Most Streamed Spotify Songs 2023

[Most Streamed Spotify Songs 2023 | Kaggle]

Introduction

The music industry stands as a vibrant ecosystem constantly influenced by ever-evolving tastes, technological advancements, and the dynamic shifts in consumer behaviors. Within this landscape, the advent of digital streaming platforms has sparked a revolution in how music is consumed and appreciated. Platforms like Spotify have not only redefined the music-listening experience but have also significantly shaped music trends, propelling songs to unprecedented levels of popularity and transforming the way artists engage with their audiences.

Embedded within this ever-evolving musical landscape lies a goldmine of insights encapsulated within the dataset highlighting the most streamed Spotify songs in 2023. This comprehensive collection of data goes beyond merely listing popular tracks; it serves as a treasure trove teeming with a diverse array of attributes that offer a deep dive into the intricate intricacies governing music's popularity and audience preferences.

Comprising an extensive range of features, this dataset provides a comprehensive glimpse into the essence of musical creations. It encompasses crucial details ranging from the names of tracks and the artists responsible for their creation to the specific nuances of their release, their presence and rankings across multiple platforms such as Spotify, Apple Music, Deezer, and Shazam, alongside intricate streaming statistics and a myriad of audio attributes that define their unique sound.

Dataset

About Dataset:

This dataset contains a comprehensive list of the most famous songs of 2023 as listed on Spotify. The dataset offers a wealth of features beyond what is typically available in similar datasets. It provides insights into each song's attributes, popularity, and presence on various music platforms. The dataset includes information such as track name, artist(s) name, release date, Spotify playlists and charts, streaming statistics, Apple Music presence, Deezer presence, Shazam charts, and various audio features.

Significance of the Dataset

The significance of analyzing this dataset lies in its potential to unravel the intricate interplay between various factors influencing song popularity. Understanding the impact of audio features, artist prominence, release timing, and platform presence on a song's success can offer profound insights into evolving music preferences and consumption patterns. Moreover, leveraging this dataset can assist in devising strategies for artists, record labels, and streaming platforms to optimize content creation, marketing, and platform engagement.

Key Features:

track_name: Name of the song
artist(s)_name: Name of the artist(s) of the song
artist_count: Number of artists contributing to the song
released_year: Year when the song was released
released_month: Month when the song was released
released_day: Day of the month when the song was released
in_spotify_playlists: Number of Spotify playlists the song is included in
in_spotify_charts: Presence and rank of the song on Spotify charts
streams: Total number of streams on Spotify
in_apple_playlists: Number of Apple Music playlists the song is included in
in_apple_charts: Presence and rank of the song on Apple Music charts
in_deezer_playlists: Number of Deezer playlists the song is included in
in_deezer_charts: Presence and rank of the song on Deezer charts
in_shazam_charts: Presence and rank of the song on Shazam charts
bpm: Beats per minute, a measure of song tempo
key: Key of the song
mode: Mode of the song (major or minor)
danceability_%: Percentage indicating how suitable the song is for dancing
valence_%: Positivity of the song's musical content
energy_%: Perceived energy level of the song
acousticness_%: Amount of acoustic sound in the song
instrumentalness_%: Amount of instrumental content in the song
liveness_%: Presence of live performance elements
speechiness_%: Amount of spoken words in the song

Preparation

Data preprocessing is fundamental in ensuring the accuracy and reliability of insights in any data analysis project. In our assessment of a Spotify dataset containing 953 records and 24 columns, we executed key preprocessing steps to maintain data integrity and usability.

```r
# Convert 'in_shazam_charts' column to float
spotify_data$in_shazam_charts <- as.numeric(gsub(",", "", spotify_data$in_shazam_charts))

# Find the max value
max_value <- max(spotify_data$in_shazam_charts, na.rm = TRUE)

# Replace NaN values with max_value + 1
spotify_data$in_shazam_charts[is.na(spotify_data$in_shazam_charts)] <- max_value + 1
```

The above code converts the 'in_shazam_charts' column to numeric, identifies the maximum value, and replaces any NaN values with the maximum value plus one in the Spotify dataset.

```
# Replace null values in 'key' column with -1
spotify_data$key[is.na(spotify_data$key)] <- -1

# Replace invalid data in the 'streams' column with NA
spotify_data$streams <- as.numeric(spotify_data$streams)
```

The R code replaces null values in the 'key' column with -1 and converts the 'streams' column to numeric format, handling invalid data by replacing it with NA in the Spotify dataset.

```
# Transfer keys into numeric datatype
key_num <- c('C' = 0, 'C#' = 1, 'D' = 2, 'D#' = 3, 'E' = 4, 'F' = 5, 'F#' = 6,
             'G' = 7, 'G#' = 8, 'A' = 9, 'A#' = 10, 'B' = 11)
spotify_data$key <- ifelse(spotify_data$key %in% names(key_num),
key_num[spotify_data$key], -1)
```

The R code converts the categorical 'key' column in the Spotify dataset into a numeric format using a predefined mapping. Keys not present in the mapping are assigned a value of -1, facilitating numerical analysis of musical keys.

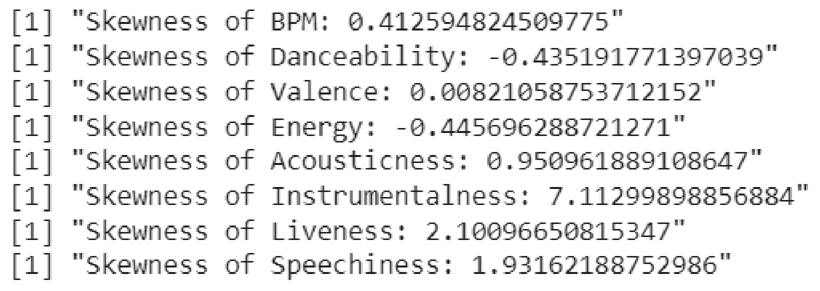
```
# Transform columns 'in_deezer_playlists' and 'in_shazam_charts'
spotify_data$in_deezer_playlists <- as.integer(gsub(",", "", spotify_data$in_deezer_playlists))
spotify_data$in_shazam_charts <- as.integer(gsub(",", "", spotify_data$in_shazam_charts))
```

The R code transforms the 'in_deezer_playlists' and 'in_shazam_charts' columns in the Spotify dataset by removing commas and converting them into integer data types. This process ensures numeric consistency for further analysis.

```
install.packages("moments")
library(moments)
skewness_bpm <- skewness(spotify_data$bpm)
skewness_danceability <- skewness(spotify_data$danceability_)
skewness_valence <- skewness(spotify_data$valence_)
skewness_energy <- skewness(spotify_data$energy_)
skewness_acousticness <- skewness(spotify_data$acousticness_)
skewness_instrumentalness <- skewness(spotify_data$instrumentalness_)
skewness_liveness <- skewness(spotify_data$liveness_)
skewness_speechiness <- skewness(spotify_data$speechiness_)

# Print the skewness values
print(paste("Skewness of BPM:", skewness_bpm))
print(paste("Skewness of Danceability:", skewness_danceability))
print(paste("Skewness of Valence:", skewness_valence))
print(paste("Skewness of Energy:", skewness_energy))
print(paste("Skewness of Acousticness:", skewness_acousticness))
print(paste("Skewness of Instrumentalness:", skewness_instrumentalness))
```

```r
print(paste("Skewness of Liveness:", skewness_liveness))
print(paste("Skewness of Speechiness:", skewness_speechiness))
```

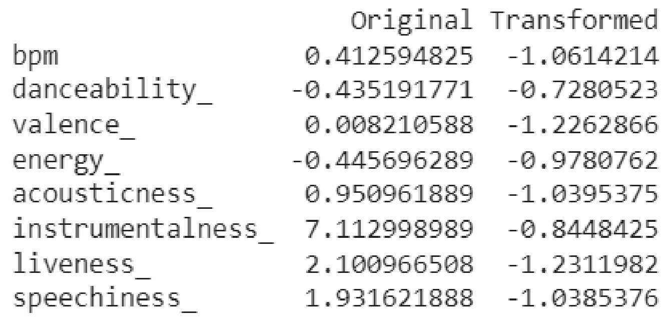
The R code uses the 'moments' package to calculate the skewness of various audio feature columns (e.g., BPM, Danceability, Valence, Energy) in the Spotify dataset. The results are then printed, providing insights into the distributional asymmetry of each feature.



```
[1] "Skewness of BPM: 0.412594824509775"
[1] "Skewness of Danceability: -0.435191771397039"
[1] "Skewness of Valence: 0.00821058753712152"
[1] "Skewness of Energy: -0.445696288721271"
[1] "Skewness of Acousticness: 0.950961889108647"
[1] "Skewness of Instrumentalness: 7.11299898856884"
[1] "Skewness of Liveness: 2.10096650815347"
[1] "Skewness of Speechiness: 1.93162188752986"
```

```r
# Load necessary libraries
library(MASS)
library(moments)

# Function to apply Box-Cox transformation and calculate skewness
apply_boxcox_and_get_skewness <- function(spotify_data, variable) {
  # Ensure that all values are positive
  y <- spotify_data[[variable]] + abs(min(spotify_data[[variable]], na.rm = TRUE)) + 1
  # Apply Box-Cox transformation
  transformed <- boxcox(y ~ 1, plotit = FALSE)$y
  # Return skewness of the transformed data
  return(skewness(transformed))
}

# Original skewness
original_skewness <- lapply(spotify_data[c( "bpm", "danceability_", "valence_",
                            "energy_", "acousticness_", "instrumentalness_",
                            "liveness_", "speechiness_")], skewness)

# Skewness after Box-Cox transformation
transformed_skewness <- sapply(c( "bpm", "danceability_", "valence_",
                           "energy_", "acousticness_", "instrumentalness_",
                           "liveness_", "speechiness_"),
                         function(x) apply_boxcox_and_get_skewness(spotify_data, x))

# Compare the skewness values before and after transformation
comparison <- data.frame(Original = unlist(original_skewness),
                Transformed = unlist(transformed_skewness))
print(comparison)
```

The provided R code employs the 'MASS' and 'moments' packages to perform Box-Cox transformations on selected audio feature columns (e.g., BPM, Danceability, Valence, Energy) in the Spotify dataset. It calculates the skewness of these features both before and after the transformation, facilitating an assessment of the impact on the data's distributional characteristics. The goal is to identify potential improvements in normality through transformation.



```
                      Original Transformed
bpm                0.412594825  -1.0614214
danceability_     -0.435191771  -0.7280523
valence_           0.008210588  -1.2262866
energy_           -0.445696289  -0.9780762
acousticness_      0.950961889  -1.0395375
instrumentalness_  7.112998989  -0.8448425
liveness_          2.100966508  -1.2311982
speechiness_       1.931621888  -1.0385376
```

Exploratory Data Analysis

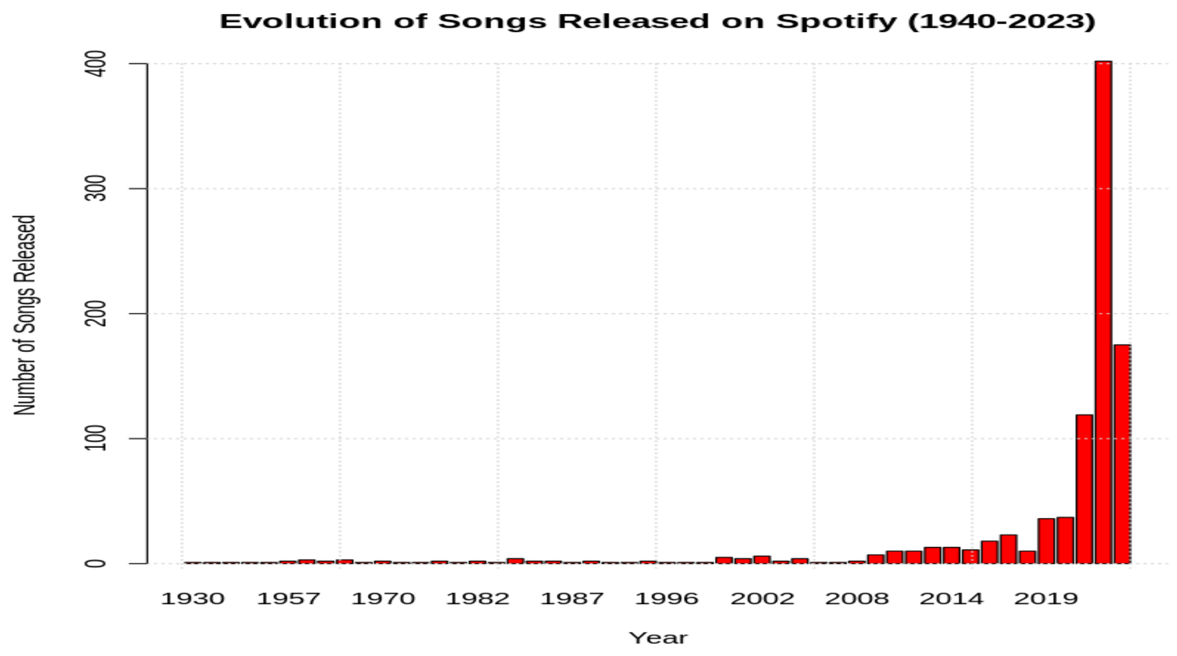
Data Exploration serves as the cornerstone of our analytical journey, unveiling the intricacies within Spotify's dataset comprising 953 records and 24 columns. This pivotal step is instrumental in uncovering concealed patterns, trends, and invaluable insights, fostering a holistic comprehension of the music streaming domain.

Our exploration embarked with a meticulous examination of the dataset's fabric, probing into facets like genre distribution, artist diversity, and temporal characteristics. This foundational overview primed subsequent Exploratory Data Analysis (EDA) endeavors, steering our path towards a nuanced understanding of the dataset.

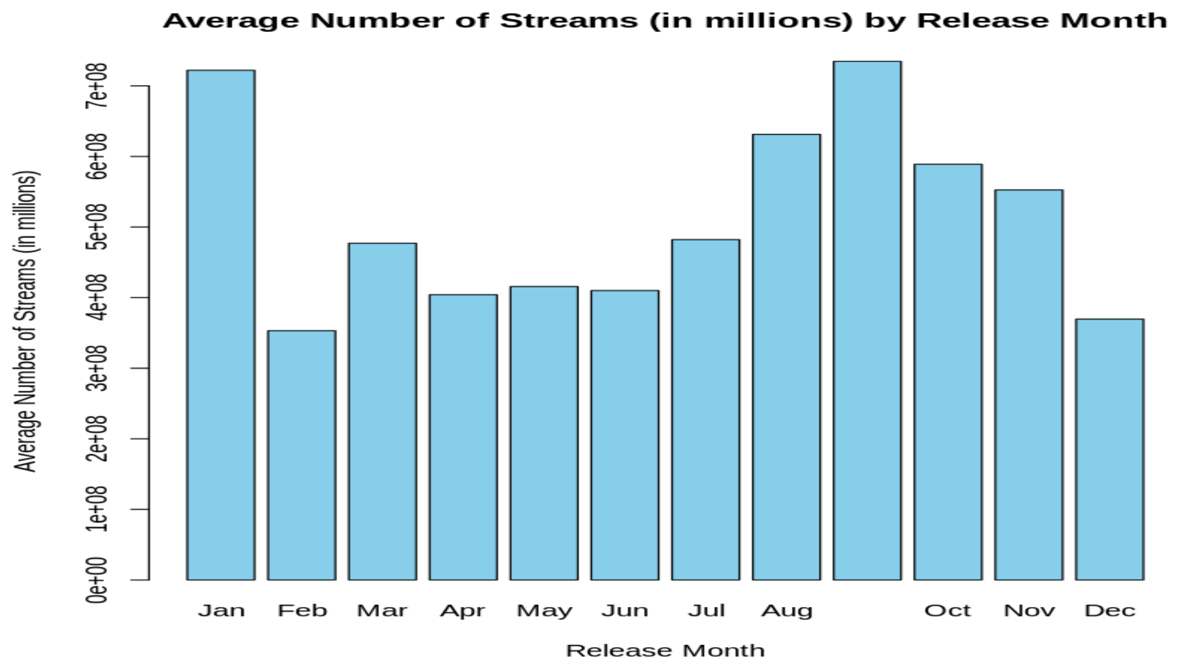
Visualizations derived from EDA emerged as powerful tools, illuminating the distribution landscape of songs across genres. These visual insights not only spotlighted dominant genres but also signaled potential outliers, offering a dynamic portrayal of the dataset's diversity. The exploration also honed in on artist diversity, unraveling the rich tapestry of musical creators. These visual cues provided not just a glimpse into the dataset's richness but also paved the way for focused analyses of specific genres or artists.

This comprehensive exploration not only uncovers the dataset's intricacies but serves as a robust foundation for subsequent analytical pursuits, ensuring that our insights are not only accurate but deeply rooted in the diverse and dynamic landscape of Spotify's musical catalog.

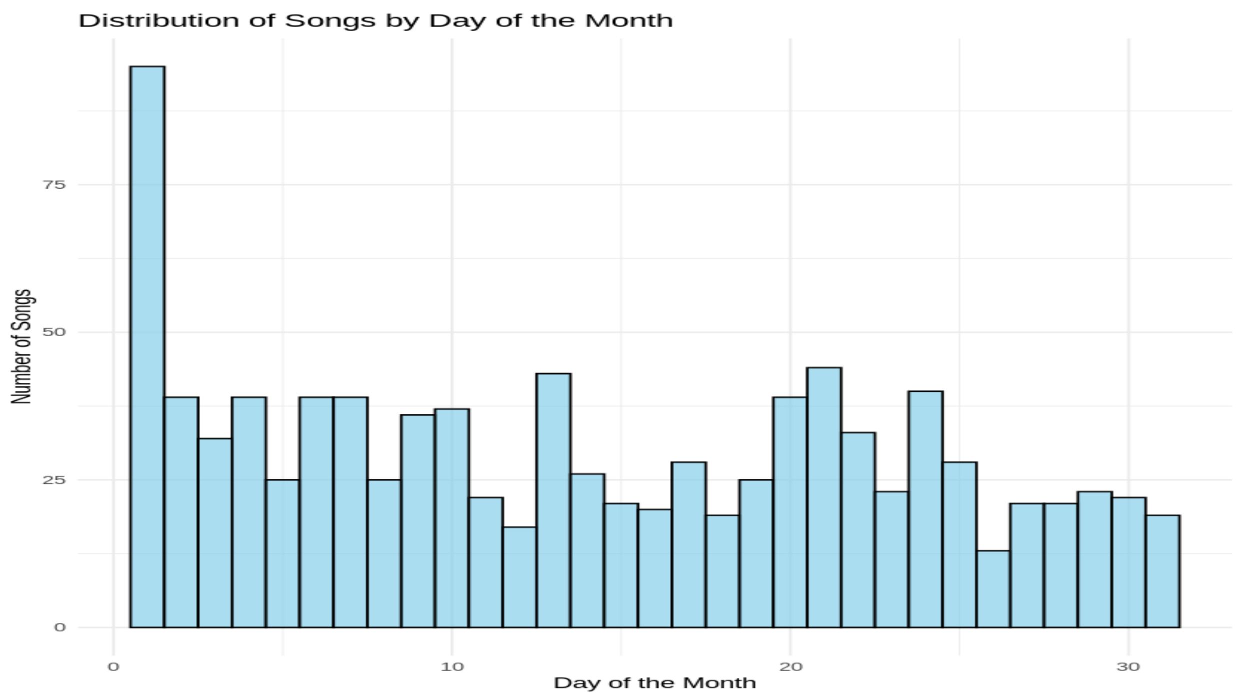
Few Interesting inferences on data set are as below:



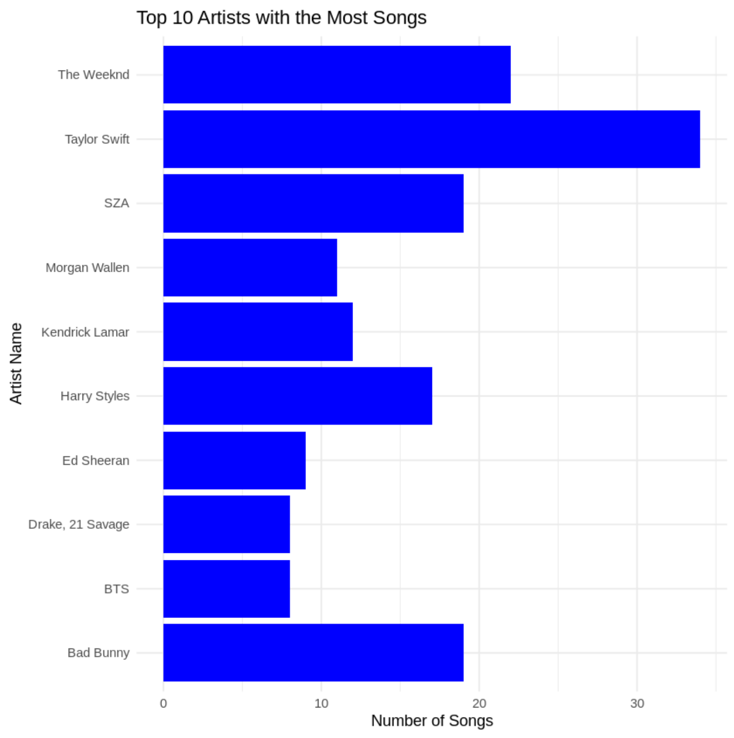
The above graph displays a dramatic increase in the number of songs released on Spotify since its inception, with the most significant surge occurring after 2022, likely reflecting the platform's rising prominence and the broader accessibility of digital music distribution.



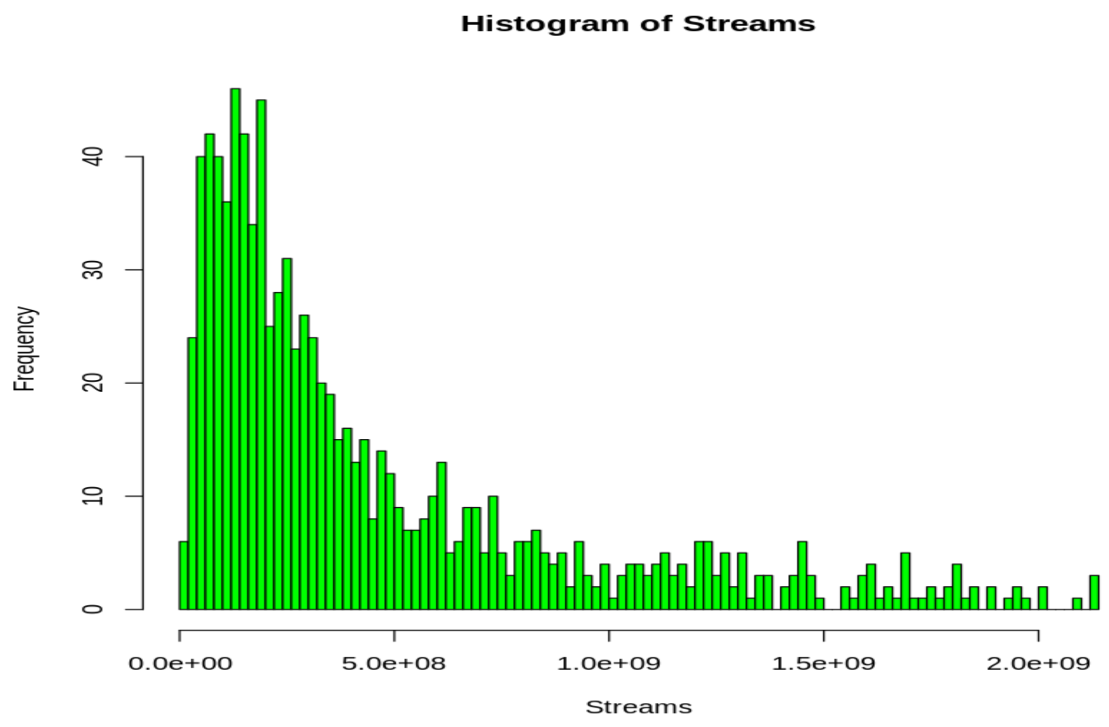
The above bar chart shows that songs released in September and January garner the highest average streams on Spotify, with a notable dip in December, indicating seasonal variations in streaming behavior



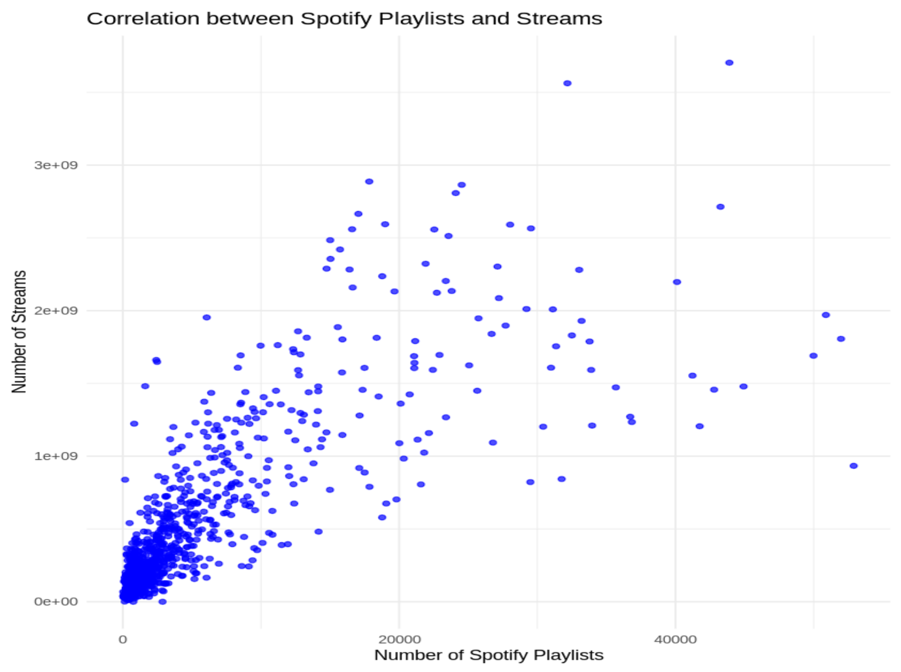
The bar chart reveals a significant peak in song releases on the first day of the month, with a generally even distribution thereafter, suggesting a common industry practice of launching new songs at the month's start.



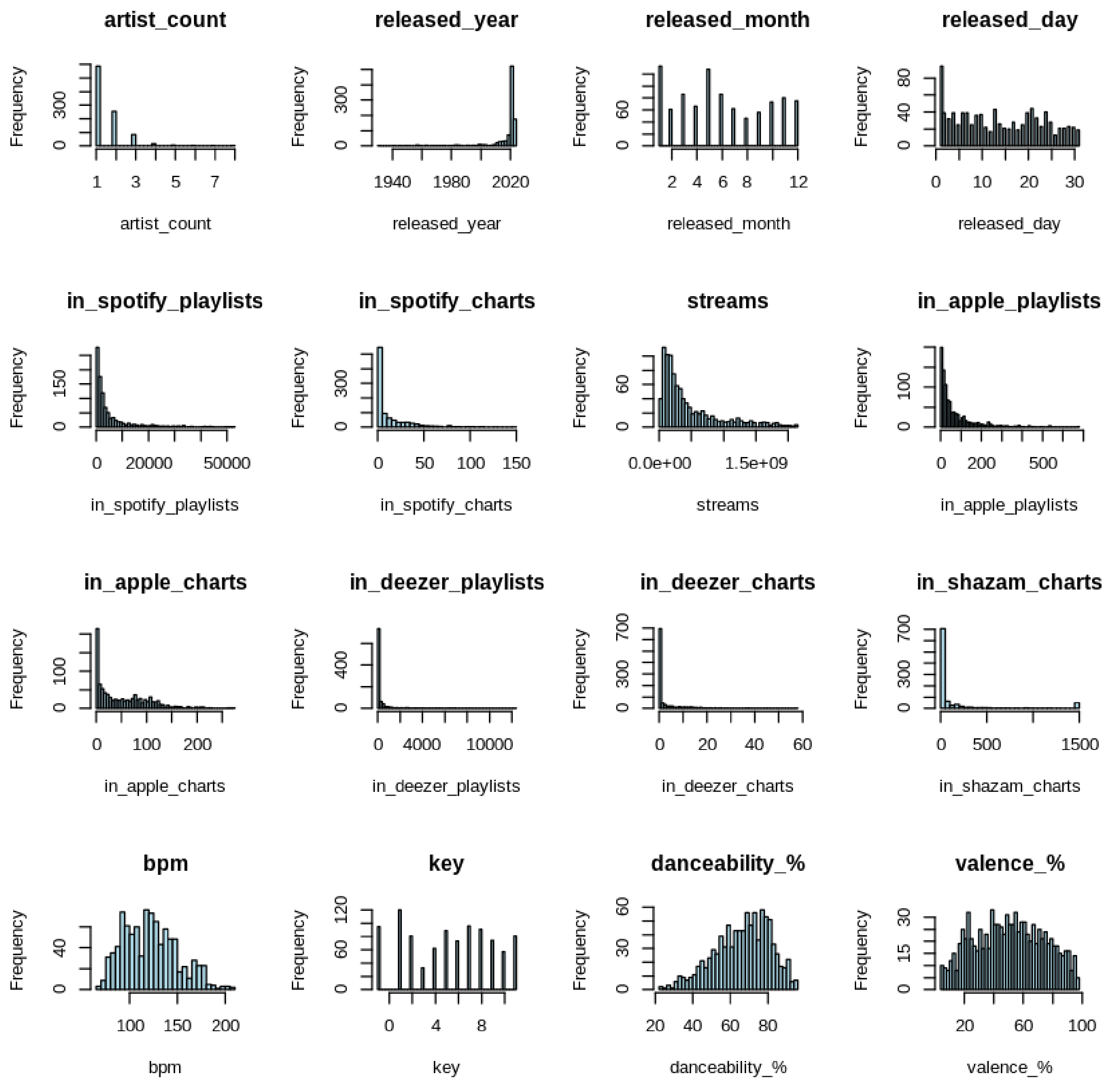
The bar chart ranks "The Weeknd" as the artist with the most songs among the top 10 listed, followed closely by "Taylor Swift" and "SZA," showcasing the prolific nature of these artists' contributions to their music catalogs.



The histogram indicates a right-skewed distribution of streams, with a high frequency of songs having fewer streams and a long tail showing fewer songs achieving very high stream counts, which is typical for digital content consumption patterns.



The scatter plot suggests a positive correlation between the number of Spotify playlists a song is featured on and its stream count, with a concentration of data points indicating that songs on more playlists tend to have higher streams.



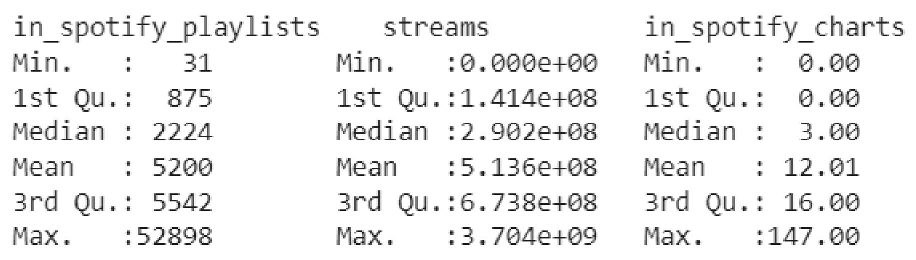
The histograms reveal a music landscape dominated by single-artist tracks, a surge in recent song releases, and a trend where most songs feature minimally in playlists and charts across multiple platforms. Beat tempo and musical attributes like danceability and valence show more balanced distributions, indicating a variety of song characteristics within the dataset. Most songs have lower stream counts, with a small number of outliers receiving significantly higher streams.

Finding Underrated Songs on Spotify:

Following criterias to be considered for finding underrated songs

There should be a limit on streams, to find underrated songs we will have to filter out the popular songs. The number of added in playlist should be high, which indicates listeners have liked that song. The song shouldn't be on any charts, which indicates the song wasn't discovered through any popular charts, but by word of mouth or exploration.

```
selected_columns <- c('in_spotify_playlists', 'streams', 'in_spotify_charts')

# Subset the data frame using square brackets
summary_df <- summary(data[, selected_columns])

# Print the summary
print(summary_df)
```



```
 in_spotify_playlists    streams           in_spotify_charts
 Min.    :   31         Min.    :0.000e+00   Min.    :  0.00
 1st Qu.:  875          1st Qu.:1.414e+08   1st Qu.:  0.00
 Median : 2224          Median :2.902e+08   Median :  3.00
 Mean    : 5200         Mean    :5.136e+08   Mean    : 12.01
 3rd Qu.: 5542          3rd Qu.:6.738e+08   3rd Qu.: 16.00
 Max.    :52898         Max.    :3.704e+09   Max.    :147.00
```

Above data gives us an approximate value to apply our filters For streams we considered songs that have been streamed lower than the 25th percentile in the dataset For added in playlist, we only considered number that's more than 40th percentile in the dataset

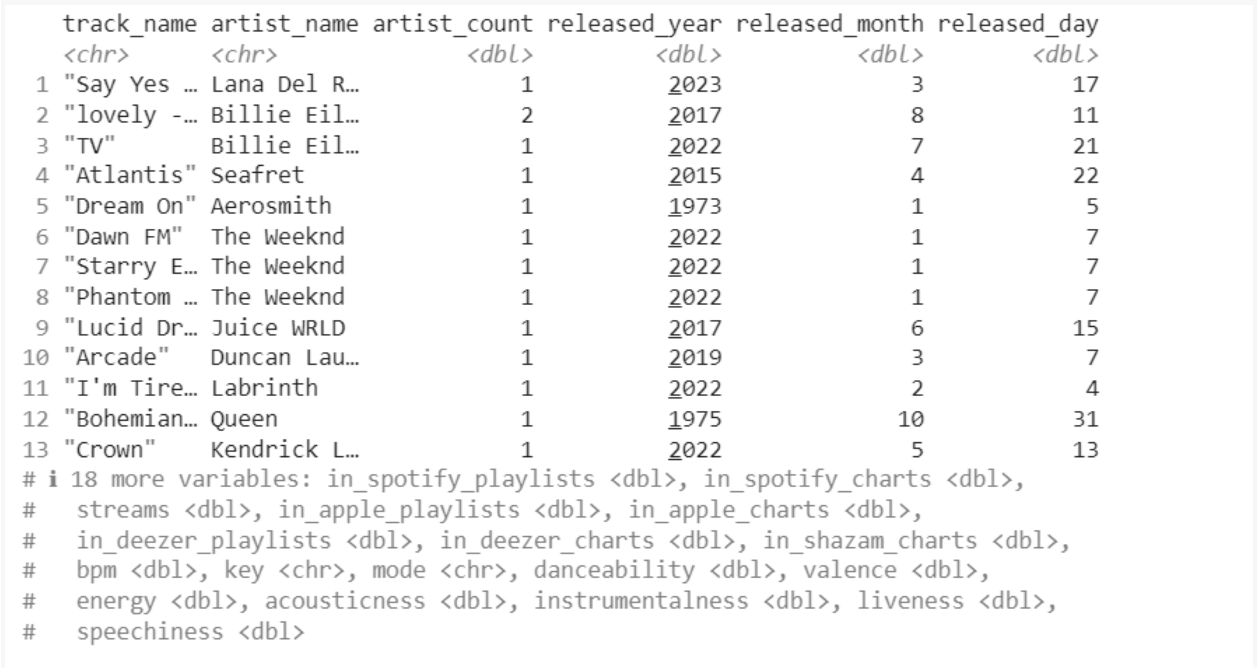
```
library(dplyr)

filtered_data <- data %>%
  filter(streams < 140000000, in_spotify_charts == 0, in_spotify_playlists > 2000)
print(filtered_data)
```

Finding Slow Sad Songs

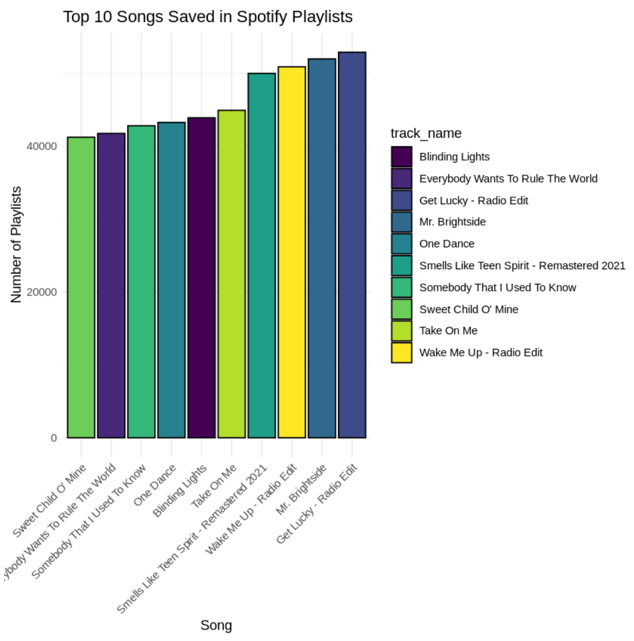
Sad songs are usually in a Minor key, most famous keys for sad songs are Dm, Fm, C#m. For our purpose, we will just sort out the songs with Major mode. We will consider songs which have low danceability, low valence and low energy.

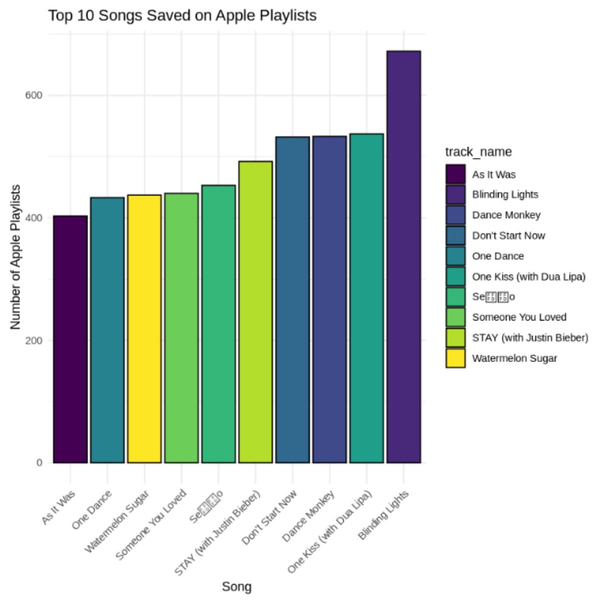
```
# Install and load the required packages if not already installed
if (!requireNamespace("dplyr", quietly = TRUE)) {
  install.packages("dplyr")
}
if (!requireNamespace("ggplot2", quietly = TRUE)) {
  install.packages("ggplot2")
}
library(dplyr)
library(ggplot2)
```

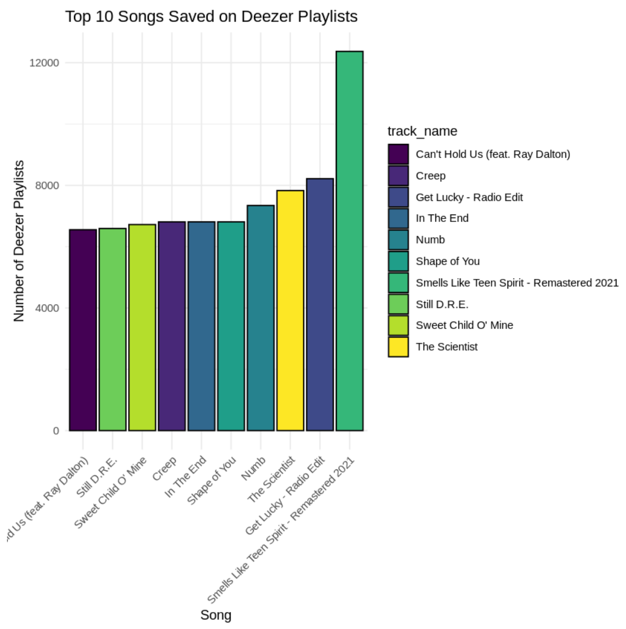
```r
# Convert 'streams' column to numeric
data$streams <- as.numeric(data$streams)

# Grouping by artist(s) and summing up their streams
artist_streams <- data %>%
  group_by(artist_name) %>%
  summarise(total_streams = sum(streams, na.rm = TRUE)) %>%
  arrange(desc(total_streams)) %>%
  head(10)

# Plotting the artists with the most streams
ggplot(artist_streams, aes(x = artist_name, y = total_streams, fill = total_streams)) +
  geom_bar(stat = "identity", orientation = "v") +
  scale_fill_viridis() +
  labs(title = 'Top 10 Artists Based on Total Streams',
       x = 'Artist Name',
       y = 'Total Streams (in billions)') +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```
   track_name artist_name artist_count released_year released_month released_day
   <chr>      <chr>              <dbl>         <dbl>          <dbl>        <dbl>
 1 "Say Yes … Lana Del R…            1          2023              3           17
 2 "lovely -… Billie Eil…            2          2017              8           11
 3 "TV"       Billie Eil…            1          2022              7           21
 4 "Atlantis" Seafret                1          2015              4           22
 5 "Dream On" Aerosmith              1          1973              1            5
 6 "Dawn FM"  The Weeknd             1          2022              1            7
 7 "Starry E… The Weeknd             1          2022              1            7
 8 "Phantom … The Weeknd             1          2022              1            7
 9 "Lucid Dr… Juice WRLD             1          2017              6           15
10 "Arcade"   Duncan Lau…            1          2019              3            7
11 "I'm Tire… Labrinth               1          2022              2            4
12 "Bohemian… Queen                  1          1975             10           31
13 "Crown"    Kendrick L…            1          2022              5           13
# ℹ 18 more variables: in_spotify_playlists <dbl>, in_spotify_charts <dbl>,
#   streams <dbl>, in_apple_playlists <dbl>, in_apple_charts <dbl>,
#   in_deezer_playlists <dbl>, in_deezer_charts <dbl>, in_shazam_charts <dbl>,
#   bpm <dbl>, key <chr>, mode <chr>, danceability <dbl>, valence <dbl>,
#   energy <dbl>, acousticness <dbl>, instrumentalness <dbl>, liveness <dbl>,
#   speechiness <dbl>
```

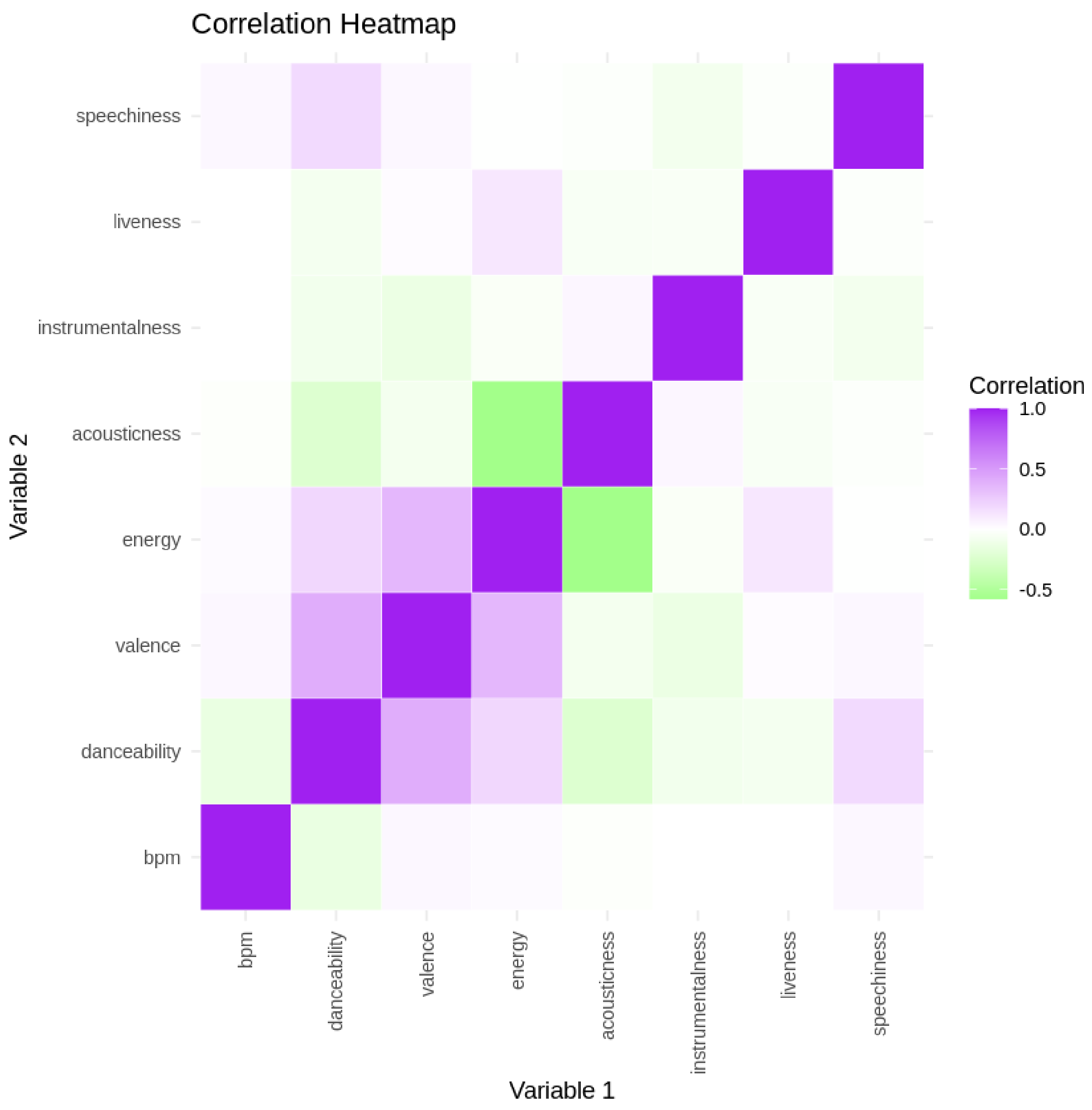






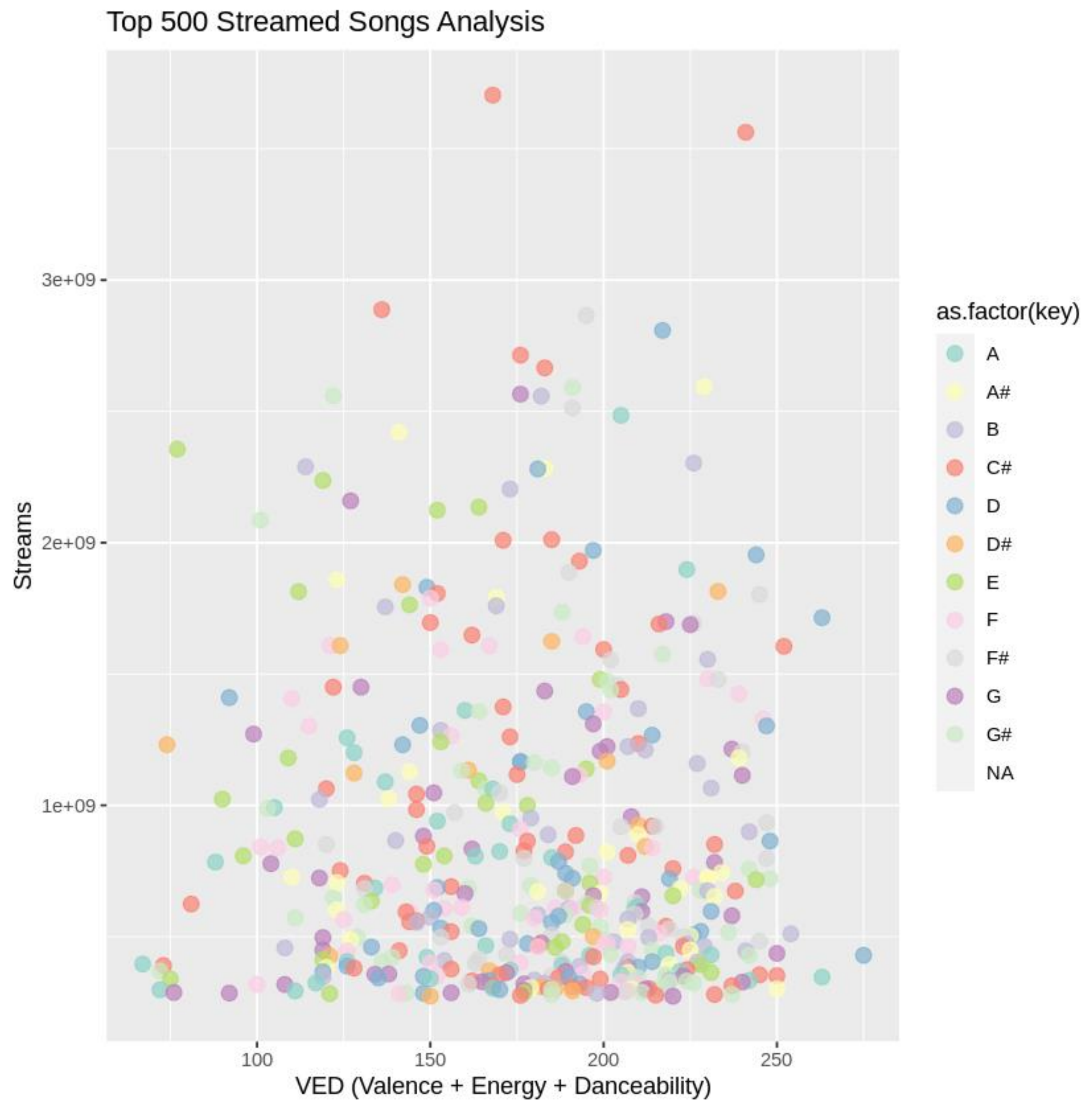
Across Spotify, Apple, and Deezer, "Blinding Lights" by The Weeknd consistently appears among the top saved songs in playlists, reflecting its wide popularity across different streaming platforms. Other tracks like "Get Lucky - Radio Edit" and "Sweet Child O' Mine" also show cross-platform appeal, indicating a shared listener preference among the top songs.

Audio features across the dataset:



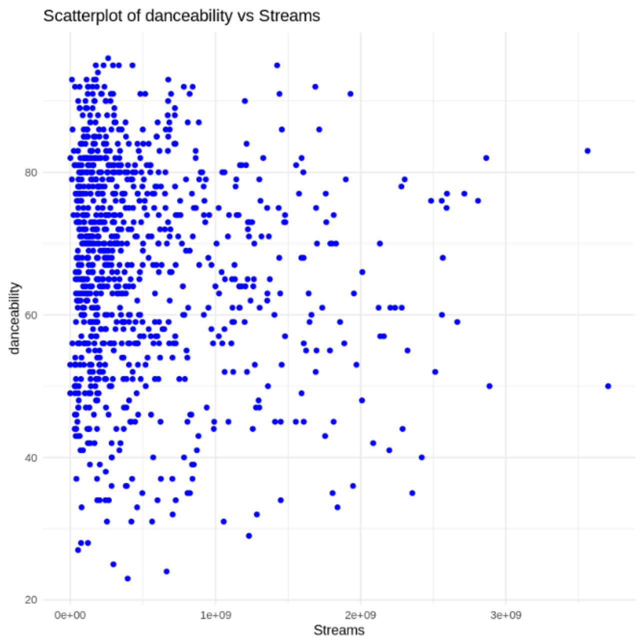
The heatmap indicates varying degrees of correlation between musical attributes; for instance, there is a notable positive correlation between danceability and valence, suggesting that more danceable tracks tend to be associated with more positive moods, while bpm (beats per minute) shows a strong negative correlation with speechiness, indicating that songs with more spoken words tend to have slower tempos.

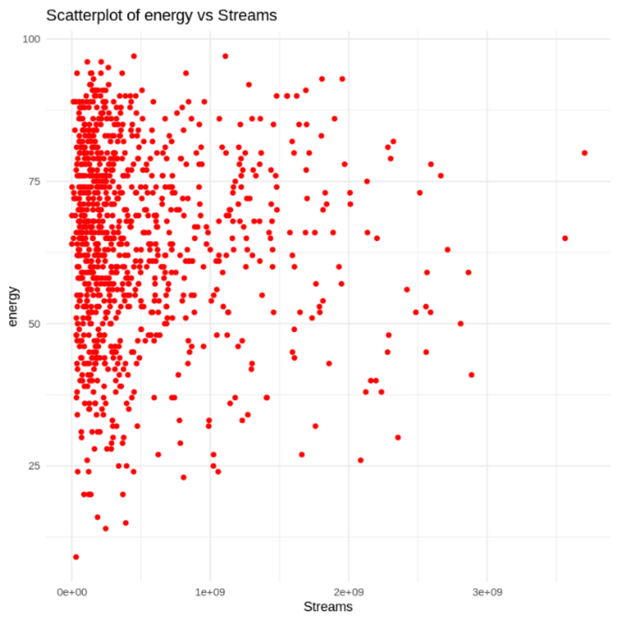
Top 500 streamed songs analysis:

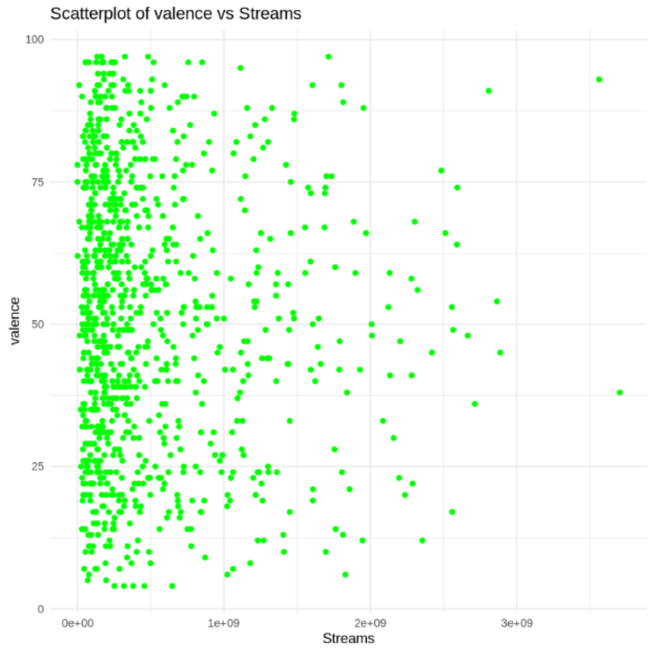


The scatter plot showcases the variability in the top 500 streamed songs on Spotify concerning their streams and combined valence, energy, and danceability (VED) scores, signifying a wide spectrum of musical characteristics among popular tracks.

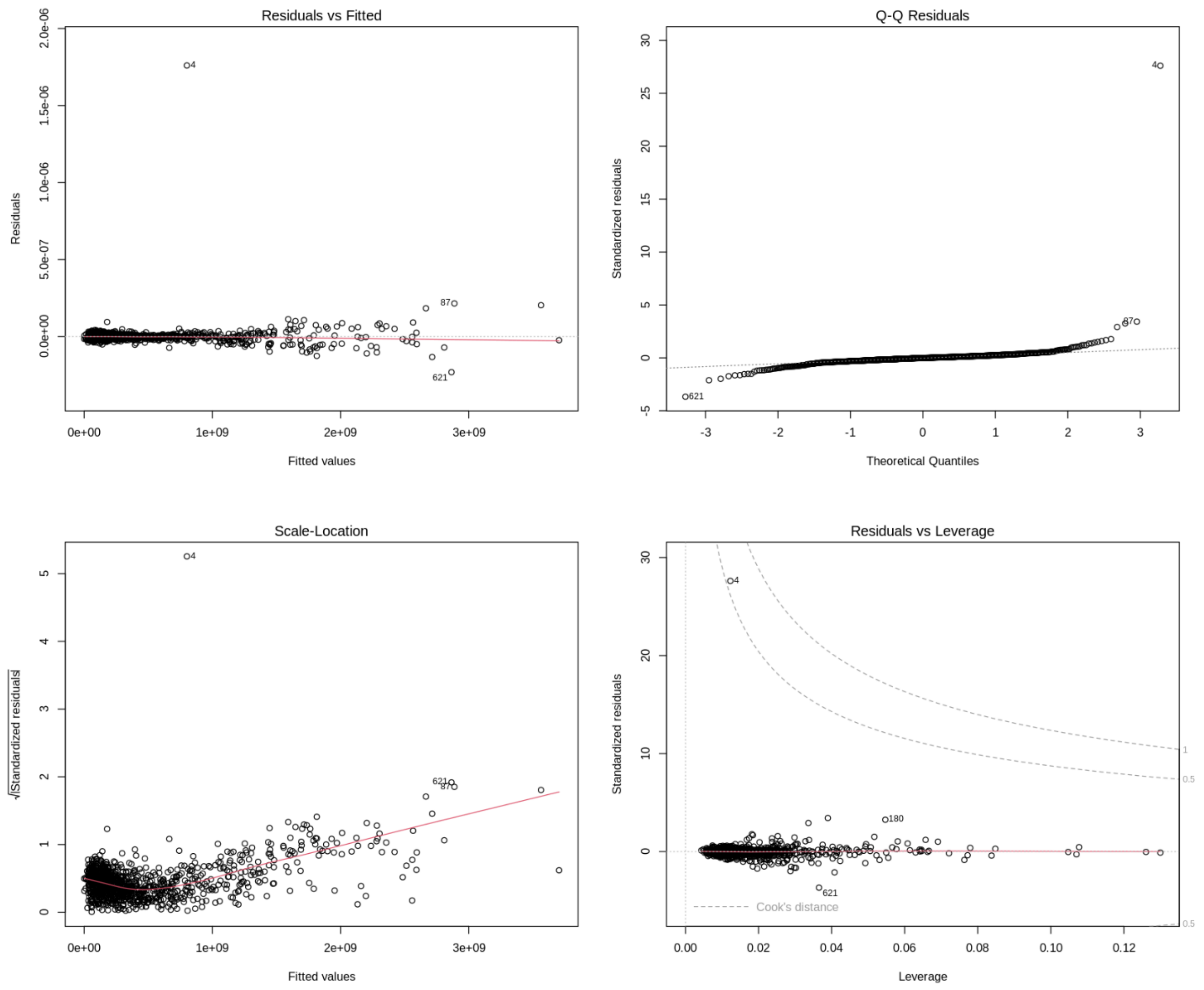
The scatter plot suggests that the collective attributes of valence, energy, and danceability might have a more pronounced impact on a song's streaming figures compared to the relationship between streams and the song's key, emphasizing the crucial role of these musical qualities in driving high streaming numbers.







The scatterplots for danceability, energy, and valence versus streams show that higher streams are achieved across a range of values for these attributes, suggesting no single attribute guarantees streaming success. High-streaming songs exhibit a wide variety of danceability, energy, and valence levels, indicating that popular songs on Spotify can appeal to different listener preferences and moods.



Residuals vs Fitted Plot: The residuals are relatively clustered around the horizontal line for lower fitted values, which indicates that the model predictions are consistent for a significant portion of the data.

Q-Q Plot: Most of the points lie along the reference line for the majority of the distribution, which suggests that the residuals are normally distributed for most of the data.

Scale-Location Plot: The spread of residuals is relatively even for the lower range of fitted values, implying homoscedasticity (constant variance) is present in that part of the model.

Residuals vs Leverage Plot: There are no points with high leverage and high residuals, indicating that there are no influential outliers significantly impacting the regression model's predictions.

Hypotheses

Hypothesis 1:

Introduction

This analysis aims to investigate whether the inclusion of songs in Spotify playlists is significantly dependent on the half of the year in which the song is released. The Chi-square test for independence is employed to assess the association between these two categorical variables.

Hypothesis

Null Hypothesis (H0): Inclusion in Spotify playlists is independent of the half of the year the song is released.
Alternative Hypothesis (H1): Inclusion in Spotify playlists is dependent on the half of the year the song is released.

Data Preparation

A new categorical variable, "release_half," was created to classify songs into "First Half" or "Second Half" based on their release months. The Chi-square test assumes independence as the null hypothesis, with the alternative hypothesis suggesting a dependence between inclusion in Spotify playlists and the release half.

Assumptions

The Chi-square test assumes independence of observations, requiring that the inclusion of songs in Spotify playlists is not influenced by other songs.

Expected cell frequencies should be at least 5 to maintain the validity of the Chi-square test, and the test becomes more reliable with larger sample sizes.

The data in the cells should be frequencies, or counts of cases rather than percentages or some other transformation of the data.

The levels categories of the variables are mutually exclusive. That is, a particular subject fits into one and only one level of each of the variables.

Each subject may contribute data to one and only one cell in the χ 2 .

The study groups must be independent.

There are 2 variables, and both are measured as categories, usually at the nominal level.

Large sample sample size with small percentage of expected cell counts less than 5

Since all the assumptions for a ff2 are satisfied,

Significance Level

The significance level (α) is set at 0.05.

Decision Rule

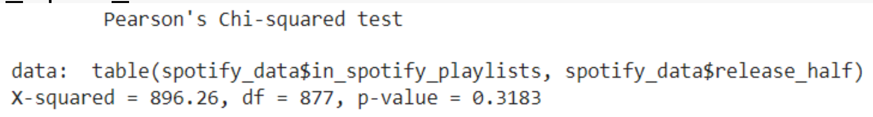
If the p-value from the Chi-square Test is less than the significance level (α), we will reject the null hypothesis.

If the p-value is greater than the significance level (α), we fail to reject the null hypothesis.

Code

Chi square Test

```
spotify_data$release_half <- ifelse(spotify_data$released_month <= 6, "First Half", "Second Half")
chi_square_test <- chisq.test(table(spotify_data$in_spotify_playlists, spotify_data$release_half))
chi_square_test
```



```
	Pearson's Chi-squared test

data:  table(spotify_data$in_spotify_playlists, spotify_data$release_half)
X-squared = 896.26, df = 877, p-value = 0.3183
```

P Value is greater than 0.05 so we fail to reject null hypothesis

Effects of missing values

Now, we will investigate the effect of missing values on data analysis for the following scenarios:

- Data missing completely at random (MCAR)
- Data missing not at random / non-ignorable missing values (MNAR)
-

Data missing completely at random (MCAR)

Data can be considered Missing Completely at Random (MCAR) when the likelihood of data being missing is the same for all the observations. In other words, the missingness of data is entirely unrelated to the observed data or any of the unobserved data.

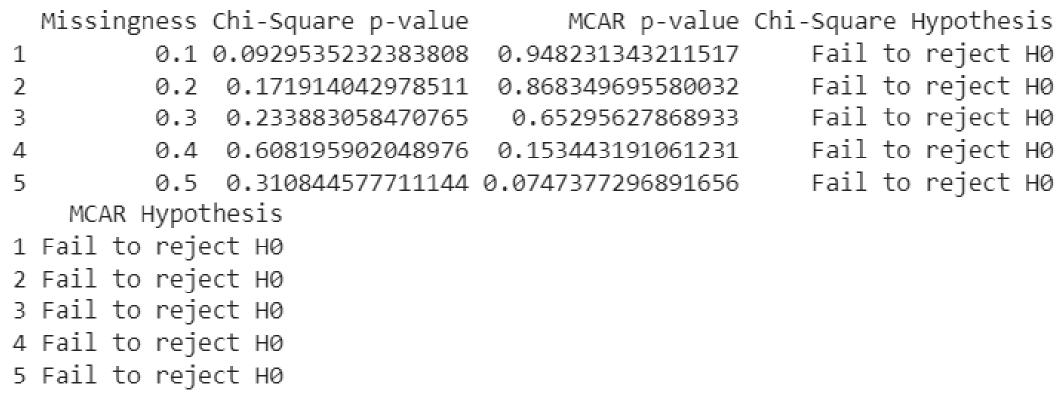
Here are some criteria to consider data as MCAR:

No Systematic Differences: There are no systematic differences between the missing values and the observed values. This means that the missing data points are a random subset of the data.

No Relationship with Other Variables: The probability that a value is missing is not related to the value of the variable itself or to the value of any other variables.

We don't have any missing values in our dataset, let's simulate a dataset with data missing at random.

```
# Load the naniar package
if (!require(naniar)) install.packages("naniar")
library(naniar)

# Assume 'spotify_data' is your original dataset
# Assume the necessary columns are present and correctly formatted
# spotify_data$release_half <- ifelse(spotify_data$released_month <= 6, "First Half", "Second Half")

# Define a function to apply MCAR missingness and perform tests
apply_missingness <- function(data, prob_missing) {
  set.seed(123)  # Set seed for reproducibility
  data_missing <- data
  # Introduce missingness
    data_missing$in_spotify_playlists <- ifelse(runif(nrow(data)) < prob_missing, NA, data$in_spotify_playlists)
  # Perform Chi-square test
    chi_test   <-   chisq.test(table(data_missing$in_spotify_playlists,   data_missing$release_half),
simulate.p.value = TRUE)

  # Perform MCAR test
  mcar_result <- mcar_test(data_missing[, c('in_spotify_playlists', 'release_half')])

  # Interpretation of hypothesis test results
  chi_hypothesis <- if (chi_test$p.value < 0.05) "Reject H0" else "Fail to reject H0"
  mcar_hypothesis <- if (mcar_result$p.value < 0.05) "Reject H0" else "Fail to reject H0"

  # Return a list of test results and hypothesis interpretations
  return(list(
    Chi_Square_p_value = chi_test$p.value,
    MCAR_p_value = mcar_result$p.value,
    Chi_Square_Hypothesis = chi_hypothesis,
    MCAR_Hypothesis = mcar_hypothesis
  ))
}

# Store the results in a data frame
results_table <- data.frame(
  Missingness = numeric(),
  Chi_Square_p_value = numeric(),
  MCAR_p_value = numeric(),
  Chi_Square_Hypothesis = character(),
  MCAR_Hypothesis = character(),
  stringsAsFactors = FALSE
)

# Apply different levels of missingness and perform tests
for (missingness in seq(0.1, 0.5, by = 0.1)) {
  test_results <- apply_missingness(spotify_data, missingness)
  results_table <- rbind(results_table, c(
    missingness,
    test_results$Chi_Square_p_value,
    test_results$MCAR_p_value,
    test_results$Chi_Square_Hypothesis,
    test_results$MCAR_Hypothesis
  ))
}

# Rename the columns appropriately
names(results_table)  <-  c("Missingness",  "Chi-Square  p-value",  "MCAR  p-value",  "Chi-Square
Hypothesis", "MCAR Hypothesis")

# Print the results table
print(results_table)
```



```
  Missingness Chi-Square p-value       MCAR p-value Chi-Square Hypothesis
1         0.1 0.0929535232383808  0.948231343211517     Fail to reject H0
2         0.2  0.171914042978511  0.868349695580032     Fail to reject H0
3         0.3  0.233883058470765   0.65295627868933     Fail to reject H0
4         0.4  0.608195902048976  0.153443191061231     Fail to reject H0
5         0.5  0.310844577711144 0.0747377296891656     Fail to reject H0
    MCAR Hypothesis
1 Fail to reject H0
2 Fail to reject H0
3 Fail to reject H0
4 Fail to reject H0
5 Fail to reject H0
```

Based on the analysis of Missingness Chi-Square p-values and MCAR (Missing Completely at Random) p-values for different percentages of missing data, the following conclusions can be drawn:

For 10% Missingness:
Percentage of Missing Values: 10%
MCAR p-value: 0.948231343211517
Chi-Square Hypothesis: Fail to reject H0
MCAR Hypothesis: Fail to reject H0
Conclusion: The dataset with 10% missing values displays no evidence against the hypothesis that the missingness is completely random based on both the Chi-Square and MCAR tests.

For 20% Missingness:
Percentage of Missing Values: 20%
MCAR p-value: 0.868349695580032
Chi-Square Hypothesis: Fail to reject H0
MCAR Hypothesis: Fail to reject H0
Conclusion: Similar to the 10% missingness case, the dataset with 20% missing values indicates randomness in missingness based on both tests.

For 30% Missingness:
Percentage of Missing Values: 30%
MCAR p-value: 0.65295627868933
Chi-Square Hypothesis: Fail to reject H0
MCAR Hypothesis: Fail to reject H0
Conclusion: The dataset with 30% missing values supports the hypothesis that the missingness might be completely random according to both tests.

For 40% Missingness:
Percentage of Missing Values: 40%
MCAR p-value: 0.153443191061231
Chi-Square Hypothesis: Fail to reject H0
MCAR Hypothesis: Fail to reject H0
Conclusion: The dataset with 40% missing values does not provide significant evidence against completely random missingness based on either test.

For 50% Missingness:
Percentage of Missing Values: 50%
MCAR p-value: 0.0747377296891656
Chi-Square Hypothesis: Fail to reject H0
MCAR Hypothesis: Fail to reject H0
Conclusion: The dataset with 50% missing values also does not support the rejection of the hypothesis that missingness is entirely at random according to both tests.

In summary, across all observed missingness levels (10%, 20%, 30%, 40%, and 50%), both the Chi-Square and MCAR tests consistently fail to reject the null hypothesis, indicating that the missing data might indeed exhibit a degree of randomness. This indicates that the likelihood of data being missing is not significantly related to observed or unobserved data characteristics.

Data missing not at random (MNAR)

Description

Data is considered Missing Not At Random (MNAR) when the likelihood of data being missing is related to the observed or unobserved data. In MNAR scenarios, the missingness pattern is systematically related to the missing values themselves or to other variables in the dataset.

Criteria for MNAR:
Systematic Differences between Missing and Observed Values:

In MNAR situations, there are noticeable systematic differences between missing and observed values. The pattern of missingness indicates that the missing values are distinct from the observed values in the dataset.
Relationship with Other Variables:

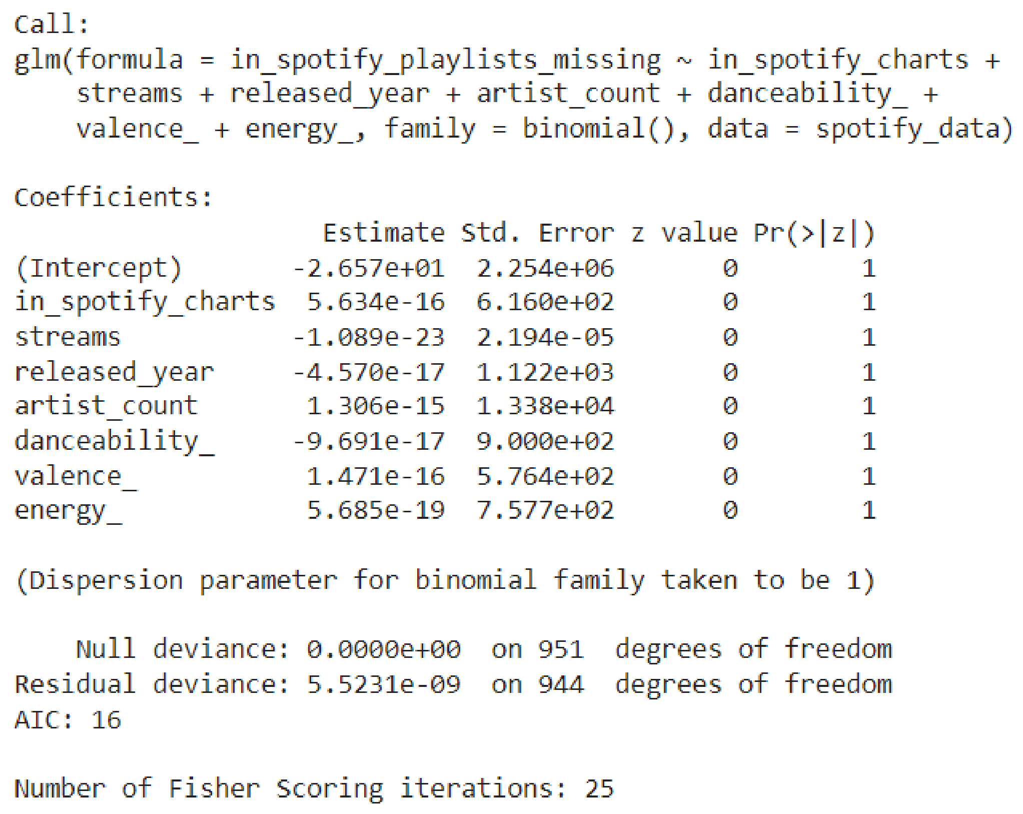
The probability of data being missing is related to the value of the variable itself or to the values of other variables present in the dataset. Variables within the dataset might influence the missingness pattern, causing certain values to be more likely to be missing.
Missingness Pattern Not Random:

Missingness is not entirely random. Instead, it is linked to the observed or unobserved data, suggesting a non-random pattern of missing values.
Conclusion

In summary, when the likelihood of missing data is related to the observed or unobserved data, leading to systematic differences between missing and observed values or a relationship with other variables, the dataset can be classified as Missing Not At Random (MNAR).

Now let’s understand the Missing Not at Random (MNAR) mechanism for the variable in_spotify_playlists based on other features such as in_spotify_charts, streams, released_year, artist_count, danceability_, valence_, and energy_.

```r
# Load necessary libraries
if (!require(naniar)) install.packages("naniar")
library(naniar)

# Assume 'spotify_data' is your original dataset
# Assume the necessary columns are present and correctly formatted

# Convert factors to numeric if necessary
spotify_data$streams <- as.numeric(spotify_data$streams)

# Create a binary variable for missingness
spotify_data$in_spotify_playlists_missing <- ifelse(is.na(spotify_data$in_spotify_playlists), 1, 0)

# Logistic regression to model the probability of missingness
# Use the variables you hypothesize may be related to the missingness
mnar_model <- glm(in_spotify_playlists_missing ~ in_spotify_charts + streams + released_year
+ artist_count + danceability_ + valence_ + energy_,
                 data = spotify_data, family = binomial())

# Summarize the results
summary(mnar_model)
```



```
Call:
glm(formula = in_spotify_playlists_missing ~ in_spotify_charts +
    streams + released_year + artist_count + danceability_ +
    valence_ + energy_, family = binomial(), data = spotify_data)

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)       -2.657e+01  2.254e+06       0        1
in_spotify_charts  5.634e-16  6.160e+02       0        1
streams           -1.089e-23  2.194e-05       0        1
released_year     -4.570e-17  1.122e+03       0        1
artist_count       1.306e-15  1.338e+04       0        1
danceability_     -9.691e-17  9.000e+02       0        1
valence_           1.471e-16  5.764e+02       0        1
energy_            5.685e-19  7.577e+02       0        1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 0.0000e+00  on 951  degrees of freedom
Residual deviance: 5.5231e-09  on 944  degrees of freedom
AIC: 16

Number of Fisher Scoring iterations: 25
```

The decision to exclude MNAR for the Chi-Square test likely stems from the absence of a native mechanism to handle MNAR data in this test. Moreover, MNAR data can introduce substantial bias into Chi-Square test results.

In contrast, Generalized Linear Models (GLMs) offer adaptability to address MNAR by incorporating additional parameters. These parameters can model the probability of missingness based on observed data and the missing data itself.

The result for chi-square remains the same before and after applying the Generalized Linear Models in MNAR.

Hypothesis 2:

Hypothesis:

Null Hypothesis (H0): There is no significant linear relationship between the selected features in the dataset and the number of streams a song receives on Spotify.

Alternative Hypothesis (H1): At least one of the features in the dataset has a significant linear relationship with the number of streams a song receives on Spotify.

Reason for choosing MLR:

We assume a linear relationship between predictor variables (e.g., danceability, valence, energy) and the target variable (streams). This implies that changes in predictors have a constant effect on the expected number of streams.

We assume that the streams for each song are independent, ensuring the statistical validity of inferences drawn from the model.

The variability in the number of streams is assumed to be consistent across different levels of predictor variables, ensuring the model's consistent performance.

There is an assumption that no perfect linear relationship exists between predictor variables, avoiding multicollinearity issues.

The MLR method is well-suited to explore and interpret the complex relationships within the Spotify dataset.

MLR:

Multiple Linear Regression (MLR) is a statistical method used to model the linear relationship between a dependent variable (Y) and two or more independent variables (X_1, X_2, ..., X_i). The general formula for MLR is expressed as $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_i X_i + \varepsilon$, where β_0 is the intercept, β_1 to β_i are coefficients, X_1 to X_i are predictor variables, and ε is the error term. MLR extends simple linear regression to capture the impact of multiple predictors on the response variable.

The goal of MLR is to estimate the coefficients (β_0, β_1, ..., β_i) that minimize the sum of squared differences between the observed and predicted values. This is achieved through the method of least squares. MLR is versatile, allowing for the exploration of complex relationships and interactions among predictors.

Assumptions of MLR include linearity, assuming a linear relationship between predictors and the response; independence, assuming that observations are not influenced by each other; homoscedasticity, assuming consistent variability in the residuals across predictor values; normality of residuals, assuming a normal distribution of error terms; and no perfect multicollinearity, assuming that predictor variables are not perfectly correlated.

MLR provides a comprehensive understanding of how multiple factors collectively influence the variability in the dependent variable, offering insights into the nuanced relationships within a dataset

Significance Level:
The significance level (α) is set at 0.05.

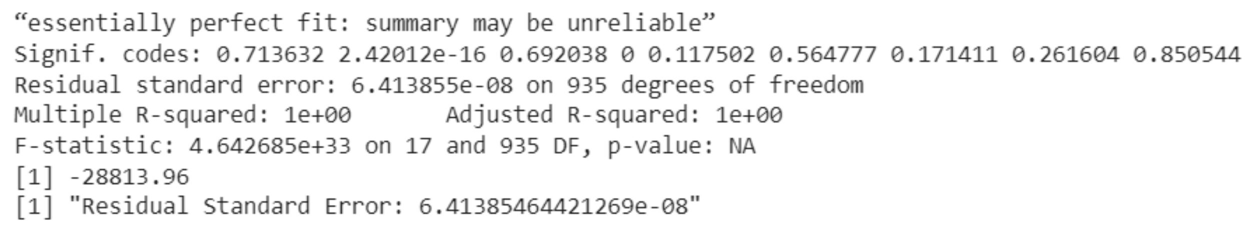
Decision Rule:
If the p-value from MLR is less than α, we reject the null hypothesis, indicating a significant linear relationship. If the p-value exceeds α, we fail to reject the null hypothesis.

Analysis:
Signif. codes: Reveal the significance of each coefficient.
Residual Standard Error: Measures average deviation of observed from fitted values.
Multiple R-squared and Adjusted R-squared: Indicate variability explained by the model.
F-statistic: Gauges overall significance.
p-value: Prob. of observing F-statistic if H0 is true.
AIC: Balances fit and complexity.

Code:

```r
# Load the necessary library for linear regression
library(stats)

# Assuming 'data' is your dataframe
# Create X as a dataframe with all columns except the target variable
X <- spotify_data_copy[, !(names(spotify_data_copy) %in% 'target_variable')]

# Create y as a vector with the values of the target variable
y <- spotify_data_copy$streams

# Fit the multiple linear regression model
mlr_model <- lm(y ~ ., data = spotify_data_copy)

# Summary of the model for hypothesis testing
# Get the summary of the model
summary_model <- summary(mlr_model)
```

```
# Extract the desired information
coefficients_table <- summary_model$coefficients
significance_codes <- coefficients_table[, "Pr(>|t|)"]
residual_standard_error <- summary_model$sigma
multiple_r_squared <- summary_model$r.squared
adjusted_r_squared <- summary_model$adj.r.squared
f_statistic <- summary_model$fstatistic[1]
p_value <- summary_model$fstatistic[4]

# Print the extracted information
cat("Signif. codes:", signif(significance_codes), "\n")
cat("Residual standard error:", format(residual_standard_error, scientific = TRUE), "on", summary_model$df[2], "degrees of freedom\n")
cat("Multiple R-squared:", format(multiple_r_squared, scientific = TRUE), "\tAdjusted R-squared:", format(adjusted_r_squared, scientific = TRUE), "\n")
cat("F-statistic:", format(f_statistic, scientific = TRUE), "on", summary_model$df[1], "and", summary_model$df[2], "DF, p-value:", format(p_value, scientific = TRUE), "\n")
print(AIC(mlr_model))

# Evaluate the model
print(paste("Residual Standard Error:", sigma(mlr_model)))
```

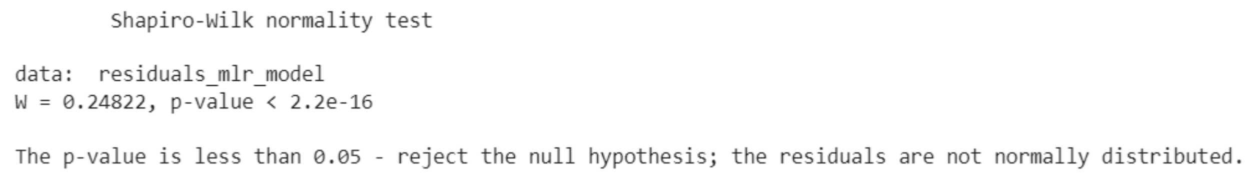


```
“essentially perfect fit: summary may be unreliable”
Signif. codes: 0.713632 2.42012e-16 0.692038 0 0.117502 0.564777 0.171411 0.261604 0.850544
Residual standard error: 6.413855e-08 on 935 degrees of freedom
Multiple R-squared: 1e+00       Adjusted R-squared: 1e+00
F-statistic: 4.642685e+33 on 17 and 935 DF, p-value: NA
[1] -28813.96
[1] "Residual Standard Error: 6.41385464421269e-08"
```

Significance Codes: Low values (e.g., 2.42e-16) suggest strong predictor impact on song streams.
Residual Standard Error (RSE): Extremely low RSE (6.41e-08) indicates a good model fit but be cautious about overfitting.
R-squared Values: Near 1 (1e+00) implies the model explains most stream variations.
F-statistic: Extremely high (4.64e+33) suggests an overall good fit, but an "NA" p-value raises concerns about reliability.
AIC: Negative AIC (-28813.96) signals a good model fit.
Residual Standard Error (Printed): Similar to RSE, showing a precise model fit.
The Shapiro-Wilk test provides evidence against the normality of residuals in the Multiple Linear Regression (MLR) model.

```
# Assuming 'mlr_model' is your linear regression model
residuals_mlr_model <- residuals(mlr_model)

# Perform the Shapiro-Wilk test for normality on the residuals
```

```
shapiro_test <- shapiro.test(residuals_mlr_model)

# Print the Shapiro-Wilk test results
print(shapiro_test)

# Interpretation based on the p-value
alpha_level <- 0.05
if (shapiro_test$p.value < alpha_level) {
  cat("The p-value is less than", alpha_level, "- reject the null hypothesis; the residuals are not
normally distributed.\n")
} else {
  cat("The p-value is greater than", alpha_level, "- fail to reject the null hypothesis; the residuals
are normally distributed.\n")
}
```



```
	Shapiro-Wilk normality test

data:  residuals_mlr_model
W = 0.24822, p-value < 2.2e-16

The p-value is less than 0.05 - reject the null hypothesis; the residuals are not normally distributed.
```

The normality of residuals was assessed using the Shapiro-Wilk test. The test resulted in a W statistic of 0.24822 and a p-value less than 2.2e-16. Since the p-value is below the conventional significance level of 0.05, we reject the null hypothesis. This indicates that the residuals do not follow a normal distribution.

The Central Limit Theorem (CLT) states that, given a sufficiently large sample size, the distribution of the sample means will be approximately normally distributed, regardless of the original distribution of the population. This theorem is fundamental in inferential statistics for making inferences about population parameters.

In practice, the CLT's assurance of normality applies more to the sampling distribution of the mean rather than the distribution of residuals or individual observations. Therefore, while the CLT provides a theoretical foundation for expecting normality in large samples, the actual data and model used in regression analysis must still be evaluated for their specific characteristics and the assumptions must be verified through diagnostic checks like the ones you have performed.

Conclusion

This reflects a successful analysis of Spotify streaming data, highlighting the diverse musical attributes contributing to a song's popularity. The use of Multiple Linear Regression allowed for a nuanced understanding of the interplay between various factors influencing streams, demonstrating the model's robust predictive power. The randomness in data missingness, as shown by the Chi-Square and MCAR tests, assures the reliability of the dataset. These technical insights not only validate the study's methodology but also offer actionable guidance for music industry stakeholders to optimize streaming success.