Data Visualizaton Project 1 Report

2024-04-03

Introduction

The selected dataset comprises comprehensive data on Taylor Swift's discography, including three subsets: `taylor_album_songs`, `taylor_all_songs`, and `taylor_albums`. These subsets provide a detailed look at Taylor Swift's studio albums, EPs, singles, and re-releases, covering audio features, album release dates, Metacritic scores, and user scores. This dataset originates from the Genius lyrics database and Spotify's API, offering a rich field for exploring the relationship between musical elements and album reception. Dataset link here.

Data Dictionary

`taylor_album_songs.csv`

variable	class	description
album_name	character	Album name
ep	logical	Is it an EP
album_release	double	Album release date
track_number	integer	Track number
track_name	character	Track name
artist	character	Artists
featuring	character	Artists featured
bonus_track	logical	Is it a bonus track
promotional_release	double	Date of promotional release
single_release	double	Date of single release
track_release	double	Date of track release
danceability	double	Spotify danceability score. A value of 0.0 is least danceable and 1.0 is most danceable.
energy	double	Spotify energy score. Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.
key	integer	The key the track is in.
loudness	double	Spotify loudness score. The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track.
mode	integer	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.

variable	class	description
speechiness	double	Spotify speechiness score. 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.
acousticness	double	Spotify acousticness score. A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
instrumentalness	double	Spotify instrumentalness score. Predicts 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	double	Spotify liveness score. 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	double	Spotify valence score. A 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).
tempo	double	The 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.
time_signature	integer	An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4".
duration_ms	integer	The duration of the track in milliseconds.
explicit	logical	Does the track have explicit lyrics.
key_name	character	The key the track is in. 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.
mode_name	character	Modality of the track.

variable	class	description
key_mode	character	The key of the track.
lyrics	list	Track lyrics. These values are all NA. To get the lyrics in nested tibbles, `install.packages("taylor")` and use the source data.

`taylor_all_songs.csv`

variable	class	description
album_name	character	Album name
ep	logical	Is it an EP
album_release	double	Album release date
track_number	integer	Track number
track_name	character	Track name
artist	character	Artists
featuring	character	Artists featured
bonus_track	logical	Is it a bonus track
promotional_release	double	Date of promotional release
single_release	double	Date of single release
track_release	double	Date of track release
danceability	double	Spotify danceability score. A value of 0.0 is least danceable and 1.0 is most danceable.
energy	double	Spotify energy score. Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.
key	integer	The key the track is in.
loudness	double	Spotify loudness score. The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track.
mode	integer	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	double	Spotify speechiness score. 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.
acousticness	double	Spotify acousticness score. A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.

variable	class	description
instrumentalness	double	Spotify instrumentalness score. Predicts 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	double	Spotify liveness score. 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	double	Spotify valence score. A 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).
tempo	double	The 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.
time_signature	integer	An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4".
duration_ms	integer	The duration of the track in milliseconds.
explicit	logical	Does the track have explicit lyrics.
key_name	character	The key the track is in. 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.
mode_name	character	Modality of the track.
key_mode	character	The key of the track.
lyrics	list	Track lyrics. These values are all NA. To get the lyrics in nested tibbles, `install.packages("taylor")` and use the source data.

`taylor_albums.csv`

variable	class	description
album_name	character	Album name
ep	logical	Is it an EP
album_release	double	Album release date
metacritic_score	integer	Metacritic score
user_score	double	User score

Reasons to choose the dataset

Taylor Swift has revolutionized the music industry with her timeless and relatable songwriting, winning numerous prestigious prizes and becoming one of our time's most influential and beloved artists. In light of this, we hope that analyzing a dataset about Taylor Swift's songs can provide valuable insights into the strategies behind her success. The dataset offers comprehensive features related to her albums and songs, so we expect to unveil interesting undiscovered patterns that contribute to her widespread appeal. This knowledge can provide valuable lessons in music production, marketing, and cultural impact, which benefits aspiring artists and music industry professionals.

Importing neccesary packages

```
suppressMessages({
  library(tidyverse)
  library(ggcorrplot)
  library(dplyr)
  library(corrplot)
  library(ggplot2)
  library(gridExtra)
  library(cluster)
  library(RColorBrewer)
})
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3

## Warning: package 'ggcorrplot' was built under R version 4.3.3

## Warning: package 'corrplot' was built under R version 4.3.3

## Warning: package 'gridExtra' was built under R version 4.3.3
```

Question 1: How do features such as `danceability`, `energy`, `valence`, and `tempo` correlate with Taylor Swift's albums' critical and public reception?

Introduction

In light of Taylor Swift's success, it can be beneficial to analyze how the characteristics of her songs influence their success. By analyzing features such as lyrics' explicitness, tempo and key, we hope to uncover patterns in Taylor Swift's songs that contribute to her music's popularity.

To simplify unnecessarily complexity, we propose excluding these columns. We believe they are not particularly interesting or relevant to how people score the albums. Our focus is on studying the statistics of Taylor Swift's songs based on audio features and how these features influence album scoring by listeners.

Column Name	Data Type	Description
ep	logical	Is it an EP?
album_release	double	Album release date
artist	character	Artists
featuring	character	Artists featured
bonus_track	logical	Is it a bonus track?
promotional_release	double	Date of promotional release
single_release	double	Date of single release
track_release	double	Date of track release

The overall dataset is as follows:

- Three columns represent the index
- 11 numerical features
- 6 categorical features
- 2 target numerical variables

Approach

The features incorporated in our analysis are split into two types: numerical and categorical features.

For the former, we utilize a correlation matrix. This is a very common way to reveal relationships between numerical or binary variables, which is exactly our goal.

For the latter, faceted box plots are used instead as using a correlation matrix is not recommended for non-binary categorical variables. Here, we also use bar charts to visualize class distribution, since class imbalance can impact our observations.

Analysis

Data preprocessing

Firstly, we need to read the two csv files. The first file contain the information about audio features (numerical and categorical) for each song whereas the second file contain the information about the album name, release date and scores:

```
album_song_df = read.csv("taylor_album_songs.csv")
album_df = read.csv("taylor_albums.csv")
```

Dropping unnecessary columns:

```
album_df_subset <- album_df[c("album_name", "metacritic_score", "user_score")]
album_song_df_subset <- album_song_df[c("album_name", "track_number",
                                        "track_name", "danceability",
                                        "energy", "key", "loudness",
```

```
                                        "mode", "speechiness",
                                        "acousticness", "instrumentalness",
                                        "liveness", "valence", "tempo",
                                        "time_signature", "duration_ms",
                                        "explicit", "key_name",
                                        "mode_name", "key_mode")]
```

We further remove rows with NaN values since we believe it is best to have a clean dataset. We don't think there is a reasonable way to interpolate these missing values, especially since they are subjective (for example, the user scores and Metacritic scores).

```
album_df <- na.omit(album_df)
```

Assuming that metacritic score and user score are calculated as their definition, which are averaged over the songs and acts as a mean, we can use it to represent the score for each song in the same album. We remove all NaN rows since there is no reasonable way to interpolate these scores based on the other albums. And the number of NaN rows is insignificant.

```
album_song_with_scores_df <- merge(album_song_df_subset, album_df_subset,
                                   by = "album_name", all.x = TRUE)

album_song_with_scores_df <- na.omit(album_song_with_scores_df)
```

The constructed dataframe is now as follows: - Three columns represent the index - 11 numerical features - 6 categorical features - 2 target numerical variables

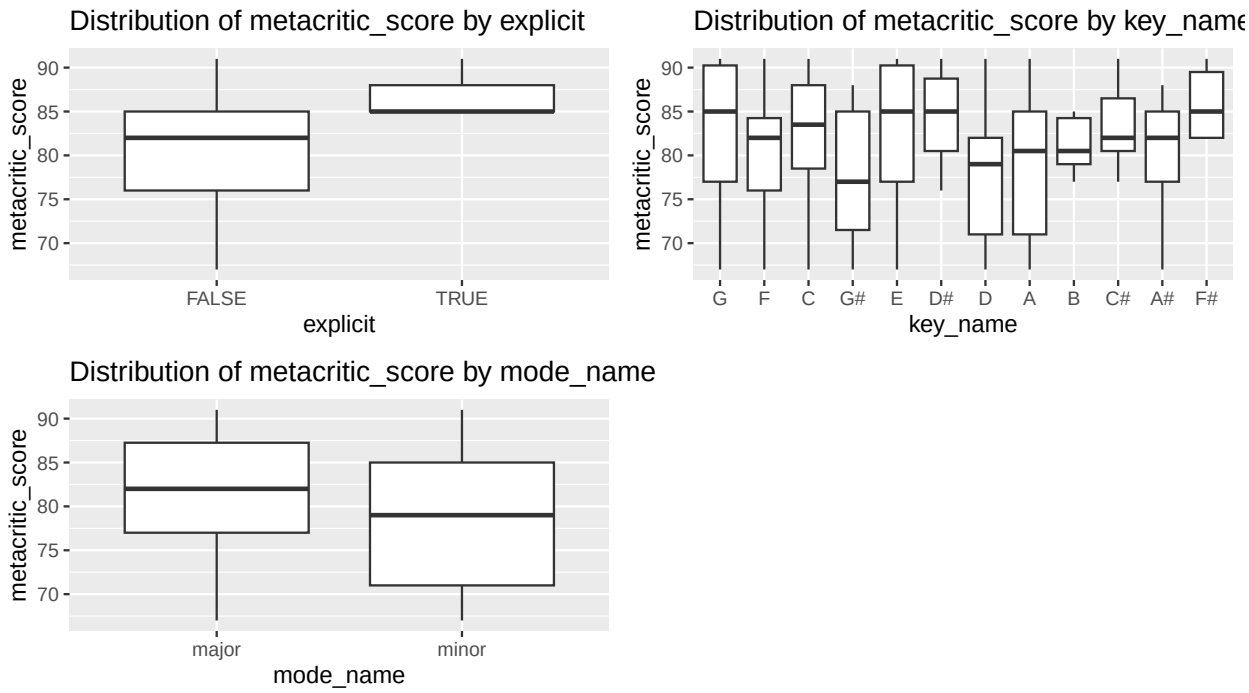
```
# 3 first index columns

# 11 numerical columns
numerical_feats = c("danceability", "energy", "loudness", "speechiness",
                    "acousticness", "instrumentalness", "liveness",
                    "valence", "tempo", "time_signature", "duration_ms")

# 6 categorical columns
categorical_feats = c("explicit", "key", "key_name", "mode",
                      "mode_name", "key_mode")

# 2 target numerical columns: metacritic_score VS user_score
```

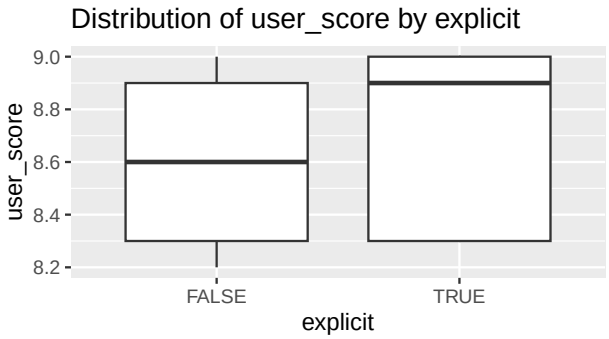
Box plots for categorical variables

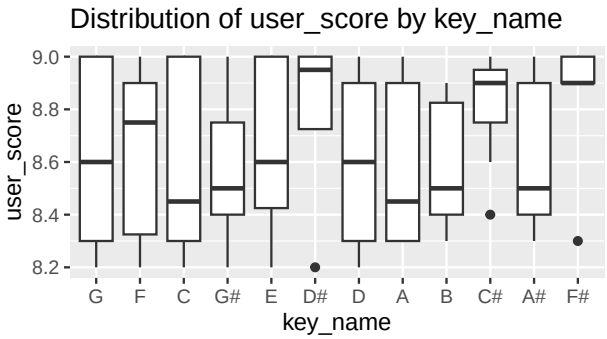
```
# Define the list of categorical columns
# categorical_cols <- c("key", "mode", "explicit", "key_name", "mode_name")
categorical_cols <- c("explicit", "key_name", "mode_name") # remove "key" and "mode" because they're ju

# Create facet grid plots for each score
for (score in c("metacritic_score", "user_score")) {
  # Initialize a list to store individual plots
  plots <- list()
```

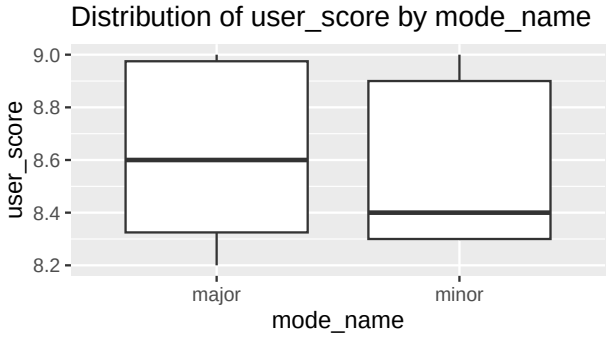
```r
  # Loop through each categorical column
  for (col in categorical_cols) {
    # Convert the column to factor with explicit levels
    album_song_with_scores_df[[col]] <-
      factor(album_song_with_scores_df[[col]],
             levels = unique(album_song_with_scores_df[[col]]))

    # Create a facet grid plot for the current categorical column
    p <- ggplot(album_song_with_scores_df, aes_string(x = col, y = score)) +
      geom_boxplot() +
      labs(x = col, y = score) +
      ggtitle(paste("Distribution of", score, "by", col))

    # Add the plot to the list
    plots[[col]] <- p
  }

  # Combine plots into a single facet grid
  grid.arrange(grobs = plots, nrow = 2, ncol = 2)
  # Adjust nrow and ncol as needed
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



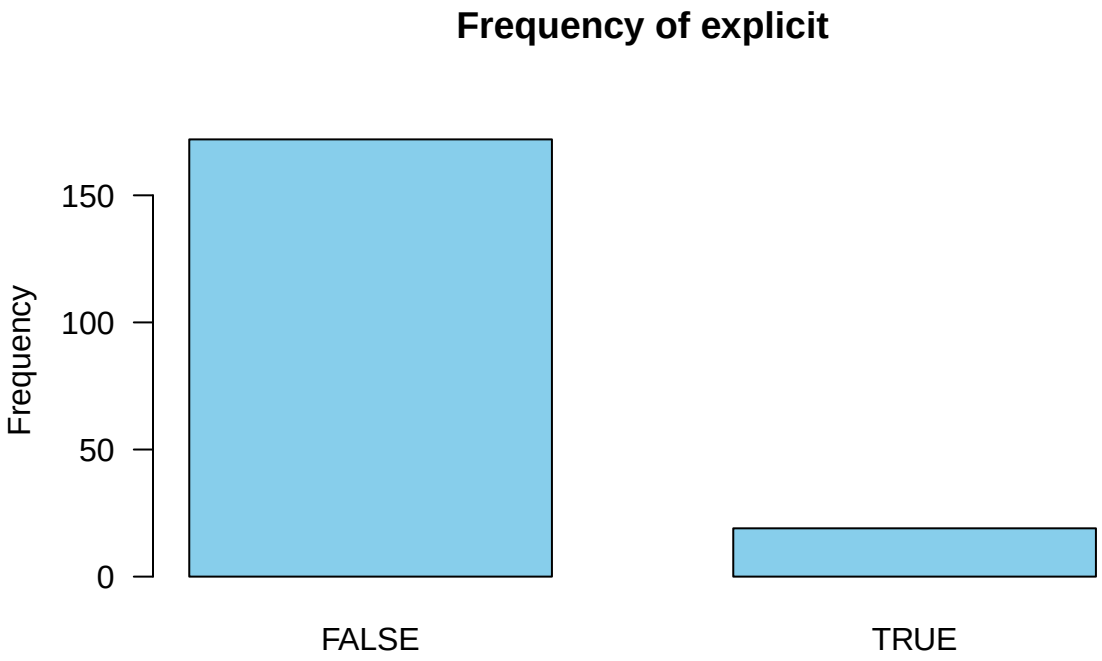


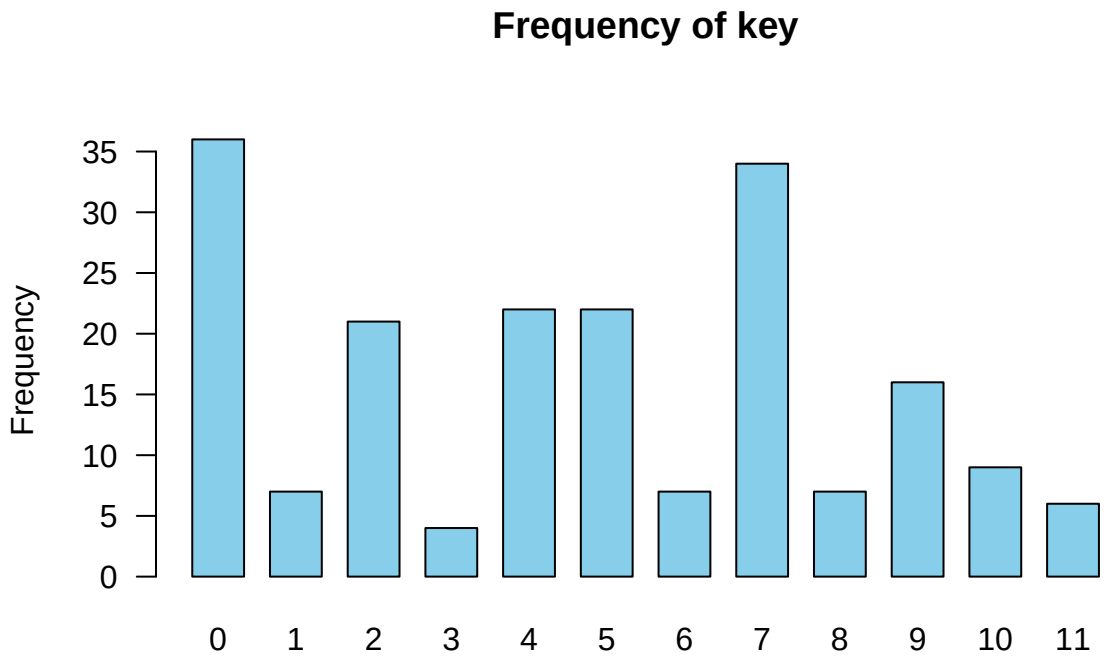


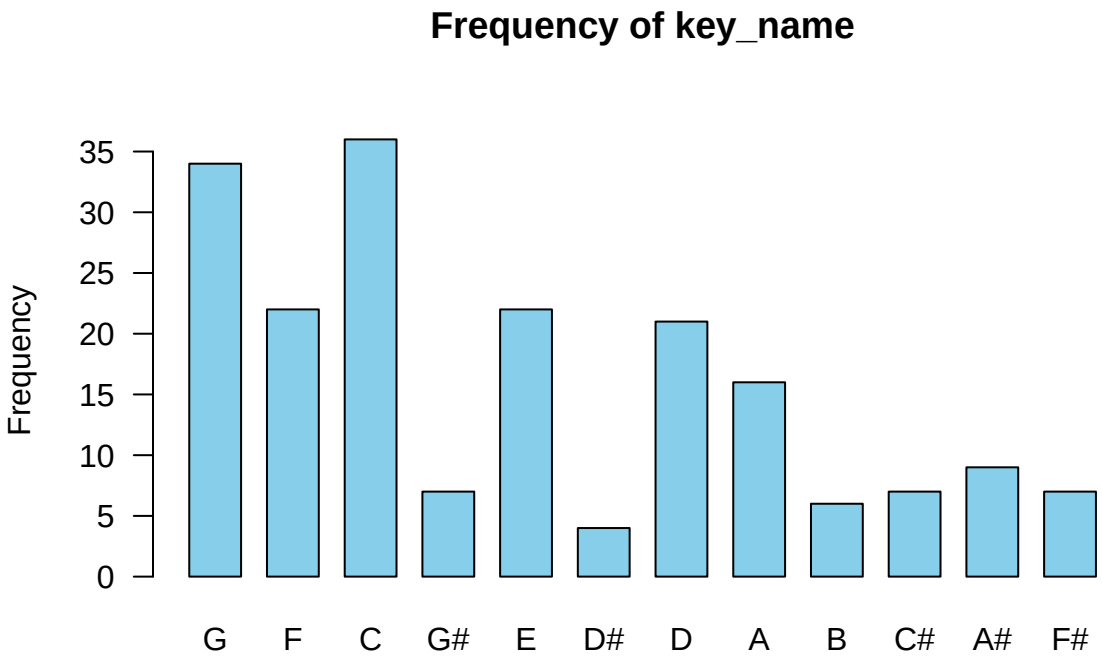


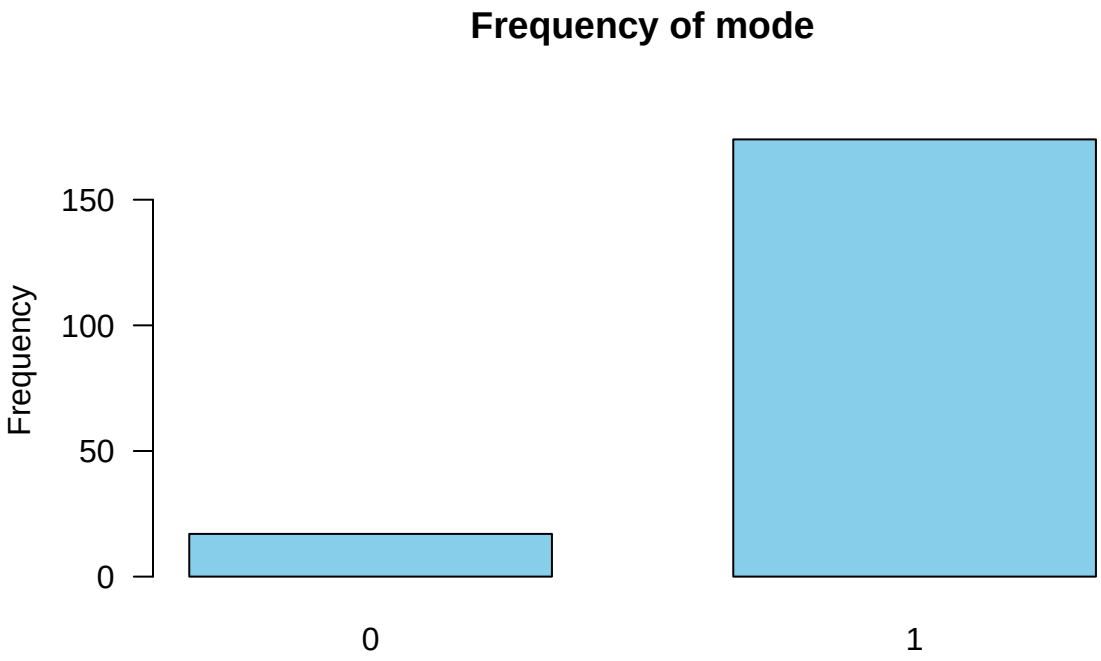
Bar plots for categorical variable distribution

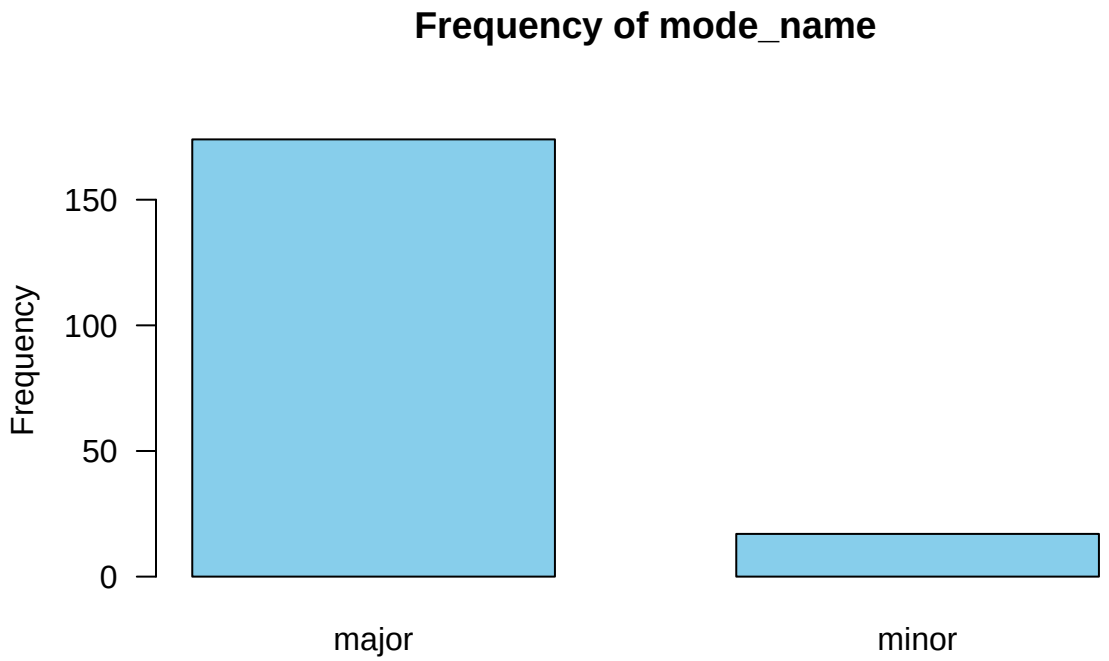
```
categorical_feats_df <- album_song_with_scores_df[categorical_feats]

categorical_feats_df <- na.omit(categorical_feats_df)


# Convert categorical variables to factors
categorical_feats_df <- lapply(categorical_feats_df, factor)

# Get frequency tables for each categorical variable
frequency_tables <- lapply(categorical_feats_df, table)

# Plot bar charts for each categorical variable
for (i in seq_along(frequency_tables)) {
  # Set up plotting area
  par(mar = c(5, 5, 6, 2))  # Adjust margins for x-axis label

  # Plot bar chart with wider spacing between categories and
  # horizontal x-axis labels
  barplot(frequency_tables[[i]],
          main = paste("Frequency of", names(frequency_tables)[i]),
          ylab = "Frequency",
          col = "skyblue",
          border = "black",
          space = 0.5,  # Adjust the spacing between bars
          las = ifelse(names(frequency_tables)[i] == "key_mode", 2, 1))
  # Rotate x-axis labels vertically if it's "key_mode"
}
```

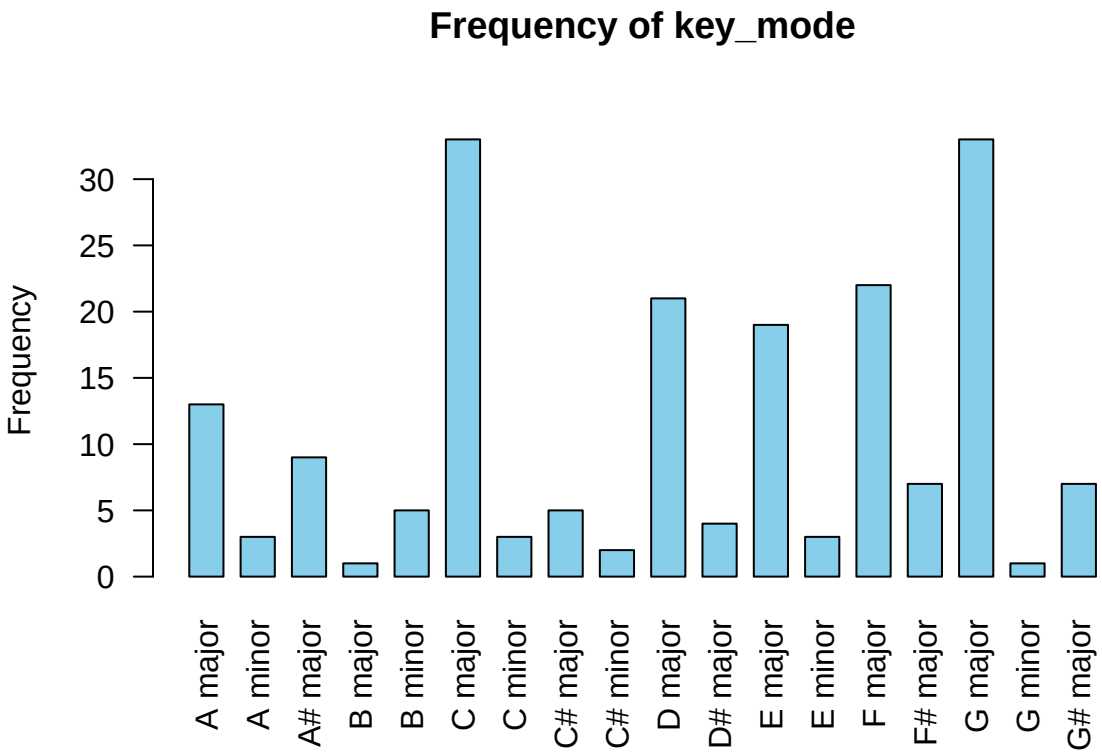

Frequency of explicit
Frequency
150
100
50
0
FALSE
TRUE


Frequency of key
Frequency
35
30
25
20
15
10
5
0
0
1
2
3
4
5
6
7
8
9
10
11


Frequency of key_name
Frequency
35
30
25
20
15
10
5
0
G
F
C
G#
E
D#
D
A
B
C#
A#
F#

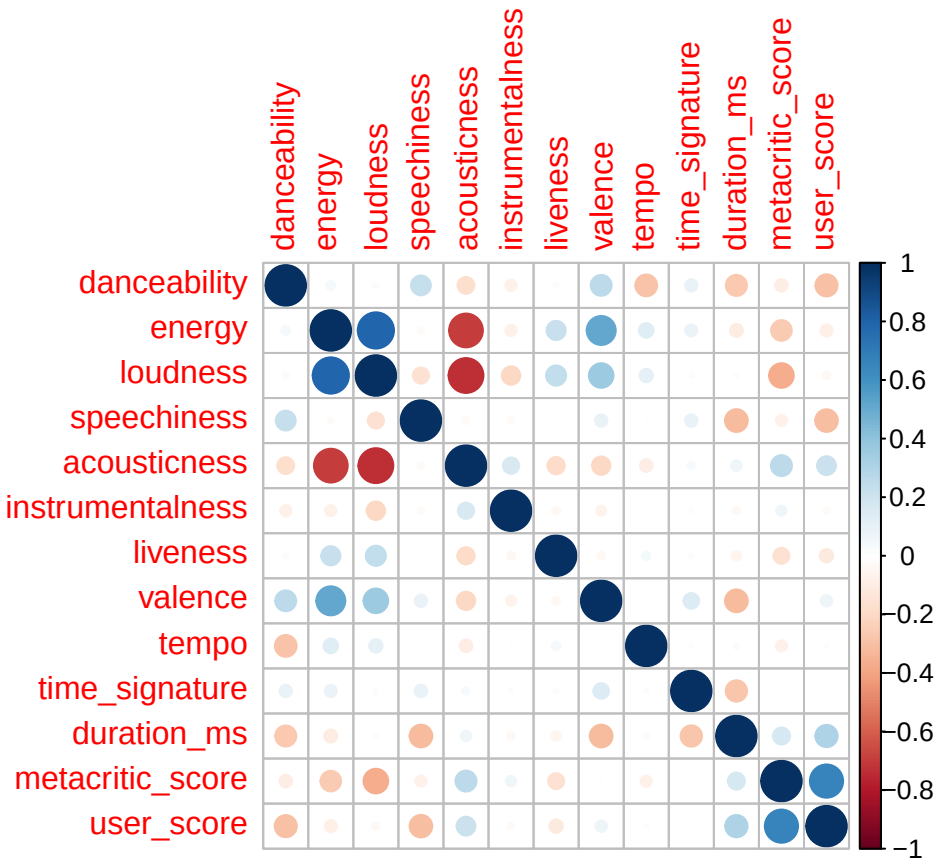

Frequency of mode
Frequency
150
100
50
0
0
1


Frequency of mode_name
Frequency
150
100
50
0
major
minor



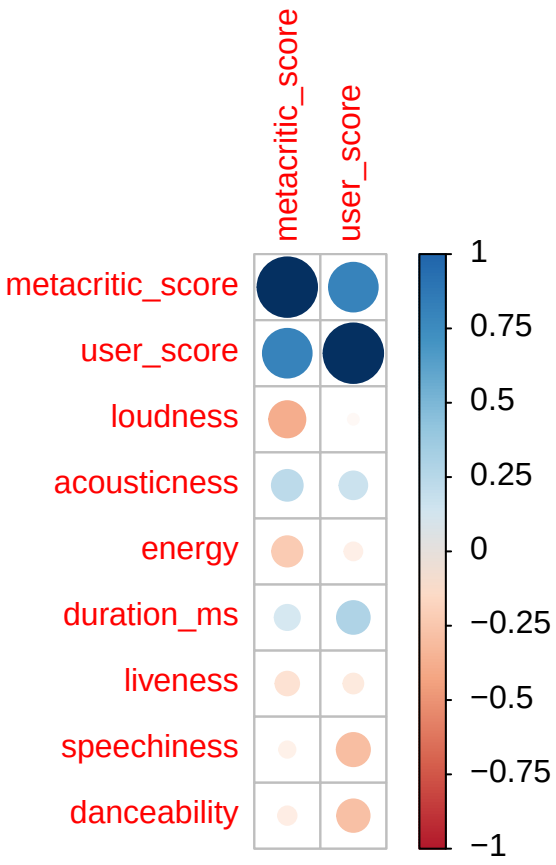
Correlation matrix for numerical variables

```
numerical_feats_and_scores_df <- album_song_with_scores_df[c("danceability",
                                                             "energy",
                                                             "loudness",
                                                             "speechiness",
                                                             "acousticness",
                                                             "instrumentalness",
                                                             "liveness",
                                                             "valence",
                                                             "tempo",
                                                             "time_signature",
                                                             "duration_ms",
                                                             "metacritic_score",
                                                             "user_score")]

numerical_feats_and_scores_df <- na.omit(numerical_feats_and_scores_df)

# Compute the correlation matrix
correlation_matrix <- cor(numerical_feats_and_scores_df)

# Visualize the correlation matrix using corrplot
corrplot(correlation_matrix, method = "circle")
```



Since a number of variables were showing negligible correlation to the scores, we decided to limit the set of features shown to the 5 most correlated to each score, totaling 7 after removing duplicates.

```
# Filter out the most correlated columns with respect to "metacritic_score"
metacritic_correlation <- abs(correlation_matrix["metacritic_score", ])
most_correlated_metacritic <- names(sort(metacritic_correlation,
                                         decreasing = TRUE)[2:7])
# Excluding self-correlation

# Filter out the most correlated columns with respect to "user_score"
user_correlation <- abs(correlation_matrix["user_score", ])
most_correlated_user <- names(sort(user_correlation, decreasing = TRUE)[2:7])

highly_correlated = unique(c(c("metacritic_score", "user_score"), most_correlated_metacritic, most_corre
salient_numerical_feats_and_scores_df = numerical_feats_and_scores_df[highly_correlated]

# Compute the correlation matrix
correlation_matrix <- cor(salient_numerical_feats_and_scores_df)[, c(1, 2)]

# Visualize the correlation matrix using corrplot
corrplot(correlation_matrix, method = "circle", cl.pos = "n")

colorlegend(xlim=c(3,4), ylim=c(0.5,9.5), colorRampPalette(brewer.pal(8, "RdBu"))(200), c(seq(-1,1,.25)
```



Discussion

A crucial insight is that songs in the major mode, typically associated with a cheerful and upbeat mood, consistently receive higher ratings from both users and Metacritic. In contrast, songs in the minor mode, often characterized by a more melancholic and somber tone, tend to score lower.

Surprisingly, songs with explicit lyrics also outperform those without. This is more evident in both scores, although the difference is much more pronounced looking at the Metacritic score.

In terms of keys, songs in C#, D# and F# tend to get the highest scores with little variance, while those in keys such as A, C and G often score lower and have more variance.

It should be noted that data imbalance is present regarding all three attributes mentioned above, as evidenced by the supplementing bar charts. As such, the above claims should not be taken at face value.

In terms of numerical features, there are no cases where the opinions of Metacritic reviewers and those of users oppose. However, each group places different weights on different attributes, positive or negative.

Certain attributes often lead to a lower score. Metacritic has a noticeable bias against loudness, energy and liveness compared to users; in particular, users are virtually indifferent towards loudness. On the other hand, users tend to score “speech” and “danceable” songs more harshly compared to Metacritic itself.

In terms of favourable attributes, both groups show a preference for longer and more acoustic songs. However, while Metacritic places more emphasis on acousticness, users are more interested in the duration of a song.