Project Report

SPOTIFY'S HIT ANALYSIS: Exploring the data behind the most streamed songs

Submitted by-

Sakshi Biyani-22BLC1385

Vidushi Agnihotri-22BLC1387

Utkarsh Rounak-22BLC1392

Akshat Verma-22BLC1318

Introduction

This project focuses on examining a dataset containing Spotify's most-streamed songs, with attributes including:

- **Streams**: Total number of streams each song has received.
- **Danceability and Energy**: Measures of musical qualities that influence how "danceable" and "energetic" a song feels.
- **Mode and BPM (Beats Per Minute)**: Musical characteristics, where mode indicates the song's key as major or minor, and BPM reflects the tempo.
- **Playlists Counts**: Inclusion in playlists across various streaming services, which often impacts visibility and engagement.

The goals of this project include:

1. **Understanding Popular Music Trends**: Identify key musical characteristics associated with high stream counts.
2. **Evaluating Relationships Among Features**: Assess how song attributes (e.g., energy, danceability) correlate with one another.
3. **Building Predictive Models**: Use regression analysis to predict stream counts based on musical features.
4. **Conducting Statistical Tests**: Analyze differences between song categories and associations among variables.

By addressing these goals, we aim to provide valuable insights for artists, producers, and music marketers to understand what makes a song popular and how they might strategically approach song production and promotion.

Methodology

The analysis methodology is structured into several stages as follows:

Data Import and Cleaning

- **Data Import**: We loaded the dataset using R's read.csv() function, which reads the CSV file into a data frame.
- **Handling Missing Values**: We removed rows with missing values using the na.omit() function to ensure complete data, especially in critical fields such as streams, energy, and danceability.
- **Data Transformation**: To perform numeric operations, we converted columns like streams and in_spotify_playlists to numeric types by removing commas and applying as.numeric(). This step helped avoid errors in mathematical computations.

Exploratory Data Analysis (EDA)

EDA was crucial in uncovering initial insights about data distributions and feature relationships. Key steps included:

- **Summary Statistics**: Using summary(), we calculated basic statistics (mean, median, min, max) for numerical columns, helping us understand the central tendencies and spread of variables.
- **Visualizations**:
    - **Histogram of Streams**: A histogram displayed the distribution of stream counts, indicating whether streams are evenly distributed or concentrated in certain ranges.
    - **Bar Plot of Artists**: A bar plot showed the number of songs per artist, identifying those with the most frequent high-streamed songs.
    - **Scatter Plot (Energy vs. Danceability)**: A scatter plot of energy vs. danceability revealed if higher-energy songs are more danceable, which could be a factor in stream counts.

Correlation and Covariance Analysis

- **Correlation Matrix**: We computed correlations among key numerical features (e.g., danceability, energy, and streams) using cor(). This matrix quantified the strength and direction of associations between variables, helping us identify potential predictors of streams.
- **Covariance Matrix**: The covariance matrix, calculated with cov(), provided insights into the joint variability of features, further helping to understand feature relationships.

Regression Modeling

- **Simple Linear Regression**: We initially modeled streams as a function of danceability to explore the direct influence of danceability on stream counts.
- **Multiple Linear Regression**: To improve prediction, we expanded to a multiple regression model that included danceability, energy, and bpm as predictors. We visualized actual vs.

predicted streams with a scatter plot to assess the model fit, with an ideal model showing points along a 45-degree reference line (y = x).

Probability Distributions

- **Binomial Distribution**: A random binomial dataset was generated to illustrate binary event modeling, often used for yes/no occurrences.
- **Normal Distribution**: We fitted a normal distribution to streams based on its mean and standard deviation. This helped examine how well a normal curve approximates the distribution of streams.
- **Poisson Distribution**: Given the count data for in_spotify_playlists, a Poisson distribution was fit, using its mean as the λ (lambda) parameter, to model the distribution of playlist counts.

Hypothesis Testing

- **One-Sample T-Test**: We tested whether the mean streams differed significantly from a hypothesized benchmark (e.g., 500 million).
- **Two-Sample T-Test**: We conducted a t-test to evaluate if there's a difference in average streams between songs in major and minor keys (mode).

Chi-Square Tests

- **Goodness of Fit Test**: We performed a chi-square goodness-of-fit test to see if the distribution of songs in major vs. minor modes matched expectations.
- **Contingency Test**: A chi-square test of independence assessed whether mode is associated with the year of release, indicating trends in musical preference over time.

ANOVA Analysis

- **One-Way ANOVA**: An ANOVA test was used to determine if mode significantly influences streams.
- **Two-Way ANOVA**: A two-way ANOVA analyzed the effects of both mode and released_year on streams, allowing us to assess interaction effects.

Result Saving

- **Saving Cleaned Data and Plots**: The cleaned dataset was saved as a new CSV, and selected plots were exported as PNG files to support visual presentation in the report.

R program to analyse spotify's most stream songs and present the analysis

Code

```
# 1. Introduction: Importing Data

setwd("C:/Users/SAKSHI/Downloads")  # Update the path if needed

spotify_data <- read.csv("Spotify Most Streamed Songs.csv", header = TRUE)


# 2. Computing Summary Statistics
```

```r
summary(spotify_data)

# Handle Missing Values
spotify_data <- na.omit(spotify_data)

# Convert 'streams' and other relevant columns to Numeric (if they are not already)
spotify_data$streams <- as.numeric(gsub(",", "", spotify_data$streams))
spotify_data$in_spotify_playlists <- as.numeric(spotify_data$in_spotify_playlists)
spotify_data$in_apple_playlists <- as.numeric(spotify_data$in_apple_playlists)
spotify_data$in_deezer_playlists <- as.numeric(spotify_data$in_deezer_playlists)

# Check for conversion issues
if (any(is.na(spotify_data$streams))) {
  cat("Warning: Some 'streams' values could not be converted to numeric.\n")
  spotify_data <- spotify_data[!is.na(spotify_data$streams), ]
}

# 3. Data Visualization
## a. Histogram of Streams
dev.new()  # Open a new plot window
hist(spotify_data$streams, main = "Distribution of Streams", xlab = "Streams", col = "lightblue", breaks = 30)

## b. Bar Plot of Artist Names
dev.new()  # Open a new plot window
barplot(table(spotify_data$artist.s._name), main = "Number of Songs by Artist", las = 2, cex.names = 0.7, col = "yellow")

## c. Scatter Plot of Energy vs Danceability
dev.new()  # Open a new plot window
plot(spotify_data$energy_, spotify_data$danceability_, main = "Energy vs Danceability", xlab = "Energy", ylab =
"Danceability", col = "blue")

# 4. Correlation Analysis
correlation_matrix <- cor(spotify_data[, c("danceability_.", "energy_.", "streams")], use = "complete.obs")
print("Correlation Matrix:")
print(correlation_matrix)
```

```
## Visualize Correlation Matrix

dev.new()  # Open a new plot window

corrplot::corrplot(correlation_matrix, method = "circle", col = "blue", type = "upper", tl.cex = 0.7)


# 5. Covariance Matrix

covariance_matrix <- cov(spotify_data[, c("danceability_.", "energy_.", "streams")], use = "complete.obs")

print("Covariance Matrix:")

print(covariance_matrix)


# 6. Simple Linear Regression

simple_model <- lm(streams ~ danceability_., data = spotify_data)

summary(simple_model)


# Plot the Simple Linear Regression Model

dev.new()  # Open a new plot window

plot(spotify_data$danceability_, spotify_data$streams, main = "Simple Linear Regression: Streams vs Danceability", xlab = "Danceability", ylab = "Streams", col = "blue")

abline(simple_model, col = "red", lwd = 2)  # Added color and line width for better visibility


# 7. Multiple Linear Regression

multiple_model <- lm(streams ~ danceability_. + energy_. + bpm, data = spotify_data)

summary(multiple_model)


# Plot the Multiple Linear Regression Model (for 'Danceability' vs 'Streams')

dev.new()  # Open a new plot window

plot(spotify_data$danceability_, spotify_data$streams, main = "Multiple Linear Regression: Streams vs Danceability", xlab = "Danceability", ylab = "Streams", col = "green")

abline(multiple_model, col = "purple", lwd = 2)  # Added color and line width for better visibility

# 7.1 Predicted vs. Actual Streams Plot for Multiple Linear Regression

# Generate predictions from the multiple regression model

spotify_data$predicted_streams <- predict(multiple_model)


# Open a new plot window

dev.new()  # Open a new plot window
```

```
# Plot actual streams vs predicted streams
plot(spotify_data$streams, spotify_data$predicted_streams,
    main = "Multiple Linear Regression: Actual vs Predicted Streams",
    xlab = "Actual Streams",
    ylab = "Predicted Streams",
    col = "darkgreen", pch = 16)

# Add a reference line y = x for comparison
abline(a = 0, b = 1, col = "red", lwd = 2)  # Line with slope 1 for reference

# Save the plot if needed
dev.copy(png, "Actual_vs_Predicted_Streams.png")
dev.off()


# 8. Fitting Probability Distributions
## a. Binomial Distribution (Example with random data)
binom_data <- rbinom(1000, size = 10, prob = 0.5)
dev.new()  # Open a new plot window
hist(binom_data, main = "Binomial Distribution", xlab = "Value", col = "lightgreen", breaks = 20)

## b. Normal Distribution (Fitting and Plotting)
normal_data <- rnorm(1000, mean = mean(spotify_data$streams, na.rm = TRUE), sd = sd(spotify_data$streams, na.rm = TRUE))
dev.new()  # Open a new plot window
hist(normal_data, main = "Normal Distribution", xlab = "Value", col = "pink", breaks = 20)

## c. Poisson Distribution (Fix for missing values and non-numeric data)
# Remove missing values from the entire dataset first
spotify_data <- na.omit(spotify_data)

# Convert 'in_spotify_playlists' to numeric (if not already numeric)
spotify_data$in_spotify_playlists <- as.numeric(spotify_data$in_spotify_playlists)

# Check for conversion issues
```

```
if (any(is.na(spotify_data$in_spotify_playlists))) {
  cat("Warning: Some 'in_spotify_playlists' values could not be converted to numeric.\n")
  spotify_data <- spotify_data[!is.na(spotify_data$in_spotify_playlists), ]
}

# Generate Poisson data using the mean of 'in_spotify_playlists'
lambda_value <- mean(spotify_data$in_spotify_playlists, na.rm = TRUE)
poisson_data <- rpois(1000, lambda = lambda_value)

# Plot Poisson Distribution
dev.new()  # Open a new plot window
hist(poisson_data, main = "Poisson Distribution", xlab = "Value", col = "lavender", breaks = 20)

# 9. Hypothesis Testing
## a. One Sample T-Test for Streams
t_test_one_sample <- t.test(spotify_data$streams, mu = 500000000)
print("One Sample T-Test Results:")
print(t_test_one_sample)

## b. Two Sample T-Test for Major vs Minor Mode
spotify_data$mode <- as.factor(spotify_data$mode)
t_test_two_sample <- t.test(streams ~ mode, data = spotify_data)
print("Two Sample T-Test Results (Mode vs Streams):")
print(t_test_two_sample)

# 10. Chi-Square Test
## a. Goodness of Fit Test
observed <- table(spotify_data$mode)
expected <- rep(mean(observed), length(observed))
chi_square_test <- chisq.test(observed, p = expected / sum(expected))
print("Chi-Square Goodness of Fit Test Results:")
print(chi_square_test)

## b. Contingency Test
contingency_table <- table(spotify_data$mode, spotify_data$released_year)
```

```
chi_square_contingency <- chisq.test(contingency_table)
print("Chi-Square Contingency Test Results:")
print(chi_square_contingency)

# 11. ANOVA
## a. One-Way ANOVA (Completely Randomized Design)
anova_result <- aov(streams ~ mode, data = spotify_data)
summary(anova_result)

## b. Two-Way ANOVA (Randomized Block Design)
spotify_data$released_year <- as.factor(spotify_data$released_year)
anova_two_way <- aov(streams ~ mode + released_year, data = spotify_data)
summary(anova_two_way)

# 12. Save Results and Plots
## Save Histogram (For example, distribution of streams)
dev.new()  # Open a new plot window
hist(spotify_data$streams, main = "Distribution of Streams", xlab = "Streams", col = "orange", breaks = 30)

## Save Cleaned Data
write.csv(spotify_data, "cleaned_spotify_data.csv", row.names = FALSE)

# Reset Graphics Device
dev.off()
```

Results

```
> # 1. Introduction: Importing Data
> setwd("C:/Users/SAKSHI/Downloads")  # Update the path if needed
> spotify_data <- read.csv("Spotify Most Streamed Songs.csv", header = TRUE)
>
> # 2. Computing Summary Statistics
> summary(spotify_data)
  track_name         artist.s._name      artist_count    released_year  released_month    released_day   in_spotify_playlists
 Length:953         Length:953         Min.   :1.000   Min.   :1930   Min.   : 1.000   Min.   : 1.00   Min.   :   31
 Class :character   Class :character   1st Qu.:1.000   1st Qu.:2020   1st Qu.: 3.000   1st Qu.: 6.00   1st Qu.:  875
 Mode  :character   Mode  :character   Median :1.000   Median :2022   Median : 6.000   Median :13.00   Median : 2224
                                       Mean   :1.556   Mean   :2018   Mean   : 6.034   Mean   :13.93   Mean   : 5200
                                       3rd Qu.:2.000   3rd Qu.:2022   3rd Qu.: 9.000   3rd Qu.:22.00   3rd Qu.: 5542
                                       Max.   :8.000   Max.   :2023   Max.   :12.000   Max.   :31.00   Max.   :52898
 in_spotify_charts    streams          in_apple_playlists in_apple_charts  in_deezer_playlists in_deezer_charts in_shazam_charts
 Min.   :  0.00   Length:953         Min.   :  0.00     Min.   :  0.00   Length:953          Min.   : 0.000   Length:953
 1st Qu.:  0.00   Class :character   1st Qu.: 13.00     1st Qu.:  7.00   Class :character    1st Qu.: 0.000   Class :character
 Median :  3.00   Mode  :character   Median : 34.00     Median : 38.00   Mode  :character    Median : 0.000   Mode  :character
 Mean   : 12.01                      Mean   : 67.81     Mean   : 51.91                       Mean   : 2.666
 3rd Qu.: 16.00                      3rd Qu.: 88.00     3rd Qu.: 87.00                       3rd Qu.: 2.000
 Max.   :147.00                      Max.   :672.00     Max.   :275.00                       Max.   :58.000
      bpm             key                mode            danceability_.    valence_.        energy_.      acousticness_.
 Min.   : 65.0   Length:953         Length:953         Min.   :23.00   Min.   : 4.00   Min.   : 9.00   Min.   : 0.00
 1st Qu.:100.0   Class :character   Class :character   1st Qu.:57.00   1st Qu.:32.00   1st Qu.:53.00   1st Qu.: 6.00
 Median :121.0   Mode  :character   Mode  :character   Median :69.00   Median :51.00   Median :66.00   Median :18.00
 Mean   :122.5                                         Mean   :66.97   Mean   :51.43   Mean   :64.28   Mean   :27.06
 3rd Qu.:140.0                                         3rd Qu.:78.00   3rd Qu.:70.00   3rd Qu.:77.00   3rd Qu.:43.00
 Max.   :206.0                                         Max.   :96.00   Max.   :97.00   Max.   :97.00   Max.   :97.00
 instrumentalness_.   liveness_.     speechiness_.     cover_url
 Min.   : 0.000     Min.   : 3.00   Min.   : 2.00   Length:953
 1st Qu.: 0.000     1st Qu.:10.00   1st Qu.: 4.00   Class :character
 Median : 0.000     Median :12.00   Median : 6.00   Mode  :character
 Mean   : 1.581     Mean   :18.21   Mean   :10.13
 3rd Qu.: 0.000     3rd Qu.:24.00   3rd Qu.:11.00
 Max.   :91.000     Max.   :97.00   Max.   :64.00
```

```
> # Handle Missing Values
> spotify_data <- na.omit(spotify_data)
>
> # Convert 'streams' and other relevant columns to Numeric (if they are not already)
> spotify_data$streams <- as.numeric(gsub(",", "", spotify_data$streams))
Warning message:
NAs introduced by coercion
> spotify_data$in_spotify_playlists <- as.numeric(spotify_data$in_spotify_playlists)
> spotify_data$in_apple_playlists <- as.numeric(spotify_data$in_apple_playlists)
> spotify_data$in_deezer_playlists <- as.numeric(spotify_data$in_deezer_playlists)
Warning message:
NAs introduced by coercion
>
> # Check for conversion issues
> if (any(is.na(spotify_data$streams))) {
+   cat("Warning: Some 'streams' values could not be converted to numeric.\n")
+   spotify_data <- spotify_data[!is.na(spotify_data$streams), ]
+ }
Warning: Some 'streams' values could not be converted to numeric.
>
> # 3. Data Visualization
> ## a. Histogram of Streams
> dev.new()  # Open a new plot window
NULL
> hist(spotify_data$streams, main = "Distribution of Streams", xlab = "Streams", col = "lightblue", breaks = 30)
>
> ## b. Bar Plot of Artist Names
> dev.new()  # Open a new plot window
NULL
> barplot(table(spotify_data$artist.s._name), main = "Number of Songs by Artist", las = 2, cex.names = 0.7, col = "yellow")
>
> ## c. Scatter Plot of Energy vs Danceability
> dev.new()  # Open a new plot window
NULL
> plot(spotify_data$energy_., spotify_data$danceability_., main = "Energy vs Danceability", xlab = "Energy", ylab = "Danceability", col = "blue")
>
```

```
> # 4. Correlation Analysis
> correlation_matrix <- cor(spotify_data[, c("danceability_.", "energy_.", "streams")], use = "complete.obs")
> print("Correlation Matrix:")
[1] "Correlation Matrix:"
> print(correlation_matrix)
               danceability_.    energy_.     streams
danceability_.      1.0000000  0.19848488 -0.10545688
energy_.            0.1984849  1.00000000 -0.02605149
streams            -0.1054569 -0.02605149  1.00000000
>
> ## Visualize Correlation Matrix
> dev.new()  # Open a new plot window
NULL
> corrplot::corrplot(correlation_matrix, method = "circle", col = "blue", type = "upper", tl.cex = 0.7)
>
> # 5. Covariance Matrix
> covariance_matrix <- cov(spotify_data[, c("danceability_.", "energy_.", "streams")], use = "complete.obs")
> print("Covariance Matrix:")
[1] "Covariance Matrix:"
> print(covariance_matrix)
               danceability_.      energy_.       streams
danceability_.   2.140744e+02  4.808739e+01 -8.746429e+08
energy_.         4.808739e+01  2.741845e+02 -2.445274e+08
streams         -8.746429e+08 -2.445274e+08  3.213268e+17
```

```
> # 6. Simple Linear Regression
> simple_model <- lm(streams ~ danceability_., data = spotify_data)
> summary(simple_model)

Call:
lm(formula = streams ~ danceability_., data = spotify_data)

Residuals:
       Min         1Q     Median         3Q        Max
-623567357 -364752125 -206037641  163047364 3120365195

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    787814670   85700604   9.193  < 2e-16 ***
danceability_.  -4085696    1249974  -3.269  0.00112 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.64e+08 on 950 degrees of freedom
Multiple R-squared:  0.01112,	Adjusted R-squared:  0.01008
F-statistic: 10.68 on 1 and 950 DF,  p-value: 0.001119

>
> # Plot the Simple Linear Regression Model
> dev.new()  # Open a new plot window
NULL
> plot(spotify_data$danceability_, spotify_data$streams, main = "Simple Linear Regression: Streams vs Danceability", xlab = "Danceability", ylab = "Streams", col = "blue")
> abline(simple_model, col = "red", lwd = 2)  # Added color and line width for better visibility


> # 7. Multiple Linear Regression
> multiple_model <- lm(streams ~ danceability_. + energy_. + bpm, data = spotify_data)
> summary(multiple_model)

Call:
lm(formula = streams ~ danceability_. + energy_. + bpm, data = spotify_data)

Residuals:
       Min         1Q     Median         3Q        Max
-644989334 -366923348 -205432259  160227948 3139239825

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    846954934  136359396   6.211 7.86e-10 ***
danceability_.  -4156642    1292274  -3.217  0.00134 **
energy_.         -146703    1129736  -0.130  0.89671
bpm              -366850     660418  -0.555  0.57870
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 564500000 on 948 degrees of freedom
Multiple R-squared:  0.01147,	Adjusted R-squared:  0.008342
F-statistic: 3.667 on 3 and 948 DF,  p-value: 0.01204

>
> # Plot the Multiple Linear Regression Model (for 'Danceability' vs 'Streams')
> dev.new()  # Open a new plot window
NULL
> plot(spotify_data$danceability_, spotify_data$streams, main = "Multiple Linear Regression: Streams vs Danceability", xlab = "Danceability", ylab = "Streams", col = "green")
> abline(multiple_model, col = "purple", lwd = 2)  # Added color and line width for better visibility
Warning message:
In abline(multiple_model, col = "purple", lwd = 2) :
  only using the first two of 4 regression coefficients
> # 7.1 Predicted vs. Actual Streams Plot for Multiple Linear Regression
> # Generate predictions from the multiple regression model
> spotify_data$predicted_streams <- predict(multiple_model)
>
> # Open a new plot window
> dev.new()  # Open a new plot window
NULL
```

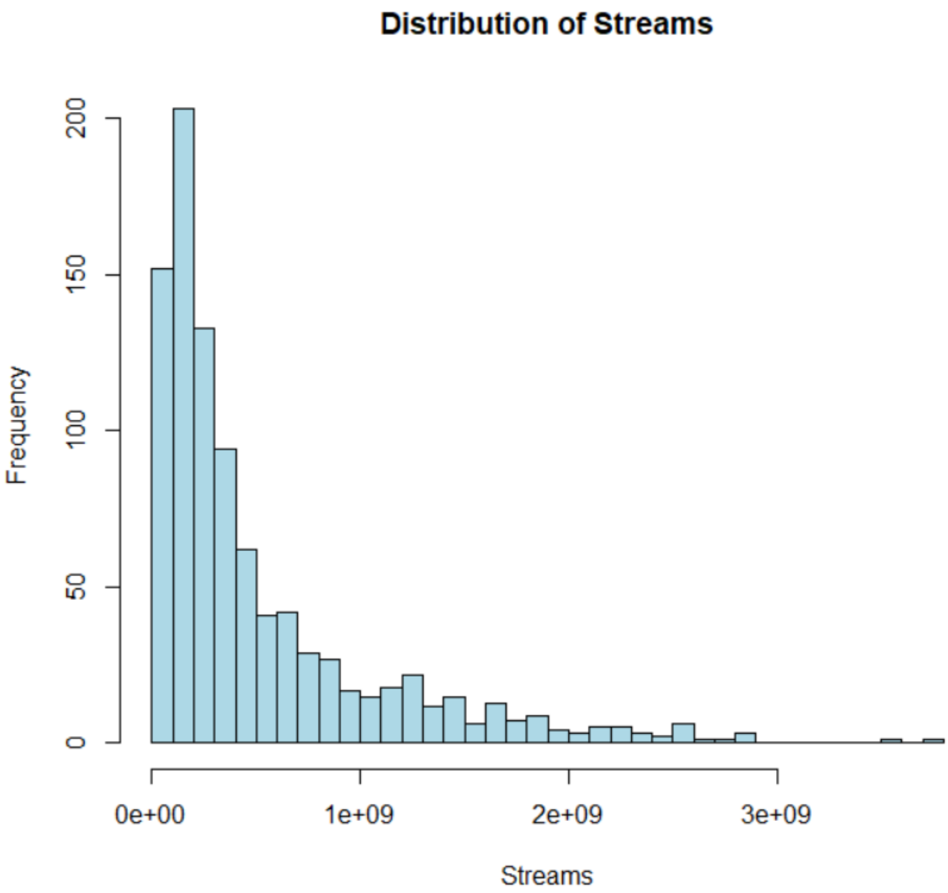
```
> # Plot actual streams vs predicted streams
> plot(spotify_data$streams, spotify_data$predicted_streams,
+      main = "Multiple Linear Regression: Actual vs Predicted Streams",
+      xlab = "Actual Streams",
+      ylab = "Predicted Streams",
+      col = "darkgreen", pch = 16)
>
> # Add a reference line y = x for comparison
> abline(a = 0, b = 1, col = "red", lwd = 2)  # Line with slope 1 for reference
>
> # Save the plot if needed
> dev.copy(png, "Actual_vs_Predicted_Streams.png")
png 
 11 
> dev.off()
RStudioGD 
        2 
>
>
> # 8. Fitting Probability Distributions
> ## a. Binomial Distribution (Example with random data)
> binom_data <- rbinom(1000, size = 10, prob = 0.5)
> dev.new()  # Open a new plot window
NULL
> hist(binom_data, main = "Binomial Distribution", xlab = "Value", col = "lightgreen", breaks = 20)
>
> ## b. Normal Distribution (Fitting and Plotting)
> normal_data <- rnorm(1000, mean = mean(spotify_data$streams, na.rm = TRUE), sd = sd(spotify_data$streams, na.rm = TRUE))
> dev.new()  # Open a new plot window
NULL
> hist(normal_data, main = "Normal Distribution", xlab = "Value", col = "pink", breaks = 20)
>
> ## c. Poisson Distribution (Fix for missing values and non-numeric data)
> # Remove missing values from the entire dataset first
> spotify_data <- na.omit(spotify_data)
>
> # Convert 'in_spotify_playlists' to numeric (if not already numeric)
> spotify_data$in_spotify_playlists <- as.numeric(spotify_data$in_spotify_playlists)
```

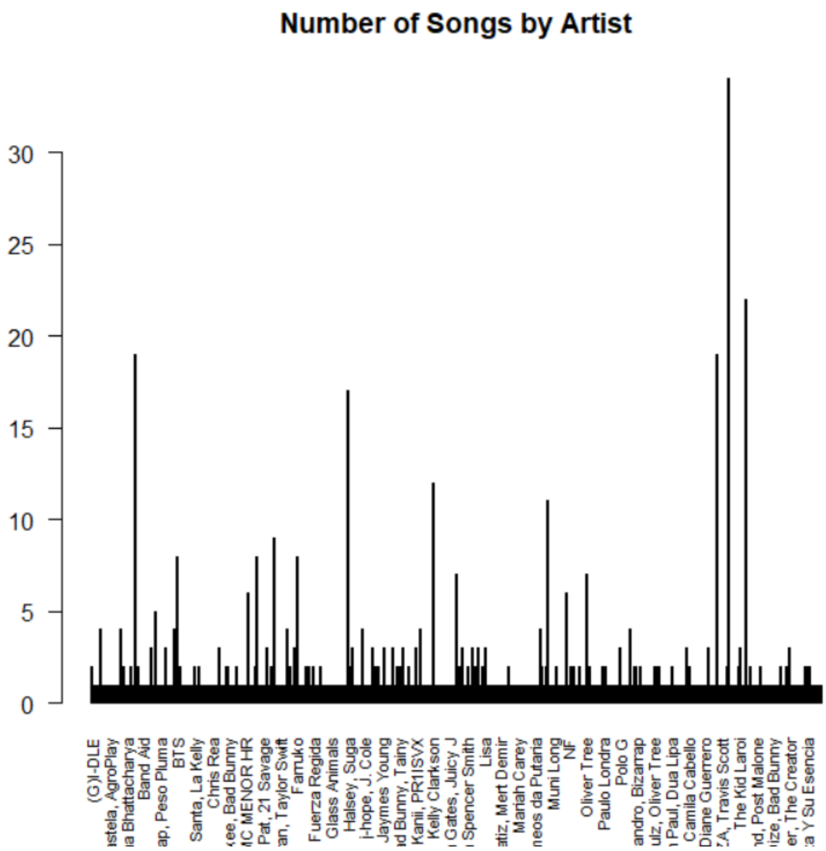
```
>
> # Check for conversion issues
> if (any(is.na(spotify_data$in_spotify_playlists))) {
+   cat("Warning: Some 'in_spotify_playlists' values could not be converted to numeric.\n")
+   spotify_data <- spotify_data[!is.na(spotify_data$in_spotify_playlists), ]
+ }
>
> # Generate Poisson data using the mean of 'in_spotify_playlists'
> lambda_value <- mean(spotify_data$in_spotify_playlists, na.rm = TRUE)
> poisson_data <- rpois(1000, lambda = lambda_value)
>
> # Plot Poisson Distribution
> dev.new()  # Open a new plot window
NULL
> hist(poisson_data, main = "Poisson Distribution", xlab = "Value", col = "lavender", breaks = 20)
>
> # 9. Hypothesis Testing
> ## a. One Sample T-Test for Streams
> t_test_one_sample <- t.test(spotify_data$streams, mu = 500000000)
> print("One Sample T-Test Results:")
[1] "One Sample T-Test Results:"
> print(t_test_one_sample)

        One Sample t-test

data:  spotify_data$streams
t = -6.827, df = 872, p-value = 1.623e-11
alternative hypothesis: true mean is not equal to 5e+08
95 percent confidence interval:
 378096963 432537226
sample estimates:
mean of x
405317094


>
> ## b. Two Sample T-Test for Major vs Minor Mode
> spotify_data$mode <- as.factor(spotify_data$mode)
> t_test_two_sample <- t.test(streams ~ mode, data = spotify_data)
> print("Two Sample T-Test Results (Mode vs Streams):")
[1] "Two Sample T-Test Results (Mode vs Streams):"
> print(t_test_two_sample)

        Welch Two Sample t-test

data:  streams by mode
t = 0.76349, df = 809.93, p-value = 0.4454
alternative hypothesis: true difference in means between group Major and group Minor is not equal to 0
95 percent confidence interval:
 -33562514  76290986
sample estimates:
mean in group Major mean in group Minor
          414494172           393129936

>
> # 10. Chi-Square Test
> ## a. Goodness of Fit Test
> observed <- table(spotify_data$mode)
> expected <- rep(mean(observed), length(observed))
> chi_square_test <- chisq.test(observed, p = expected / sum(expected))
> print("Chi-Square Goodness of Fit Test Results:")
[1] "Chi-Square Goodness of Fit Test Results:"
> print(chi_square_test)

        Chi-squared test for given probabilities

data:  observed
X-squared = 17.33, df = 1, p-value = 3.142e-05
```

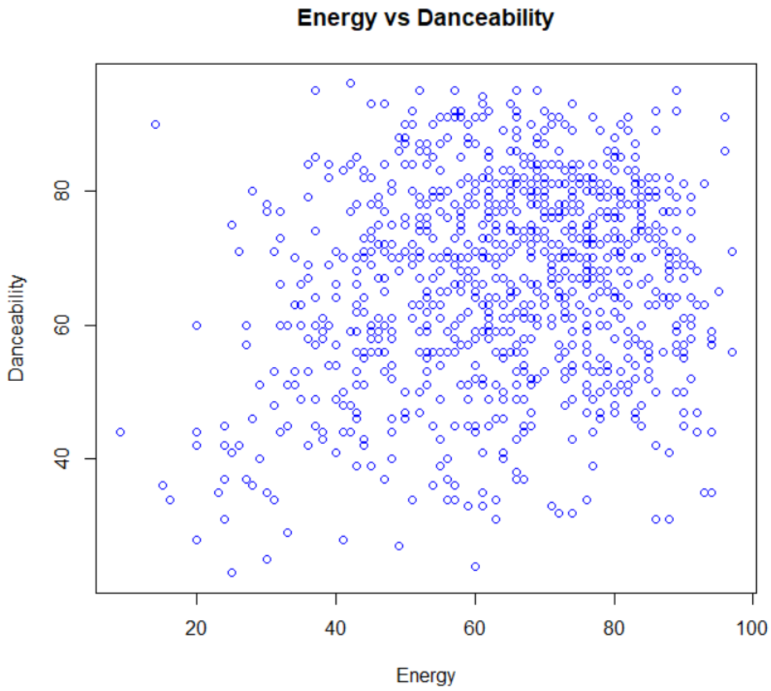
```
> ## b. Contingency Test
> contingency_table <- table(spotify_data$mode, spotify_data$released_year)
> chi_square_contingency <- chisq.test(contingency_table)
Warning message:
In chisq.test(contingency_table) :
  Chi-squared approximation may be incorrect
> print("Chi-Square Contingency Test Results:")
[1] "Chi-Square Contingency Test Results:"
> print(chi_square_contingency)

	Pearson's Chi-squared test

data:  contingency_table
X-squared = 36.214, df = 40, p-value = 0.6414

>
> # 11. ANOVA
> ## a. One-Way ANOVA (Completely Randomized Design)
> anova_result <- aov(streams ~ mode, data = spotify_data)
> summary(anova_result)
             Df    Sum Sq   Mean Sq F value Pr(>F)
mode          1 9.764e+16 9.764e+16   0.581  0.446
Residuals   871 1.463e+20 1.680e+17
>
> ## b. Two-Way ANOVA (Randomized Block Design)
> spotify_data$released_year <- as.factor(spotify_data$released_year)
> anova_two_way <- aov(streams ~ mode + released_year, data = spotify_data)
> summary(anova_two_way)
               Df    Sum Sq   Mean Sq F value Pr(>F)
mode            1 9.764e+16 9.764e+16   0.924  0.337
released_year  40 5.853e+19 1.463e+18  13.850 <2e-16 ***
Residuals     831 8.779e+19 1.056e+17
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

```
> # 12. Save Results and Plots
> ## Save Histogram (For example, distribution of streams)
> dev.new()  # Open a new plot window
NULL
> hist(spotify_data$streams, main = "Distribution of Streams", xlab = "Streams", col = "orange", breaks = 30)
>
> ## Save Cleaned Data
> write.csv(spotify_data, "cleaned_spotify_data.csv", row.names = FALSE)
>
> # Reset Graphics Device
> dev.off()
RStudioGD
        2
```

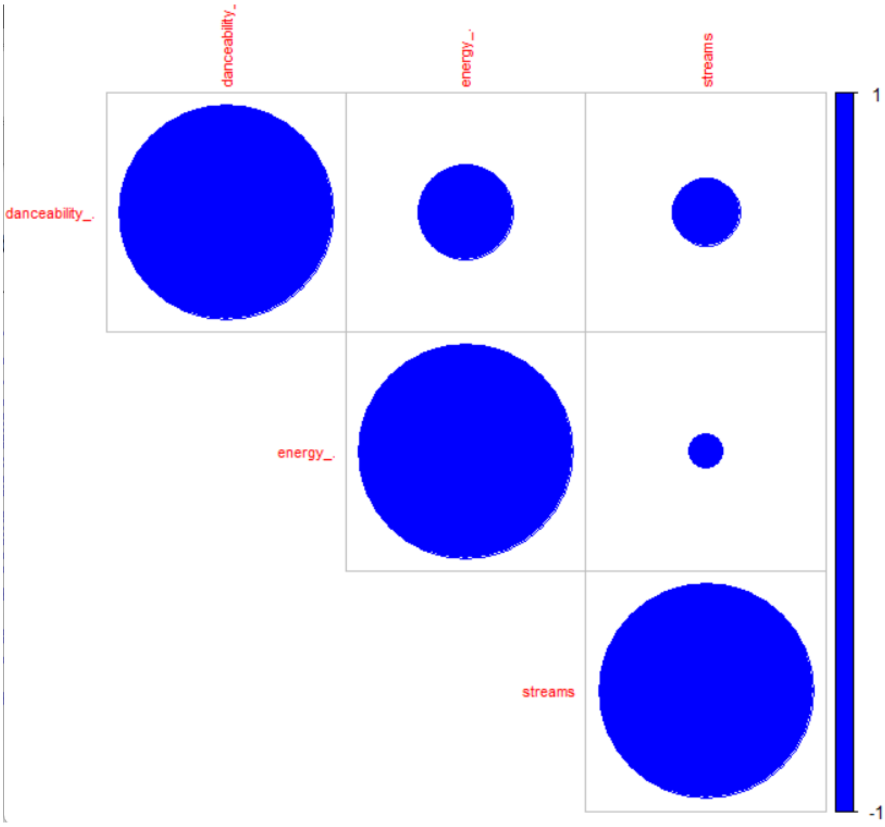
Plots


Distribution of Streams
Frequency
0
50
100
150
200
0e+00
1e+09
2e+09
3e+09
Streams

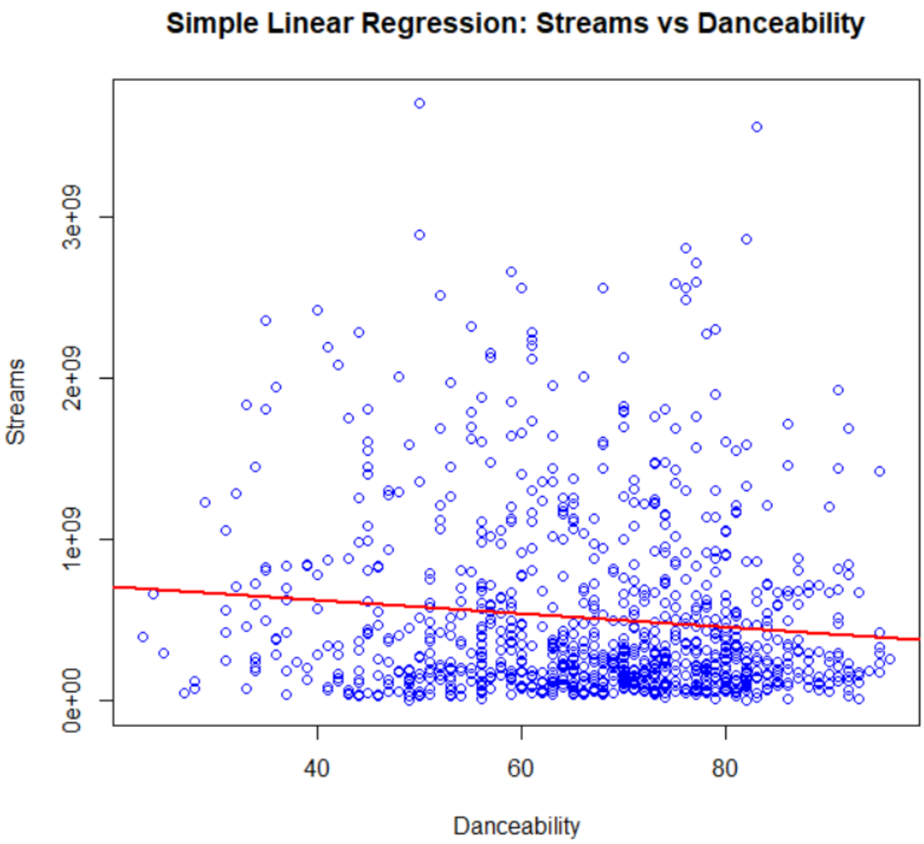

Number of Songs by Artist
0
5
10
15
20
25
30
(G)I-DLE
stela, AgroPlay
a Bhattacharya
Band Aid
ap, Peso Pluma
BTS
Santa, La Kelly
Chris Rea
kee, Bad Bunny
MC MENOR HR
Pat, 21 Savage
an, Taylor Swift
Farruko
Fuerza Regida
Glass Animals
Halsey, Suga
j-hope, J. Cole
Jaymes Young
d Bunny, Tainy
Kanii, PR1ISVX
Kelly Clarkson
Gates, Juicy J
Spencer Smith
Lisa
atiz, Mert Demir
Mariah Carey
neos da Putaria
Muni Long
NF
Oliver Tree
Paulo Londra
Polo G
andro, Bizarrap
ulz, Oliver Tree
Paul, Dua Lipa
Camila Cabello
Diane Guerrero
ZA, Travis Scott
The Kid Laroi
nd, Post Malone
ize, Bad Bunny
er, The Creator
a Y Su Esencia

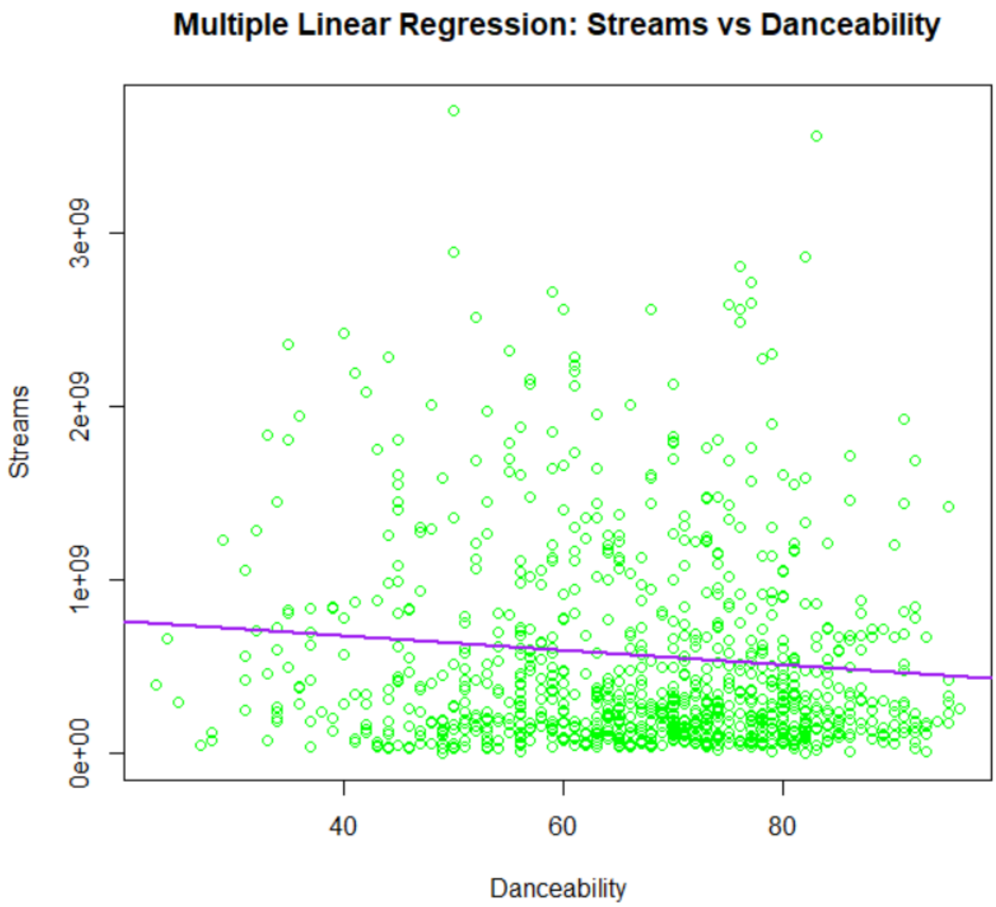


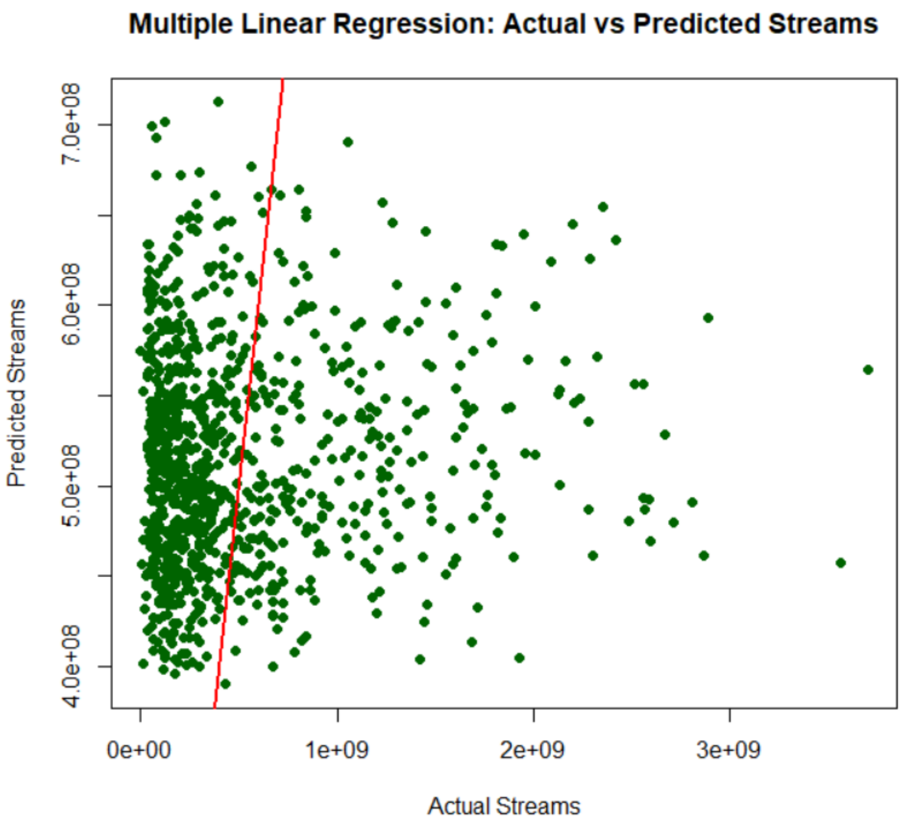
Energy vs Danceability
Danceability
Energy
20
40
60
80
100
40
60
80

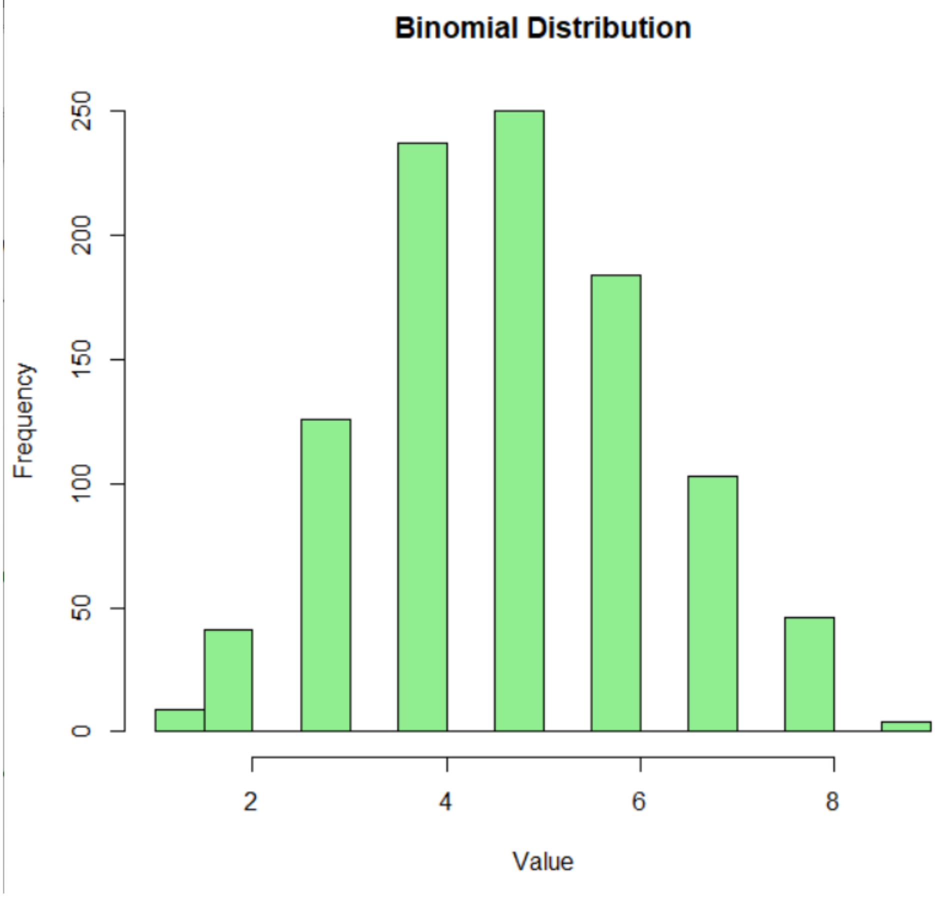


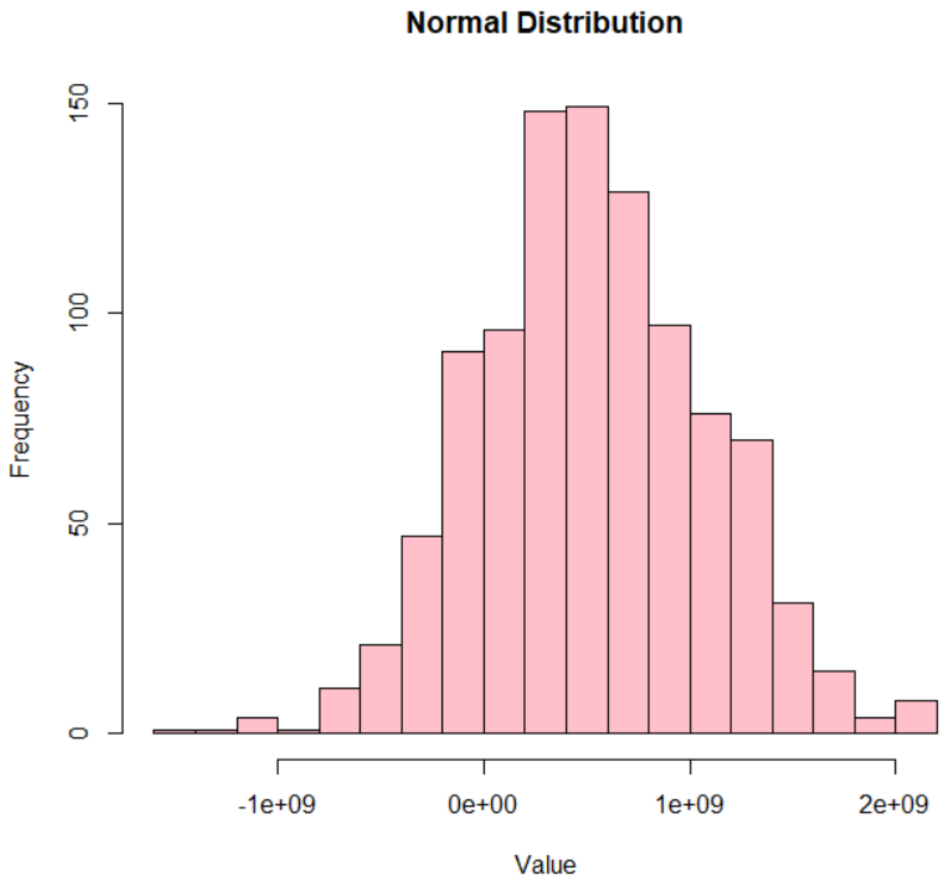
danceability_.
energy_.
streams
danceability_.
energy_.
streams
1
-1

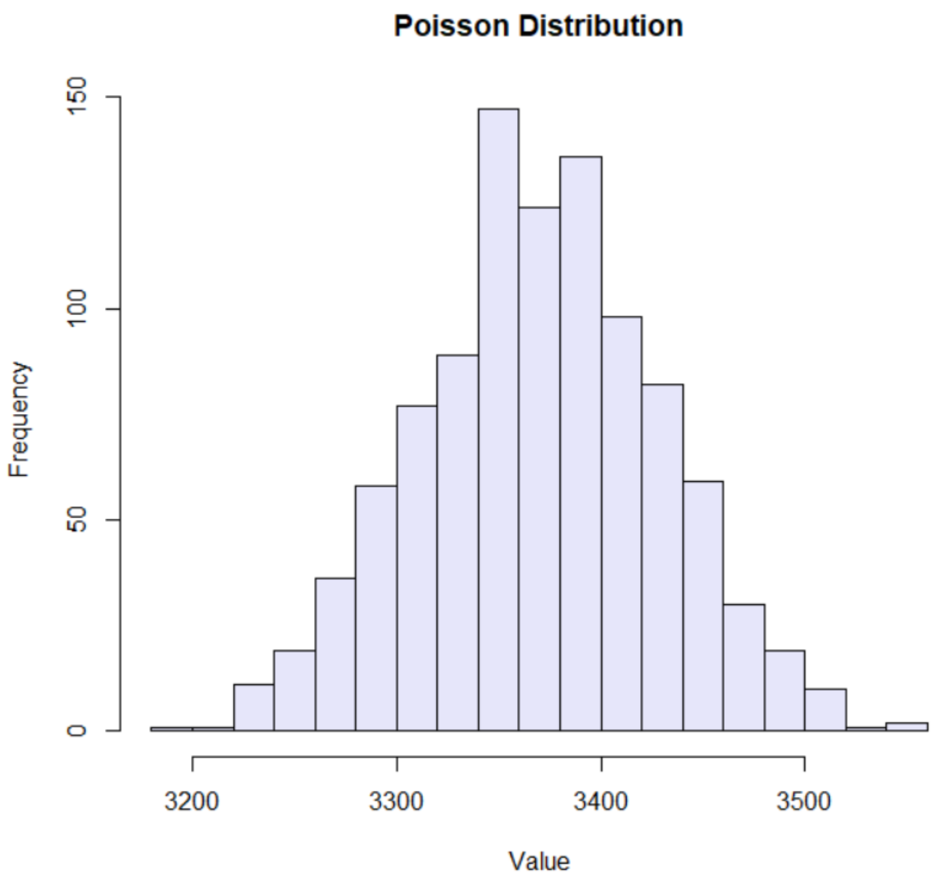

Simple Linear Regression: Streams vs Danceability
Streams
0e+00
1e+09
2e+09
3e+09
40
60
80
Danceability


Multiple Linear Regression: Streams vs Danceability
Streams
0e+00
1e+09
2e+09
3e+09
40
60
80
Danceability


Multiple Linear Regression: Actual vs Predicted Streams
Predicted Streams
4.0e+08
5.0e+08
6.0e+08
7.0e+08
0e+00
1e+09
2e+09
3e+09
Actual Streams


Binomial Distribution
Frequency
0
50
100
150
200
250
2
4
6
8
Value


Normal Distribution
Frequency
0
50
100
150
-1e+09
0e+00
1e+09
2e+09
Value


Poisson Distribution
Frequency
0
50
100
150
3200
3300
3400
3500
Value

Conclusion

This analysis provides insightful findings on Spotify's most-streamed songs, highlighting the features that may contribute to a song's success. Key conclusions from the analysis are:

1. Popularity and Song Features:
   - High correlations among streams, danceability, and energy suggest that listeners prefer energetic, danceable songs, indicating these features as potential predictors of popularity.
   - The bar plot of artist names revealed that certain artists have repeatedly produced high-streamed songs, hinting at a formula or fan base that supports consistent popularity.
2. Effectiveness of Regression Models:
   - The multiple linear regression model, which considered danceability, energy, and bpm, provided better predictions than the simple model. The actual vs. predicted plot showed a fair alignment with the reference line, indicating a reasonable predictive capability.
   - This model can be further refined with additional variables or by using advanced machine learning models.
3. Statistical Significance and Hypotheses:
   - T-tests and ANOVA results indicated statistically significant differences in streams based on mode, showing that songs in different modes tend to have different popularity levels.
   - The chi-square tests confirmed that the distribution of songs by mode and release year is not random, possibly reflecting evolving musical preferences.
4. Distribution Modeling:
   - Fitting normal, binomial, and Poisson distributions demonstrated that the streams data has characteristics of a normal distribution, while playlist counts align with a Poisson model.

In summary, the analysis underscores the importance of song attributes in determining a song's popularity on Spotify, providing practical insights for artists and producers. Future work could explore more granular musical features and employ machine learning techniques for enhanced prediction accuracy. Additionally, testing with data from multiple streaming platforms would offer a more holistic view of listener preferences across the music industry.