Cluster Analysis on Spotify's Top 200 songs' Audio Features

Dania Bawab

Introduction and Motivation

The music industry is constantly evolving, with new genres emerging and existing genres evolving over time. Understanding the underlying patterns and characteristics of different music genres can provide valuable insights into their popularity and appeal among listeners. In this study, I aim to conduct a cluster analysis on the audio features of songs to distinguish genres and explore their popularity in the Spotify top 200 charts over time.

Through cluster analysis on audio features, I can discern distinct patterns that differentiate genres, aiding music enthusiasts, industry professionals, and researchers in better comprehending genre classifications. Moreover, tracking genre popularity trends provides insights into market dynamics and audience preferences, assisting in predicting future trends and guiding content recommendation systems. For artists and producers, insights into genre characteristics and popularity can inform creative decisions, enhance audience engagement, and improve success rates in the ever-evolving music industry landscape.

Research Question

How can clustering techniques be utilized to identify distinct music genres and analyze the relationships between these genres and their respective durations in the top 200 charts over time*?*

Introduction to Clustering

Clustering can be explained by, "clustering… is to divide the population into several groups in such a way that the data points in the same groups are more similar to each other than the data points in other groups. In short, it is a collection of objects based on their similarities and dissimilarities" (Sharma, 2024). K-means clustering, is a cluster algorithm that assigns the points based on the distance between it and a chosen centroid. The points are clustered based on their closeness to their centroid. Hierarchical clustering, is a method of cluster analysis that seeks to build a hierarchy of clusters. It does so by iteratively merging or dividing clusters based on their similarity or distance from each other.

In my methodology, I'll start by implementing min-max scaling to standardize feature ranges between 0 and 1. This preprocessing step ensures that each feature contributes equally to the subsequent clustering analysis, enhancing comparability across the data set. By transforming each feature individually, I'll mitigate the influence of features with larger scales, preventing scale bias and facilitating the identification of meaningful clusters based on audio feature similarities.

Following scaling, I'll employ k-means clustering to partition the scaled data set into K clusters, with K representing the chosen number of clusters. Through iterative assignment

of observations to clusters, I'll minimize the total within-cluster sum of squares, effectively grouping similar observations together. By varying the number of clusters and evaluating the resulting inertia values, I'll determine an optimal number of clusters using the elbow method or other relevant criteria.

Simultaneously, I'll explore the dataset's clustering structure using hierarchical clustering as an alternative technique. Hierarchical clustering organizes observations into a hierarchical tree structure (dendrogram), revealing hierarchical relationships between observations and clusters. Using the "ward.D2" method for merging clusters, I'll construct the dendrogram and assess the resulting inertia values to gain insights into the hierarchical clustering patterns.

To validate the clustering results, I'll do a comparison with existing genre annotations. Comparison with existing genre annotations will validate the identified clusters against established genre classifications, ensuring their relevance and interpretability. Through visualizations, statistical analysis, and interpretation of clustering results, I'll derive insights into genre relationships and their durations in the top 200 charts over time.

Data Collection and Preparation

```
#Reading in the Data
top_songs = read.csv("~/Downloads/spotify_top_songs_audio_features 2.csv")
head(top_songs)
```

```
##                        id               artist_names
## 1 000xQL6tZNLJzIrtIgxqSl        ZAYN, PARTYNEXTDOOR
## 2 003eoIwxETJujVWmNFMoZy               Alessia Cara
## 3 003vvx7Niy0yvhvHt4a68B                The Killers
## 4 00B7TZ0Xawar6NZ00JFomN Cardi B, Chance the Rapper
## 5 00Blm7zeNqgYLPtW6zg8cj    Post Malone, The Weeknd
## 6 00EPIEnX1JFjff8sC6bccd      Thalia, NATTI NATASHA
##                             track_name             source   key  mode
## 1 Still Got Time (feat. PARTYNEXTDOOR)  RCA Records Label     G Major
## 2                       Growing Pains Def Jam Recordings C#/Db Minor
## 3                      Mr. Brightside     Island Records C#/Db Major
## 4  Best Life (feat. Chance The Rapper)       Atlantic/KSR     A Major
## 5      One Right Now (with The Weeknd)   Republic Records C#/Db Major
## 6                       No Me Acuerdo   Sony Music Latin     G Minor
##   time_signature danceability energy speechiness acousticness
instrumentalness
## 1        4 beats        0.748  0.627      0.0639      0.13100
0
## 2        4 beats        0.353  0.755      0.7330      0.08220
0
## 3        4 beats        0.352  0.911      0.0747      0.00121
0
## 4        4 beats        0.620  0.625      0.5530      0.28700
0
## 5        4 beats        0.687  0.781      0.0530      0.03610
0
```

```
## 6        4 beats        0.836  0.799      0.0873      0.18700
0
##   liveness valence loudness   tempo duration_ms weeks_on_chart   streams
## 1   0.0852   0.524   -6.029 120.963      188491             17 107527761
## 2   0.3900   0.437   -6.276 191.153      193680              2   9944865
## 3   0.0995   0.236   -5.230 148.033      222973            125 512388123
## 4   0.3140   0.665   -7.438 167.911      284856              2  11985346
## 5   0.0755   0.688   -4.806  97.014      193507             30 301860377
## 6   0.0920   0.772   -4.247  94.033      217653             16  98123727
```

Exploratory Data Analysis (EDA):

```
#Checking for NA values and duplicates
colSums(is.na(top_songs))
```

```
##               id     artist_names       track_name           source
##                0                0                0                0
##              key             mode   time_signature     danceability
##                0                0                0                0
##           energy      speechiness     acousticness instrumentalness
##                0                0                0                0
##         liveness          valence         loudness            tempo
##                0                0                0                0
##      duration_ms   weeks_on_chart          streams
##                0                0                0
```

```
sum(duplicated(top_songs))
```

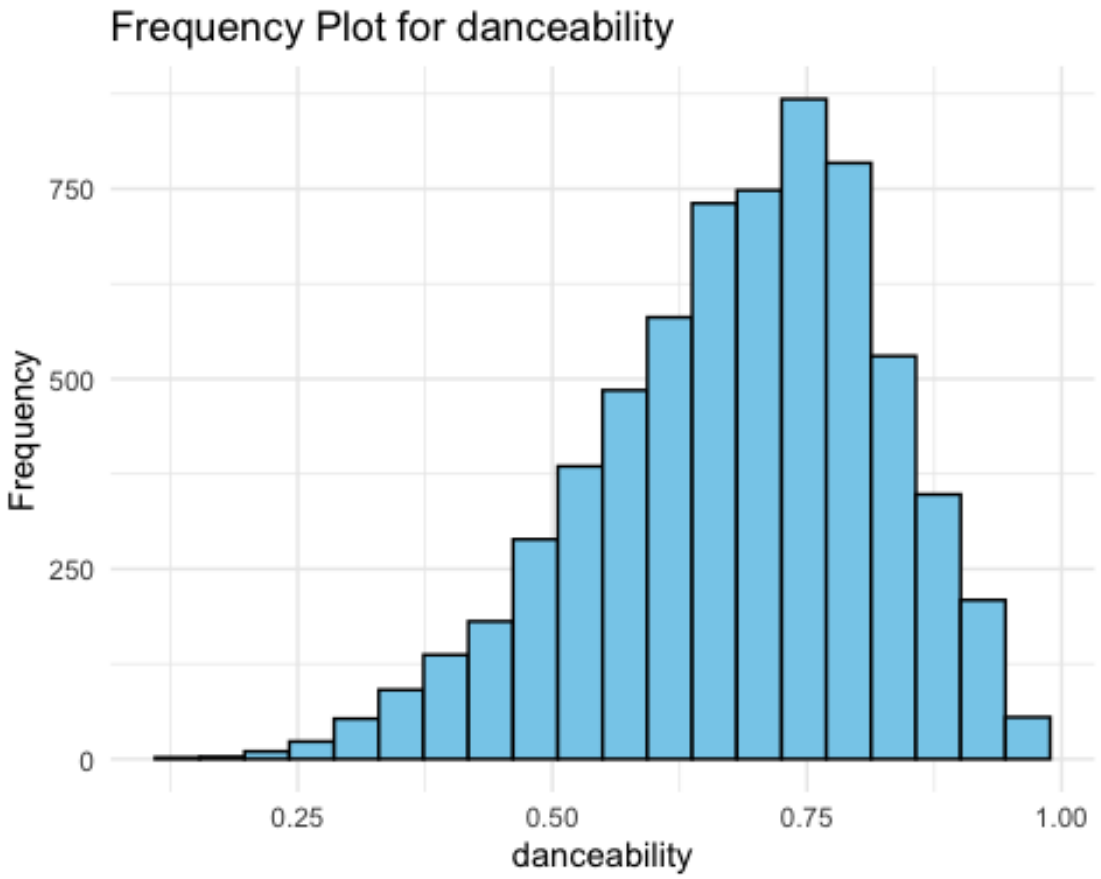
```
## [1] 0
```

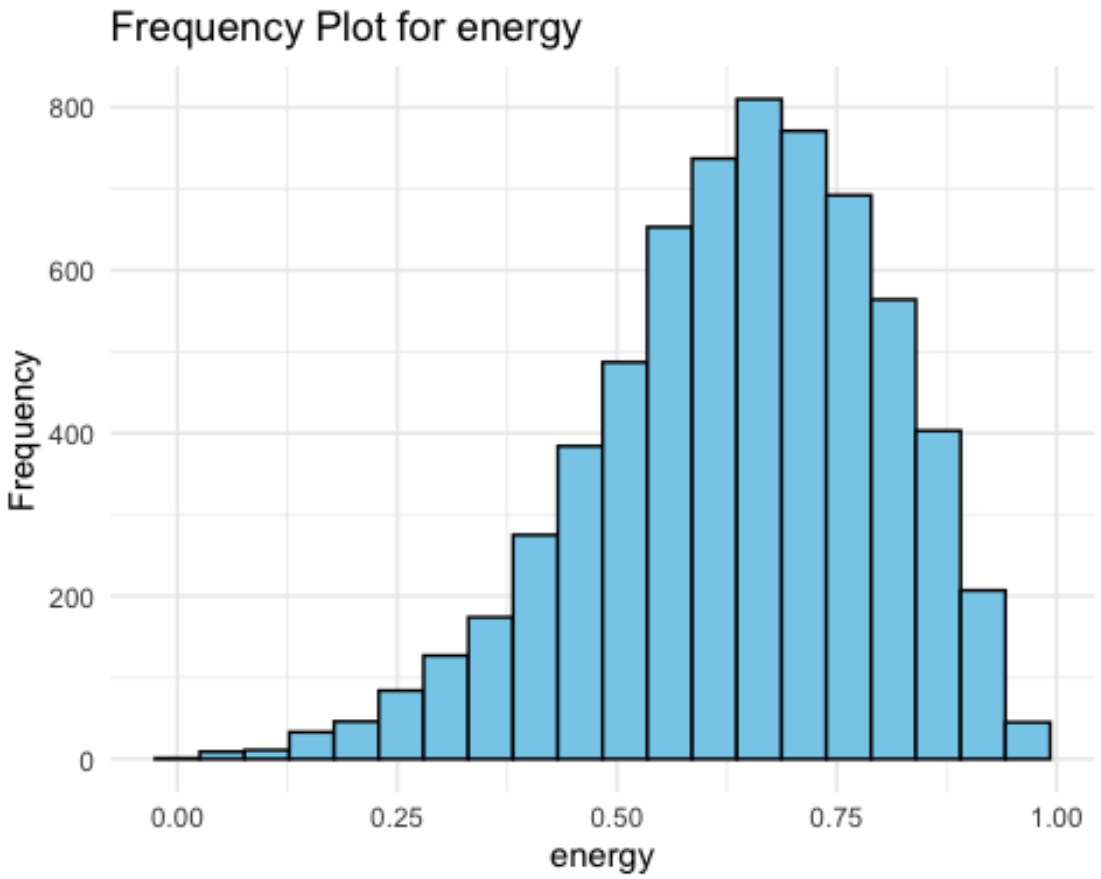
The first step in my exploratory data analysis is checking my data set for NA values and duplicates. Missing data points can skew the analysis, and duplicates can distort findings. These checks set a solid foundation for my analysis, ensuring that the data is clean and reliable for deeper exploration.

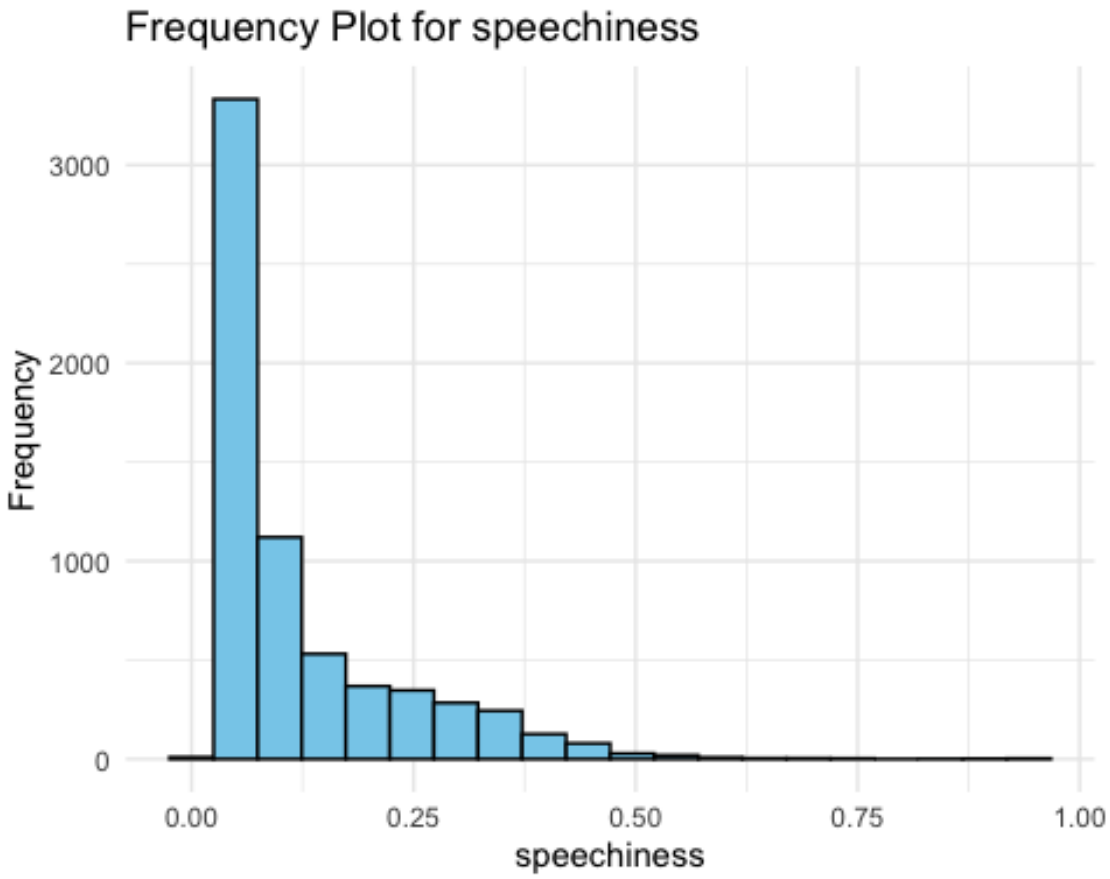
As seen above, there are no NA values or duplicates in this data set.

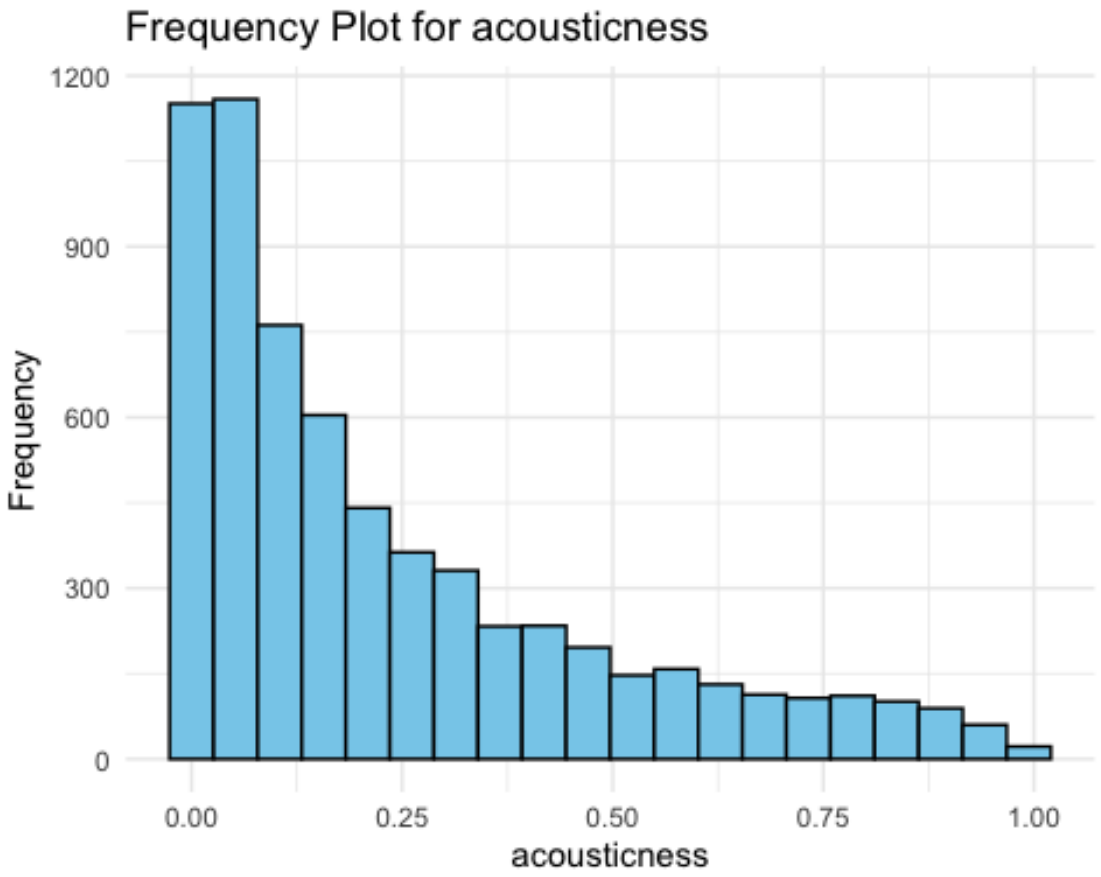
```
#Frequency plots for all the audio features
features_df <- top_songs[, 8:16]
features_names <- names(features_df)

for (feature in features_names) {
  p <- ggplot(top_songs, aes_string(x = feature)) +
    geom_histogram(bins = 20, fill = "skyblue", color = "black") +
    labs(x = feature, y = "Frequency") +
    ggtitle(paste("Frequency Plot for", feature)) +
    theme_minimal()

  print(p)  # Print the plot inside the loop
}
```

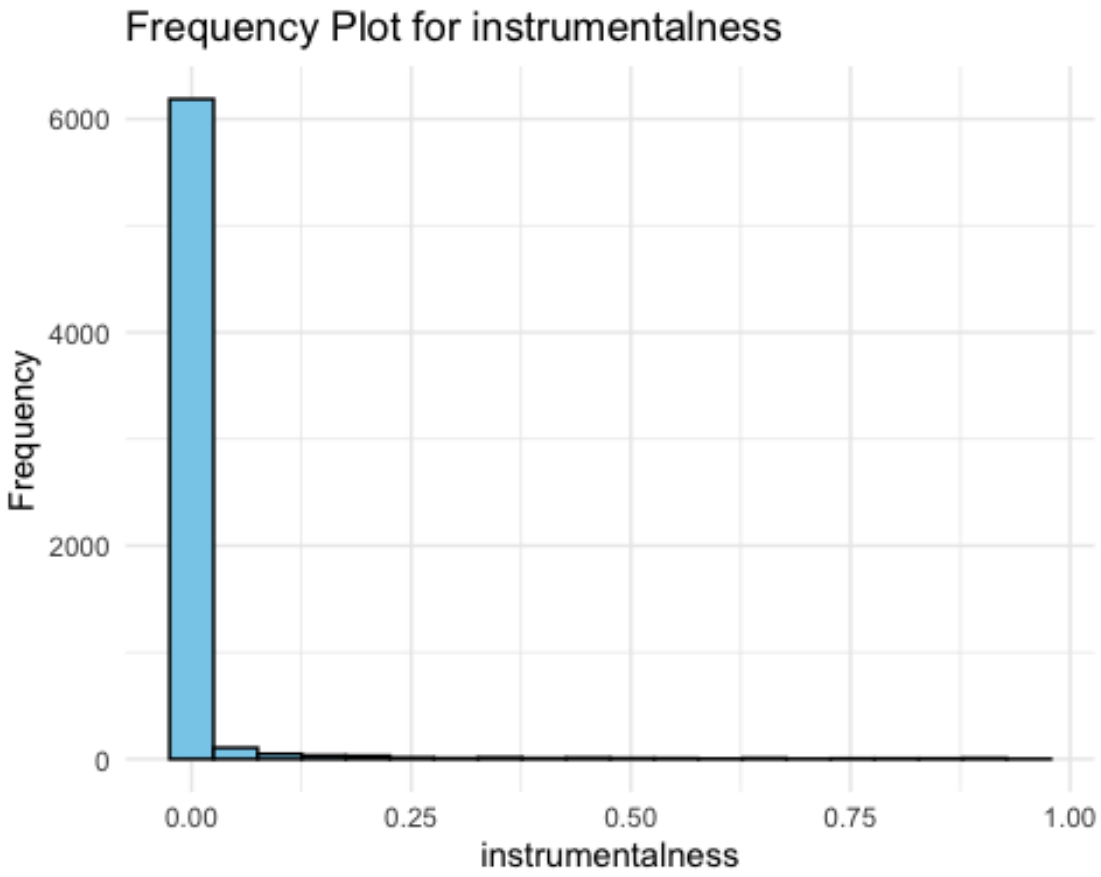
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this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

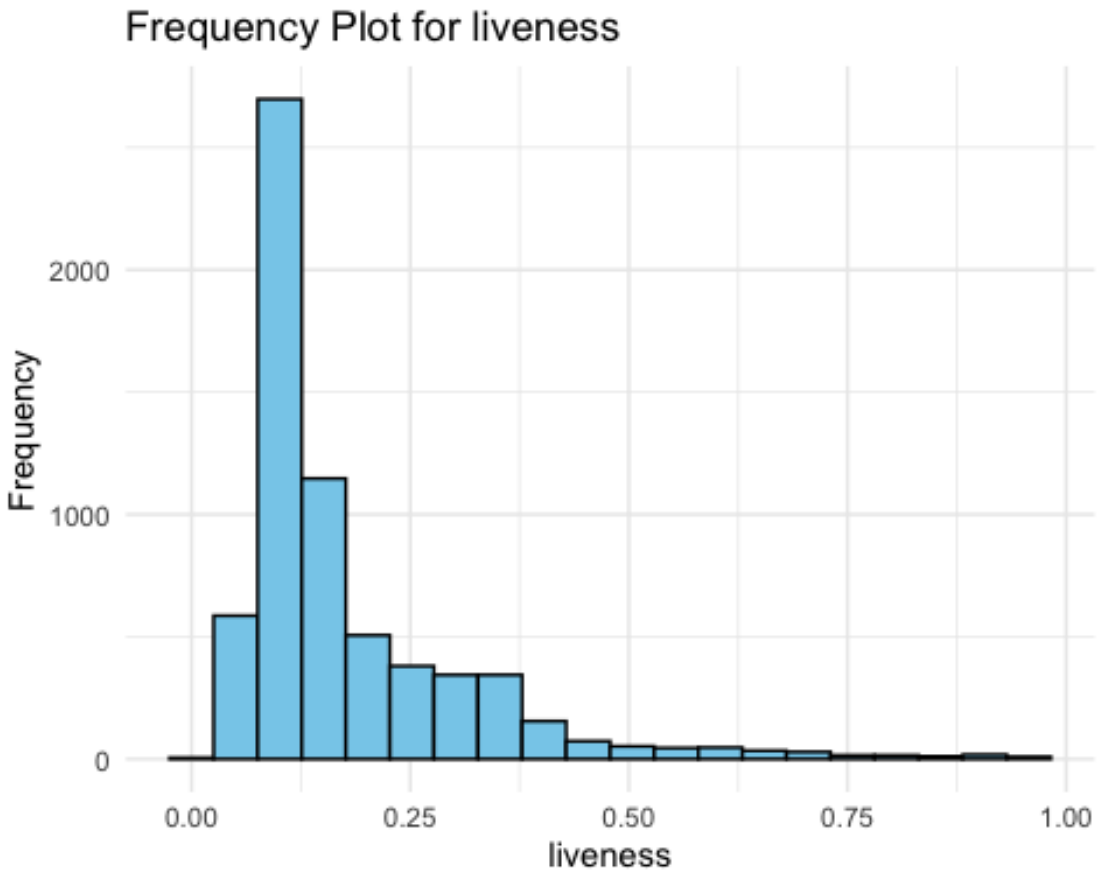

Frequency Plot for danceability
750
500
250
0
Frequency
0.25
0.50
0.75
1.00
danceability

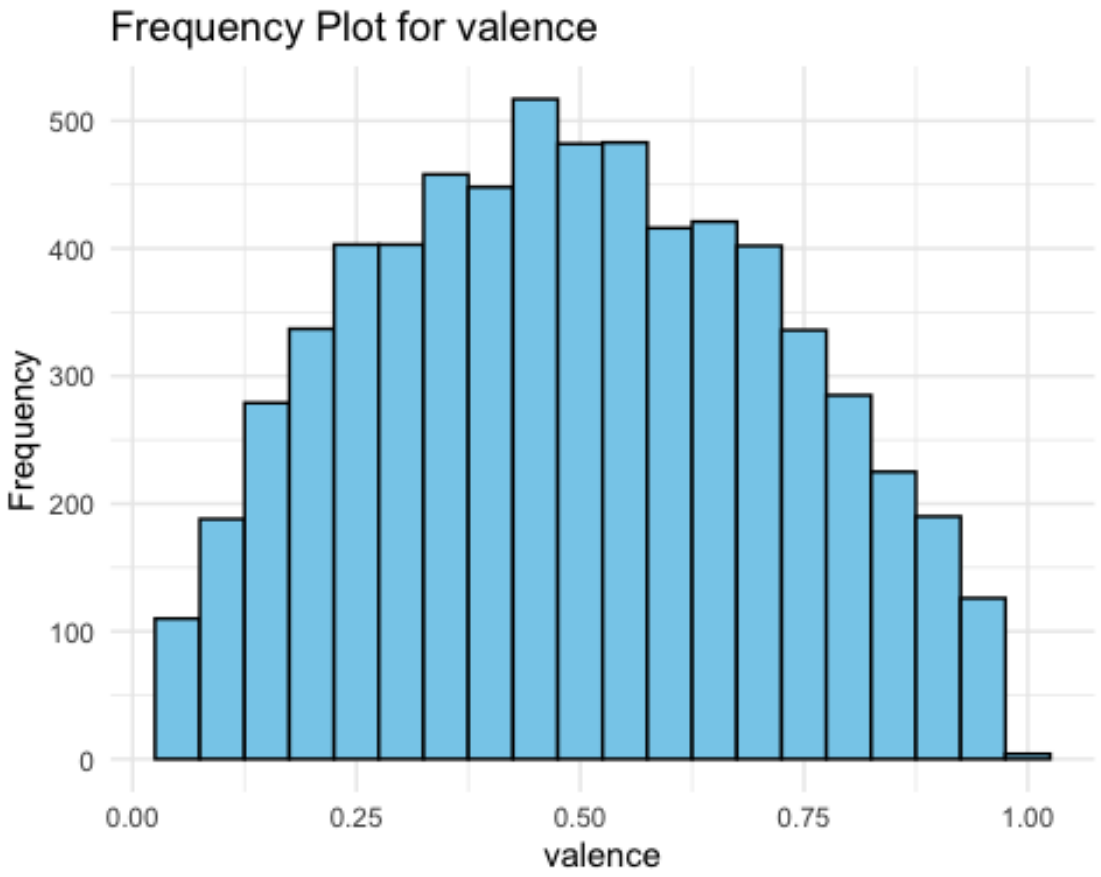

Frequency Plot for energy
800
600
400
200
0
Frequency
0.00
0.25
0.50
0.75
1.00
energy

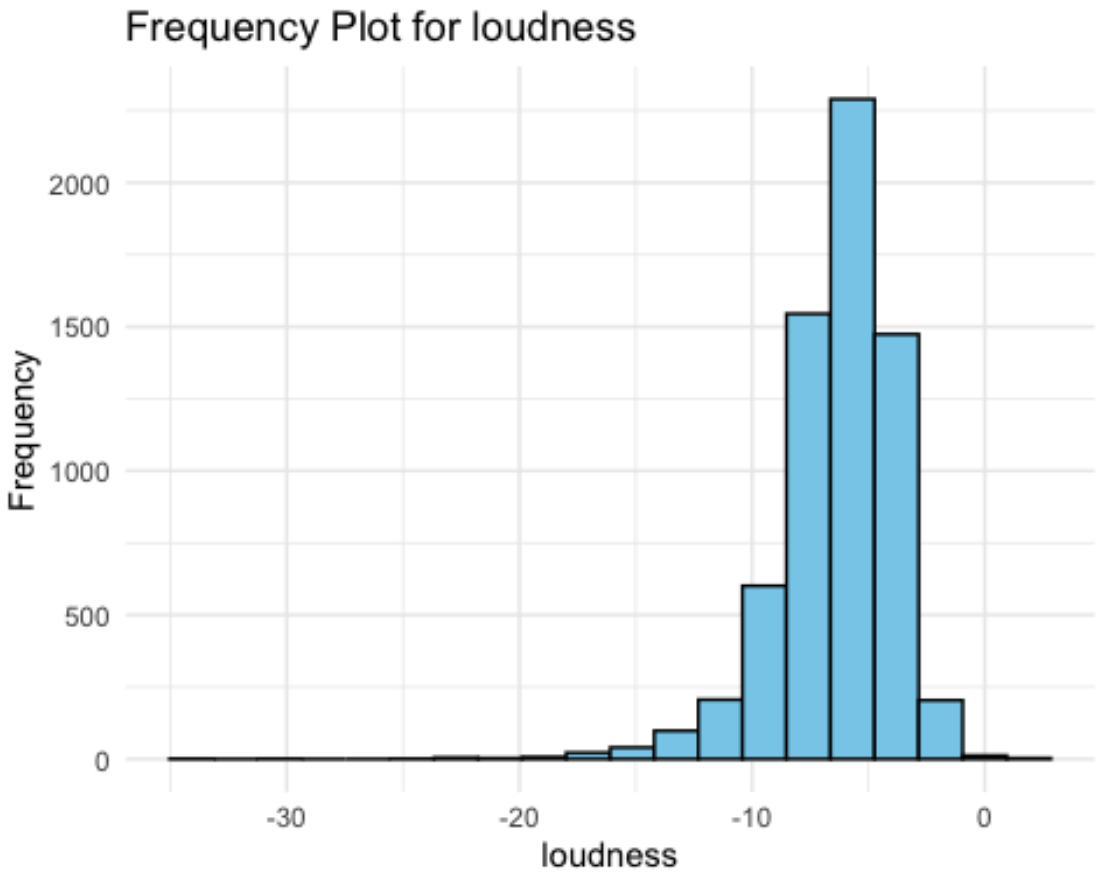

Frequency Plot for speechiness
3000
2000
1000
0
Frequency
0.00
0.25
0.50
0.75
1.00
speechiness

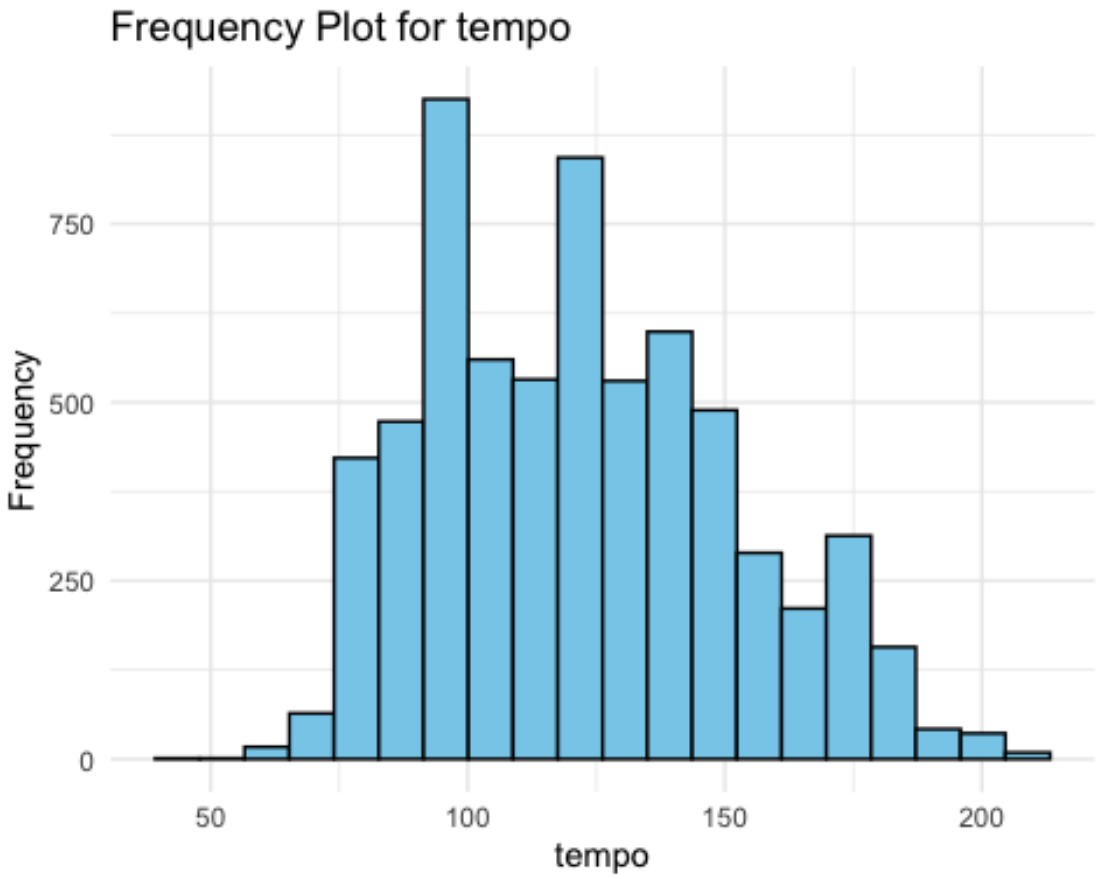

Frequency Plot for acousticness
1200
900
600
300
0
Frequency
0.00
0.25
0.50
0.75
1.00
acousticness


Frequency Plot for instrumentalness
6000
4000
2000
0
Frequency
0.00
0.25
0.50
0.75
1.00
instrumentalness


Frequency Plot for liveness
Frequency
2000
1000
0
0.00
0.25
0.50
0.75
1.00
liveness


Frequency Plot for valence
Frequency
500
400
300
200
100
0
0.00
0.25
0.50
0.75
1.00
valence


Frequency Plot for loudness
Frequency
2000
1500
1000
500
0
-30
-20
-10
0
loudness

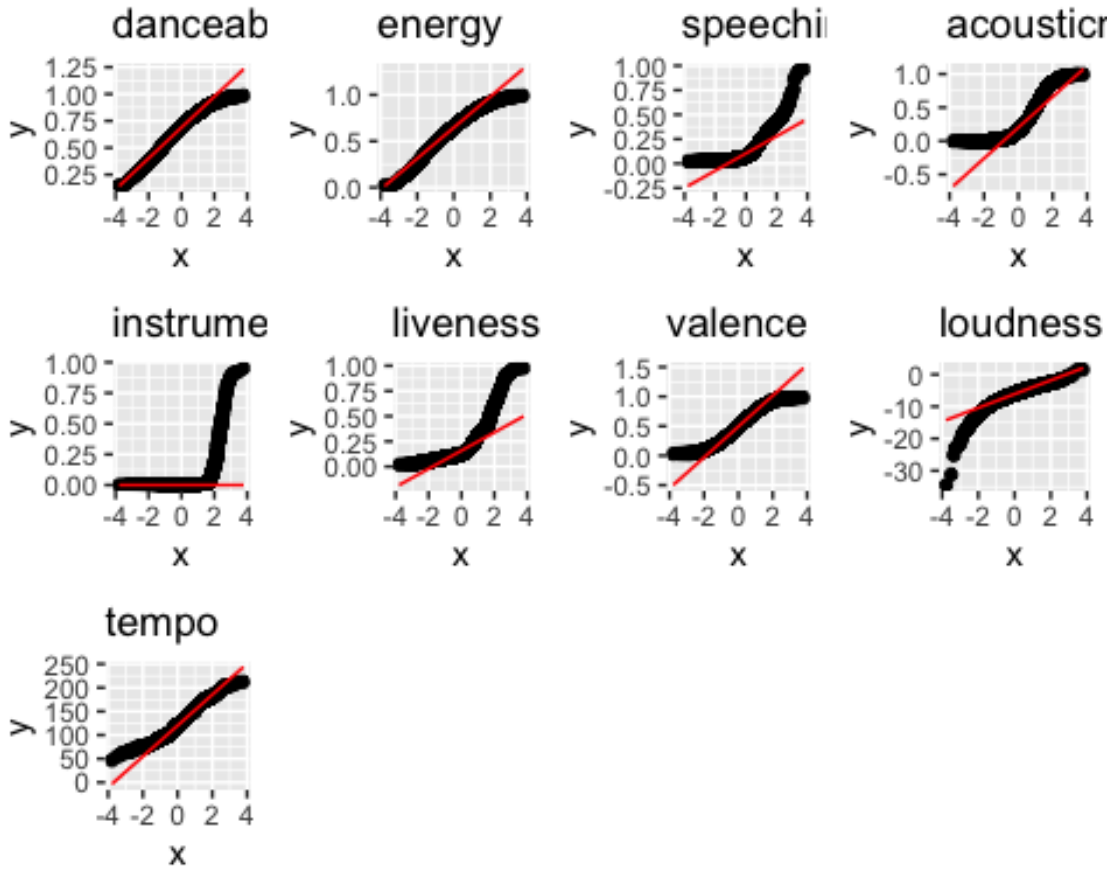


Frequency plots provide a visual representation of the distribution of variables within a data set, offering valuable insights into the characteristics of the data. In the context of audio features, such as danceability, energy, speechiness, acousticness, instrumentalness, liveness, valence, loudness, and tempo, these plots serve as powerful tools for understanding the inherent patterns and trends present in the music data.

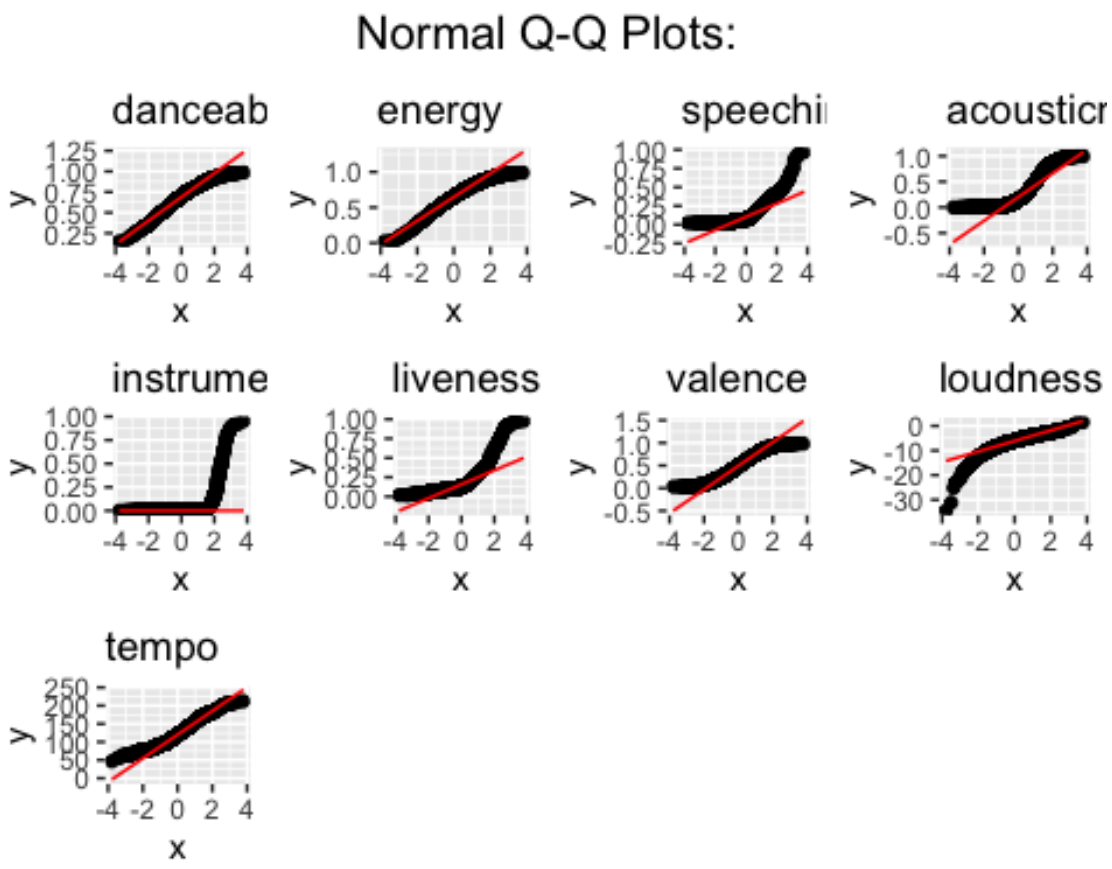
Among these features, valence stands out as the most normally distributed, with data points clustered around the mean of 0.5 and spanning the range between 0 and 1. Tempo, energy, and danceability exhibit distributions that appear somewhat normal but display slight skewness, potentially due to the presence of outliers in the data set. The remaining audio features demonstrate high levels of skewness in their distributions. This skewness suggests that these features are not evenly distributed across the data set and may be influenced by specific characteristics or attributes present in the music samples.

```
#Normal Q-Q plots for all the audio features
plots <- list()

for (feature in features_names) {
  p <- ggplot(top_songs, aes(sample = .data[[feature]])) +
    geom_qq() +
    geom_qq_line(color = "red") +
    labs(title = feature)
```

```
  plots[[feature]] <- p
}

grid_plot = grid.arrange(grobs = plots, ncol = 4)
```




```
grid_plot_with_title <- ggdraw() +
  draw_plot(grid_plot, width = 1, height = 0.9) +
  draw_label("Normal Q-Q Plots:", size = 15, x = 0.5, y = 0.95)

grid_plot_with_title
```



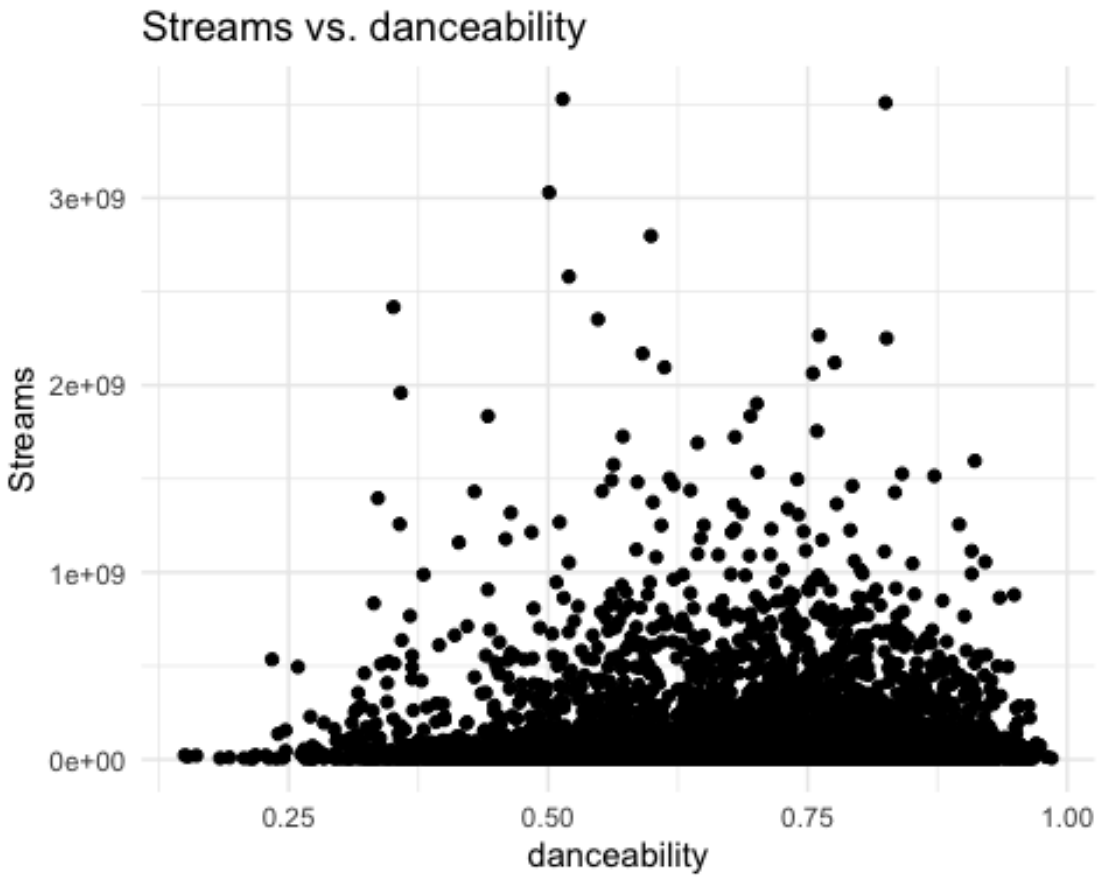
Normal QQ plots provide a graphical comparison between the distribution of data and a theoretical normal distribution. In the context of audio features such as danceability, energy, speechiness, acousticness, instrumentalness, liveness, valence, loudness, and tempo, these plots offer insights into how closely each feature follows a normal distribution.

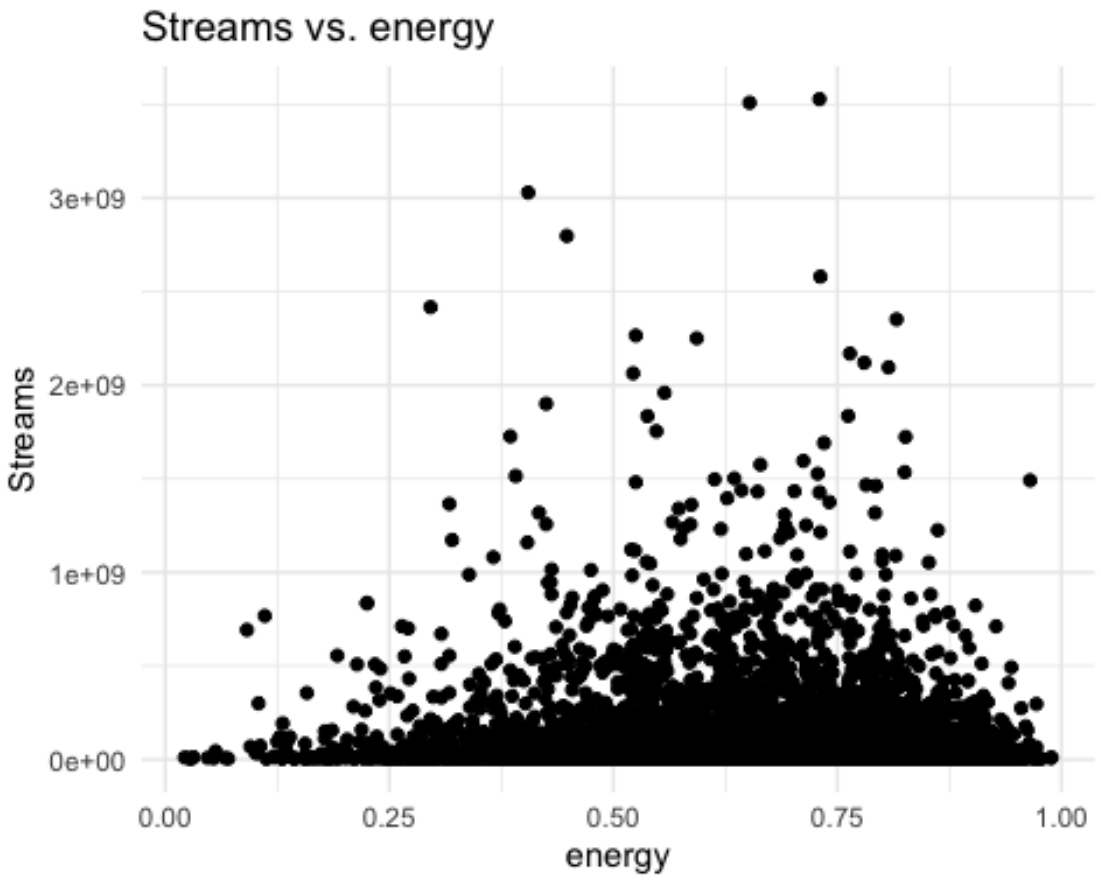
For tempo, loudnes, energy, danceability, and valence appear to closely adhere to a normal distribution, as evidenced by the points aligning closely along the diagonal line on the QQ plot. This suggests that their values are distributed in a manner consistent with a normal distribution.

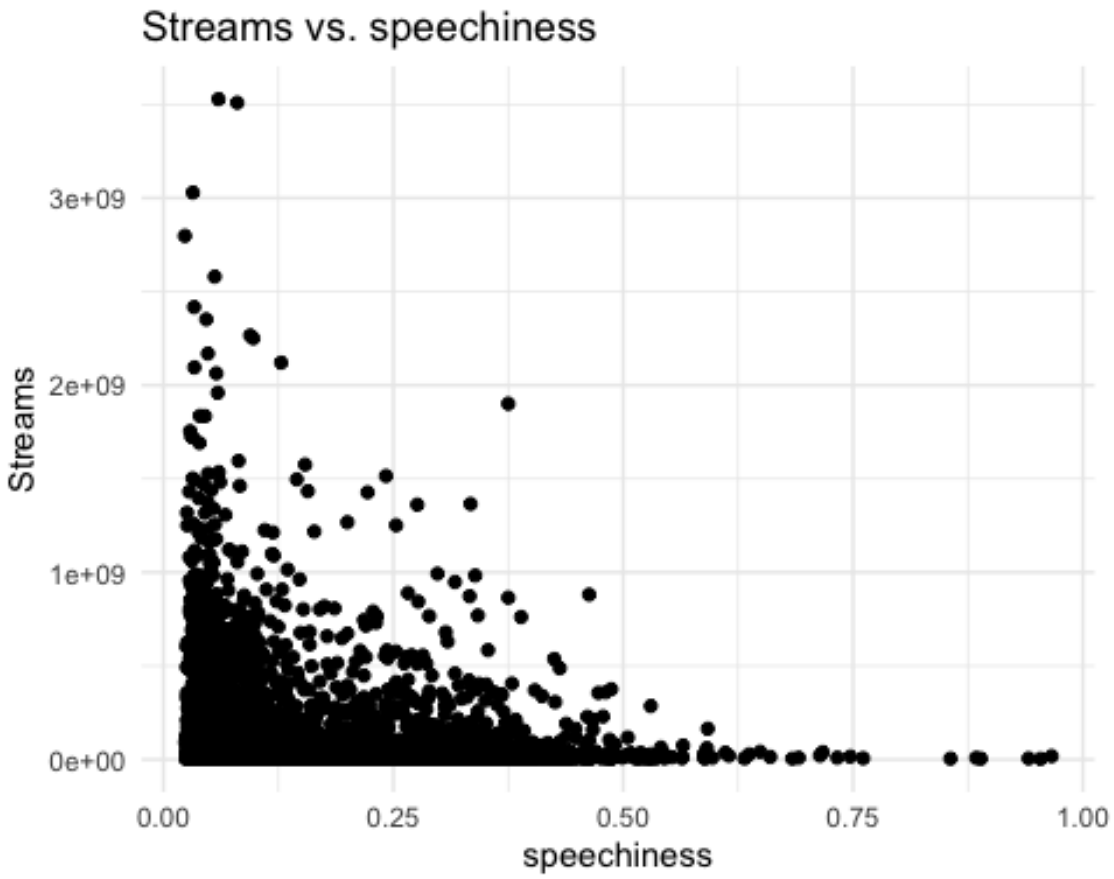
In contrast, the QQ plots for other audio features reveal more pronounced deviations from the diagonal line, indicating significant departures from normality. This suggests that these features are not distributed in a manner consistent with a normal distribution, highlighting potential complexities or biases within the data set.

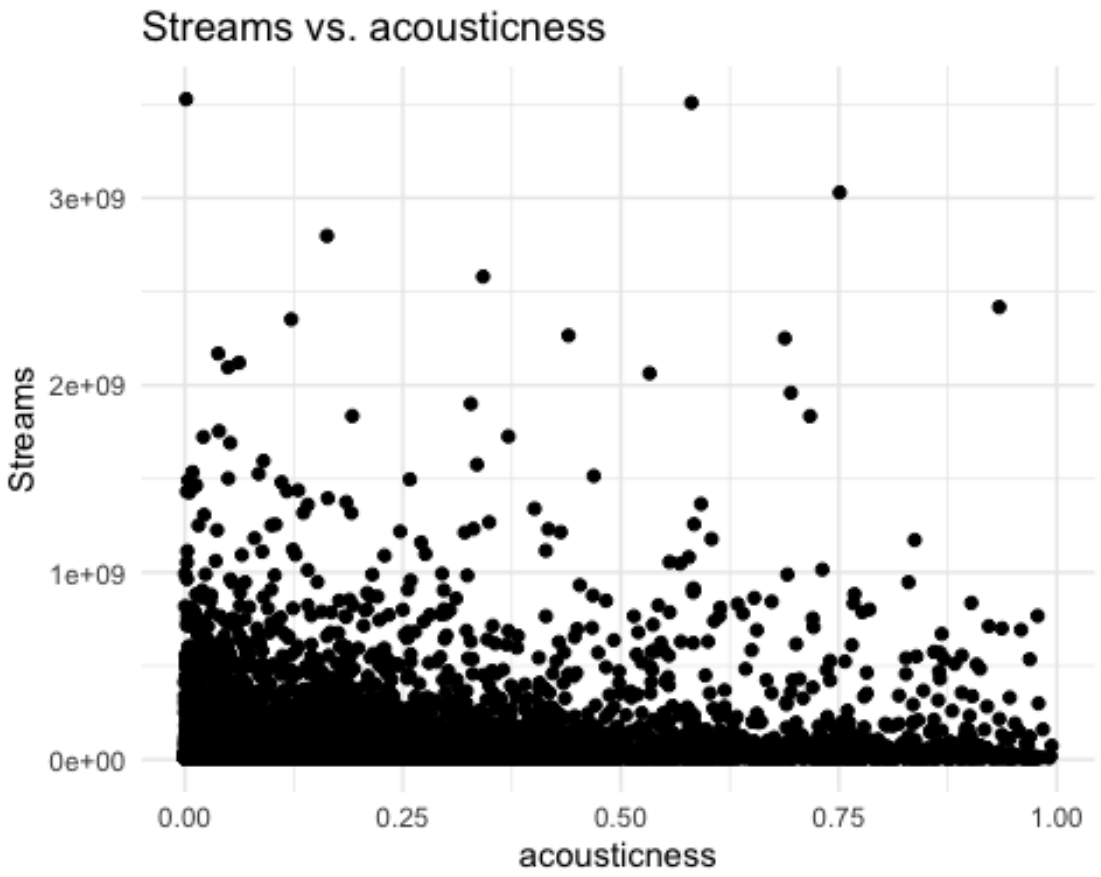
```
#Correlation plots for all audio features vs. streams
for (feature in features_names) {
  correlation_streams <- ggplot(top_songs, aes_string(x = feature, y =
'streams')) +
    geom_point() +
    scale_color_viridis(discrete = TRUE, option = "D") +
    labs(title = paste("Streams vs.", feature),
```

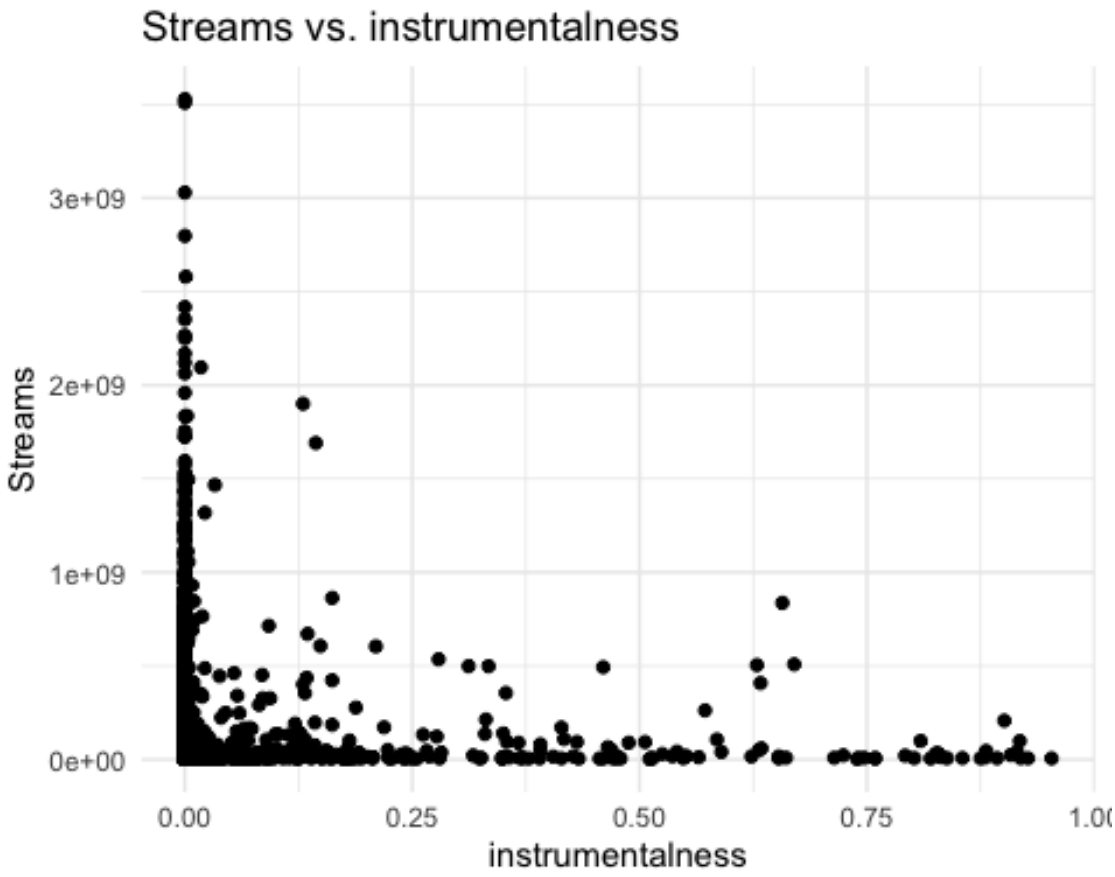
```
        x = feature,
        y = 'Streams') +
    theme_minimal() +
    theme(legend.position.inside = c(1, 1), legend.justification = c(1, 1))

  print(correlation_streams)
}
```

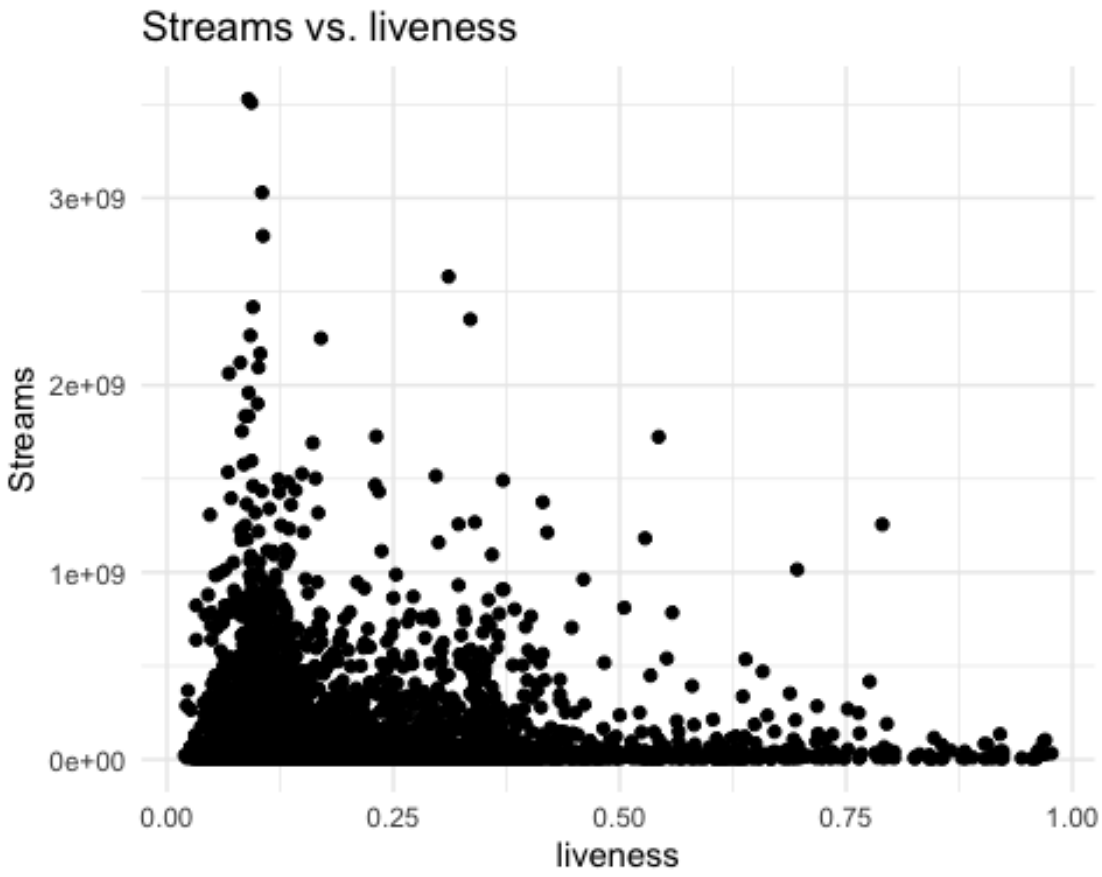

Streams vs. danceability
Streams
3e+09
2e+09
1e+09
0e+00
0.25
0.50
0.75
1.00
danceability

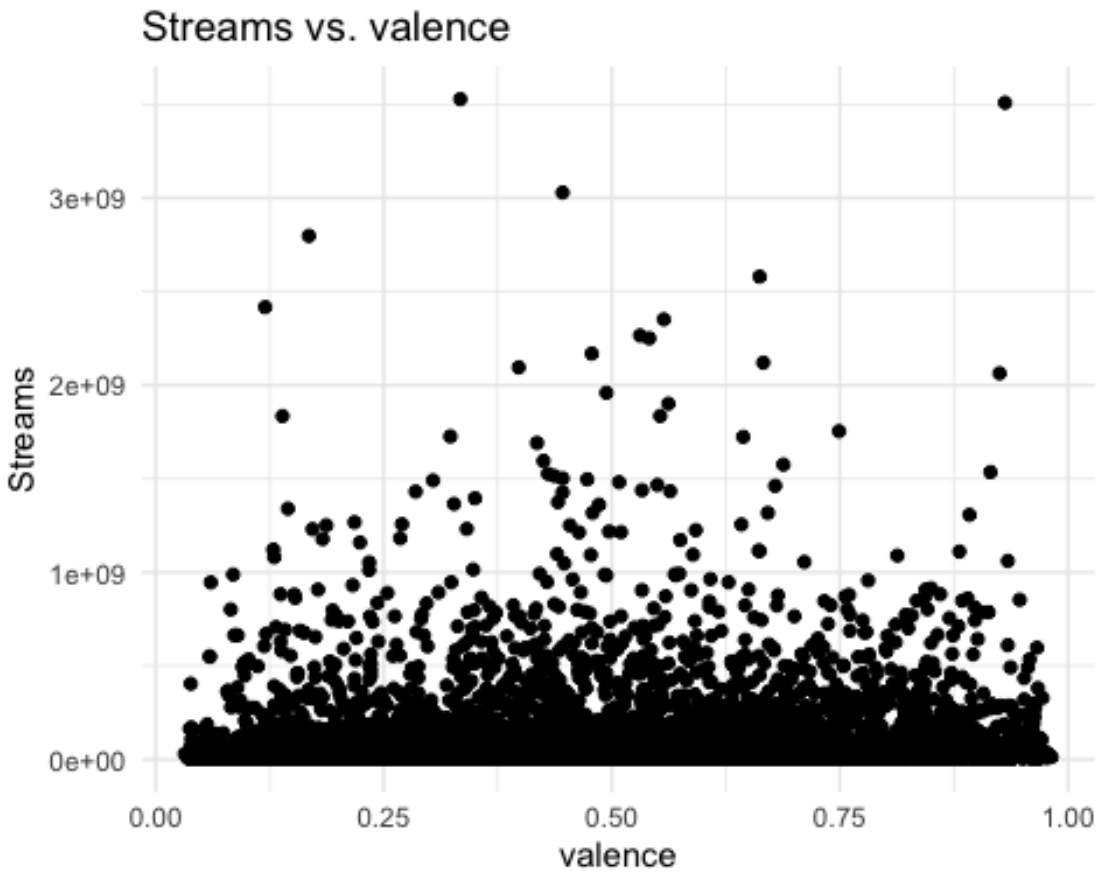

Streams vs. energy
Streams
3e+09
2e+09
1e+09
0e+00
0.00
0.25
0.50
0.75
1.00
energy

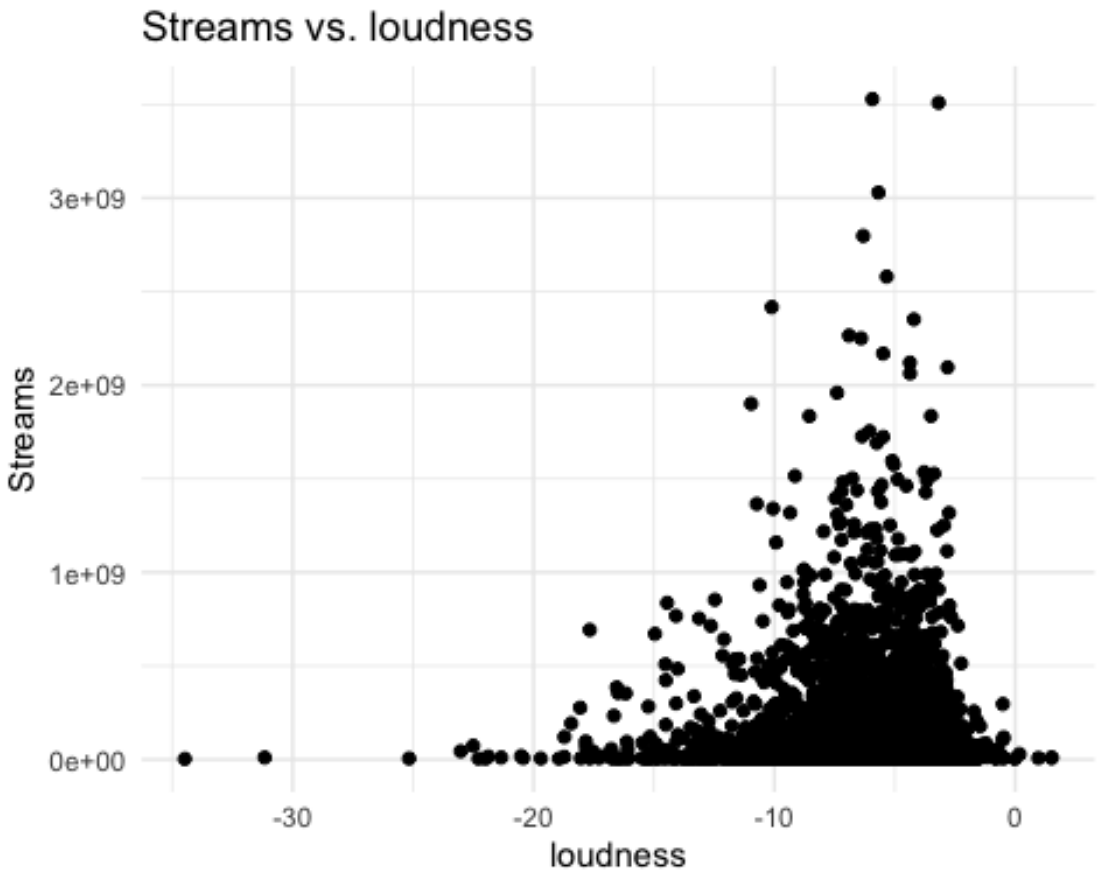

Streams vs. speechiness
Streams
3e+09
2e+09
1e+09
0e+00
0.00
0.25
0.50
0.75
1.00
speechiness

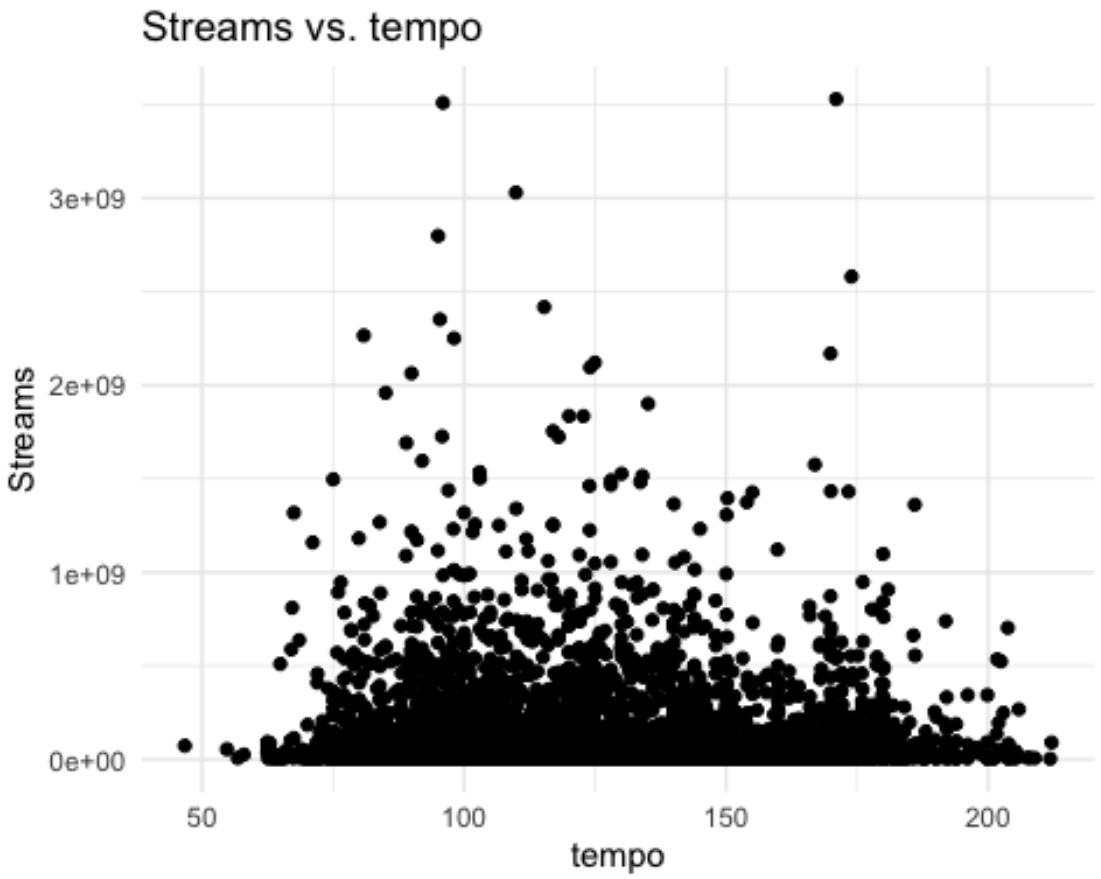

Streams vs. acousticness
Streams
3e+09
2e+09
1e+09
0e+00
0.00
0.25
0.50
0.75
1.00
acousticness


Streams vs. instrumentalness
Streams
3e+09
2e+09
1e+09
0e+00
0.00
0.25
0.50
0.75
instrumentalness


Streams vs. liveness
Streams
3e+09
2e+09
1e+09
0e+00
0.00
0.25
0.50
0.75
1.00
liveness


Streams vs. valence
Streams
3e+09
2e+09
1e+09
0e+00
0.00
0.25
0.50
0.75
1.00
valence


Streams vs. loudness
Streams
3e+09
2e+09
1e+09
0e+00
-30
-20
-10
0
loudness



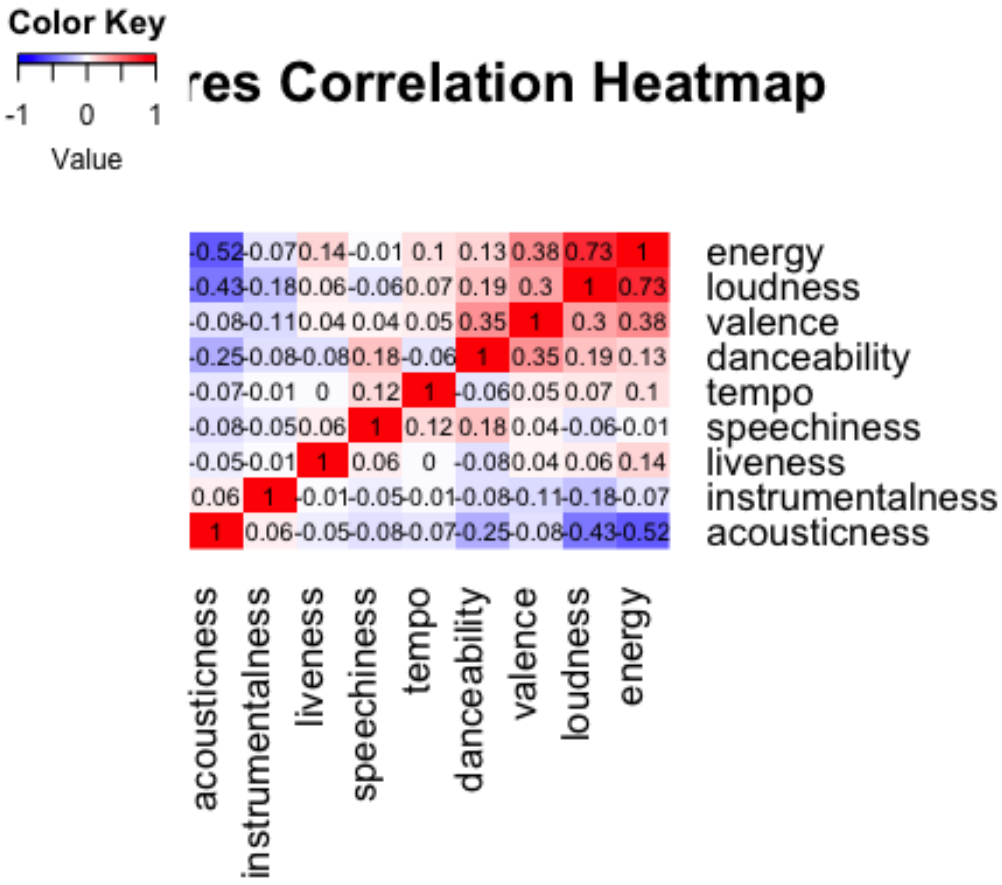
The correlation plots between features and streams provide valuable insights into the relationships between various attributes of the data set and the popularity of the music tracks, as measured by stream counts. These plots serve as visual representations of the degree and direction of correlation between each feature and the number of streams, guiding our understanding of the factors that contribute to a song's popularity.

Upon examination, several notable patterns emerge from the correlation plots. Firstly, features such as danceability, energy and loudness, exhibit a strong positive correlation with stream counts, indicating that songs with higher values of these features tend to attract more streams. Conversely, features like speechiness, instrumentalness, and liveness show a negative correlation with stream counts, suggesting that the higher values of these features attract less streams.

Furthermore, valence, acousticness, and tempo features display weak or negligible correlations with stream counts, indicating that they may have minimal influence on a song's popularity. These findings underscore the multifaceted nature of music preferences and consumption patterns, highlighting the importance of considering a diverse range of factors when assessing a song's appeal.

```
#Correlation matrix for all audio features
corr_matrix <- round(cor(top_songs[,features_names]), 2)
```

```r
heatmap.2(corr_matrix,
          col = colorRampPalette(c("blue", "white", "red"))(100),
          symm = TRUE,
          trace = "none",
          margins = c(9, 9),
          main = "Features Correlation Heatmap",
          cex.main = 1.2,
          cex.axis = 1.1,
          cex.lab = 1.1,
          key = TRUE,
          keysize = 1.5,
          key.title = NULL,
          cellnote = corr_matrix,
          notecol = "black",
          notecex = 0.8,
          density.info = "none",
          dendrogram = "none")
```



The correlation matrix reveals significant associations between various audio features, offering insights into the interplay of musical attributes within the data set. Notably, the highest positive correlation of 0.73 is observed between energy and loudness. This strong positive correlation suggests that songs with higher energy levels tend to be perceived as louder, reflecting a cohesive relationship between these two characteristics. This finding

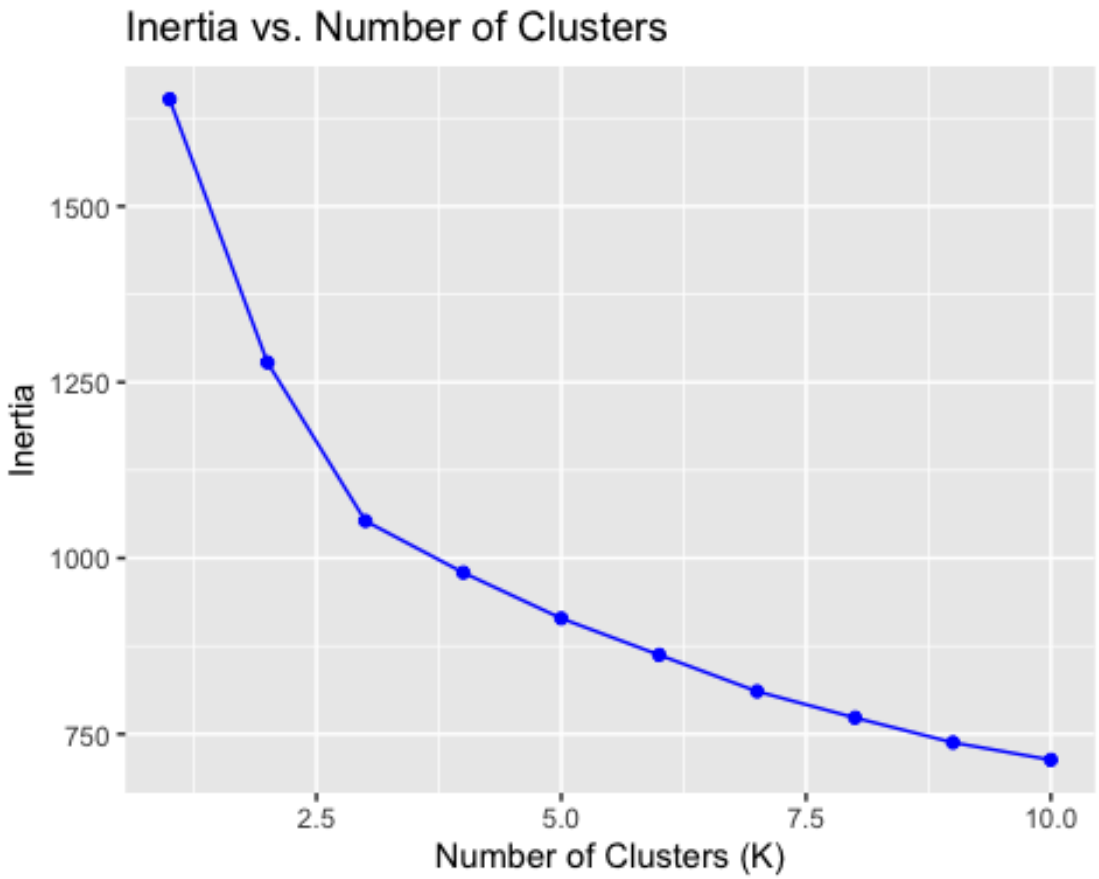
aligns with expectations, as energetic tracks often feature louder instrumentation and more dynamic arrangements, contributing to their perceptual intensity and impact.

Conversely, the highest negative correlation of -0.52 is identified between energy and acousticness. This substantial negative correlation implies an inverse relationship between energy levels and acousticness in music tracks. Specifically, songs characterized by high energy levels are more likely to exhibit lower acousticness, indicating a preference for electronically-produced or heavily processed sounds over acoustic instrumentation. This observation underscores the contrast between energetic, electrified music genres and more subdued, acoustic-oriented styles, reflecting diverse listener preferences and consumption patterns within the music landscape.

Cluster Analysis

```r
# Set seed for reproducibility
set.seed(123)

# Defining a function for min-max scaling
min_max_scale <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

# Scaling the features of the dataset
scaled_ts <- apply(top_songs[, features_names], 2, min_max_scale)
scaled_ts_df <- as.data.frame(scaled_ts)

# Initializing a vector to store inertia values
inertia <- numeric(length = 9)

# Defining the range of K (number of clusters)
K <- 1:10

# Looping over different values of K
for (k in K) {
  # Perform K-means clustering with the current value of K
  kmeans_model <- kmeans(scaled_ts_df, centers = k)
  # Store the total within-cluster sum of squares (inertia)
  inertia[k] <- kmeans_model$tot.withinss
}

# Creating a data frame to store the inertia values
inertia_data <- data.frame(K = K, inertia = inertia)

# Plotting inertia against the number of clusters (K)
ggplot(inertia_data, aes(x = K, y = inertia)) +
  geom_point(color = "blue") +  # Blue points representing inertia values
  geom_line(color = "blue") +    # Connect points with a blue line
  labs(title = "Inertia vs. Number of Clusters",  # Title and axis labels
```

```
       x = "Number of Clusters (K)",
       y = "Inertia")
```



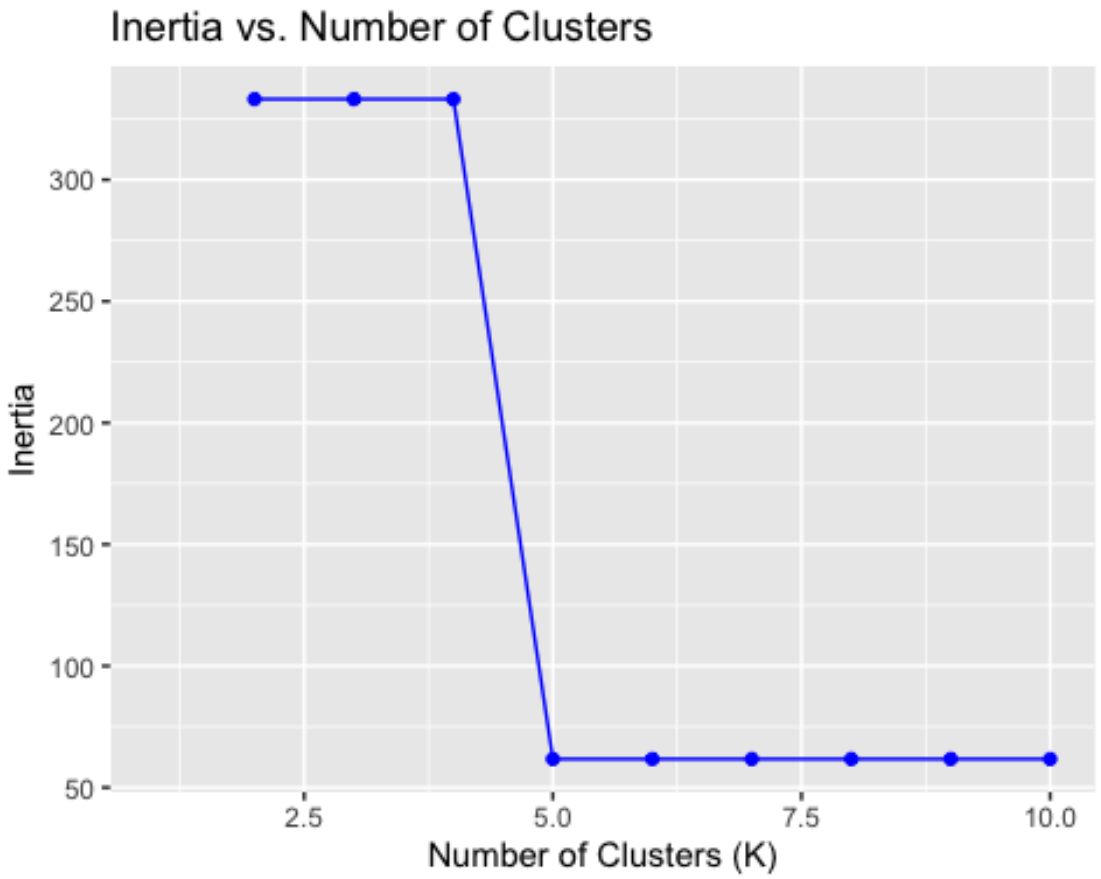
The resulting inertia plot depicts the relationship between the number of clusters (**K**) and the corresponding inertia values. Notably, as the number of clusters increases, the inertia tends to decrease. However, we observe a pivotal point, often referred to as the "elbow," between 2.5 and 5. This point signifies a balance between maximizing cluster granularity and minimizing redundancy. Around this point we might expect to find the ideal number of clusters.

```r
# Set seed for reproducibility
set.seed(123)

# Initializing a vector to store inertia values
inertia_hclust <- numeric(length = 9)

# Defining number of clusters (K)
K <- 1:10

for (k in K) {
  # Performing hierarchical clustering with the current value of K
  hclust_model <- hclust(dist(scaled_ts_df), method = "ward.D2")
```

```r
  # Cutting the dendrogram at k clusters
  cluster_labels <- cutree(hclust_model, k)

  # Storing the total within-cluster sum of squares (inertia)
  inertia_hclust[k] <- sum(hclust_model$height[which(cluster_labels == k)])
}

# Creating a data frame to store the inertia values
inertia_data_hclust <- data.frame(K = K, inertia = inertia_hclust)

# Plotting inertia against the number of clusters (K)
ggplot(inertia_data_hclust, aes(x = K, y = inertia)) +
  geom_point(color = "blue") +  # Blue points representing inertia values
  geom_line(color = "blue") +    # Connect points with a blue line
  labs(title = "Inertia vs. Number of Clusters",  # Title and axis labels
       x = "Number of Clusters (K)",
       y = "Inertia")
```

```
## Warning: Removed 1 row containing missing values or values outside the
scale range
## (`geom_point()`).
```

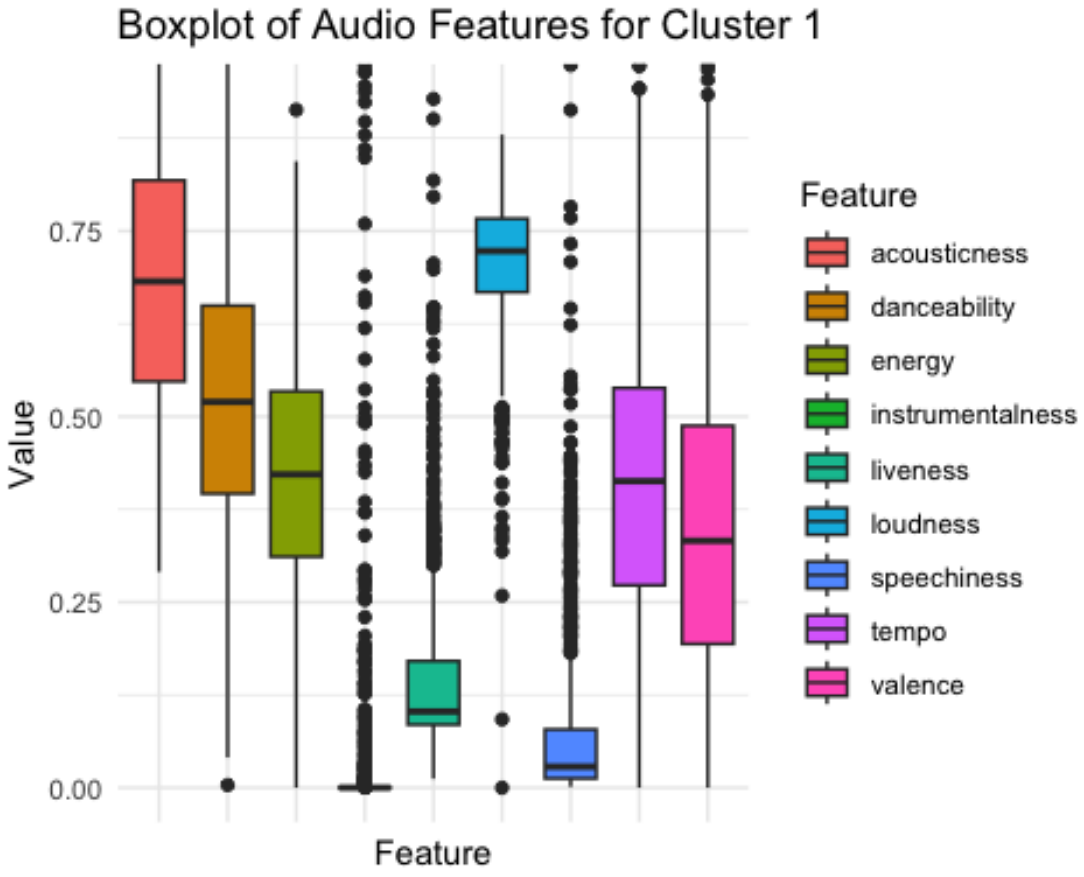
```
## Warning: Removed 1 row containing missing values or values outside the
scale range
## (`geom_line()`).
```

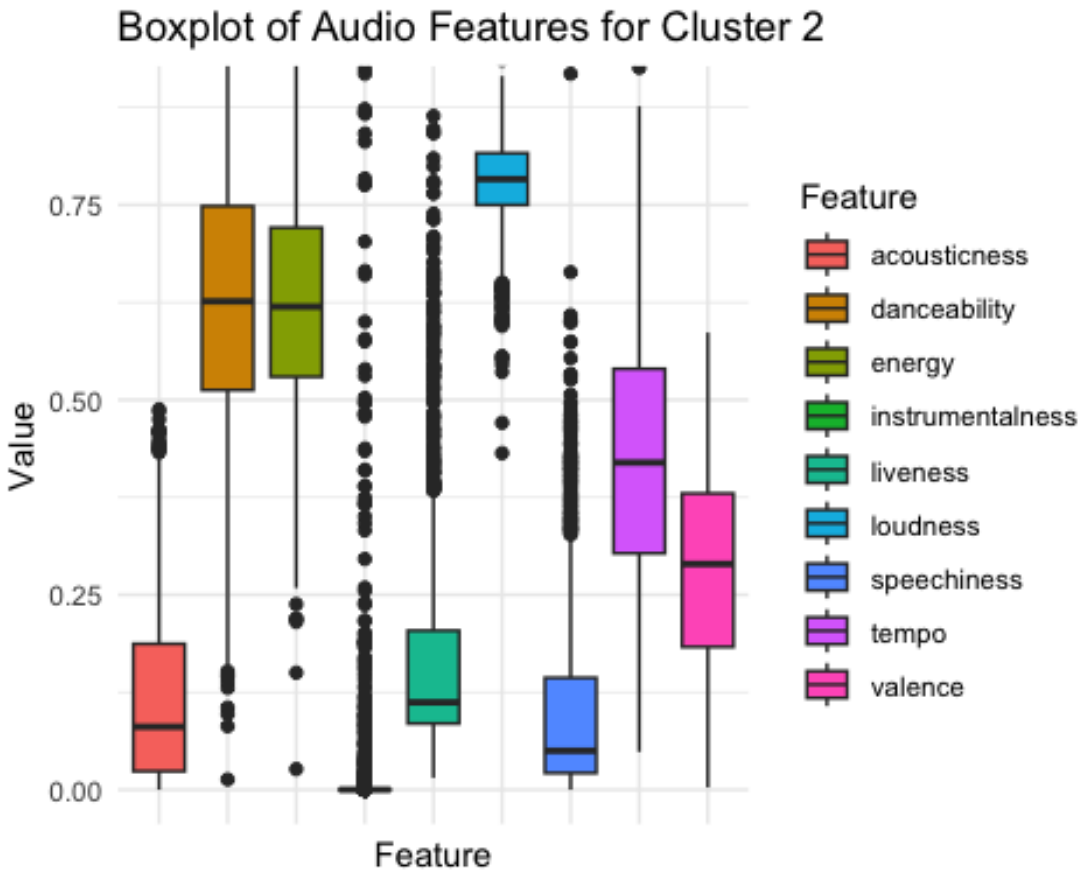


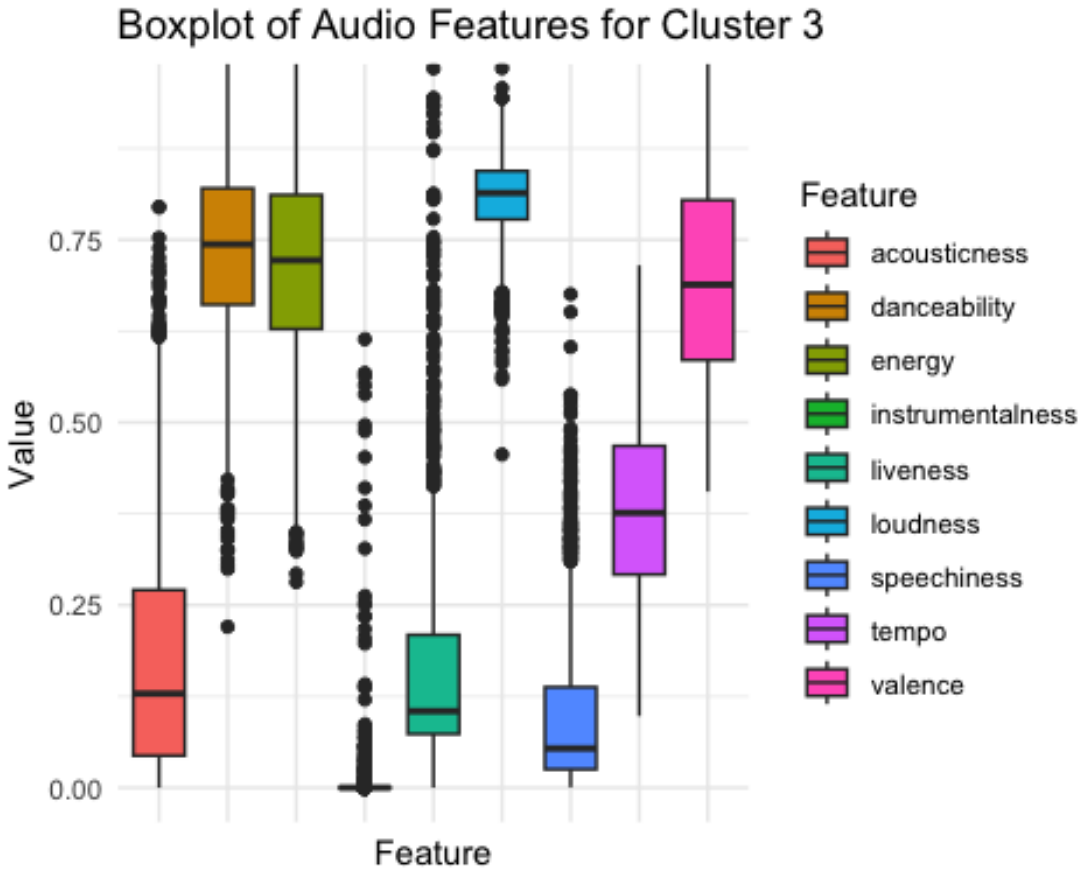
The resultant inertia plot exhibits a similar trend to that observed in the K-means clustering analysis. As the number of clusters increases, the inertia generally decreases, indicating improved cluster cohesion. Notably, a discernible point of inflection can be observed between the points 2.5 and 5. Again, this is the around point where we might expect to have ideal number of clusters.

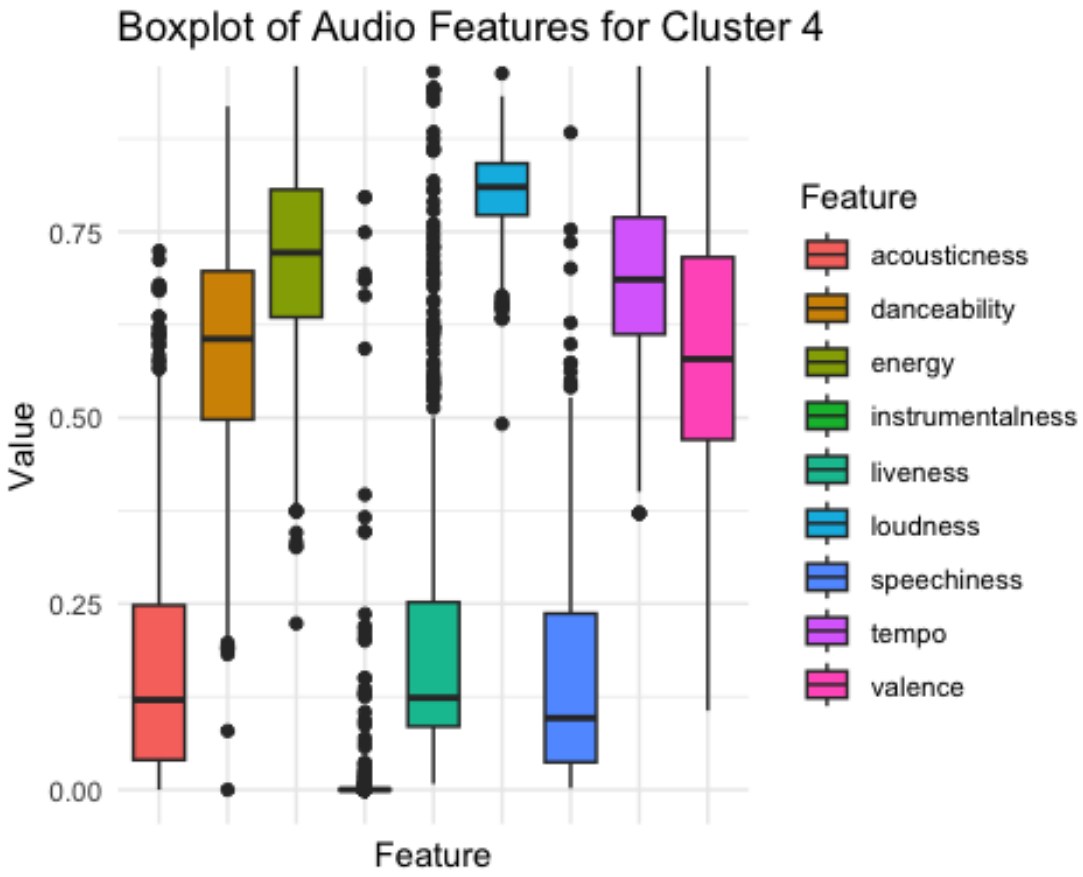
```r
# Set seed for reproducibility
set.seed(123)

# Setting the chosen number of clusters
num_clusters <- 4

# Computing k-means clustering
kmeans_result <- kmeans(scaled_ts_df, centers = num_clusters)

# Addding cluster labels to the original data frame
top_songs$cluster <- kmeans_result$cluster
scaled_ts_df$cluster <- kmeans_result$cluster

# Set seed for reproducibility
set.seed(123)

# Reshape the data to long format
```

```r
data_long <- pivot_longer(scaled_ts_df, cols = all_of(features_names),
names_to = "feature", values_to = "value")

# Get unique cluster labels
unique_clusters <- sort(unique(data_long$cluster))

# Iterate over unique clusters
for (i in unique_clusters) {
  # Subset data for the current cluster
  cluster_data <- subset(data_long, cluster == i)

  # Create boxplot for the current cluster
  p <- ggplot(cluster_data, aes(x = factor(feature), y = value, fill =
feature)) +
    geom_boxplot() +
    labs(x = "Feature", y = "Value", title = paste("Boxplot of Audio Features
for Cluster", i)) +
    theme_minimal() +
    theme(axis.text.x = element_blank()) +
    scale_fill_discrete(name = "Feature") +
    coord_cartesian(ylim = quantile(cluster_data$value, c(0.01, 0.99)))

  # Print the boxplot
  print(p)
}
```


Boxplot of Audio Features for Cluster 1
Value
0.75
0.50
0.25
0.00
Feature
Feature
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence


Boxplot of Audio Features for Cluster 2
Value
0.75
0.50
0.25
0.00
Feature
Feature
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence


Boxplot of Audio Features for Cluster 3
Value
0.75
0.50
0.25
0.00
Feature
Feature
acousticness
danceability
energy
instrumentalness
liveness
loudness
speechiness
tempo
valence



When attempting to plot these boxplots, I noticed that my data was not formatted in an effective way for plotting. After many tries to plot them, I ended up aksing ChatGPT for different ways of transforming the data in order to simplify the method of plotting. The reshaping to a long format is beneficial for clustering algorithms as it organizes the data into a tidy structure where each row represents a unique combination of a feature and its corresponding value. This format facilitates the application of clustering algorithms by providing a consistent and uniform representation of the data, enabling more straightforward analysis and interpretation of cluster patterns and characteristics.

After choosing 4 as the ideal number of clusters, I generated boxplots to provide a visual depiction of the distribution of audio features within each cluster. Upon examination of the boxplots, notable patterns emerge regarding the distribution of audio features across clusters. Certain clusters exhibit distinct feature profiles characterized by pronounced differences in feature values compared to other clusters.

- **Cluster 1:** This cluster shows the highest mean for acousticness, suggesting a prevalence of songs with more acoustic elements. This characteristic aligns with genres like folk or acoustic-driven music genres.
- **Cluster 4:** In contrast, Cluster 4 stands out with higher means for valence compared to other clusters. This indicates a tendency towards more positive or upbeat

emotional content in the songs belonging to this cluster, which may be associated with genres like pop or dance music.

Variability Across Clusters:

- While the means of tempo, liveness, and speechiness exhibit consistency across clusters, their variances vary slightly. This suggests that while these features may have similar average values across clusters, the range of values within each cluster may differ, indicating potential nuances in song characteristics.
- Danceability and energy also exhibit slight variations across clusters, implying differences in the rhythmic and energetic qualities of songs within each cluster. These variations may reflect genre-specific preferences or stylistic conventions.

Genre Labeling

```
#Creating the subsets of each cluster audio features excluding (acousticness,
instrumentalness,liveness)
features_to_exclude <- c("acousticness", "instrumentalness", "liveness")

# Subset the data to exclude the features
scaled_ts_df_sub <- scaled_ts_df[, !names(scaled_ts_df) %in%
features_to_exclude]

cluster_data_1 <- subset(scaled_ts_df_sub, cluster == 1)
cluster_data_2 <- subset(scaled_ts_df_sub, cluster == 2)
cluster_data_3 <- subset(scaled_ts_df_sub, cluster == 3)
cluster_data_4 <- subset(scaled_ts_df_sub, cluster == 4)

#Calculating the means of the audio features
means1 <- colMeans(cluster_data_1)
means2 <- colMeans(cluster_data_2)
means3 <- colMeans(cluster_data_3)
means4 <- colMeans(cluster_data_4)

# Print the means in a table format
means_clusters <- data.frame(Mean1 = means1,Mean2 = means2,Mean3 =
means3,Mean4 = means4)
means_clusters
```

```
##                    Mean1     Mean2      Mean3     Mean4
## danceability 0.51646594 0.6213085 0.73595129 0.5915954
## energy       0.42267923 0.6228969 0.71272599 0.7153923
## speechiness  0.07650313 0.1019897 0.09892216 0.1481108
## valence      0.35574305 0.2807099 0.69815802 0.5951206
## loudness     0.70686009 0.7799235 0.80807753 0.8052733
## tempo        0.41981457 0.4246761 0.38296692 0.6908606
## cluster      1.00000000 2.0000000 3.00000000 4.0000000
```

- **Danceability:** Cluster 3 stands out with the highest mean danceability score, indicating a preference for songs with higher dance appeal within this cluster.
- **Energy:** Clusters 3 and 4 exhibit relatively high mean energy scores, suggesting a tendency towards more energetic songs compared to Clusters 1 and 2.
- **Speechiness:** Cluster 4 has the highest mean speechiness, indicating a higher presence of spoken words or vocal content compared to other clusters.
- **Valence:** Cluster 3 shows the highest mean valence score, indicating a prevalence of more positive or upbeat emotional content in the songs belonging to this cluster.
- **Loudness:** Clusters 3 and 4 demonstrate higher mean loudness values, suggesting a preference for louder or more dynamically intense songs within these clusters.
- **Tempo:** Cluster 4 exhibits the highest mean tempo, indicating a preference for faster-paced songs within this cluster.

According to Navoda Senavirathne's research on Music Genre Prediction Using Audio feature, the scaled values of the audio features represent the following genres;

- **Danceability** — Hip hop has the highest danceability while classical has the lowest.
- **Energy** — Metal has the highest energy while classical has the lowest.
- **Loudness** — Metal has the highest loudness while classical has the lowest.
- **Tempo** — Metal music reports the highest average tempo while classical reports the lowest.
- **Valance** — Blues has the highest valance and classical has the lowest.
- **Speechiness —** Hip-hop has the highest speechiness while country music has the lowest.

Using this research, here are the suggested genres

1. **Cluster 1:** Classical Music
    - This cluster likely corresponds to classical music due to its lower energy, loudness, and potentially lower tempo and speechiness compared to other clusters.
2. **Cluster 2:** Pop Music
    - This cluster might represent pop music, as it shows moderate values across various audio features, suggesting a versatile and mainstream music style.
3. **Cluster 3:** Hip-Hop/Rap
    - This cluster could be associated with hip-hop or rap music, given its higher danceability, energy, tempo, and potentially higher speechiness.

4. **Cluster 4:** Rock/Metal Music

   - This cluster may correspond to rock or metal music, as it exhibits higher energy, tempo, and potentially higher valence, which are characteristic features of these genres.

```
#Extracting the songs names and artists from each cluster
Classical <- subset(top_songs[2:3], scaled_ts_df$cluster == 1)
Pop <- subset(top_songs[2:3], scaled_ts_df$cluster == 2)
HipHop_Rap <- subset(top_songs[2:3], scaled_ts_df$cluster == 3)
Rock_Metal <- subset(top_songs[2:3], scaled_ts_df$cluster == 4)

head(Classical)
```

```
##                                                    artist_names
## 8  Tropa do Bruxo, DJ Ws da Igrejinha, SMU, Triz, Mc Menor Thalis
## 10                                                 Brent Faiyaz
## 17                                 Taylor Swift, Phoebe Bridgers
## 28                                                   Mac Miller
## 37                                                  Calum Scott
## 41                                                     NewJeans
##                                                                 track_name
## 8                                                         Baile do Bruxo
## 10                                                          LOOSE CHANGE
## 17 Nothing New (feat. Phoebe Bridgers) (Taylor’s Version) (From The Vault)
## 28                                                  Come Back to Earth
## 37                                                   Dancing On My Own
## 41                                                       Cool With You
```

```
head(Pop)
```

```
##          artist_names           track_name
## 3         The Killers       Mr. Brightside
## 7  Kygo, Donna Summer            Hot Stuff
## 9          YBN Nahmir Bounce Out With That
## 14         Zach Bryan  East Side of Sorrow
## 15           Dua Lipa       Break My Heart
## 18          Lil Tecca         Did It Again
```

```
head(HipHop_Rap)
```

```
##                 artist_names                            track_name
## 1        ZAYN, PARTYNEXTDOOR Still Got Time (feat. PARTYNEXTDOOR)
## 5    Post Malone, The Weeknd      One Right Now (with The Weeknd)
## 6      Thalia, NATTI NATASHA                         No Me Acuerdo
## 12           Jax Jones, RAYE       You Don't Know Me (feat. RAYE)
## 13      KZA Produções, wonda                          Maria Mariah
## 16 ScHoolboy Q, Travis Scott        CHopstix (feat. Travis Scott)
```

```
head(Rock_Metal)
```

```
##                            artist_names
track_name
## 2                          Alessia Cara                        Growing
Pains
## 4            Cardi B, Chance the Rapper Best Life (feat. Chance The
Rapper)
## 11 Chance the Rapper, MadeinTYO, DaBaby                         Hot
Shower
## 20                                  BTS
Paradise
## 21                          sangiovanni
malibu
## 29     Marracash, thasup, Sfera Ebbasta                   SUPREME -
L'ego
```

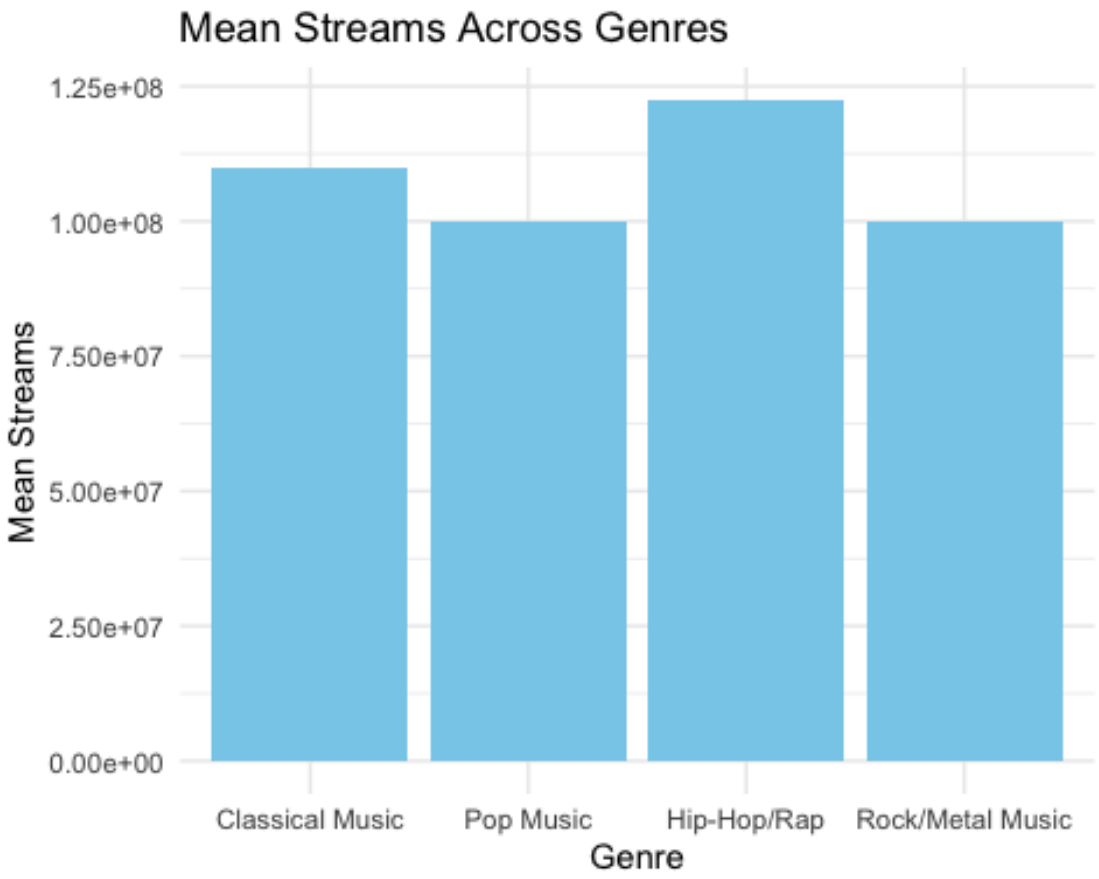
By printing out the names of the songs and artists, I've undertaken a manual inspection of the clusters. Leveraging my own knowledge of music and genres, I've been able to validate the genre assumptions to a certain extent. I've noticed distinct differences in song types and genres across the clusters, which align with the expected characteristics based on the audio features.

However, it's important to note that the assumptions are somewhat limited due to the finite number of clusters. While the identified clusters represent certain genres well, there are instances where multiple genres coexist within a single cluster. This suggests that the data we have is not optimally conducive to clustering, as the "ideal" number of clusters, determined by k-means and hierarchical clustering, did not fully account for the multitude of genres present in the dataset. Consequently, there are cases where distinct genres are grouped together in a single cluster.

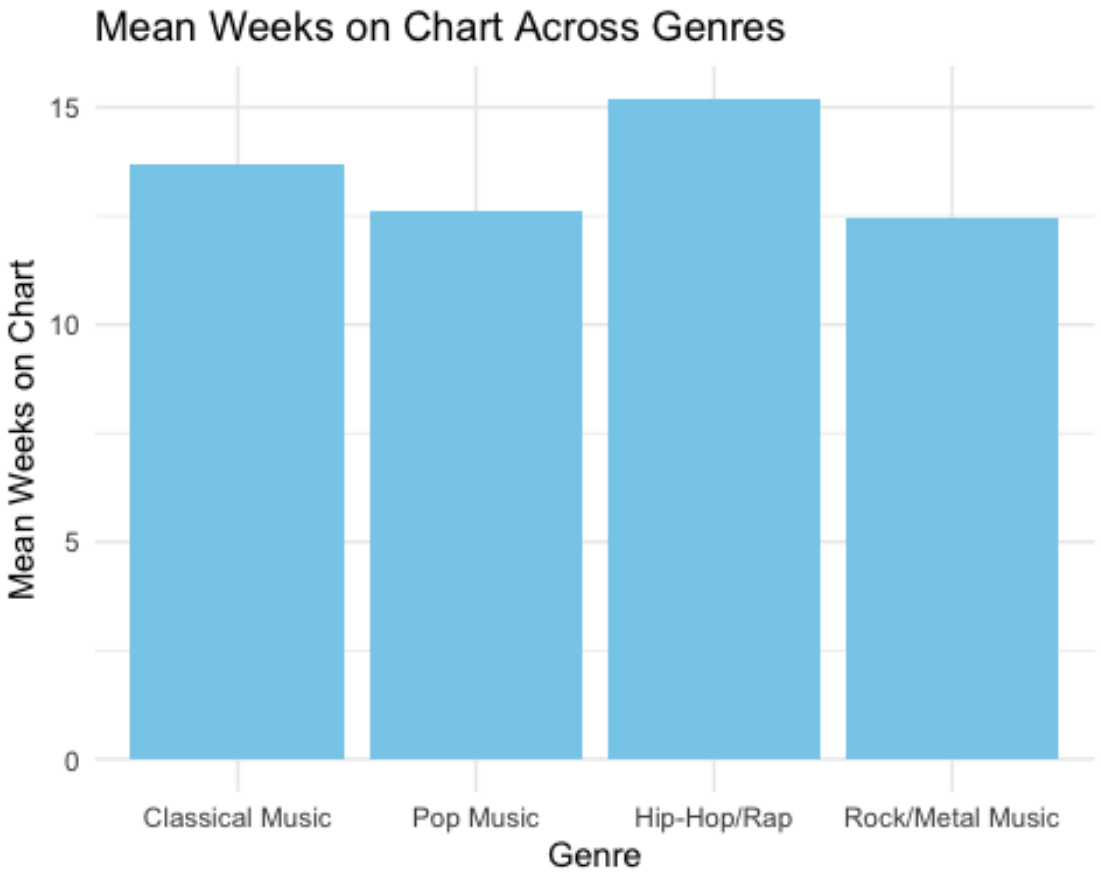
Nevertheless, despite these limitations, the genres within each cluster exhibit notable similarities, indicating that clustering can still provide valuable insights into genre relationships. With a larger and more diverse dataset, we could potentially achieve more refined clustering results that accurately capture the nuances of each genre. Thus, while the current clustering may not be perfect, it demonstrates the potential validity of the approach, especially when applied to a more comprehensive dataset encompassing a broader spectrum of music genres.

```r
# Calculating mean streams for each cluster
cluster_summary <- top_songs %>%
  group_by(cluster) %>%
  summarise(mean_streams = mean(streams))

# Defining genre labels
genre_labels <- c("Classical Music", "Pop Music", "Hip-Hop/Rap", "Rock/Metal
Music")

# Creating a bar plot with genre labels on the x-axis
ggplot(cluster_summary, aes(x = factor(cluster), y = mean_streams)) +
  geom_bar(stat = "identity", fill = "skyblue") +
```

```
  labs(x = "Genre", y = "Mean Streams", title = "Mean Streams Across Genres")
+
  theme_minimal() +
  scale_x_discrete(labels = genre_labels)
```




```
# Calculating mean weeks on chart for each cluster
cluster_summary1 <- top_songs %>%
  group_by(cluster) %>%
  summarise(mean_woc = mean(weeks_on_chart))

# Defining genre labels
genre_labels <- c("Classical Music", "Pop Music", "Hip-Hop/Rap", "Rock/Metal
Music")

# Creating a bar plot with genre labels on the x-axis
ggplot(cluster_summary1, aes(x = factor(cluster), y = mean_woc)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(x = "Genre", y = "Mean Weeks on Chart", title = "Mean Weeks on Chart
Across Genres") +
  theme_minimal() +
  scale_x_discrete(labels = genre_labels)
```



- Rock/Metal music appears to have the highest mean number of streams and weeks on charts, suggesting that songs within this cluster tend to be more popular or widely streamed compared to songs in other clusters.
- Conversely, Pop music exhibits the lowest mean streams and weeks on charts, indicating that songs in this cluster are comparatively less popular in terms of streaming metrics.

Conclusions

1. **Cluster Analysis Effectiveness:** The cluster analysis successfully identified distinct patterns in the audio features of songs, allowing for the differentiation of music genres. However, there were instances where multiple genres were grouped together within a single cluster, indicating limitations in the clustering approach due to the finite number of clusters chosen.
2. **Genre Identification:** By manually inspecting the clusters and leveraging domain knowledge of music genres, it was possible to validate the genre assumptions to a certain extent. Despite the limitations, the clusters exhibited notable similarities within each genre, suggesting the potential validity of the approach.
3. **Streaming Metrics:** The analysis of mean streams across clusters revealed variations in the popularity of songs within different genre clusters. Hip-Hop/Rap

music showed the highest mean number of streams and weeks on charts, indicating its popularity among listeners.

Recommendations for Future Analysis:

1. **Increase Dataset Diversity:** To improve the effectiveness of clustering techniques, future analysis should consider incorporating a larger and more diverse dataset encompassing a broader spectrum of music genres. This would ensure that the clustering algorithm can capture the nuances of each genre more accurately and avoid grouping multiple genres into a single cluster.

2. **Fine-Tune Clustering Parameters:** Experimentation with different clustering algorithms, distance metrics, and parameter settings could enhance the clustering process. Additionally, exploring advanced techniques such as ensemble clustering or semi-supervised clustering may yield more robust results.

3. **Validation and Refinement:** Continuous validation of clustering results against existing genre annotations and manual inspection of clusters are crucial steps in ensuring the reliability and interpretability of the analysis. Refinement of clustering algorithms based on feedback and domain knowledge can further improve the accuracy of genre classification.

4. **Integration of Additional Features:** In addition to audio features, incorporating metadata such as artist information, album details, and listener demographics could provide valuable context for genre classification. This multidimensional approach would enhance the richness of the dataset and improve the granularity of genre clustering.

5. **Longitudinal Analysis:** Conducting longitudinal analysis to track genre popularity trends over time would offer insights into evolving listener preferences and market dynamics. This could involve analyzing streaming metrics, chart performance, and genre distribution patterns across different time periods.

6. **Collaborative Research:** Collaboration with industry experts, musicologists, and data scientists could enrich the analysis by integrating diverse perspectives and methodologies. Leveraging domain expertise in musicology and machine learning techniques could lead to more sophisticated genre classification models.

Bibliography

Learn Cluster Analysis | Cluster Analysis Tutorial | Introduction to Cluster Analysis. (n.d.). Www.youtube.com. Retrieved May 2, 2024, from https://www.youtube.com/watch?v=3MnVCX94jJM

Learn R in 39 minutes. (n.d.). Www.youtube.com. https://www.youtube.com/watch?v=yZ0bV2Afkjc

Pavlik, K. (2019, December 20). *Classifying genres in R using Spotify data*. Kaylin Pavlik. https://www.kaylinpavlik.com/classifying-songs-genres/#:~:text=There%20are%2012%20audio%20features

Rau, T. E. (2021, April 9). *Clustering & Forecasting | Spotify Songs Audio Features*. Medium. https://tomasezequielrau.medium.com/clustering-forecasting-spotify-songs-audio-features-5b2c21f0a6b9

Senavirathne, N. (2022, December 12). *Music Genre Prediction Using Audio Features*. Medium. https://medium.com/@navodas88/music-genre-prediction-using-audio-features-96dea4e71ad3

Sharma, N. (2021, November 4). *K-Means Clustering Explained*. Neptune.ai. https://neptune.ai/blog/k-means-clustering

Spotify Top Songs. (n.d.). Kaggle.com. Retrieved May 2, 2024, from https://www.kaggle.com/code/rayyan123z/spotify-top-songs/notebook