorder the genres along the x-axis based on their respective average popularity values.

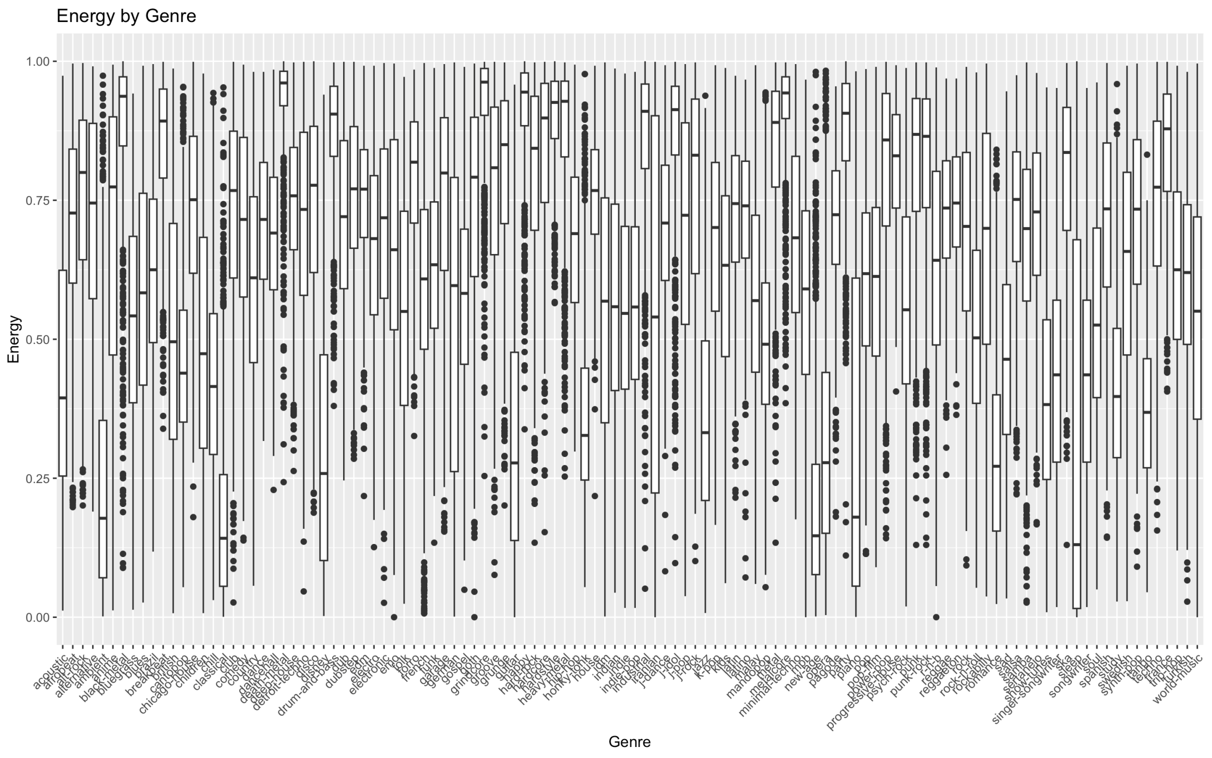


The below boxplots show variability in the feature's scores for each genre and whether certain genres will have consistently higher or lower danceability scores compared to others.

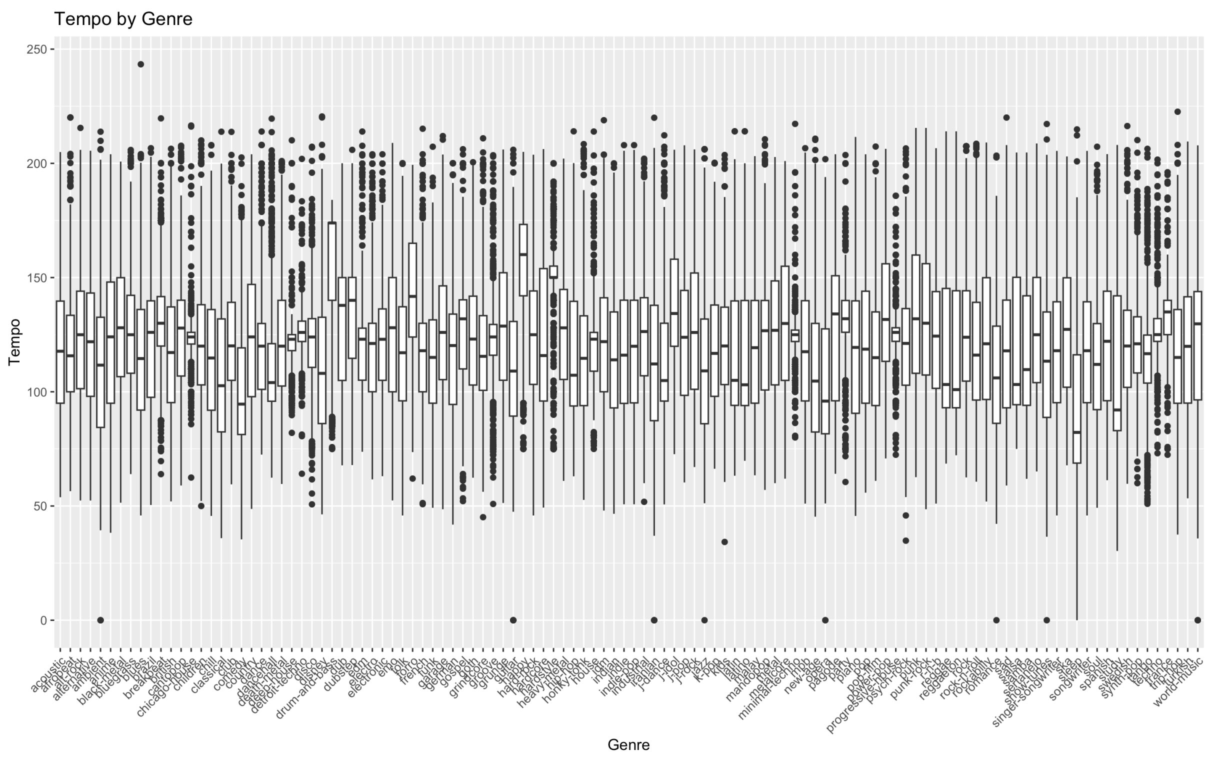
1.4.3. Danceability by Genre Boxplot



1.4.4. Energy by Genre Boxplot



1.4.5. Tempo by Genre Boxplot



1.5. Recommendation System

We shall implement a recommendation system based on both song popularity and genre.

1.5.1. Genre Method

If the selected method is `genre` the system :

- Filters for all songs that share the same genre as the input song.
- Excludes the input song itself from the recommendations.
- Selects the top 10 most popular songs within that genre.
- This logic assumes that listeners are likely to enjoy other songs that share a genre with their favorite.

```
if (method == "genre") {
  # recommend songs with the same genre
  recommendations <- spotify_data %>%
    filter(track_genre == song_info$track_genre & track_name != song_name) %>%
    select(track_name, track_genre, popularity) %>%
    arrange(desc(popularity)) %>%
    head(10)
```

1.5.2. Popularity Method

If the selected method is `popularity`, then the system :

- Searches for songs whose popularity is similar to the input song's popularity.
- Excludes the input song itself from the recommendations.
- Selects the top 10 most popular songs from the filtered results.

```
 } else if (method == "popularity") {
    #recommend songs with similar popularity
    recommendations <- spotify_data %>%
      filter(
        abs(popularity - song_info$popularity) < 5 & track_name != song_name
      ) %>%
      select(track_name, track_genre, popularity) %>%
      arrange(desc(popularity)) %>%
      head(10)
  }
```

After processing either method, the function returns a dataframe recommendations (`return(recommendations)`) containing :

`track_name` : The name of the recommended song.

`track_genre` : The genre of the recommended song.

`popularity` : The popularity score of the recommended song.

For Example :

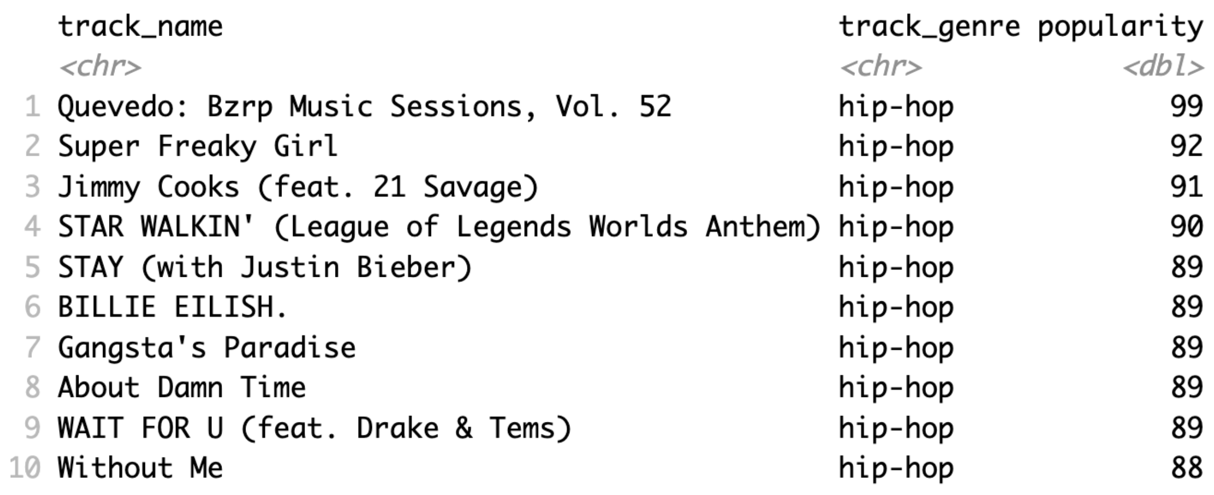
We input a song “N95” by Kendrick Lamar, the system will recommend 10 other songs that are in the same genre as “N95”.

```
recommendations_genre <- recommend_songs("N95", method = "genre")
print("Recommended Songs by Genre:")
print(recommendations_genre)
```

Or else, if we want similar popular songs to be recommended, then we use `method = popularity` When we input “White Ferrari” by Frank Ocean, it returns us songs that have a popularity score close to “White Ferrari” but are not the same song.

```
recommendations_popularity <- recommend_songs("White Ferrari", method =
"popularity")
print("Recommended Songs by Similar Popularity:")
print(recommendations_popularity)
```

Recommended Songs by Similar Genre are :



```
   track_name                                     track_genre popularity
   <chr>                                          <chr>            <dbl>
 1 Quevedo: Bzrp Music Sessions, Vol. 52          hip-hop             99
 2 Super Freaky Girl                              hip-hop             92
 3 Jimmy Cooks (feat. 21 Savage)                  hip-hop             91
 4 STAR WALKIN' (League of Legends Worlds Anthem) hip-hop             90
 5 STAY (with Justin Bieber)                      hip-hop             89
 6 BILLIE EILISH.                                 hip-hop             89
 7 Gangsta's Paradise                             hip-hop             89
 8 About Damn Time                                hip-hop             89
 9 WAIT FOR U (feat. Drake & Tems)                hip-hop             89
10 Without Me                                     hip-hop             88
```

Recommended Songs by Similar Genre are :

```
   track_name                track_genre popularity
   <chr>                     <chr>            <dbl>
 1 Little Dark Age           alt-rock            83
 2 You Get Me So High        alt-rock            83
 3 Smells Like Teen Spirit   alt-rock            83
 4 Shut Up and Dance         alt-rock            83
 5 Numb                      alternative         83
 6 Little Dark Age           alternative         83
 7 Shut Up and Dance         alternative         83
 8 Smells Like Teen Spirit   alternative         83
 9 You Get Me So High        alternative         83
10 Chop Suey!                alternative         83
```