Billboard Hot 100 Songs Data Visualization

Eunice Zhou*

University of Illinois at Urbana-Champaign

ABSTRACT

This is the report for my CS 519 Scientific Visualization final project. The goal of this project is to create a clean and user-friendly data visualization that shows the rise and fall of popular songs on Billboard over time. The visualization utilizes a variety of different JavaScript frameworks including React, Chart.js, and Semantic UI. The dataset we use is obtained from Kaggle, cleaned and processed in a Jupyter Notebook before feeding into the visualization. The finished product is a line chart hosted on a web application with different interactive elements such as tooltips, custom artist and year input, and progressive line chart animation. The final product is published at `https://eunicornbread.github.io/CS519-final-project/`.

1 INTRODUCTION

The Billboard Hot 100 is the music industry standard record chart in the United States for songs, published weekly by Billboard magazine. Chart rankings are based on sales, radio play, and online streaming in the United States. These rankings demonstrate the popularity and success of artists and their songs, and they give people insights into the trend and patterns of the music industry. It is a very interesting dataset that we want to visualize in this project. We hope that by creating a good data visualization, we can help people better understand these data.

2 DATA ACQUISITION

The dataset we use for this project is obtained from Kaggle at the following url: `https://www.kaggle.com/dhruvildave/billboard-the-hot-100-songs`. The dataset is a collection of all "The Hot 100" charts released by Billboard since its inception in 1958. The dataset contains the following fields:

- date: the date when the ranking was released in the format of "YYYY-MM-DD"
- rank: the ranking of the song in the Billboard chart
- song: the title of the song
- artist: the artist(s) of the song
- last-week: the ranking of the song in the previous week
- peak-rank: top rank achieved by the song
- weeks-on-board: number of weeks on the board

In the following section, we will briefly talk about how we obtain, clean and process the data for later use in visualization.

*e-mail: xz33@illinois.edu

2.1 Data Cleaning

We first select the data we are interested in using for the visualization. Since we only want to visualize the rank of the songs, the three fields *last-week*, *peak-rank*, and *weeks-on-board* are irrelevant for the visualization. In a Jupyter notebook (*data-cleaning-part1.ipynb*), we read in the dataset and drop the three columns. We also select only songs in the 21st century, dropping any row with date before year 2000 for a more recent song collection before outputting the dataset into a csv file.

2.2 Data Format

Since we will host our visualization on a web app and use JavaScript to implement our visualization, we want to convert the format of our data into a JSON file for easier processing. This is accomplished in *data-cleaning-part2.ipynb*.

3 IMPLEMENTATION

We use the following framework for our project:

- React for creating a web application
- Semantic UI for its JavaScript component
- Chart.js for data visualization

4 DESIGN CHOICES

One thing to note is that instead of visualizing all of the songs that get included on the Billboard Top 100 at once, we decide to visualize songs by one artist at a time. That's because the dataset with all the Hot 100 songs is large, and including all the information will make the visualization clustered and difficult to understand. In addition, song rankings change frequently and songs get replaced by new songs after a while. That's why we think it's a bad idea to visualize all the information in one giant graph. We think grouping songs by artist is a good idea because the popularity of the songs in many way reflects the success of an artist, and their popular songs are close together in terms of temporal distance. In this way, the users can focus on just a specific section of the dataset rather than getting confused by the large amount of data available.

5 FEATURES

In this section, we will go over the various features included in our project.

5.1 Basic Visualization

We visualize the rise and drop of song rankings over time in a line graph (see Figure 1). The x-axis is the date of the song and the y-axis is the ranking. We think that our choice of a line graph is a good representation of the data because it emphasizes the change of the song ranking over time.

5.2 Interactive Visualization

We add in some interactivity to allow user interaction with our visualization (see Figure 2). For example, if a user hovers its mouse over a data point, a tooltip will appear to the right of that data point showing the date, name and ranking of that song. Another feature is that a user can click inside the legend area, which is right above

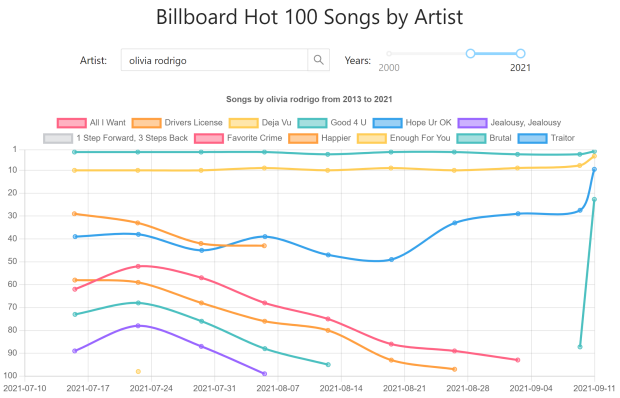

Figure 1: Basic Line Graph

the chart and below the chart title, and they can hide and show the songs they click on.

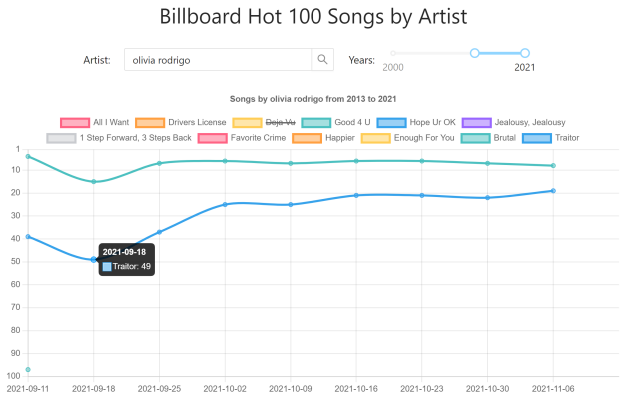

Figure 2: Interactive Line Graph

5.3 Filter by Artist

Since the visualization shows the rankings of all the songs by a certain artist, we want the users to be able to choose the artist they want to look at. To do that, we add a text input field at the top of our chart which allows the user to enter the name of the artist (see Figure 3). After the user has entered an artist, the chart will update to show all the songs by that artist. The default artist we use is Katy Perry.

5.4 Custom Time

We also allow the user to select a specific time period they want to look at (see Figure 4). In order to achieve that, we add a slider right next to the text input (see section 5.3) that allows a user to specfiy the start and end year of the visualization. That's because an artist can have a lot of songs over the years from 2000 to 2021, and too many songs in the visualization can make it messy and hard to understand.

5.5 Progress Line Chart Animation

We also accomplish our reach goal of animating the lines as a progressive line graph that runs from left to right. One advantage of this feature is better clarity of the visualization. Since an artist and his or her songs may be popular for a long time, visualizing all the songs across a large time period can make the graph look very

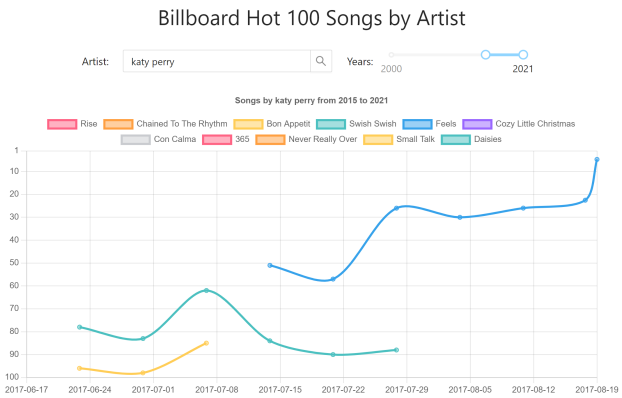

Figure 3: Custom Artist using Text Input

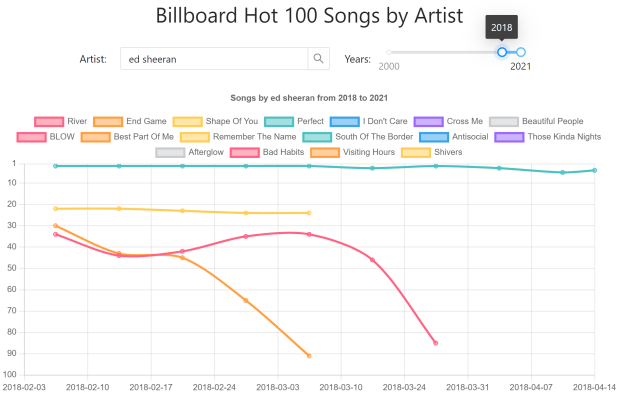

Figure 4: Custom Years using Slider

cramped, since all the data points will be compacted in a fixed-width graph. Therefore, we decide to visualize at most 10 dates in the graph, removing old data from the left and adding new data on the right. This creates the impression of a running line graph for users, as new data appear and the whole chart shifts left.

However, one drawback of adding the animation we found is that it limits user interaction when the graph is running. To be more specific, when the graph is moving, it is difficult for a user to hover his or her mouse over the data points to learn more about them. The users will have to wait until the animation finishes, but they may be interested in an earlier part of the visualization that is now inaccessible.

6 Links

The deployed web application can be found at `https://eunicornbread.github.io/CS519-final-project/`.

The GitHub repository with the code can be found at `https://github.com/eunicornbread/CS519-final-project`.

7 Future Work

One future line of work is to add in the functionality to pause and unpause the animation. This will effectively address the limitation mentioned in section 5.5. Some other improvements we can make to the visualization include adding song names right next to the data points/line and allowing the selection of multiple artists or songs.