Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Master in Data Science

Design of a new interactive data analysis tool for Spotify

Madrid, January 2025

Contents

List of Figures

List of Tables

1. Introduction

This report is a continuation of the previous first assignment of the Data Visualization subject where we needed to get our first steps using Shiny[1] to build an interactive designing tool. In this case we are going to use Shiny for the visualization of different data from the Spotify API[2] in order to have a better understanding of musical trends and patterns. This includes exploring the evolution of genre popularity, artist performance metrics, audio characteristics and other key insights derived from the data. Except for song evolution over time, songs chosen were based on the September 21st edition 2024 of Billboard Top 50 songs (TikTok).

2. Problem characterization

The final dashboard can be of course viewed and enjoyed by music-enthusiasts of all backgrounds, however the *main* audience considered for this dashboard are music producers who want to better understand the qualities and characteristics that are a part of popular songs trending on Tiktok (as well as Instagram Reels and YouTube Shorts). The consideration of music popularity in social media is a relatively new concept (beginning in around 2019 or so), but has since taken off in popularity as popular songs in social media can translate to higher amounts of streams on platforms such as Spotify and Apple Music.

With that in mind, questions a music producer may have include (but are not limited to):

- How is loudness across different sections of songs?
- How has genre popularity evolved during time globally?
- How do different genres/songs compare among key audio characteristics?
- How do a song's energy and beats per minute influence its popularity and categorization?

3. Data and tasks abstractions

This section is dedicated to all the processes and results of performing data and task abstraction to all the respective idioms.

3.1. Data abstraction

This section is dedicated to all the data abstractions that have to be done for each one of the idioms presented in this document.

3.1.1. How is loudness across different sections of songs?

We use a dataset that was handpicked from Spotify's data[2] for Idiom 1, which is specifically set up to examine how loudness is distributed over various song segments. Five features make up the dataset, which records specific details about the time of song segments inside the recording as well as their loudness and musical key.

The following table 1 summarizes the dataset features and their characteristics:

Feature	Type	Description
track_name	Categorical	Name of the song
start	Numerical	Start time of the section in seconds.
duration	Numerical	Duration of the section in seconds.
loudness	Numerical	Average loudness (dBFS) for the section. Lower values indicate quieter sections.
key	Categorical	Musical key of the section encoded as an integer.

TABLE 1: DATASET VARIABLES FOR QUESTION: HOW IS LOUDNESS ACROSS DIFFERENT SECTIONS OF SONGS?

Each row in the tabular dataset represents a distinct track segment, and the columns correspond to the characteristics or measurements of that segment. Two categories differ in the attributes:

- **Categorical Attributes**: Here are included track_name which indicates the song that goes with each segment, and key that is in charge of encoding the musical key so that portions can be categorized according to their tonal properties.
- **Numerical Attributes**: Features like start that indicates the section's temporal placement within the track, duration which shows how long the section is and loudness that is a measure of perceived loudness that is essential for comprehending dynamic changes between parts.

3.1.2. How has genre popularity evolved over time globally?

For Idiom 2, we utilize a dataset extracted from the Spotify API[2], specifically curated to analyze genre popularity over time on a global scale. The dataset comprises X instances and 7 features that capture detailed information about the most popular songs and albums of leading artists within each genre.

The following Table 2 will contain a brief resume of the table and its characteristics.

Feature	Type	Description
genre	Categorical	Genre of the song
artist_id	Categorical	Id of the song’s artist on the Spotify database
artist_name	Categorical	Name of the song’s artist
track_name	Categorical	Name of the song

track_id	Categorical	Spotify Id for the track
release_year	Numerical	Year of release of the song
popularity	Numerical	The popularity of the track. The value will be between 0 and 100, with 100 being the most popular.

TABLE 2: DATASET VARIABLES FOR QUESTION: HOW HAS GENRE POPULARITY EVOLVED OVER TIME GLOBALLY?

The dataset is tabular, where each row represents a unique track, and columns represent attributes or dimensions of the track. The attributes vary across three types:

- **Categorical Attributes**: These include *genre*, *artist_id*, *artist_name*, and *track_name*. These attributes enable grouping and categorization, crucial for understanding genre trends and artist contributions.
- **Numerical Attributes**: Attributes such as *release_year* and *popularity* provide quantitative insights. *release_year* offers a temporal sequence, while *popularity* gives a comparative measure of the tracks' success.
- **Unique Identifiers**: Features like *track_id* and *artist_id* ensure that data integrity is maintained and allow precise tracking of individual records.

3.1.3. How do different genres/songs compare among key audio characteristics?

For Idiom 3, the top 50 TikTok songs from Billboard are considered, and the Spotify Audio Features API endpoint was used.

The table below lists all key attributes used as part of Idiom 3.

Feature	Type	Description
genre	Categorical	Genre of the song, based on the artist's genres in spotify.
artist_song	Categorical	Concatenation of artist + the song name
happiness	Numerical	Score from 0-100 of song's "happiness" as reported by Spotify.
acousticness	Numerical	Score from 0-100 of song's "acousticness" as reported by Spotify.
danceability	Numerical	Score from 0-100 of song's "danceability" as reported by Spotify.

energy	Numerical	Score from 0-100 of song's "happiness" as reported by Spotify.
speechiness	Numerical	Score from 0-100 of song's "instrumentalness" as reported by Spotify.
instrumentalness	Numerical	Score from 0-100 of song's "instrumentalness" as reported by Spotify.
liveness	Numerical	Score from 0-100 of song's "liveness" as reported by Spotify.

TABLE 3: DATASET VARIABLES FOR QUESTION: HOW DO DIFFERENT GENRES COMPARE AMONG KEY AUDIO CHARACTERISTICS?

- **Categorical Attributes**: These include *genre* and *artist_song*. These attributes are used for grouping later on in the chosen visualization
- **Numerical Attributes**: Features such as *happiness, acousticness, danceability, energy, speechiness, instrumentalness, liveness*

3.1.4. How does a song's energy and beats per minute influence its popularity and categorization?

For Idiom 4, we selected a representative set of 64 songs, with several metrics that represent each song. On table 4 we can observe the different variables used, and a description of them.

Feature	Type	Description
Beats per Minute	Numerical	Beats per minute of the song
Energy	Numerical	Score from 0-100 of song's "energy" as reported by Spotify.
Popularity	Numerical	Score from 0-100 of song's "popularity" as reported by Spotify.
Key Type	Nominal Categorical	Either Major or Minor
Duration Category	Ordinal Categorical	Discretized duration of the song, in categories, 1-2, 2-3, 3-4, or more than 4 minutes
Genre	Nominal Categorical	A set of musical genres

		defined by Spotify
Explicit	Nominal Categorical	Either Explicit or not

TABLE 4: DATASET VARIABLES FOR QUESTION: HOW DO A SONG'S ENERGY AND BEATS PER MINUTE INFLUENCE ITS POPULARITY AND CATEGORIZATION?

3.2 Tasks abstraction

This section is dedicated to all the task abstractions that have to be done for each one of the idioms presented in this document.

3.2.1. How is loudness across different sections of songs?

Heatmaps are a great tool for visualizing loudness in various song segments because they make it easy to compare loudness levels over time. The user has the option of visualizing a single track or comparing several tracks according to how loud each section of the song is.

Track-level and section-level information taken from Spotify data make up the dataset utilized for this visualization. With characteristics like start time, duration, key, and loudness, each recording is divided into segments.

Exploration is the heatmap's main use case. Users can find loudness patterns in a single track or in several tracks. This makes it possible to generate and validate hypotheses like determining the loudest and quietest parts of a song or analyzing patterns in musical composition, which involves comparing the dynamics of loudness over several tracks.

Finding the loudest part of a track or a continuous loudness trend are simple jobs for individual tracks. However, because users must combine patterns from several heatmaps, analyzing loudness trends across tracks becomes more complicated, here is where the task complexity resides.

The primary drawbacks this first idiom addresses are that it is time-consuming and unobjective for users to have to listen to complete tracks in order to identify loudness trends. Without having to listen to the audio, users can obtain a quantitative and comparative understanding of loudness dynamics by employing heatmaps. Music producers and analysts that need to efficiently process vast datasets will find this especially helpful.

3.2.2. How has genre popularity evolved over time globally?

The visualization is needed to discover new knowledge, it enables users to explore and reveal trends, patterns, and changes in genre popularity over time. By visualizing data aggregated by genre and year, it allows users to identify trends, such as increases, peaks, or declines in popularity, and to generate hypotheses about the factors driving these changes, such as cultural or industry shifts. Additionally, it provides a way to verify hypotheses by comparing insights from the visualization with external data, supporting the broader goal of uncovering unexpected insights or confirming specific expectations through the analysis.

The type of search based on whether the target and the location are known or not was a Lookup. This is because your visualization allows users to focus on specific genres or years of interest, searching for patterns or trends in the data that align with their goals. For instance,

a user may want to examine the popularity of a particular genre like "hip hop" in a specific time range, such as the 2010s. The visualization facilitates this by enabling filtering and exploration of targeted data points, allowing users to efficiently locate the information they are interested in.

In relation to the type of query used in this project, the visualization focuses on “Summarize” queries. This approach aligns with the objective of providing an overarching view of genre popularity trends over time, based on aggregated data. By summarizing the popularity of different genres globally across the years, the streamgraph effectively presents a high-level perspective that allows users to observe gradual trends, major shifts, and overall patterns.

For the target the visualization visualizes how genre popularity evolves over time, focusing on identifying temporal trends in the data. The main goal of the visualization is to observe patterns in popularity changes across genres, highlighting shifts in musical preferences over the years. So while it involves multiple attributes (time, genre, and popularity), the focus on time-based changes makes trends the most appropriate target.

The most appropriate design choice for this work is “Map” because the visualization used relies on mapping data attributes to visual channels to convey information effectively. In this case, time is mapped to the main axis, representing the temporal progression of data, while average popularity is mapped to the short axis to show magnitude. Additionally, genres are mapped to unique color channels to distinguish between categories visually.

3.2.3. How do different genres/songs compare among key audio characteristics?

Parallel coordinates are a good choice for comparing data that shares a common scale. In this case, with 7 main attributes, it is easy to show all results. Furthermore, the features visualized all share a common score between 0-100, making it easy to compare different attributes. There are 50 unique songs that the user can choose from based on the song.

Color is the main visual channel to help differentiate items in the chart. It changes based on user selection of either *genre* or *artist_song*. Because of parallel coordinates’ unfamiliar nature (in comparison to visuals such as map), users are unlikely to know where to look, however the target will most likely be known (users will have a certain genre or a certain song in mind).

The primary action for this visual falls under ‘discover’. Users could potentially verify or generate new hypotheses, depending on their familiarity with the given song attributes.

What makes these song attributes particularly interesting is that all users have an intuitive sense of what they all mean, and the user could individually identify these attributes if listening to a song, but it may be more of a challenge to compare, as well assign an objective ‘score’ between 0-100. Human listening tasks are particularly time consuming, and if a user hasn’t listened to a given song before, it is impossible to know the attributes of said song.

A question a user might have with regard to using this visual is, “what does a given genre or set of songs look like in terms of its sound characteristics”. Again, this is an easy task for the user to self-answer simply by listening to a single song, but the task becomes more difficult when trying to compare more than a couple songs. Having a better picture could help the

given user make better choices when trying to help decide what characteristics to include in their own music.

3.2.4. How does a song's energy and beats per minute influence its popularity and categorization?

A bubble plot is an excellent choice for examining the relationship between energy, beats per minute (BPM), popularity, and additional song characteristics. The plot leverages three key axes: energy on the vertical axis, BPM on the horizontal axis, and bubble size to represent popularity. To add another dimension, bubbles are colored based on categorical attributes such as genre, duration category, or explicitness.

This visualization enables users to explore trends and relationships among audio features and popularity. For instance, users can filter the data to focus on specific ranges of BPM, energy, or popularity to identify patterns. By visually grouping songs with similar characteristics, users can uncover trends, such as whether higher energy levels and faster BPMs correlate with greater popularity or whether certain duration categories dominate in specific energy ranges.

The visualization serves both “Discover” and “Lookup” tasks. Users may discover general trends, such as whether energetic, high-tempo songs are more likely to be popular, or use it to locate specific data points, such as finding songs within a certain BPM range that also have high energy and popularity.

For the type of query, the visualization aligns with “Summarize” tasks, as it aggregates data across multiple attributes to reveal overall trends and patterns. At the same time, it supports targeted queries, as users can filter the dataset to zoom in on specific characteristics or combinations.

The primary target of this visualization is to uncover correlations and patterns in the data, enabling users to form hypotheses about factors that drive song popularity. It provides a multifaceted view by combining energy, BPM, popularity, and additional characteristics, helping users grasp both individual and aggregated relationships.

The most appropriate design choice for this work is “Map,” as it maps multiple data attributes to visual channels effectively. The axes map energy and BPM, bubble size maps popularity, and color encodes additional categorical attributes. This allows for an intuitive and interactive exploration of the relationships between the selected song features and popularity.

4. Interaction and visual encoding

This section is dedicated to the different approaches for the visualization and interactivity for each one of the idioms presented in this document.

4.1. How is loudness across different sections of songs?

The heatmap uses color intensity to represent loudness values in order to visualize patterns of loudness across song parts, emphasizing on the connections between start time, key, and loudness allowing users to examine changes within a track or among several songs.

The key parts of the design which the users can interact with are:

- Track Selection: Allows dynamically refresh of the heatmap, users choose tracks from a dropdown menu.
- Section Range Filtering: Users can focus on particular song segments by using a slider to filter by start time.
- Loudness Range Adjustment: Parts of a selected loudness range are highlighted by a slider.
- Tooltips: When you hover over a segment, detailed information is displayed, such as the key, loudness, and start time.
- Export Option: Heatmaps can be exported as CSV files or images.

The key visual encodings related to this first idiom are the x-axis which encodes the start time of sections, showing the progression of loudness across a track, the y-axis that represents the key, grouping sections by musical key for a better contextual understanding. The last visual encoding is the color which is the main visual channel, encoding loudness. A gradient from blue low loudness to red high loudness allows users to intuitively interpret the intensity of each section.

To summarize, this tool for the first idiom offers an easy-to-understand color gradient and distinct labels for start time and key for users who are not familiar with heatmaps. Customization according to user requirements is ensured by the presence of a volume range slider and the capability to compare several tracks. Also, the heatmap can be downloaded, allowing users to share or analyze their results. All these implementations can be seen in the figure [1].

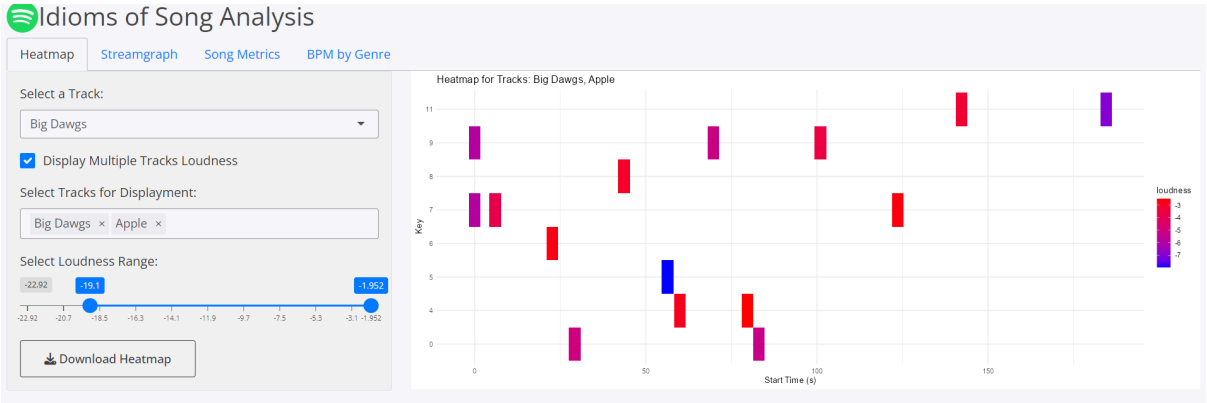

Figure 1: Heatmap for loudness across different sections of 2 different songs

4.2. How has genre popularity evolved over time globally?

The visualization used to solve this question is a streamgraph, a type of stacked area chart designed to display the flow and magnitude of values over time. It emphasizes the temporal

progression of data while maintaining the relative proportions of categories, making it ideal for visualizing trends and changes in genre popularity across years.

For the streamgraph it is first generated on the filters selected where the data is filtered first to the musical genres selected by the user and to only include the years on the inputs. Then the application does take all the songs in the data set corresponding to the selected genres. Finally the data gets grouped and ordered by its *release_year* and it's calculated the average on the attribute *popularity*. The visualization for the main axis represents the years providing a temporal scale to observe historical trends and patterns. For the short axis reflects the magnitude of the average popularity and its height indicates how popular a genre was in a given year, each genre is assigned a distinct color, making it easy to identify within the graph.

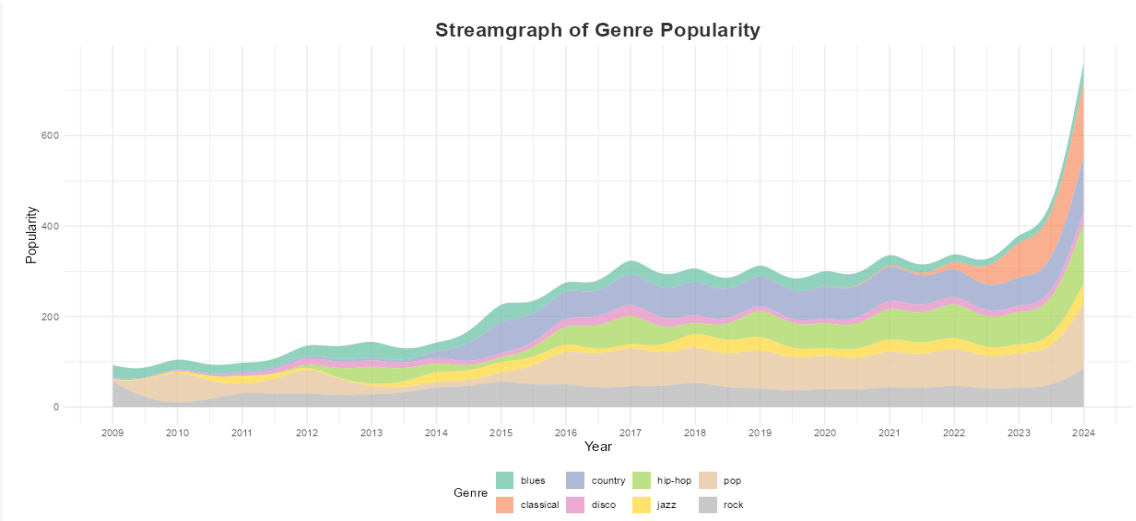

Figure 2: Streamgraph for the popularity of different genre over the years

4.3. How do different genres/songs compare among key audio characteristics?

The parallel coordinate plot allows for two main attributes to be considered at the start; *genre* and *artist_song*. From there, the code running in the server will automatically adjust the data to be used as input for the parallel coordinates plot. The color and points used in the plot will reflect the selected attribute and the dropdown menu of options will change based on the selection. By default, only one genre is considered, as space is definitely a premium for this visual. In addition to being able to select a given song or genre, the user also has the option to drag and move around the order of the columns. This is particularly helpful, as a user might want to compare two attributes directly side by side, such as *acousticness* and *instrumentalness,* but with only one default view, it would be impossible to achieve all possible comparisons. Finally, a reset button is offered if the user wishes to remove their selection and start over.

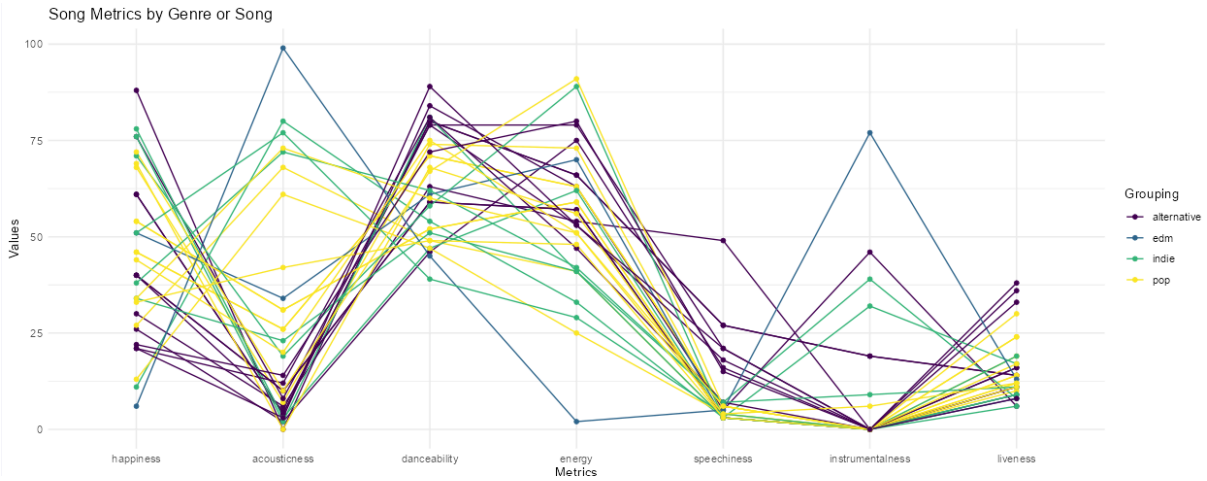

Figure 3: Parallel coordinates for different genres and songs among key audio characteristics

4.4. How does a song's energy and beats per minute influence its popularity and categorization?

The main advantage of a bubble plot over a traditional scatter plot is the inclusion of a third axis in the form of size. Using color to group data further reveals patterns within each group. Although additional groups could be represented, for instance, by using different shapes, this has not been implemented to avoid making the visualization overly complex and potentially less comprehensible.

Interactivity Features have been implemented as follows:

- **Adjustable Ranges:** Users can select specific ranges for BPM, energy, and popularity using sliders, enabling focused analysis on areas of interest. By default, the ranges for numerical data encompass all available values.

- **Customizable Grouping**: A category menu allows users to choose which variable determines the color of the bubbles, facilitating tailored interpretations of the numerical data within specific groups. By default, the selected categorical variable is the "Key."

These interactive features make the visualization both flexible and user-friendly, supporting diverse analytical goals while maintaining simplicity and readability.

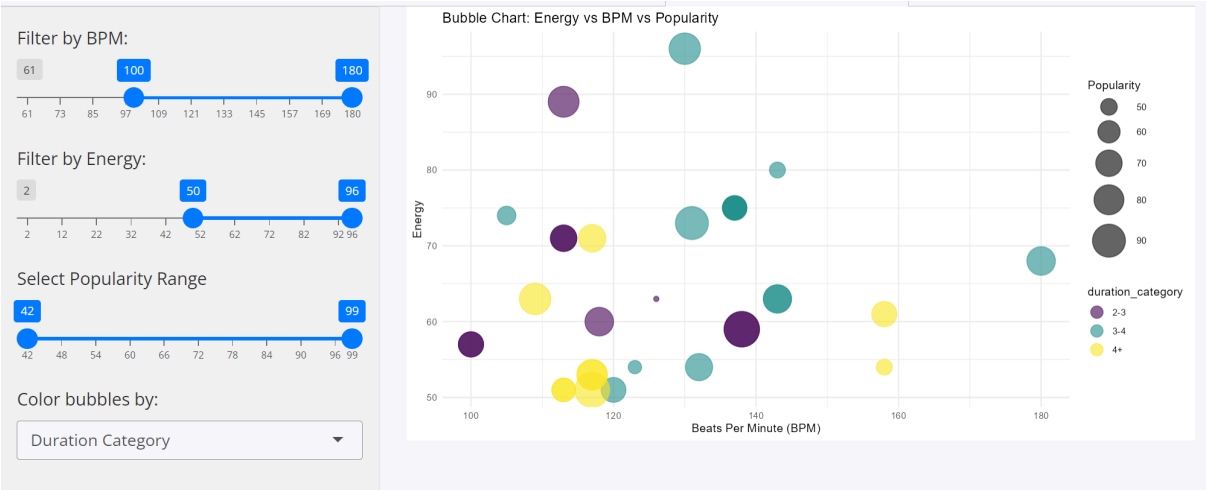

Figure 4: Bubble plot with some ranges selected and duration as the categorical variable

5. Algorithmic implementation

This section is dedicated to the different approaches for the algorithms used for each one of the idioms presented in this document, and the technologies used for this purpose.

5.1. Software

For this project first we used some scripts created in Python to extract and clean data from the Spotify API[2]. For the application we used R[4] in RStudio[5] using the Shiny[1] framework, which enables the creation of dynamic and user-friendly web applications. And the visualization is achieved through libraries such as ggplot2[6] for creating static and interactive plots

5.2. How is loudness across different sections of songs?

Data preparation for the heatmap display begins with CSV files (track_info and sections) that contain music segments and loudness parameters. Important characteristics are extracted, including loudness, key, duration, and start. Section data is filtered to remove outliers, and missing values are handled as part of the preprocessing step to guarantee consistency.

To organize parts according to their musical key, the key attribute is mapped to the y-axis, while the start attribute is mapped to the x-axis, signifying the section's commencement in seconds. In order to convey intensity, loudness levels are shown as a gradient color scale, where red indicates louder passages and blue indicates softer ones.

The Shiny framework is used to implement user interaction, where users can choose recordings, filter by section range … An interactive and adaptable exploration experience is offered by the visualization, which changes in real time in response to user interaction.

This method uses R and Shiny to provide a dynamic, user-friendly interface for examining the characteristics of song sections.

5.3. How has genre popularity evolved over time globally?

The streamgraph visualization starts with the data preprocessing from the Spotify API[2]. Key attributes such as *genre*, *release_year*, and *popularity* are retained. The data is grouped by *genre* and *release_year*, and the average popularity for each genre per year is calculated. Missing values are replaced with zero to ensure consistency, and the data is sorted chronologically to facilitate a coherent temporal narrative.

For the display the processed data is rendered using the ggplot2[6] library in R. The *release_year* attribute is mapped to the man axis to represent time, while average popularity is mapped to the other axis to determine the height of each layer in the streamgraph. The *genre* attribute is assigned unique colors to distinguish layers visually. User interactivity is implemented through Shiny[1], allowing users to filter genres and adjust the year range dynamically, with the visualization updating in real time.

5.4. How do different genres/songs compare among key audio characteristics?

The parallel coordinates plot all rests on user options selected. Default options are presented, but if the user decides to, the data that is utilized for the visual will change. The group by attribute will change between *genre* and *artist_song*, and the rows retrieved will change depending on which genres or songs the user picks. Lastly, the order of the columns can change using the selector, and the code will update the visual accordingly.

The visual is sourced from the ggally [7] library in R. The x axis in this case represents each audio attribute (7 total). The y-axis provides the score of each attribute (all attributes are between 0-100. Color is sourced from either *genre* or *artist_song*, and a point represents a single song. Lines are drawn between plots to help distinguish an individual song from another.

5.5. How does a song's energy and beats per minute influence its popularity and categorization?

The implementation of this Shiny application begins with data preparation and transformation. Three datasets (audiofeatures.csv, track_info.csv, and artists.csv) are merged to create a comprehensive dataset containing audio features, track information, and artist metadata. Specific transformations are applied to enhance the dataset's usability: a new categorical variable key_type is derived from the key column, indicating whether a song is in a major or minor key, and duration_minutes is calculated from the duration column to represent song length in minutes. A duration_category variable is then created to classify songs into categories based on their length, which are later used for grouping in the visualization.

In the server function, the filtered dataset is generated reactively based on the user-selected ranges and options. The renderPlot function dynamically creates the bubble plot using ggplot2. The x-axis represents BPM, the y-axis represents energy, the bubble size reflects song popularity, and the bubble color is determined by the user-selected categorical variable. A continuous scale is used for bubble size, while the viridis palette is employed for colors to

enhance visual clarity. Additional plot features, such as a dynamic legend and customized guides, are implemented to ensure the visualization is informative and visually appealing.

6. Shiny App

The application has been uploaded and deployed to Shinyapps, the link for the app is: *https://rcmartemp.shinyapps.io/datavisualization/*

Here are some brief instructions to run the source code of the application properly.

- First you need to have both R and RStudio installed on your computer.
- Set the working directory on the main folder of the app
- Install the following libraries and dependencies: *shiny, shinyWidgets, ggplot2, dplyr, bslib, shinyBS, GGally, viridis*
- Got to the RStudio console and type *library("shiny")* and then type *runApp()*
- A new window should appear with the application running.

7. Conclusions

In this document we have highlighted the usefulness of visual analytics to obtain insights from data. We have answered several questions over our data that might be useful for a business. Some of the advantages of using the visual representations selected are:

- Heatmaps effectively showed loudness patterns without requiring users to listen to full tracks
- Streamgraphs displayed temporal trends in genre popularity
- Parallel coordinates enabled comparison of multiple audio characteristics simultaneously
- Bubble plots revealed relationships between multiple variables while maintaining visual clarity

The project addressed real business needs by helping music producers understand:

- Song structure through loudness analysis
- Genre trends over time
- Audio characteristic patterns in successful songs
- Relationships between technical aspects and song popularity

The implemented visualizations demonstrated clear benefits of interactivity over static analysis, particularly in making complex musical data more accessible and analyzable. Through interactive features like track selection, range filtering, customizable grouping, and adjustable parameters across all visualizations, users could efficiently explore and analyze large datasets from different angles. This was particularly evident in the bubble plot visualization, where users could dynamically adjust ranges for BPM, energy, and popularity while simultaneously grouping by different categorical variables, enabling insights that would be impossible to achieve through static visualization alone. Using a static representation for the same data would mean we would have to plot the same information multiple times, with slight variations, to capture all the possible views and combinations that our interactive visualizations allow with a single plot. This would make the analysis process not only more time-consuming and resource-intensive but also less flexible.

8. References

[1] Shiny: a web application framework for R: RStudio. https://shiny.posit.co/

[2] Spotify API. https://developer.spotify.com/documentation/web-api

[3] Spotify. https://www.spotify.com/

[4] GNU: The R Project for Statistical Computing. www.r-project.org

[5] RStudio: RStudio. www.rstudio.com

[6] Ggplot2: Ggplot2. https://ggplot2.tidyverse.org/

[7] GGally: Extension to Ggplot2, https://ggobi.github.io/ggally/