Elijah Wennberg Smith

Interactive Visualization

Final Project Report

4/30/20

Apple Music Library Analyzer

Introduction:

For my final project, I wanted to visualize the genre distribution across my Apple Music library and listening history. Although genre distribution was my main goal, there are also a number of other metrics that are visualized or listed. As an avid music listener and curator, I have always been interested in visualizing how my library changes and evolves over time. For a long time, Spotify has provided its users with a yearly update of what they listened to that year. This is something that I have always wanted out of Apple Music. The target audience for this application is anyone who is interested in how their Apple Music library and listening habits evolve over time. Unfortunately, for reasons that will soon be explained, this application is only compatible with Apple Music and no other streaming services.

Throughout the time that I have been working on this project, I have been under quarantine. The 2020 coronavirus pandemic in the United States has made a tremendous impact on the daily lives of just about everyone in the world. The main research topic of this project serves to examine how this change to my daily life is shown through my music library and listening habits. I Hypothesized that I would see an increase in overall listening and library growth.

I chose to examine two main aspects of my library: the songs added to it over time, and the songs listened to over time. To achieve this, I created a few time-based graphs to document

over changes in both. Although my quarantine began in March 2020, I chose to start the time scale for these graphs in January. The reason for this was to give some reference to my "normal" data before my social isolation.

There are also a number of other visualizations that this application produces that do not test this hypothesis, but merely serve to give the user more information about their library and listening habits. These graphs include stream graphs to show genre distribution over time, tree maps to show overall genre distribution, and more.

Backend Production:

The back-end application is built in Python and MySQL. This is the part of the application that acquires the data from the XML file. There were a number of different hurdles that had to be overcome in this process. First, the XML file provided by Apple is formatted differently than a standard XML document. Data in the Apple XML document is stored in key value pairs rather than standard XML tags. This required a specialized parser to read the file and extract the data that I needed. Luckily, this type of project has been created before and I was able to find an algorithm posted in a blog that handles this exact situation. This work is cited below (Jeff, 2019).

However, the solution provided in this blog does not handle time-based data. This is an issue because the majority of the data I want to explore with this project is time-based. This is not a problem with the parsing algorithm, but rather the data that is available in the XML file. The file is essentially a list of songs and their attributes. These attributes include name, artist, genre, play count, last play date and more. Although the file provides play count, it only provides last play date. This was the second hurdle I needed to overcome. To solve this, I decided to store

my data in a MySQL database and track the changes over time. My SQL database consists of two tables. One is for the overall library that stores all the information about each song from the XML file. The second table is a listening history table. The schema is as follows:

library(*id, name, artist, album, album_artist, genre, total_time, year, date_added, play_count, skip_count*)

listening_history(*record_id, track_id, listen_date, listen_count*)

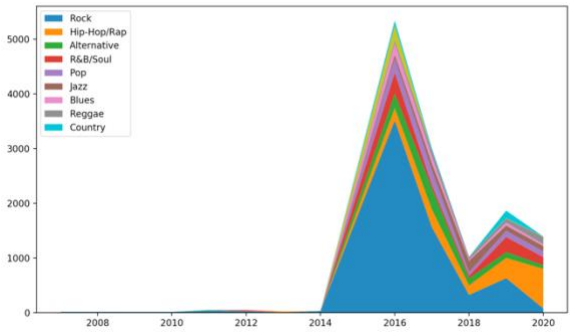
The algorithm for insertion into the database is as follows:

```
For each song in XML:
    Select song from library database
    If song exists:
        If play count changed:
            Update values in library table
            Insert play difference and last listen date into listening history table
    If song does not exist:
        Insert song into library
        If play count > 0:
            Insert play count and last listen date into listening history table
```

One flaw with this algorithm is that any songs that are new to the database and have a play count greater than 1, lack important listening history data. Only the most recent play date is available. For example, if there is a song that I have listened to 5 times, but has not been added to the library table, upon insertion, all 5 plays will be recorded with the same date, as only the last play has temporal data. There is a similar problem if a song that already exists in the library table is listened to more than once between XML uploads. If I had more time to work on this or a group member, I would have uniformly distributed the new plays between the old listen date and new listen date. This would 'fake' the data that is missing. Luckily, most of the time I do not listen to a song more than once between XML uploads, so the data for my experiment is not too badly affected.

Chart Production:

After completion of the backend code, I was shifted my focus to the analysis and visualization. In the prototyping phase I used mostly matplotlib for the charts. I initially chose matplotlib because I am familiar with it and it is very easy to use. The graphs that test my hypothesis as well as a few others remained in matplotlib because they are straightforward and simple. However, I found that the streamgraphs produced by matplotlib were limited and boring (pictured left). Instead, I transitioned to Altair which is a python library for creating visualizations using the Vega visualization grammar.



The streamgraphs created with Altair were far superior as they offer a symmetrical layout and the option to change line interpolations. I created two stream graphs, one of the monthly genre distribution of library growth (Fig. 1a) and monthly genre distribution across my listening history (Fig. 1b). The stream graphs are produced in the form of an HTML file that is automatically opened in a browser by my app. This way, the charts can have a level of interactivity. Users can zoom in and out of the chart to examine small elements, as well as use a tooltip box to provide accurate information about the region they are hovering over. (Fig. 1c)

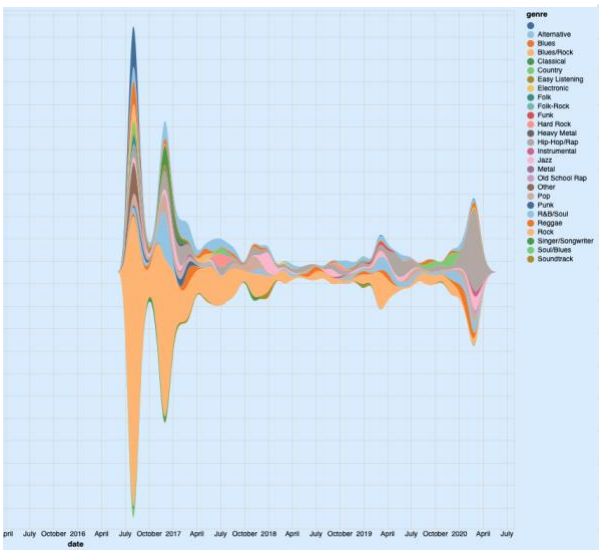


Figure 1a) library growth stream graph

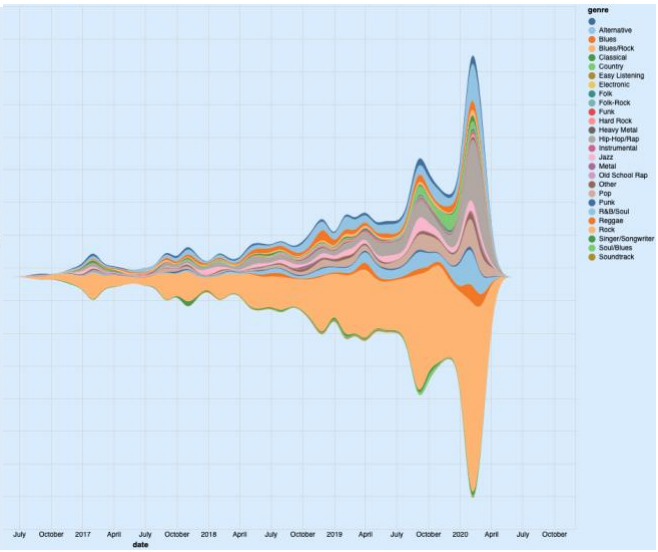


Figure 1b) listening stream graph

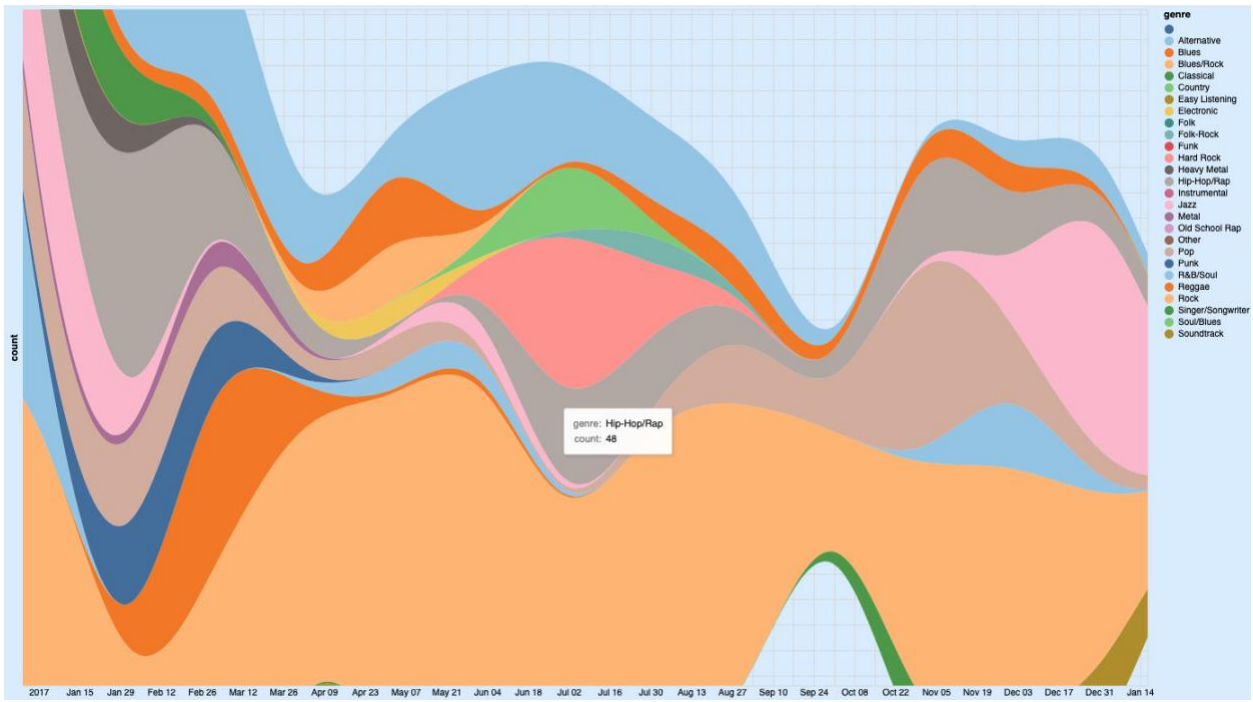


Figure 1c) zoom and tool tip

Other graphs from my application include treemap graphs. Treemap graphs are good at showing distribution of sub-elements within a dataset. I used treemaps to show the relative sizes of my top 20 genres, as well as the relative sizes of my top 20 Artists. These are shown in Figures 2a and 2b.

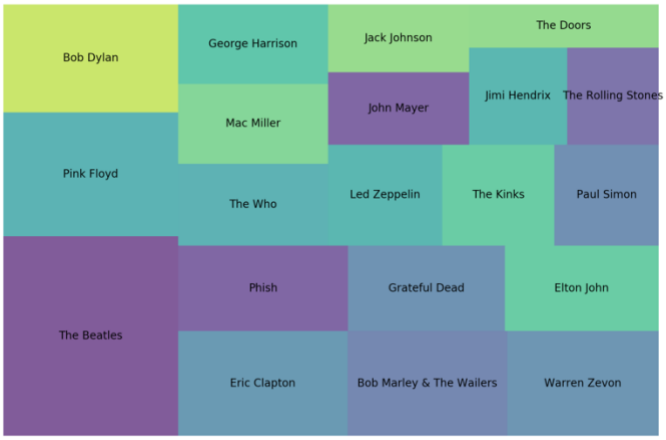


Figure 2a) Artist treemap

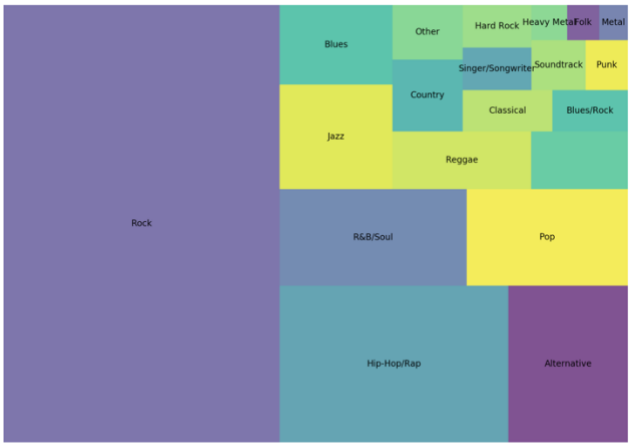


Figure 2b) Genre treemap

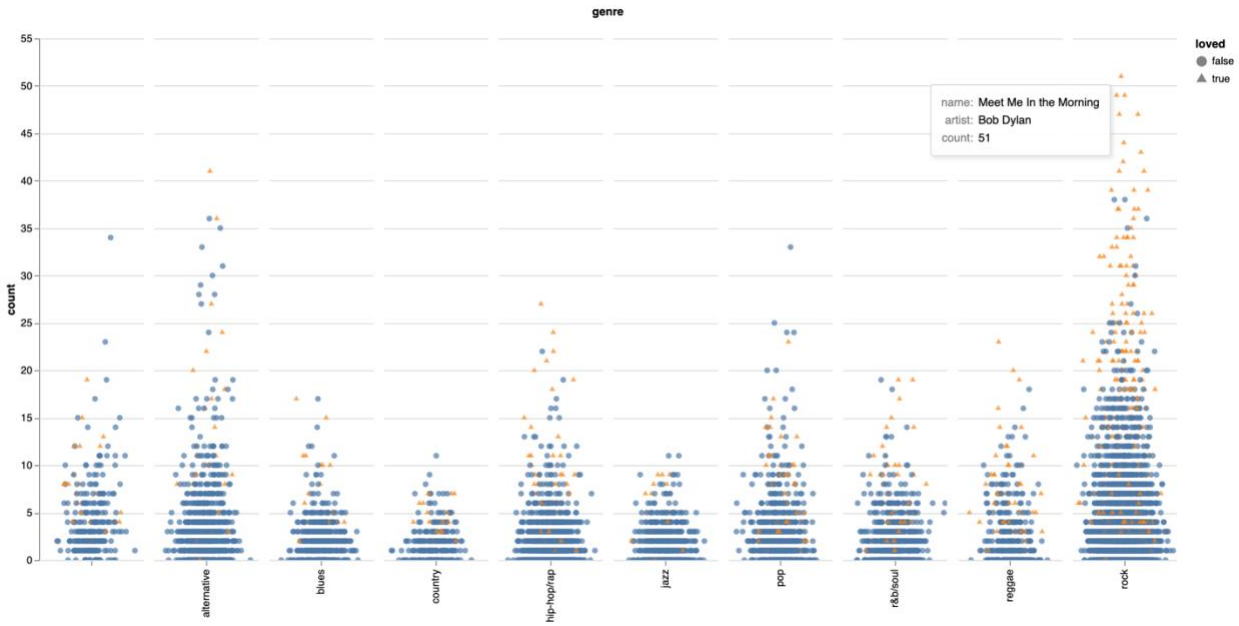


Figure 3) Strip Plot

In addition to the treemaps, I also created a visualization called a strip plot. This graph shows each song as a point grouped into columns or 'Strips' by my top ten genres (Fig. 3) The y-axis encodes the number of listens each song has. The x-axis within the columns are determined by a Box-Muller transform jitter. This uniformly distributed jitter gives the user a better idea of the number of songs that exist on the same line. Otherwise, they would all overlap into one point. The jitter is defined as follows:

$$\sqrt{-2\log * random()} * \cos(2\pi * random())$$

This graph also shows whether or not I 'loved' a song. If I 'loved' the song while listening to it, the graph encodes the point as an orange triangle. Otherwise it is a blue dot. This graph is useful for looking at what genres I really listen to and like a lot. Darker, longer horizontal lines on the graph mean that genre has a large number of songs with that number of listens. Strips that reach higher up vertically on the graph show that that genre is listened to more. This graph is also interactive. The user can zoom into each chart and examine the denser areas. There is also a tool tip that gives you the title, artist, and accurate play count for each song.

Finally, I created a number of simple line graphs to display general information. These include an overall daily listen chart, a library growth chart, and versions of both of these showing results from just 2020 for testing my hypothesis (Figures 4a-d).

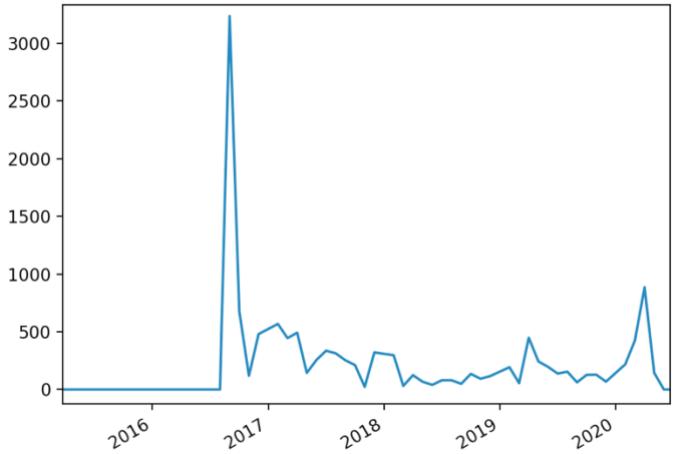


Figure 4a) Monthly library growth

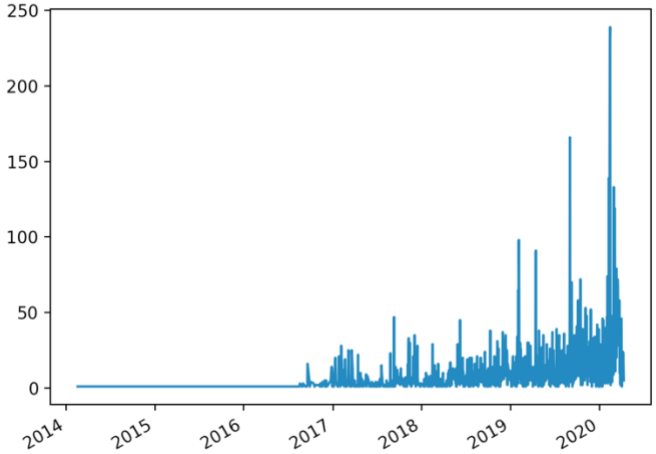


Figure 4b) Overall daily listen count

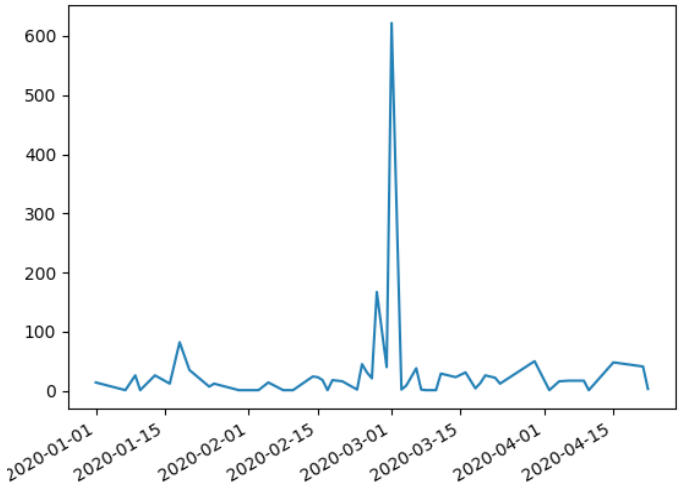


Figure 4c) 2020 Library growth (hypothesis)

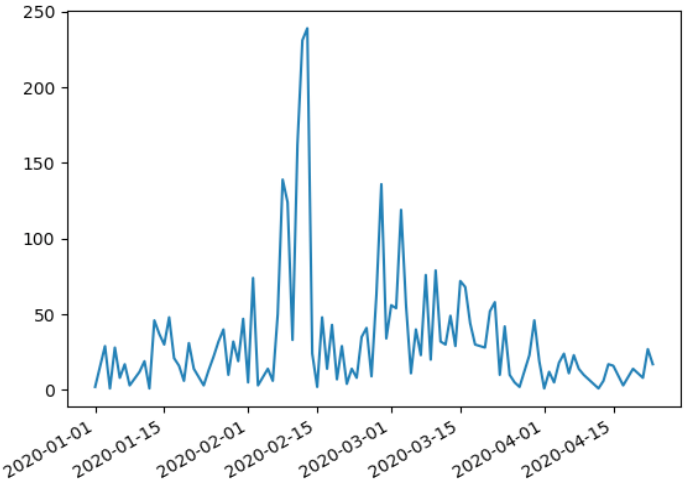


Figure 4d) 2020 daily listen count (hypothesis)

Observations:

Stream Graphs

There was some interesting information I was able to obtain from the Stream Graphs. These were by far the best visualizations for showing how my library and music tastes change over time. There is a clear point in the library stream graph (Fig. 1a) when I first purchased an Apple Music subscription. After this point, my library increased rapidly in size and diversity. Initially I downloaded mostly rock music. However, over time the beige region that shows rock music becomes thinner and thinner as other genres grow. For example, my interest in hip-hop music has grown greatly in the past few years. The grey region that represents hip hop increased greatly in the later years of the chart.

I like the listening history stream graph (Fig. 1b) also because you can clearly see some links to the library chart. At points where one genre grows greatly in the library chart, there is usually an increase in listening of that genre in the same point in the listening chart. This makes sense, as usually when I download a new album, I listen to it through once or twice. This is seen in early 2020 with the large increase in hip hop music into my library. There is a similar increase in hip hop listens at the same time in the other chart.

Strip Plot

The purpose of the strip plot was really to show what genres were liked the most. It is evident that my most liked genre is Rock. That is the genre with the greatest number of listens and songs. It stretches the highest out of any of the strips and has the greatest number of triangles. Another cool thing about this graph is it shows you which songs you have liked but haven't listened to many times. This could be useful for curating playlists.

Hypothesis

My hypothesis that I would see an increase in listening turned out to be mostly false. I started social distancing around March 7th, 2020. As seen in figure 4d, there was a very slight increase in overall listening when compared to January. However, it eventually shrunk to sub-normal levels.

Likewise, my hypothesis that I would also see an increase in library growth, has shown not to be true either. There was no noticeable increase in library growth during my time in social isolation shown in figure 4c. However, both charts have spikes in listening and library additions that do not represent the normal. For instance, there was a large increase of songs in early march that had a resulting increase in listens. These appeared to be out of the ordinary. I believe these random increases in listening are attributed to study days or days spent travelling over spring break.

Conclusion and Feedback

Professor Cutler as well as the rest of the class provided me with very useful feedback such as 'faking' the missing data by distributing the new plays. I do plan to continue working on this project in the future so these are definitely things that I will end up implementing. If I were given unlimited time on this project, I would have performed a formal user study to assess how well my graphs convey the information. I also would have liked to give this application a graphical user interface, but with the limited time and coders (just me), I was forced to stick with a command-line interface.

Overall, I really enjoyed this project, it allowed me to practice my software engineering skills and visualization skills, while working on a subject I really enjoy. Although my hypotheses did

not all turn out correct, I was still able to get useful information from the graphs produced. For instance, I will continue to use the strip plot to curate playlists of songs that I like but have not heard very many times. I also really enjoy looking through the streamgraphs and seeing how my taste in music has evolved over time. Up to date code can be found here: https://github.com/wennbergsmithe/LibraryAnalyzer

Bibliography

Jeff. “ITunes Library Analysis Using Python.” *Medium*, Medium, 6 July 2019, medium.com/@leojosefm/python-analyzing-itunes-library-97bec60e13cb.