Project

2024-04-03

Importing neccesary packages

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

Introduction

Information about the dataset can be found at: https://github.com/rfordatascience/tidytuesday/blob/master/data/2023/2023-10-17/readme.md

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

According to our proposal, we would like to conduct a series of necessary data mining and visualization tasks to extract meaningful insights from this dataset. Considering the characteristics of the dataset, we believe it is reasonable to perform the following analytics:

1. Descriptive Statistics (for both numerical and categorical features)
2. Features Distribution by scores
3. Time-series Analysis (based on the album release date)
4. Correlation Analysis (for the numerical features)
5. Regression Analysis

With the hypothesis that audio features (numerical and categorical values) and the scores for each album have an inferential relationship, we will conduct our Regression Analysis using the audio features as input variables and scores as output variables. We will focus on two aspects:

- Since we lack scores for individual songs, it is reasonable to assign each song the score from "taylor_album_songs.csv" based on the score for the entire album in "taylor_albums.csv."
- One might suggest that we can aggregate all song features to represent an album, using that representation as input features to predict the scores. For simplicity, we will use the mean and median as the statistical measures to aggregate all the songs. However, the current size of the dataset that we can retrieve is relatively small (~12 valid records), which theoretically can be challenging to fit a regression model. Hence, we would like to discuss the potential of using such representation without fitting a specific model in our implementation.

Dataset 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

```
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).

```r
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.

```r
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

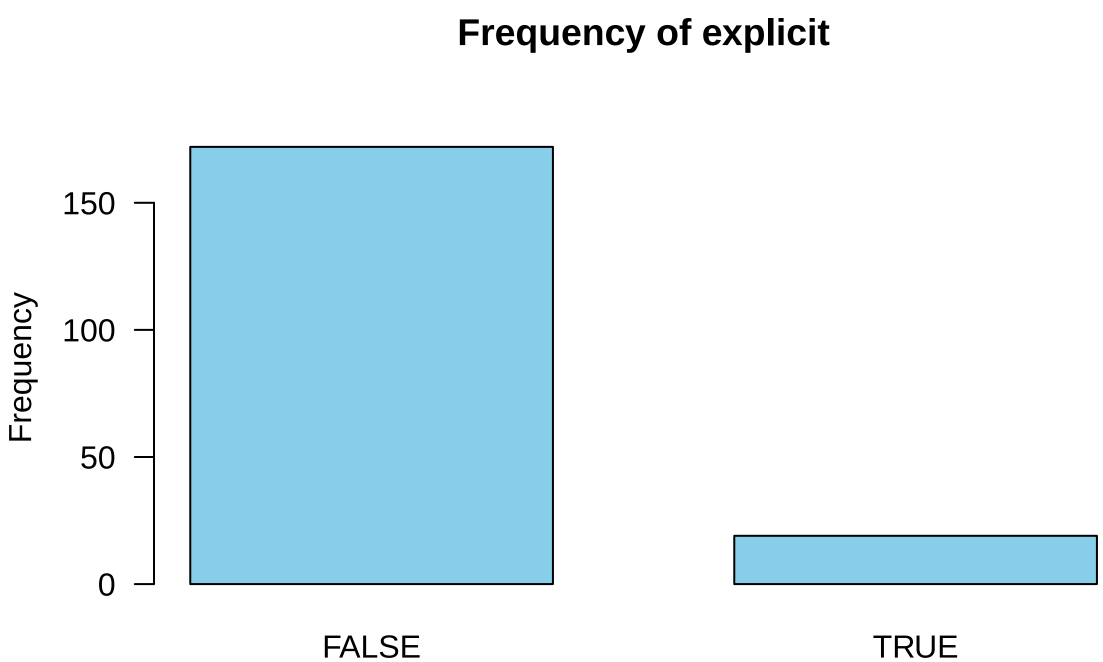
```r
# 3 first index columns

# 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
```

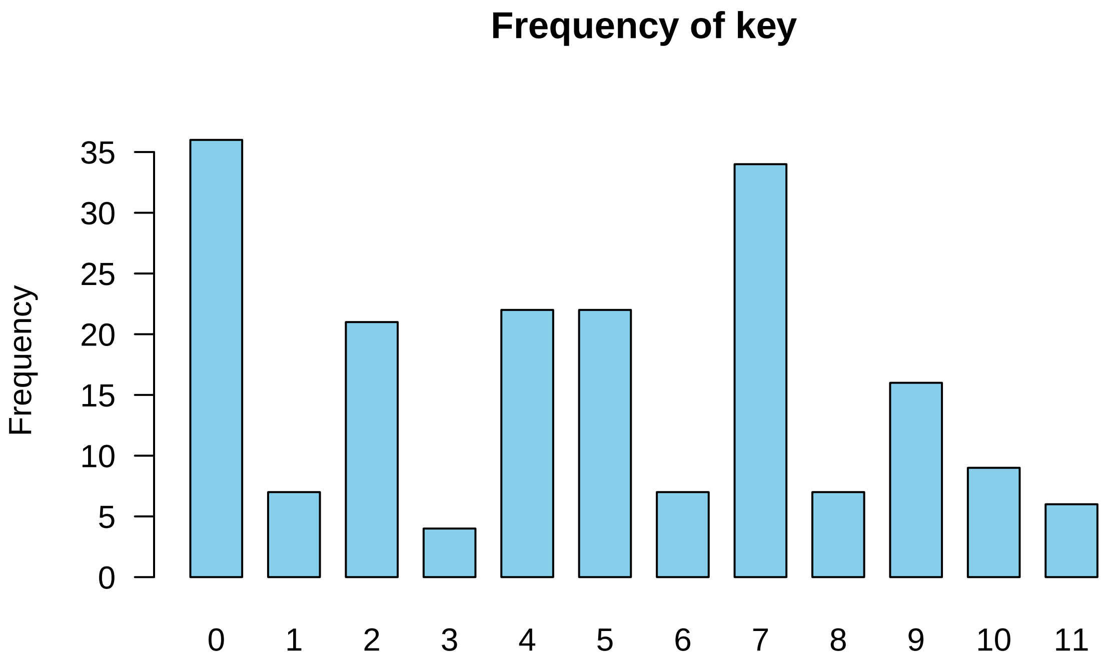
1. Descriptive Statistics

1.1. Categorical input variables

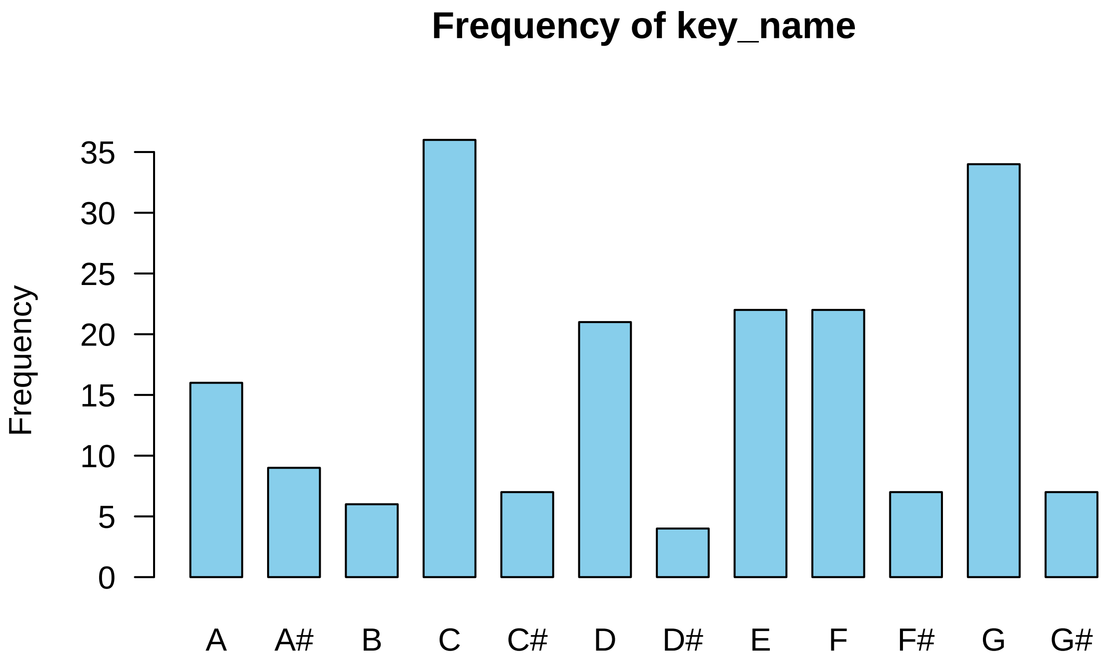
```r
categorical_feats_df <- album_song_with_scores_df[categorical_feats]

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
```

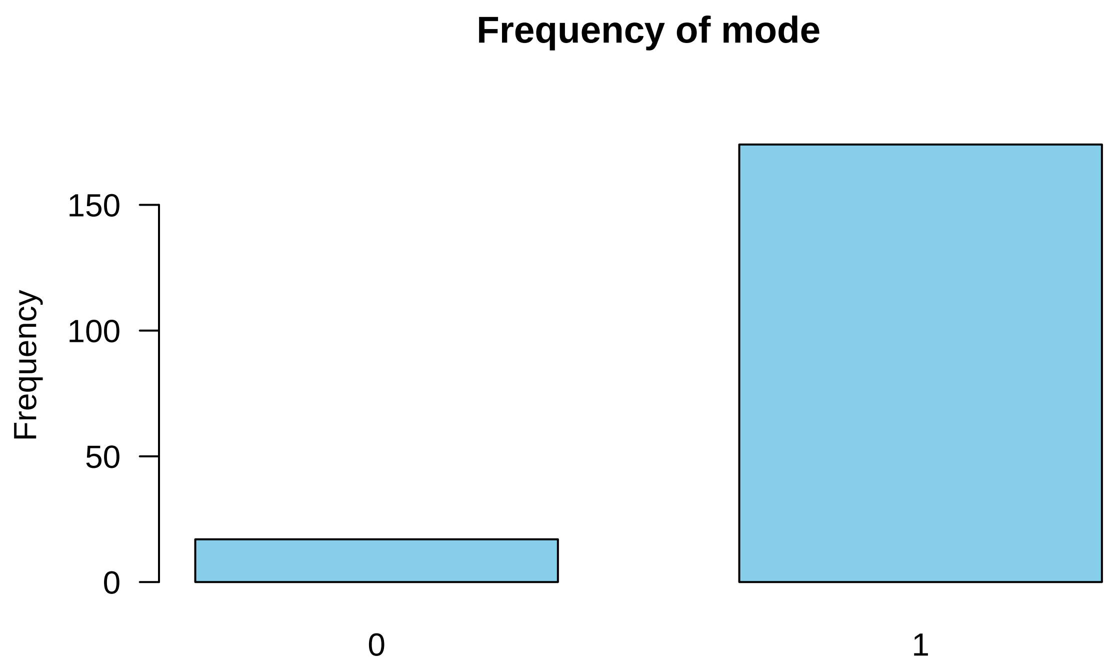
```
  # horizontal x-axis labels
  barplot(frequency_tables[[i]],
          main = paste("Frequency of", names(frequency_tables)[i]),
          ylab = "Frequency",
          col = "skyblue",
          border = "black",
          space = 0.5,  # Adjust the spacing between bars
          las = ifelse(names(frequency_tables)[i]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

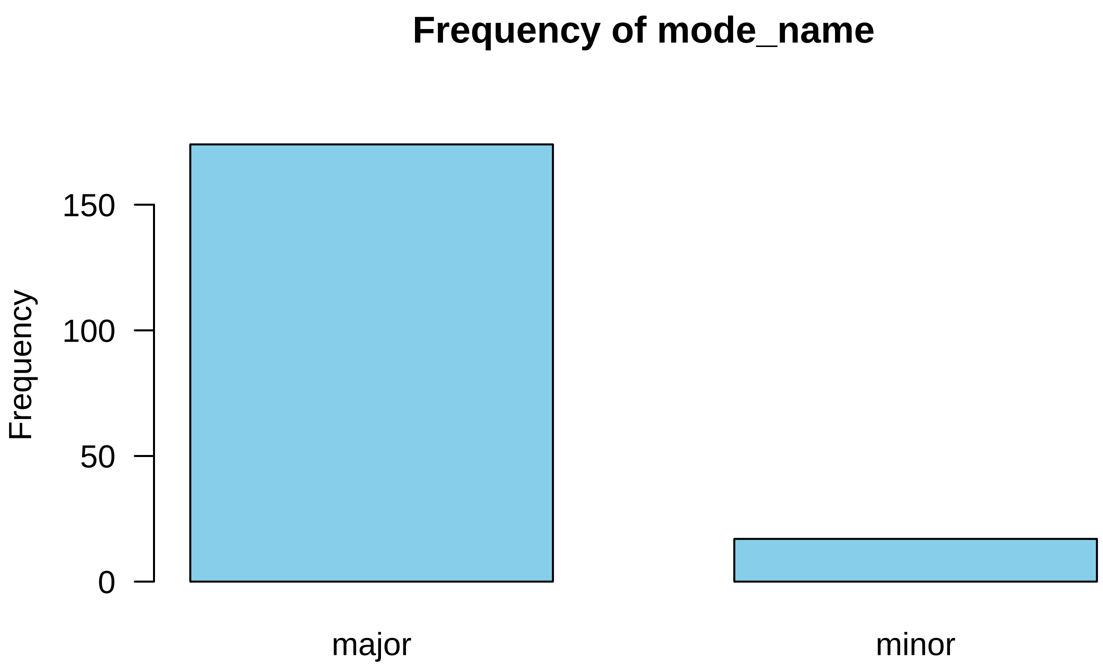


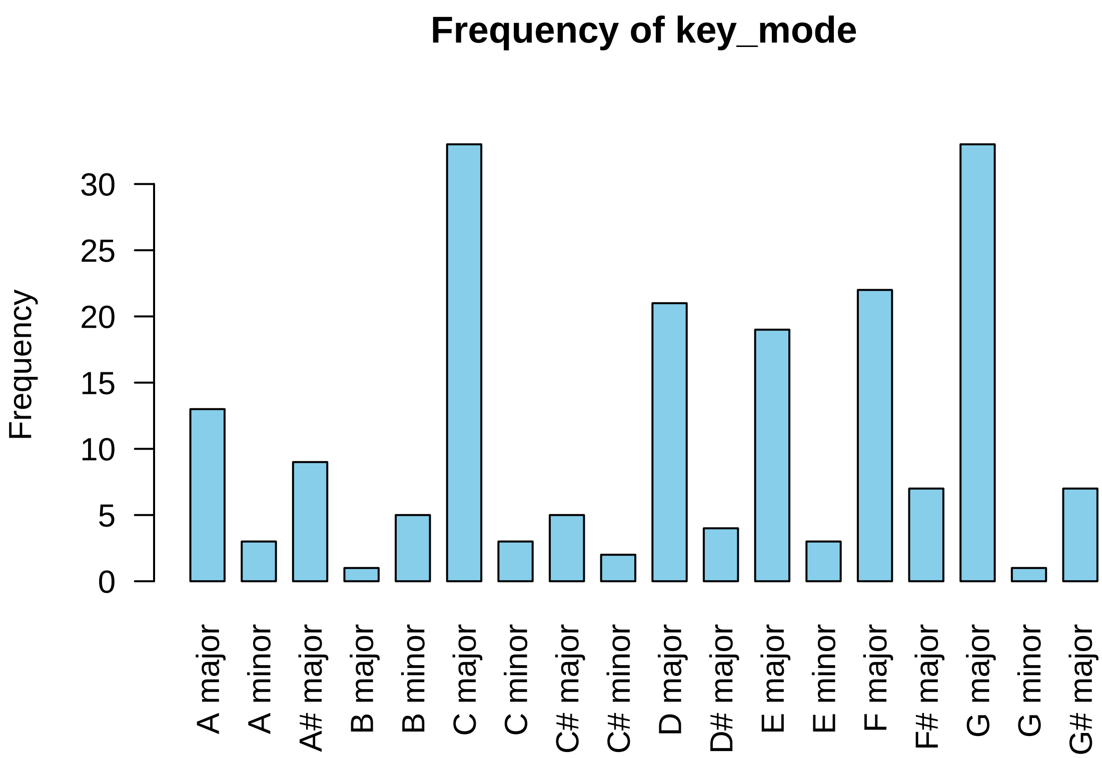


Frequency of key
Frequency
35
30
25
20
15
10
5
0
0
1
2
3
4
5
6
7
8
9
10
11


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


Frequency of mode
Frequency
150
100
50
0
0
1


Frequency of mode_name
Frequency
0
50
100
150
major
minor



1.2. Numerical input variables

```r
numerical_feats_df <- album_song_with_scores_df[numerical_feats]

numerical_feats_df <- na.omit(numerical_feats_df)

summary(numerical_feats_df)
```

```
##   danceability        energy           loudness         speechiness
##  Min.   :0.292   Min.   :0.1310   Min.   :-15.434   Min.   :0.02310
##  1st Qu.:0.511   1st Qu.:0.4465   1st Qu.: -9.326   1st Qu.:0.03080
##  Median :0.594   Median :0.5800   Median : -6.937   Median :0.03960
##  Mean   :0.584   Mean   :0.5745   Mean   : -7.518   Mean   :0.05831
##  3rd Qu.:0.652   3rd Qu.:0.7170   3rd Qu.: -5.606   3rd Qu.:0.05740
##  Max.   :0.897   Max.   :0.9500   Max.   : -2.098   Max.   :0.51900
##   acousticness       instrumentalness       liveness           valence
##  Min.   :0.000191   Min.   :0.0000000   Min.   :0.03570   Min.   :0.0382
##  1st Qu.:0.034600   1st Qu.:0.0000000   1st Qu.:0.09295   1st Qu.:0.2535
##  Median :0.162000   Median :0.0000014   Median :0.11500   Median :0.4040
##  Mean   :0.321225   Mean   :0.0039358   Mean   :0.14081   Mean   :0.4009
##  3rd Qu.:0.662000   3rd Qu.:0.0000399   3rd Qu.:0.15050   3rd Qu.:0.5345
##  Max.   :0.971000   Max.   :0.3480000   Max.   :0.59400   Max.   :0.9420
##      tempo         time_signature   duration_ms
```

```
##  Min.   : 68.53   Min.   :1.000   Min.   :148781
##  1st Qu.: 99.98   1st Qu.:4.000   1st Qu.:209326
##  Median :121.96   Median :4.000   Median :232107
##  Mean   :125.99   Mean   :3.979   Mean   :237079
##  3rd Qu.:150.03   3rd Qu.:4.000   3rd Qu.:254448
##  Max.   :208.92   Max.   :5.000   Max.   :613027
```

1.3. Target variables

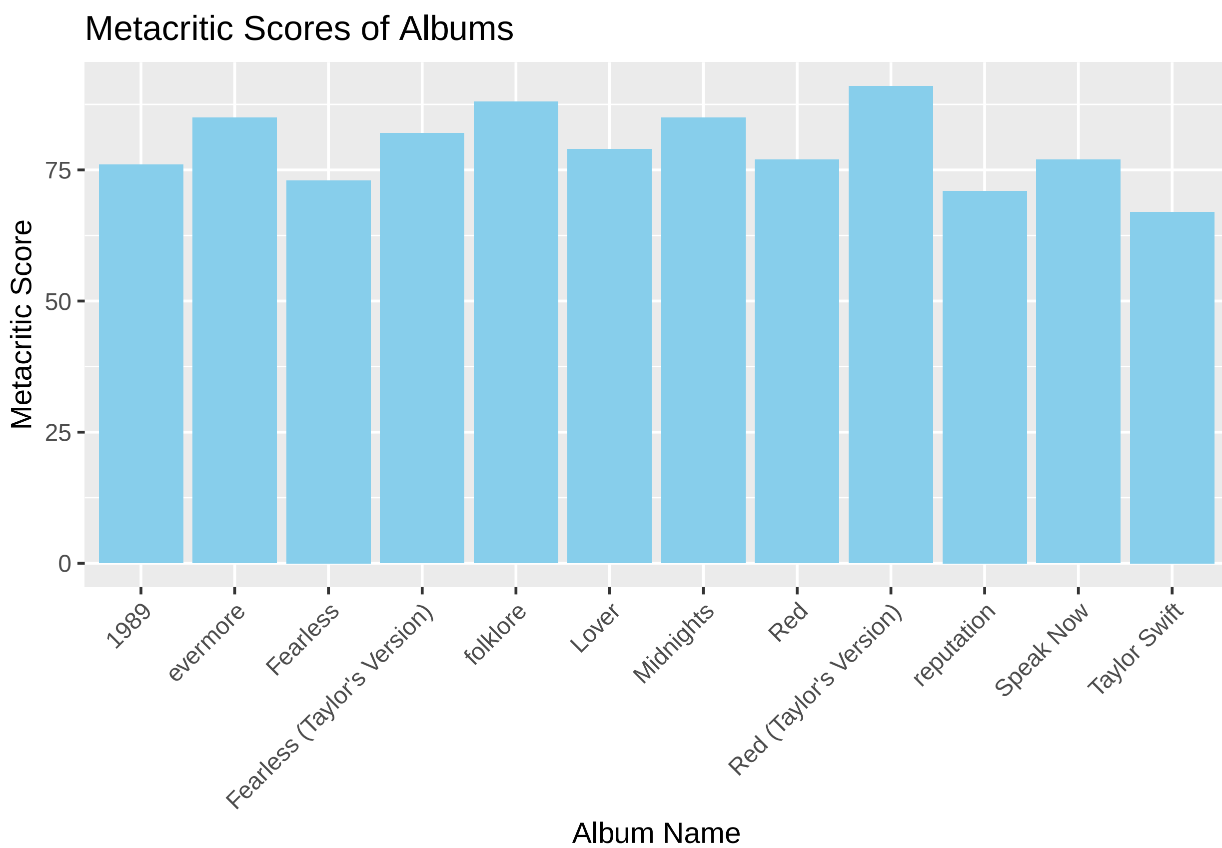
1.3.1. The descriptive STAT for the scores per album

```r
target_df <- album_df[c("metacritic_score", "user_score")]

target_df <- na.omit(target_df)

summary(target_df)
```

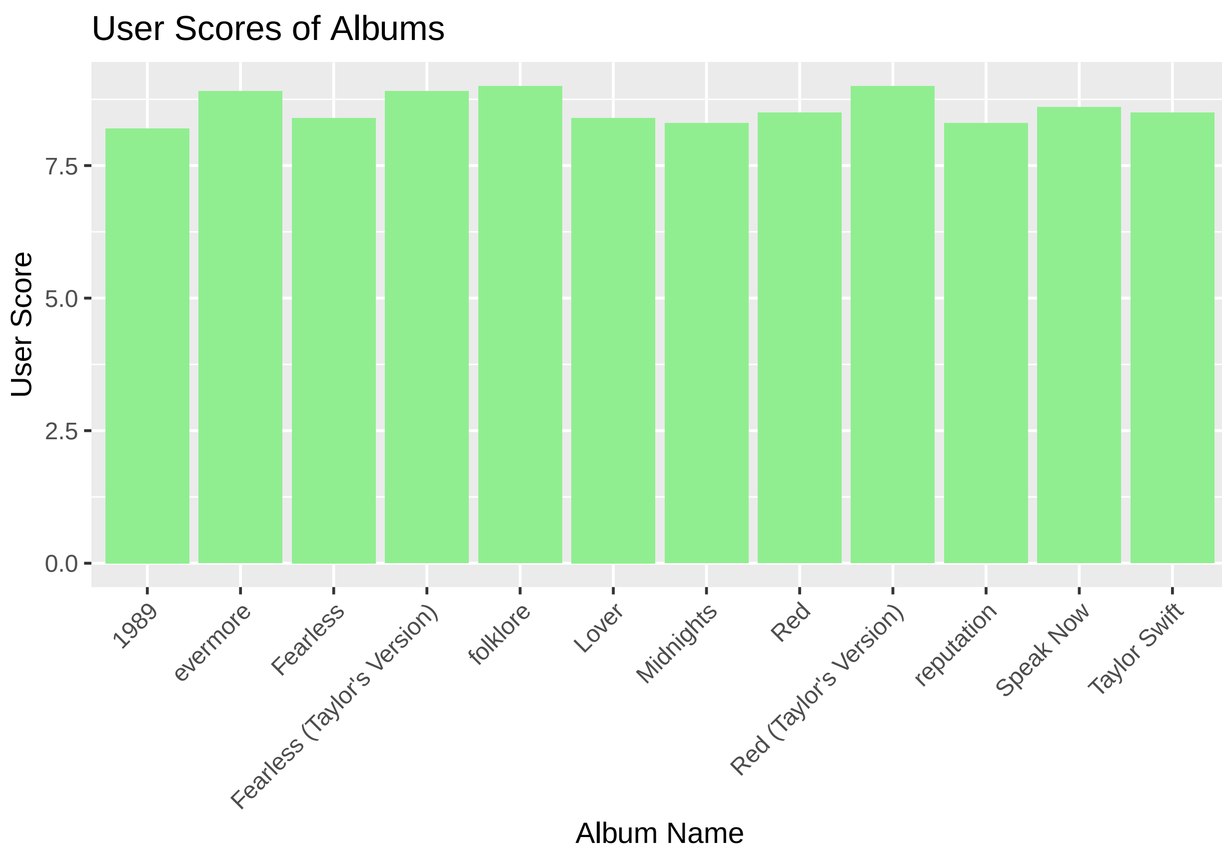
```
##  metacritic_score   user_score
##  Min.   :67.00    Min.   :8.200
##  1st Qu.:75.25    1st Qu.:8.375
##  Median :78.00    Median :8.500
##  Mean   :79.25    Mean   :8.583
##  3rd Qu.:85.00    3rd Qu.:8.900
##  Max.   :91.00    Max.   :9.000
```

1.3.2. Scores for each album

```r
# Convert album_release to Date format
album_df$album_release <- as.Date(album_df$album_release)

# Plot Metacritic scores
ggplot(album_df, aes(x = album_name, y = metacritic_score)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Metacritic Scores of Albums",
       x = "Album Name",
       y = "Metacritic Score")
```

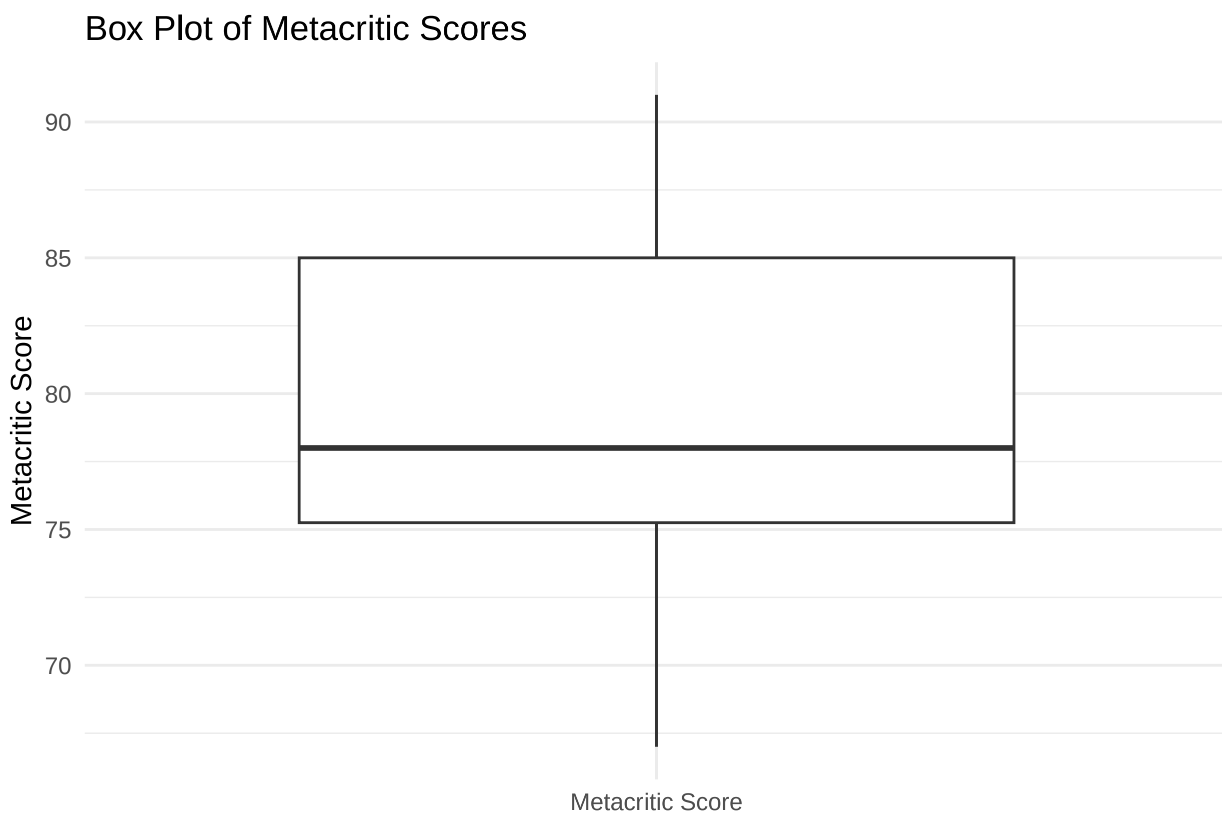



```r
# Plot User scores
ggplot(album_df, aes(x = album_name, y = user_score)) +
  geom_bar(stat = "identity", fill = "lightgreen") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "User Scores of Albums",
       x = "Album Name",
       y = "User Score")
```

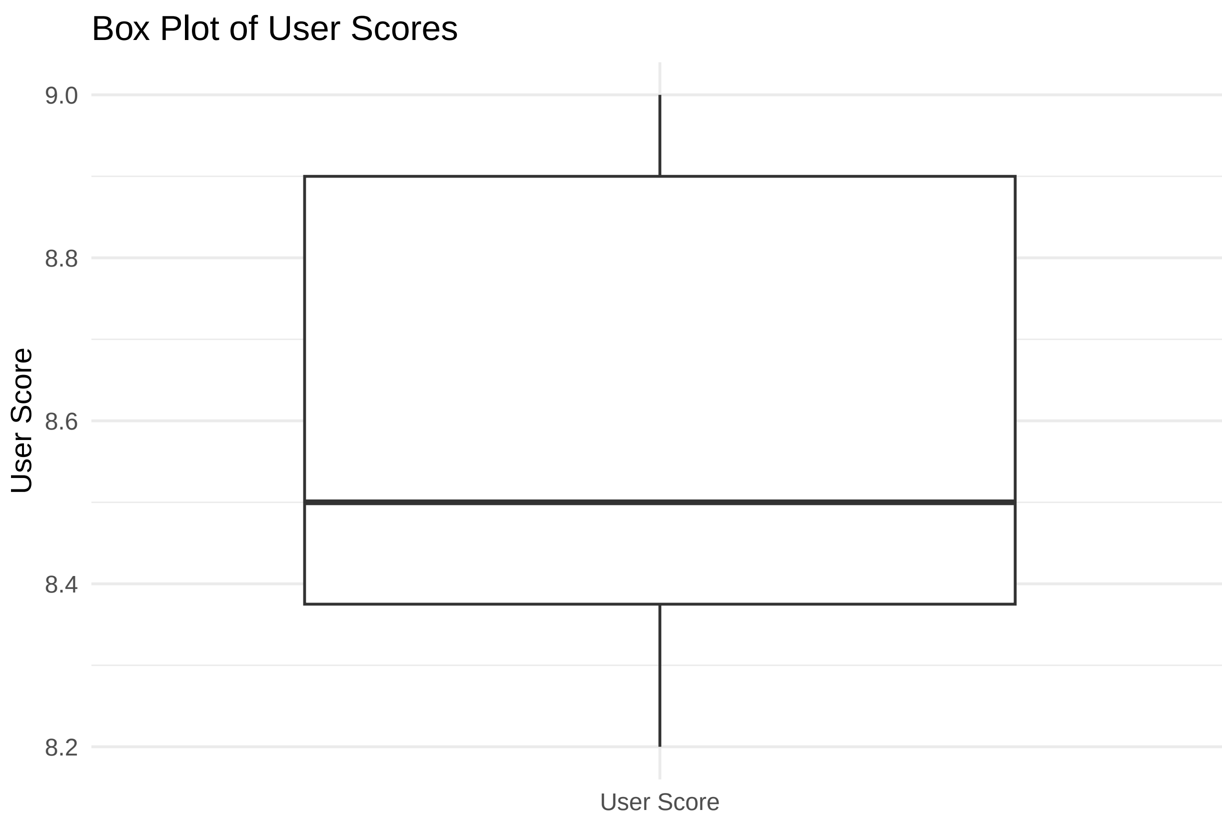


It seems that there is a strong correlation between the two scores. The patterns look almost the same, with the user scores exhibiting more consistent distribution.

```r
# Plot box plot for Metacritic scores
ggplot(album_df, aes(x = "Metacritic Score", y = metacritic_score)) +
  geom_boxplot() +
  labs(title = "Box Plot of Metacritic Scores",
       x = "",
       y = "Metacritic Score") +
  theme_minimal()
```




```r
# Plot box plot for User scores
ggplot(album_df, aes(x = "User Score", y = user_score)) +
  geom_boxplot() +
  labs(title = "Box Plot of User Scores",
       x = "",
       y = "User Score") +
  theme_minimal()
```



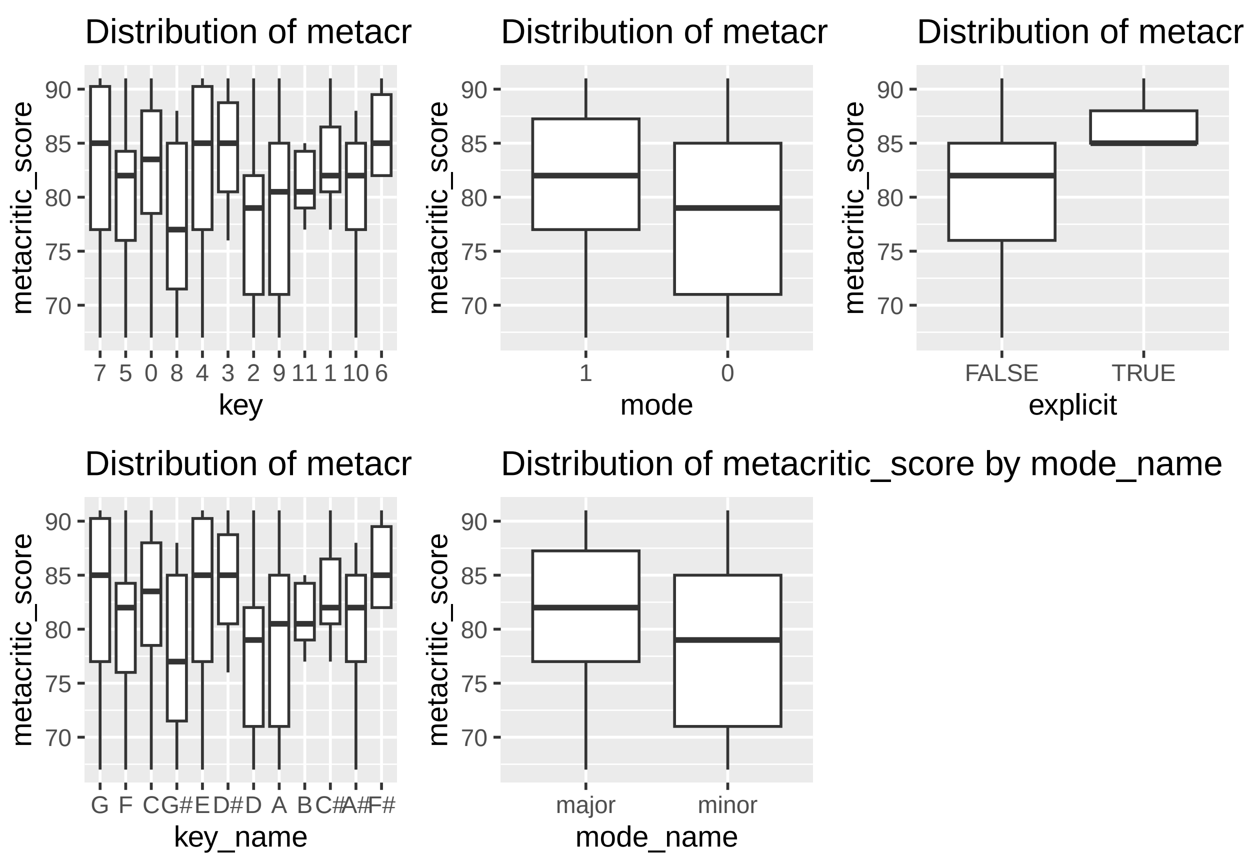
2. Feature distribution by scores

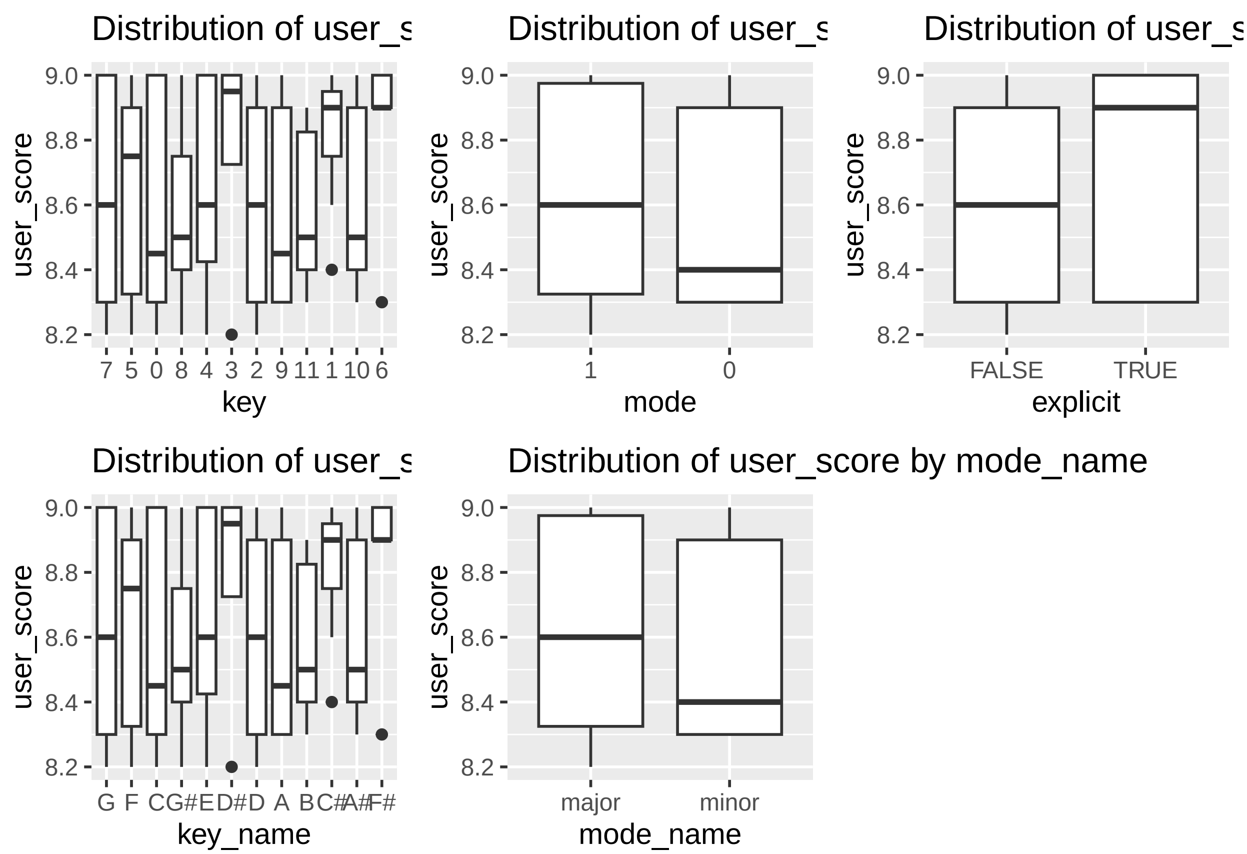
We employ a facet grids to visualize the distribution of the categorical features by scores.

```
# Define the list of categorical columns
categorical_cols <- c("key", "mode", "explicit", "key_name", "mode_name")

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

  # 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 = 3)
  # 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.
```





Some findings from the metacritic_score and the user_score graphs:

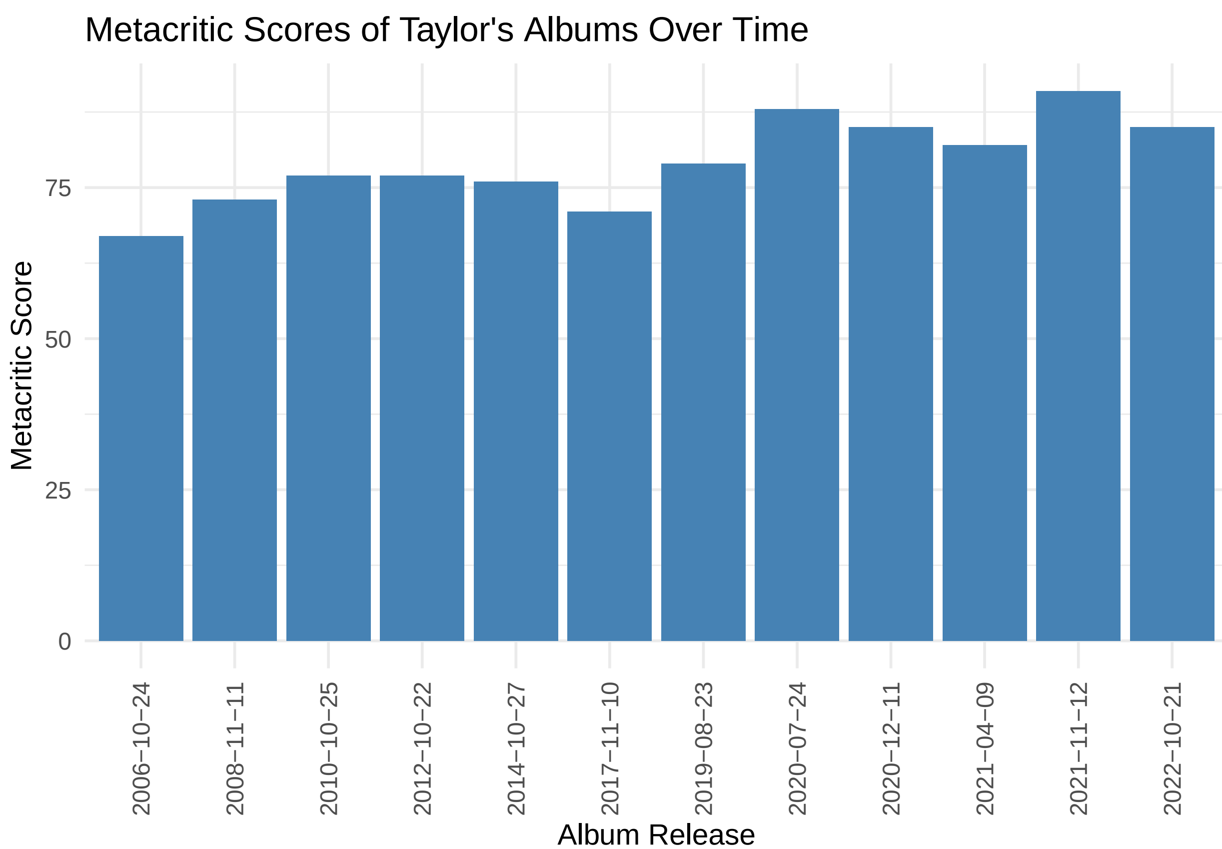
- Songs with explicit lyrics and a major mode seem to have higher scores than implicit ones.
- Songs in the keys of G, E, and F# have the highest scores.
- The most common keys used in Taylor's songs are G, C, E, D, and A, which reflect her style of using country music.

3. Time-series analysis

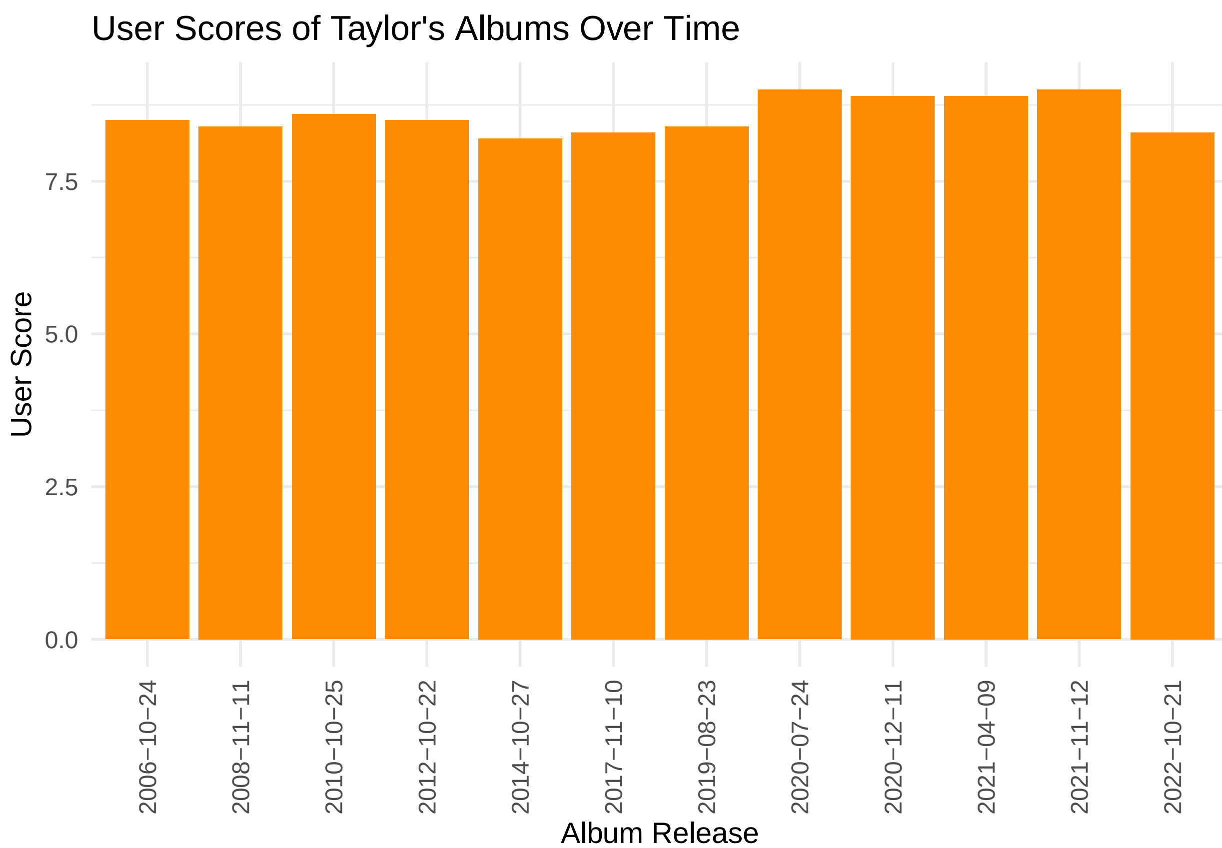
Album scores over time

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

# Convert album_release to Date format
album_df$album_release <- as.Date(album_df$album_release, format = "%Y-%m-%d")

# Convert album_release to character for the x-axis
album_df$album_release_char <- as.character(album_df$album_release)

# Plotting Metacritic scores
ggplot(album_df, aes(x = album_release_char, y = metacritic_score)) +
  geom_col(fill = "steelblue") +
```

```r
  theme_minimal() +
  labs(title = "Metacritic Scores of Taylor's Albums Over Time",
       x = "Album Release",
       y = "Metacritic Score") +
  scale_x_discrete(labels = album_df$album_release_char) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```




```r
# Plotting User scores
ggplot(album_df, aes(x = album_release_char, y = user_score)) +
  geom_col(fill = "darkorange") +
  theme_minimal() +
  labs(title = "User Scores of Taylor's Albums Over Time",
       x = "Album Release",
       y = "User Score") +
  scale_x_discrete(labels = album_df$album_release_char) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```

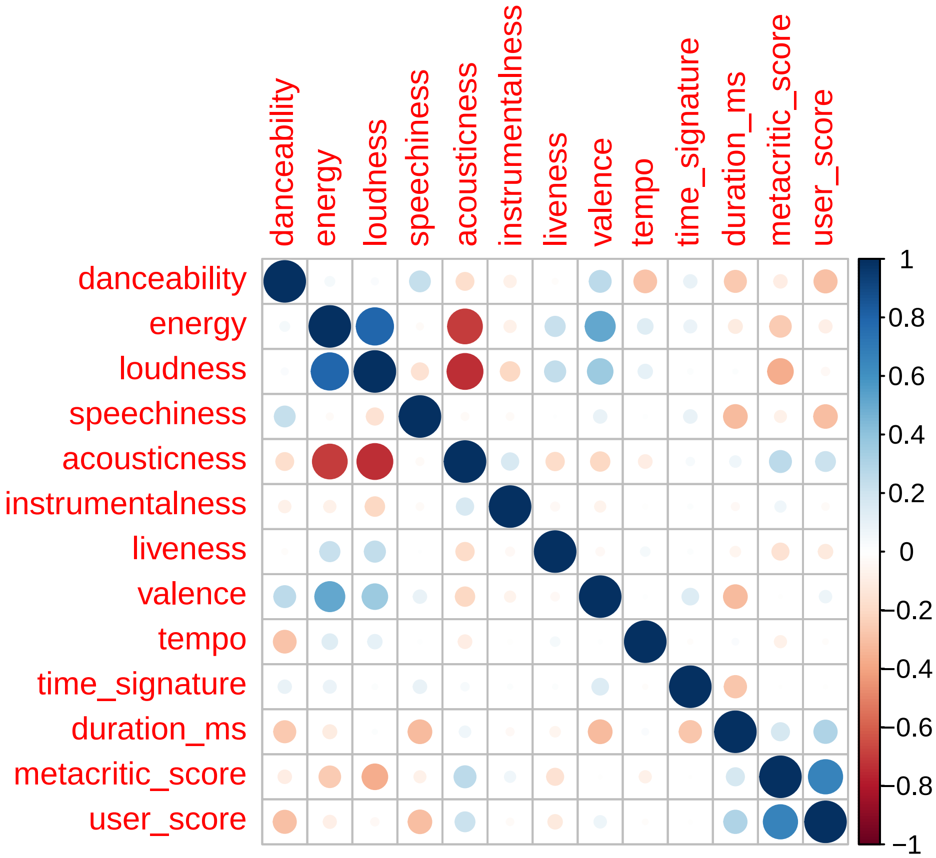


4. Correlation analysis

4.1. Correlation map

```
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)
```

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



4.2. Comments

We can see that the most correlated columns w.r.t the target columns are:

```r
# 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])
# Excluding self-correlation

# Display the most correlated columns
cat("Most correlated columns with respect to metacritic_score:\n")
```

```
## Most correlated columns with respect to metacritic_score:
```

```r
cat(most_correlated_metacritic, "\n")
```

```
## user_score loudness acousticness energy duration_ms liveness
```

```r
cat("Most correlated columns with respect to user_score:\n")
```

```
## Most correlated columns with respect to user_score:
```

```r
cat(most_correlated_user, "\n")
```

```
## metacritic_score speechiness duration_ms danceability acousticness liveness
```

```r
# Select most correlated columns with
# respect to "metacritic_score" excluding "user_score"
selected_columns_ms <- setdiff(most_correlated_metacritic,
                               "user_score")
selected_columns_ms <- c(selected_columns_ms,
                         "metacritic_score")

# Create subset dataframe with selected columns for "metacritic_score"
selected_numerical_feats_and_ms_df <- album_song_with_scores_df[selected_columns_ms]

# Select most correlated columns with respect to "user_score"
# excluding "metacritic_score"
selected_columns_user <- setdiff(most_correlated_user,
                         "metacritic_score")
selected_columns_user <- c(selected_columns_user, "user_score")

# Create subset dataframe with selected columns for "user_score"
selected_numerical_feats_and_user_score_df <-
  album_song_with_scores_df[selected_columns_user]
```

5. Regression Analysis

5.1. Regression Analysis using numerical values

In this section, we proceed with the columns that have the highest correlation with the scores to use as input features and utilize metacritic_score and user_score as our predicted variables.

We will utilize scatter plots to visualize the data points for each input variable and the target scores.

5.1.1. Regression analysis on metacritic_score

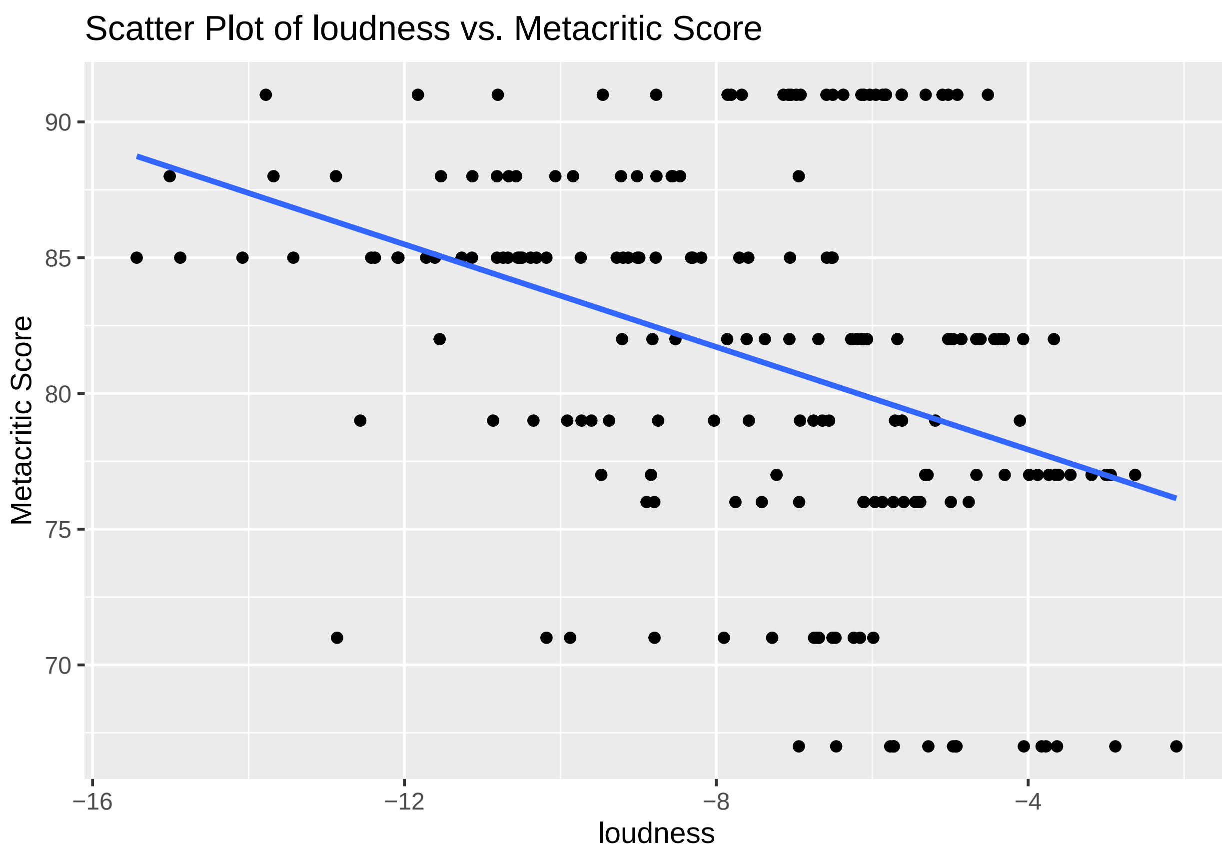
```r
# Perform linear regression analysis
regression_model <- lm(metacritic_score ~ ., data = selected_numerical_feats_and_ms_df)

# Summarize the regression results
summary(regression_model)
```

```
##
## Call:
## lm(formula = metacritic_score ~ ., data = selected_numerical_feats_and_ms_df)
```

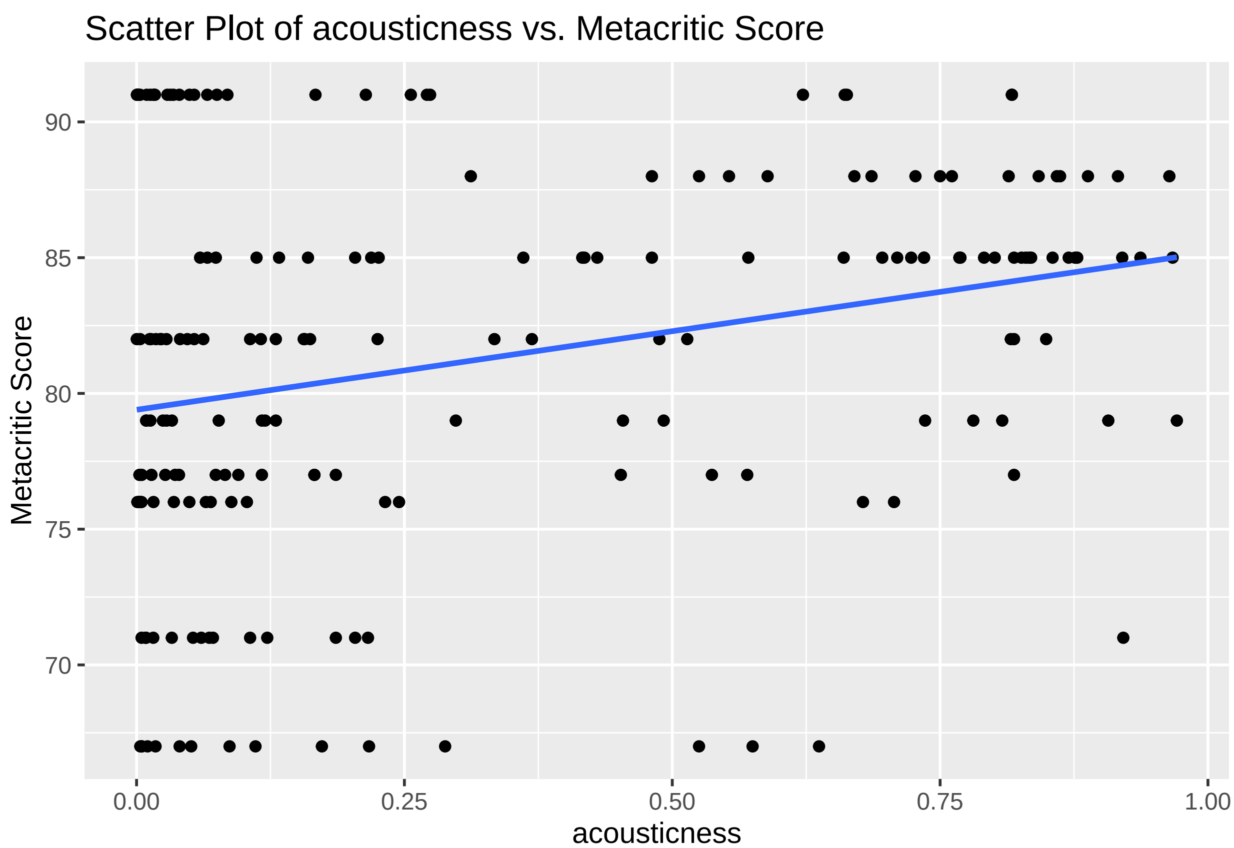
```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.2291  -3.9857   0.4283   3.8485  12.7599
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.204e+01  5.546e+00  11.187  < 2e-16 ***
## loudness     -1.254e+00  3.197e-01  -3.922 0.000124 ***
## acousticness -1.335e-01  2.214e+00  -0.060 0.952002
## energy        6.287e+00  4.487e+00   1.401 0.162838
## duration_ms   2.955e-05  1.021e-05   2.893 0.004276 **
## liveness     -5.585e+00  6.124e+00  -0.912 0.362943
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.524 on 185 degrees of freedom
## Multiple R-squared:  0.1816, Adjusted R-squared:  0.1595
## F-statistic: 8.213 on 5 and 185 DF,  p-value: 5.025e-07
```

```r
# Create a scatter plot of metacritic_score against each predictor variable
plot_data <- selected_numerical_feats_and_ms_df
predictor_variables <- setdiff(names(plot_data), "metacritic_score")

for (var in predictor_variables) {
  # Create scatter plot for each predictor variable
  plot <- ggplot(plot_data, aes_string(x = var, y = "metacritic_score")) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE) +
    labs(title = paste("Scatter Plot of", var, "vs. Metacritic Score"),
         x = var, y = "Metacritic Score")

  # Display each plot individually
  print(plot)
}
```

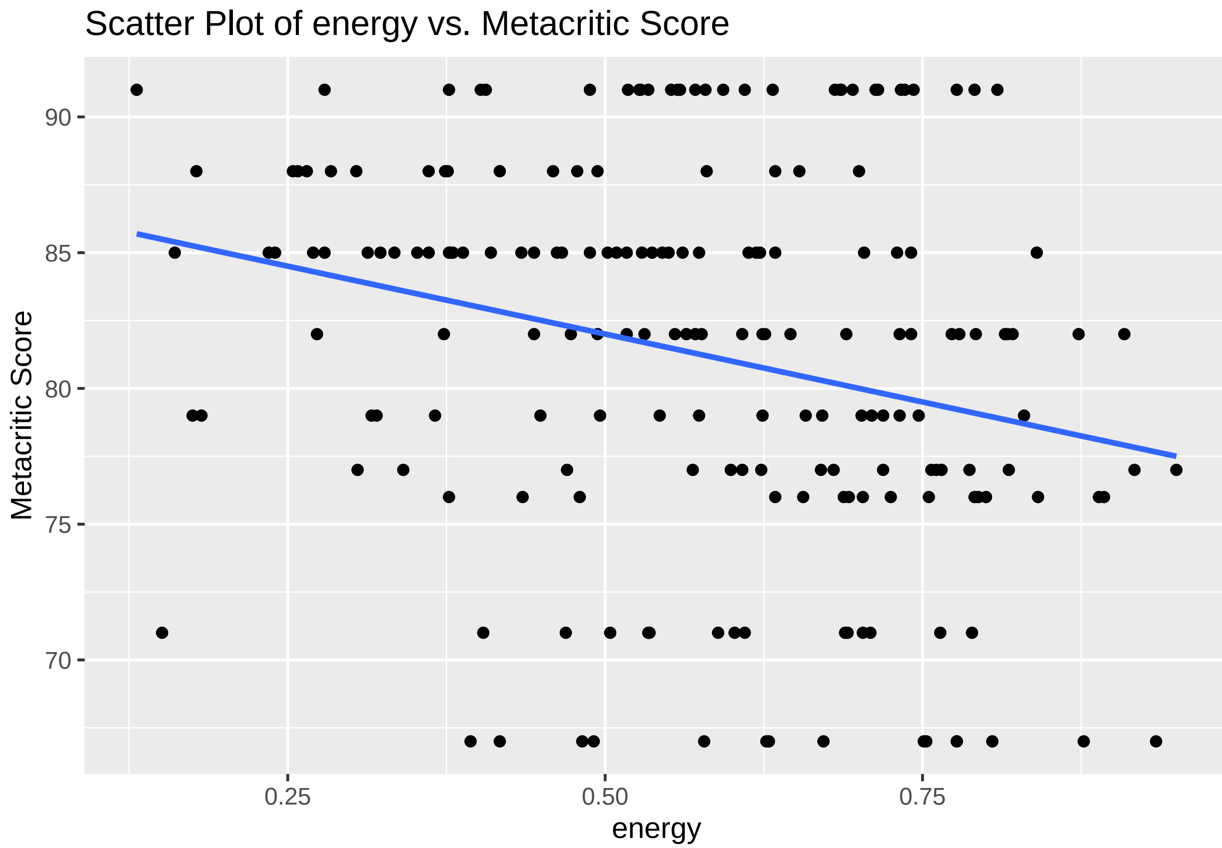
```
## `geom_smooth()` using formula = 'y ~ x'
```



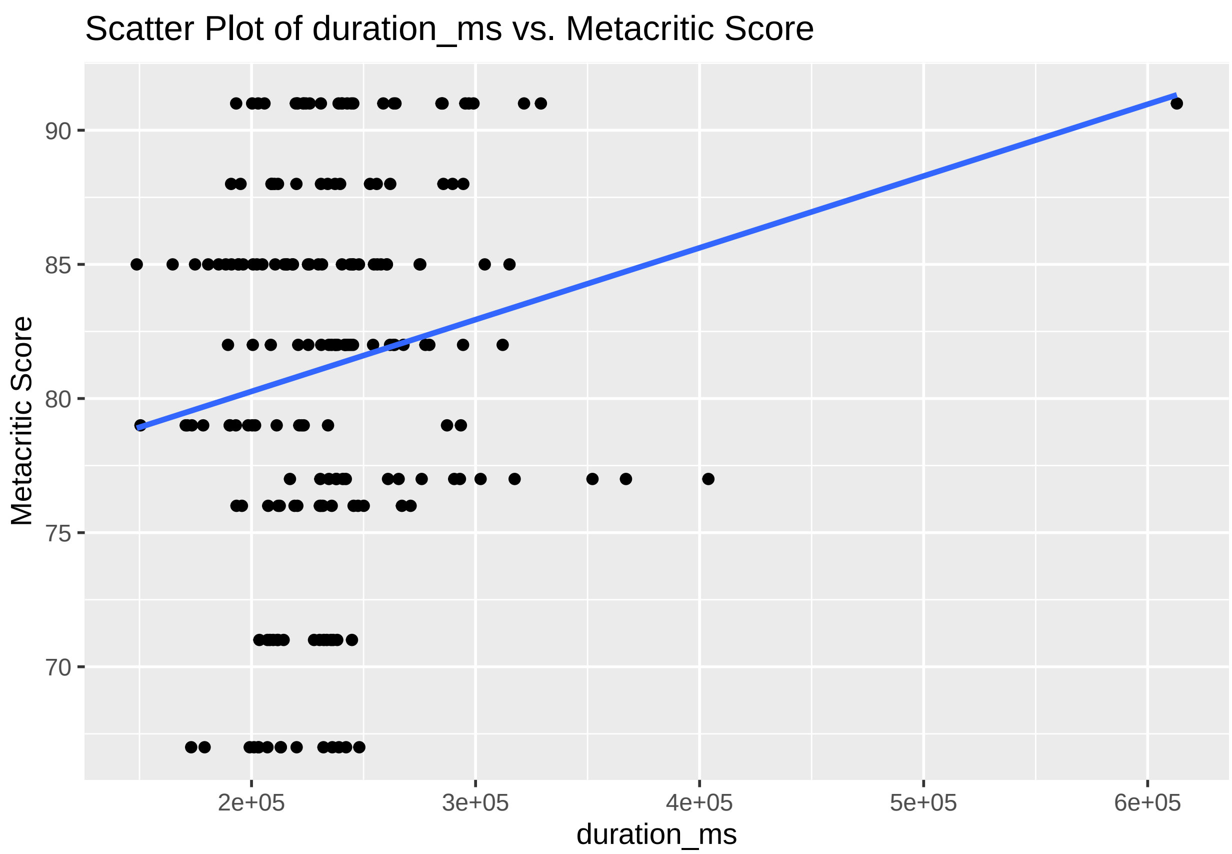

```
## `geom_smooth()` using formula = 'y ~ x'
```



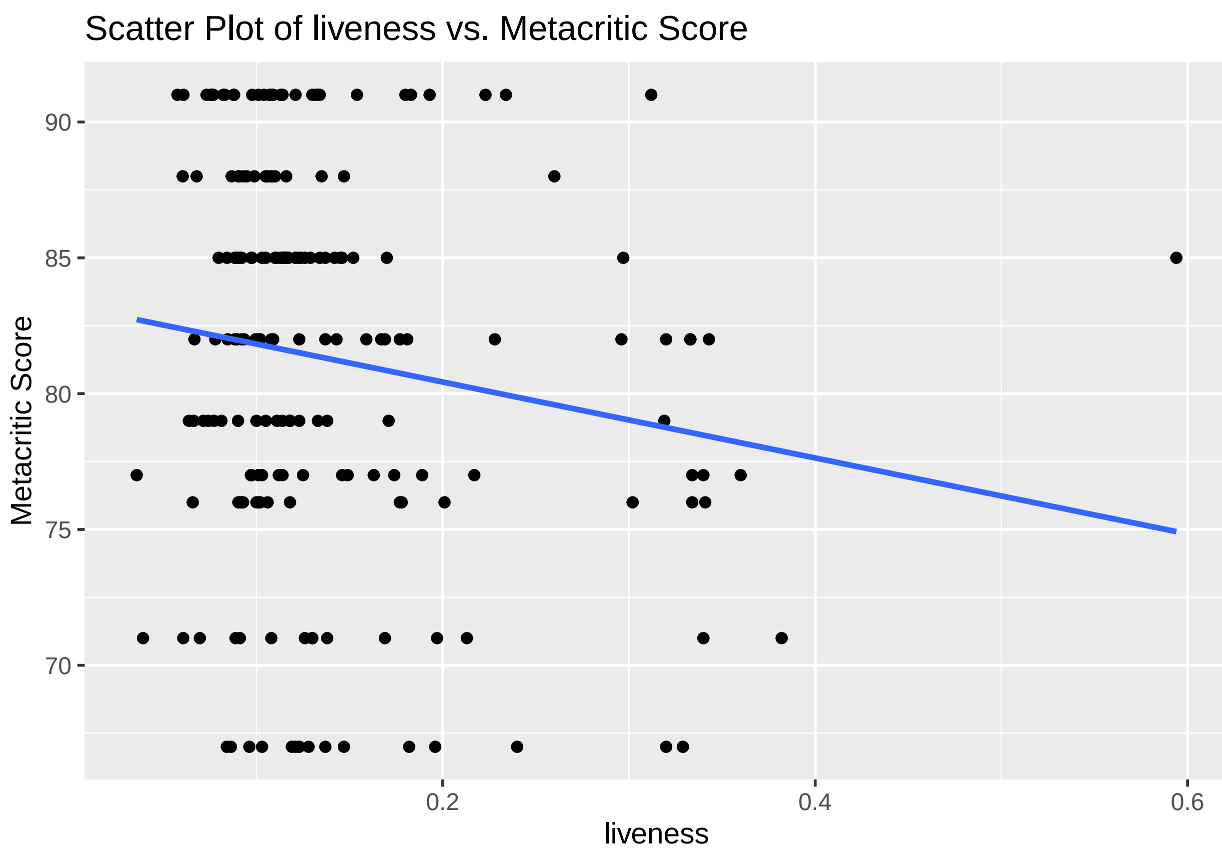

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
## `geom_smooth()` using formula = 'y ~ x'
```




```
## `geom_smooth()` using formula = 'y ~ x'
```



5.1.2. Regression analysis on user_score

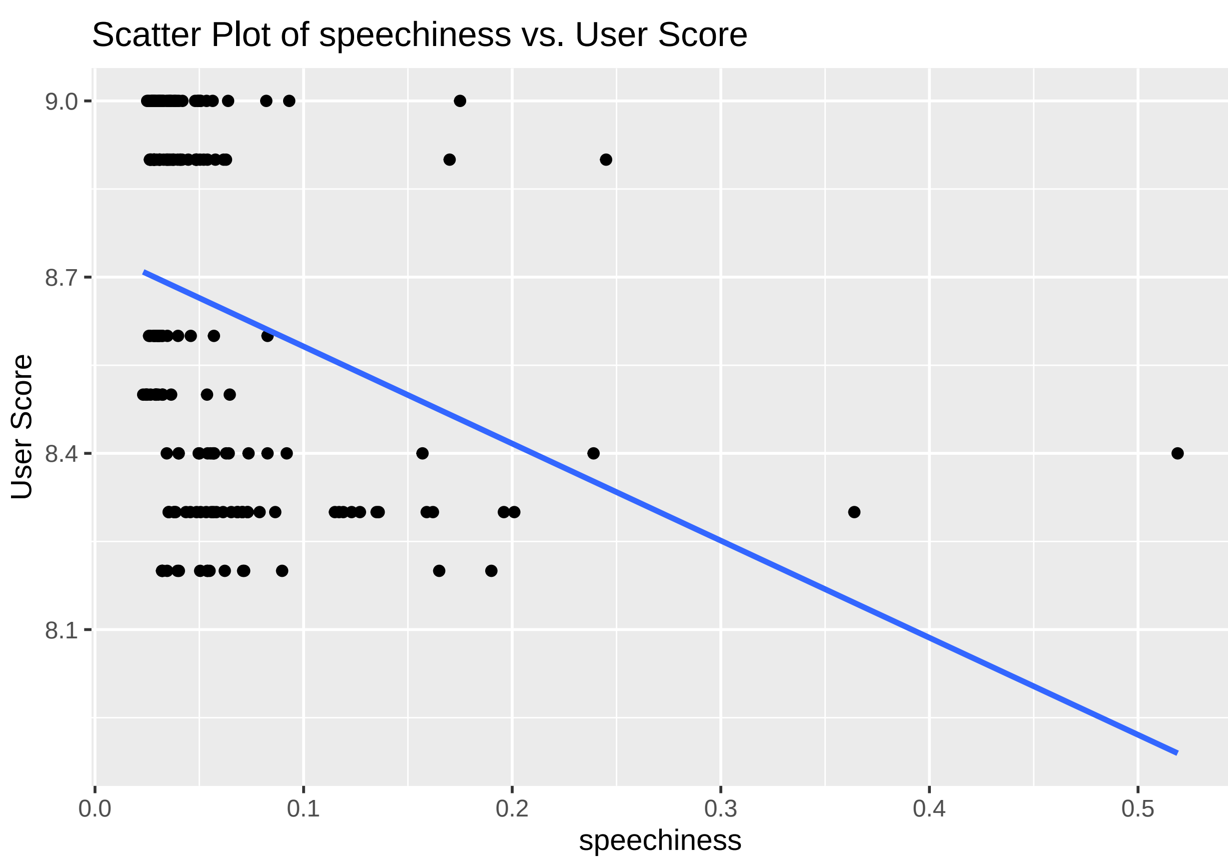
```
# Perform linear regression analysis
regression_model <- lm(user_score ~ ., data =
                          selected_numerical_feats_and_user_score_df)

# Summarize the regression results
summary(regression_model)
```

```
##
## Call:
## lm(formula = user_score ~ ., data = selected_numerical_feats_and_user_score_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.60627 -0.21217 -0.01964  0.23345  0.54260
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.711e+00  1.804e-01  48.292  < 2e-16 ***
## speechiness  -1.113e+00  3.723e-01  -2.989  0.00318 **
## duration_ms   1.121e-06  4.519e-07   2.480  0.01402 *
## danceability -4.575e-01  1.836e-01  -2.492  0.01359 *
## acousticness  1.428e-01  6.224e-02   2.294  0.02292 *
```

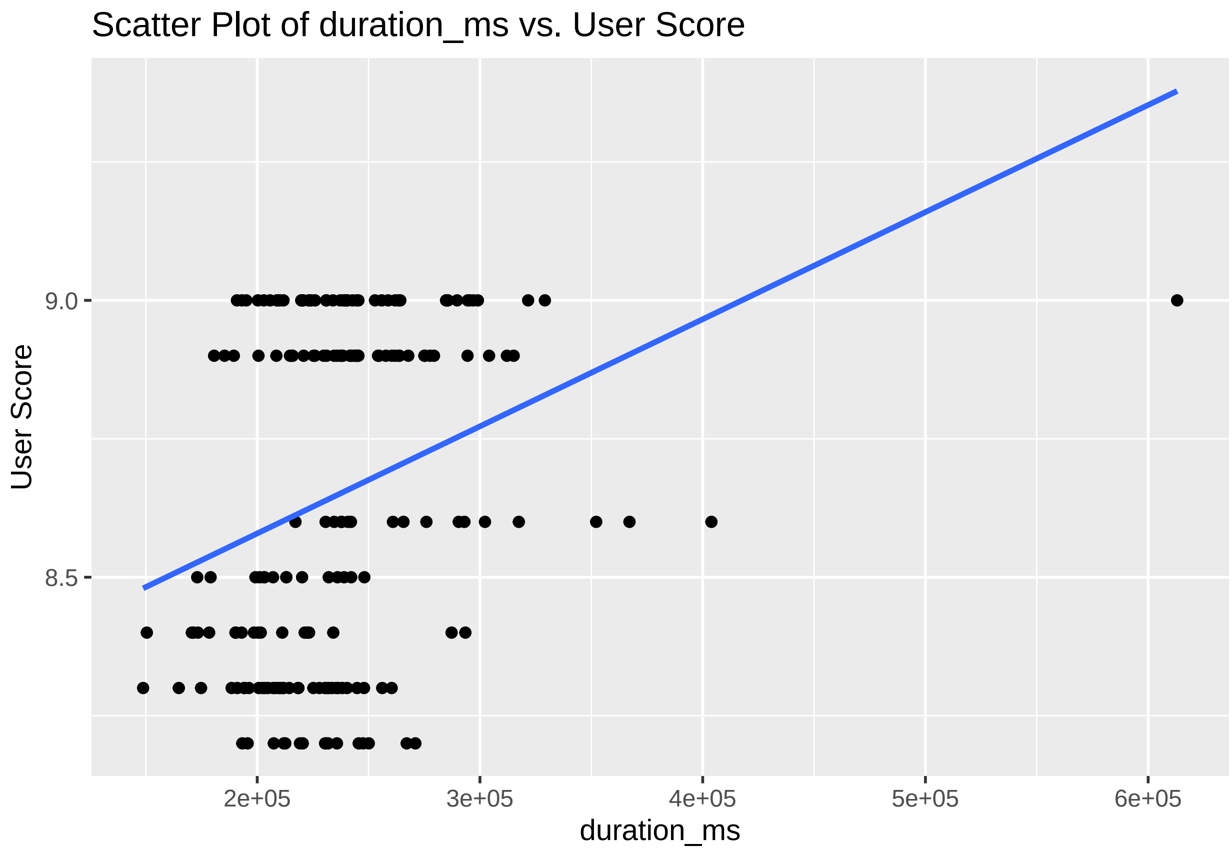
```
## liveness     -2.845e-01  2.527e-01  -1.126  0.26168
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.273 on 185 degrees of freedom
## Multiple R-squared:  0.2109, Adjusted R-squared:  0.1896
## F-statistic:  9.89 on 5 and 185 DF,  p-value: 2.14e-08
```

```r
# Create a scatter plot of metacritic_score against each predictor variable
plot_data <- selected_numerical_feats_and_user_score_df
predictor_variables <- setdiff(names(plot_data), "user_score")

for (var in predictor_variables) {
  # Create scatter plot for each predictor variable
  plot <- ggplot(plot_data, aes_string(x = var, y = "user_score")) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE) +
    labs(title = paste("Scatter Plot of", var, "vs. User Score"),
         x = var, y = "User Score")

  # Display each plot individually
  print(plot)
}
```

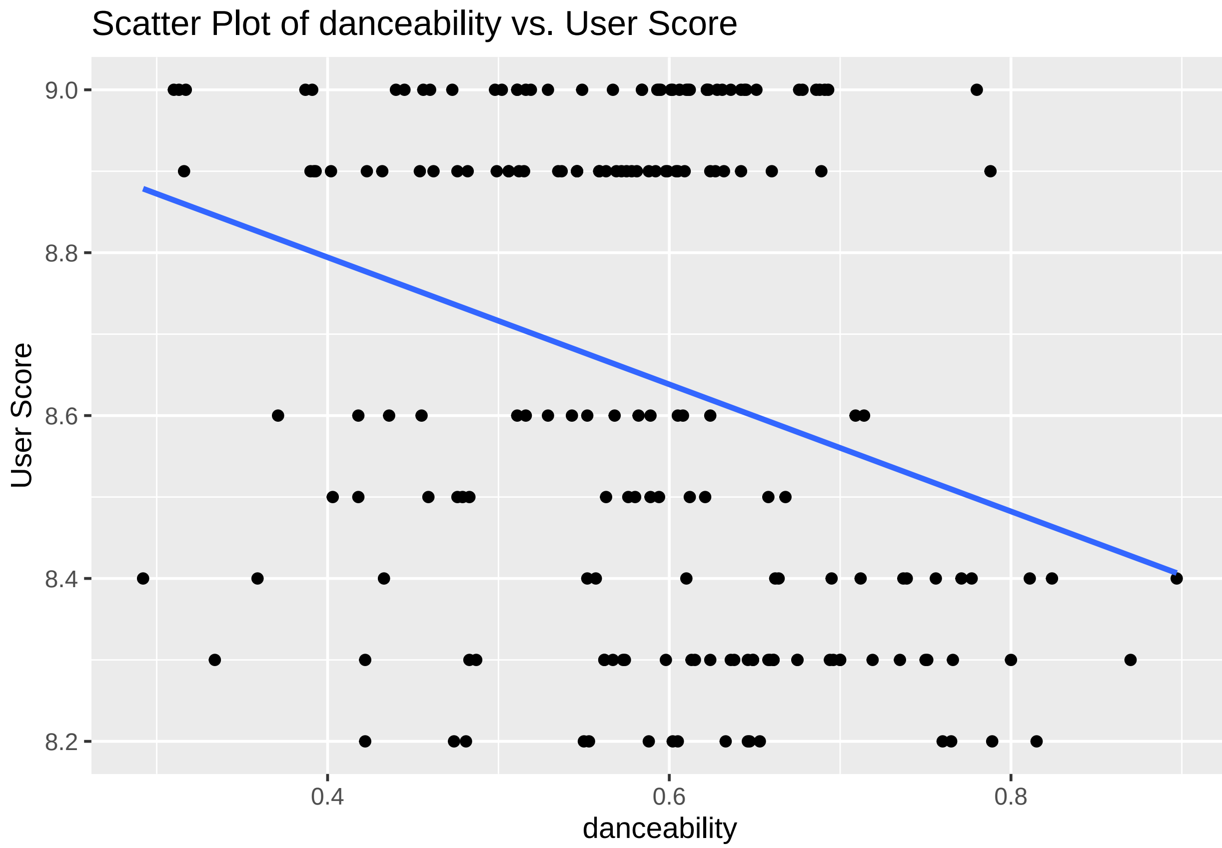
```
## `geom_smooth()` using formula = 'y ~ x'
```



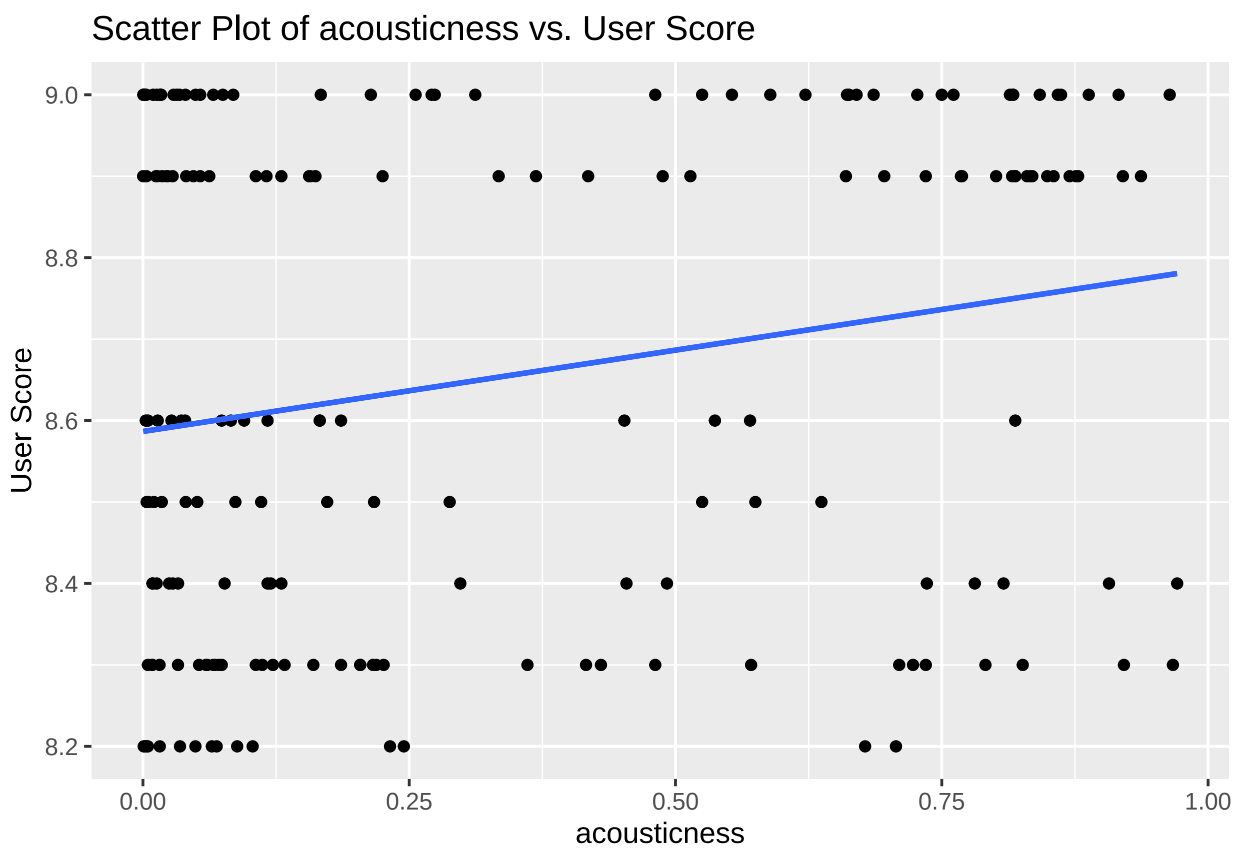
```
## `geom_smooth()` using formula = 'y ~ x'
```



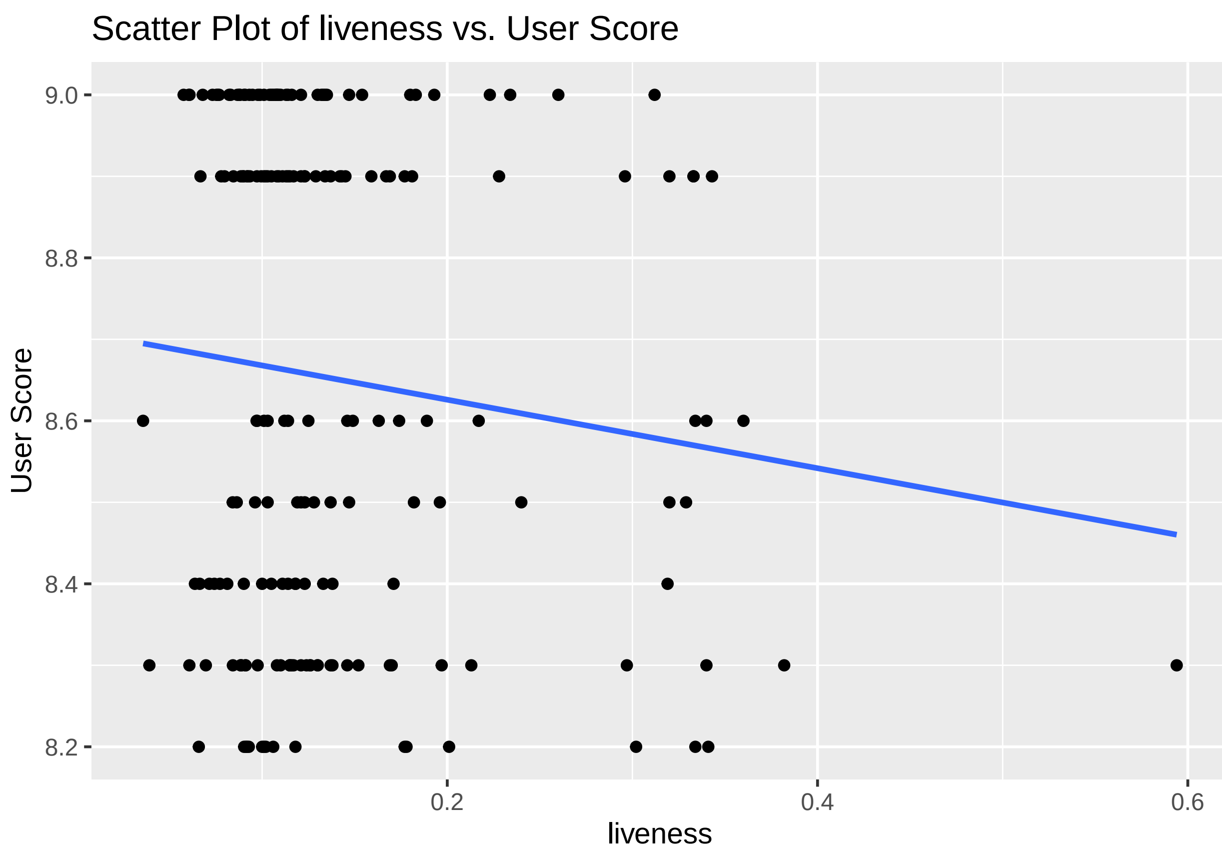

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
## `geom_smooth()` using formula = 'y ~ x'
```




```
## `geom_smooth()` using formula = 'y ~ x'
```



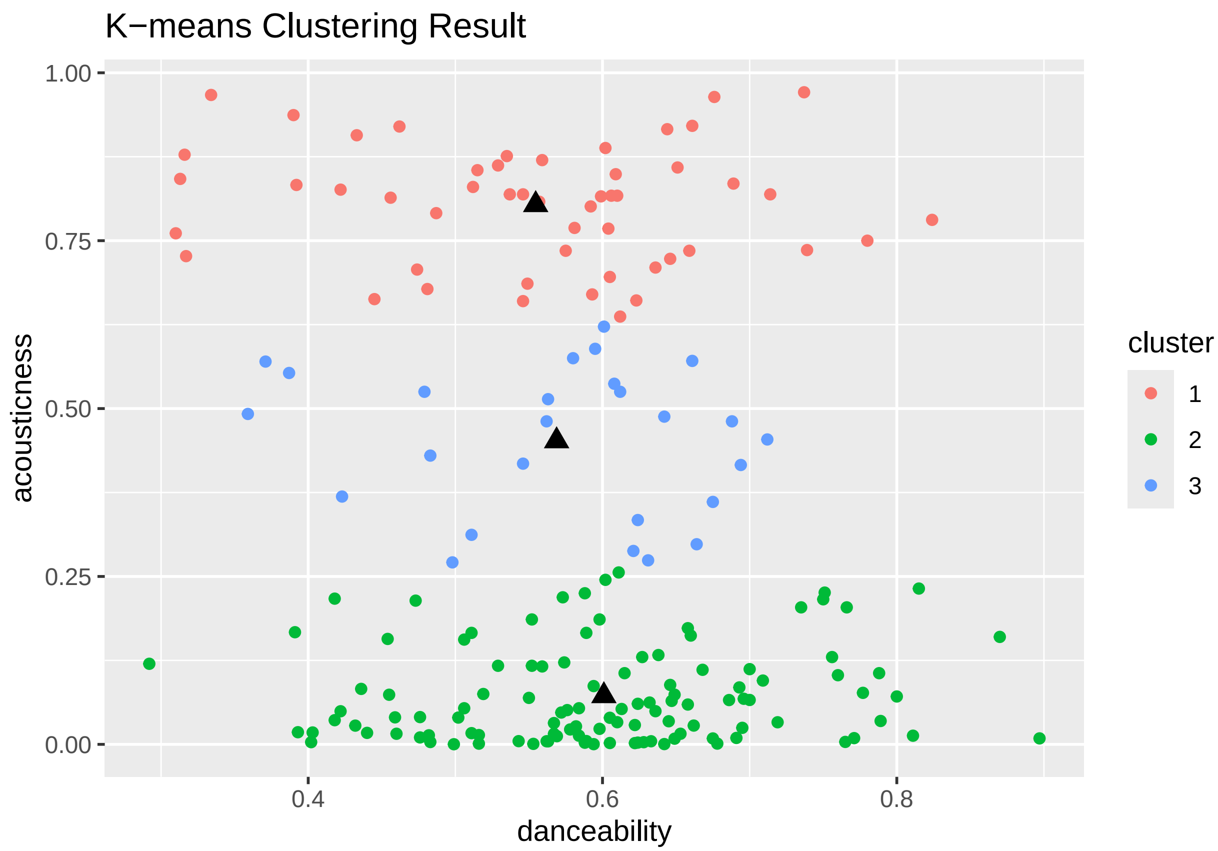
5.1.3 Comments

- The standard error is lower than that of the model predicting the metacritic score.
- There appears to be no linear relationship between the input features and user scores. It is evident that a more complex fitting method should be employed instead of a simple linear regression model.
- Acousticness and danceability can serve as decisive features in certain edge cases. When the acousticness exceeds 0.5, there is a high probability of assigning high user scores. Danceability associated with high user scores typically falls between 0.4 and 0.7.

6. Clustering

```
# Select numerical features for clustering
features <- selected_numerical_feats_and_user_score_df[, c("danceability",
                                                          "acousticness")]

# Perform K-means clustering
kmeans_result <- kmeans(features, centers = 3, nstart = 30)

# Add cluster labels to the data
selected_numerical_feats_and_user_score_df$cluster <-
  as.factor(kmeans_result$cluster)

# Plot the clustered data
```

```r
ggplot(selected_numerical_feats_and_user_score_df, aes(x = danceability,
                                                        y = acousticness,
                                                        color = cluster)) +
  geom_point() +
  geom_point(data = as.data.frame(kmeans_result$centers),
             aes(x = danceability, y = acousticness), color = "black",
             size = 3, shape = 17) +
  labs(title = "K-means Clustering Result")
```




```r
# Print cluster centers
print(kmeans_result$centers)
```

```
##   danceability acousticness
## 1    0.5545882   0.80352941
## 2    0.6009292   0.07233615
## 3    0.5688148   0.45185185
```