Exploring Spotify Data

Mila Lewis-Peinado

May 15th 2024

Introduction

Spotify has been the #1 digital music streaming service for a few years now. One of the main reasons as to why it is so popular is because of its many personalization algorithms. More specifically, Spotify has "Daily Mix" (playlists created every day) and "Discover Weekly" (playlist that has new songs you haven't heard) playlists that are curated by algorithms for each user based on a user's listening history. The general public doesn't know specifically how the algorithms work but we know the general process as to what the outcomes from the analyses are. It would be really convenient and less mysterious if these algorithms were publicized, but if they were then Spotify's competitors could improve their own recommendation systems. Once Spotify processes a new song, an algorithm analyzes it to get different metrics and reports this data into the database. We also don't know the details of how the algorithm gets these different metrics but we can get a general idea of what the algorithm looks for by the name of the different metrics. The different metrics are: danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, energy, and tempo. It can be inferred that most of them are based on the composition of a track. For example, "instrumentalness" most likely just looks at the ratio of vocals to instrumental in a song and "speechiness" most likely uses some natural language processing analysis to see how much spoken word is in a song. These metrics help algorithms pick which songs to recommend to a user and recommend songs with similar metrics to previous listening history. All of these metrics that are measured are put into a Spotify database that we can access using the Spotify API, which acts as a mediator between us and the database. In this project, I will teach you how to analyze some of your favorite music on Spotify.

Demonstration

We can first load the necessary packages needed in R to do this activity. We need the `spotifyr` package for its functions that help us analyze and get data, the `tidyverse` package to plot our data and manipulate it, and the `ggridges` package to make distribution plots. If you don't have any of these packages installed, then you can install them first.

```
library(spotifyr)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggridges)
```

In order to look at Spotify data and do this full activity, you need a Spotify account To access the Spotify API in R, you must get your client ID and secret ID. These two IDs are needed to get an access token which is what the API will input when taking in our queries. You can get these IDs from the Spotify for Developers webpage (https://developer.spotify.com/) by logging in and going to the dashboard. From here, you press create an app and fill in the necessary required fields and press the save button; for the redirect URI you can put whatever you want and I put: http://localhost:1410. After this, it will create the app you requested and it will show up on your dashboard by the name you titled it. By clicking on it and going to settings on the top right, you can see your client ID and client secret ID. Assign these two IDs to their own variables and run the bottom 3 lines of code. These bottom 3 lines of code set the IDs as environmental variables and use the `spotifyr` package functions to get your access token.

```
#these are my IDs, use your own!
id = '9343fb75cad24dacb1b245b036fa82f0'
secret = '702ca111c0ed4711967a464f718140d8'

#don't change the code for this, just run it
Sys.setenv(SPOTIFY_CLIENT_ID = id)
Sys.setenv(SPOTIFY_CLIENT_SECRET = secret)
access_token <- get_spotify_access_token()
```

Now, we can look into a specific artist. Most artists are on Spotify, but if you aren't sure if one is you can just Google it. For this exercise, I am looking into Lana Del Rey's music since she is one of my favorite artists and is pretty "mainstream". To get an artist's discography and the previously mentioned metrics for each song, we can use the function `get_artist_audio_features` from the `spotifyr` package. This function inputs an artist's name as a string and returns a list of all the artist's songs with the variables that the Spotify algorithm measures. I chose to only select the variables I care about: artist_name, danceability, album_release_date, energy, valence, track_name, and album_name. Danceability, energy, and valence interested me the most out of all the metrics because I am the least certain about how they measured these variables in specific. I inferred that the other variables are more about the song composition, but I'm not sure what is measured when seeing how "danceable", "energetic", or "happy" a song is and I feel like it could involve a lot of different factors. There are a few of Lana Del Rey's albums from her earlier releases that are just two albums semi-combined so I removed those since they are really similar to each other.

```
LDR <- get_artist_audio_features('lana del rey') %>%
  select(artist_name, album_release_date, danceability, energy, valence, track_name, album_name) %>%
  filter(album_name!='Born To Die - The Paradise Edition' & album_name!='Born To Die (Bonus Track Versi
           album_name!='Born To Die - Paradise Edition (Special Version)')
head(LDR)
```

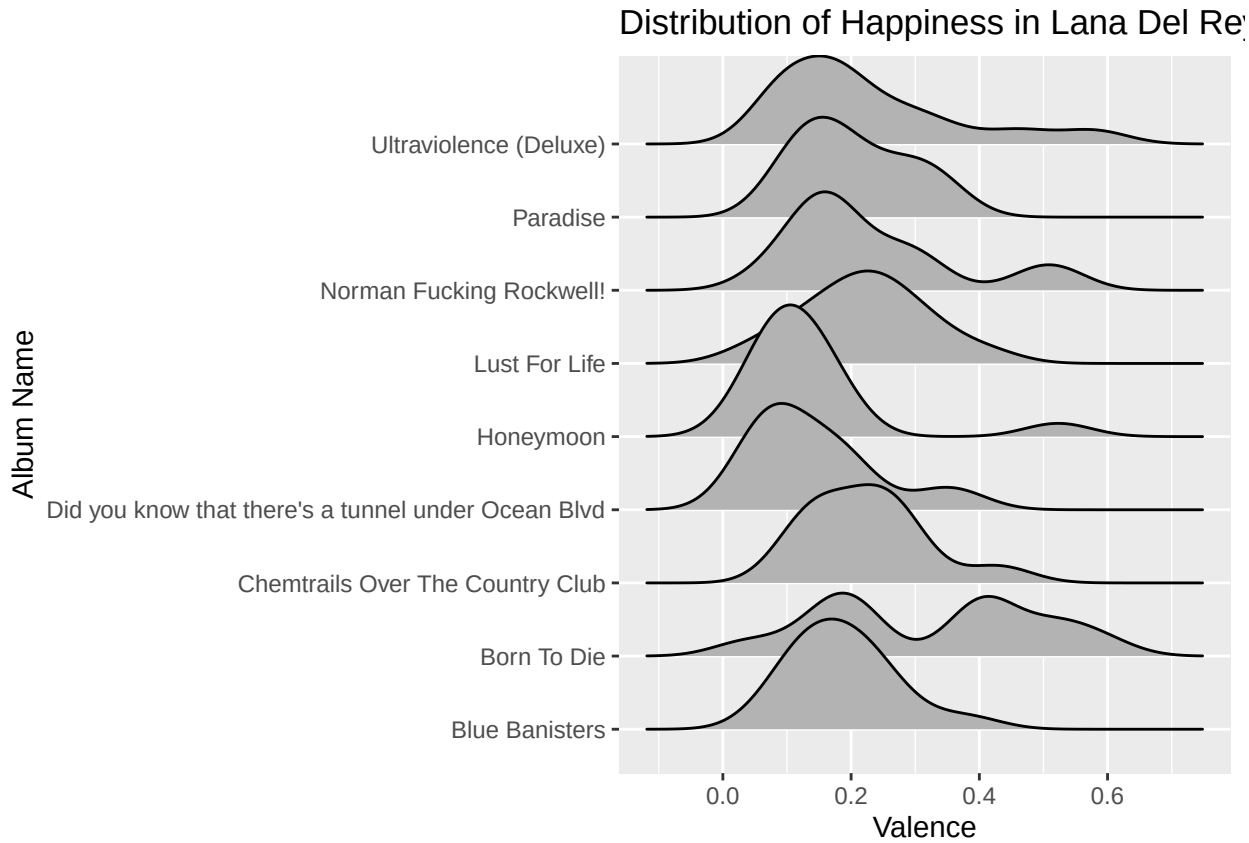
```
##    artist_name album_release_date danceability energy valence
## 1 Lana Del Rey         2023-03-24        0.366  0.156  0.1310
## 2 Lana Del Rey         2023-03-24        0.441  0.326  0.0526
## 3 Lana Del Rey         2023-03-24        0.305  0.241  0.0688
## 4 Lana Del Rey         2023-03-24        0.429  0.239  0.1370
## 5 Lana Del Rey         2023-03-24        0.474  0.324  0.3350
## 6 Lana Del Rey         2023-03-24        0.391  0.140  0.0937
##                                           track_name
## 1                                         The Grants
## 2 Did you know that there's a tunnel under Ocean Blvd
## 3                                              Sweet
## 4                                                A&W
## 5                                Judah Smith Interlude
## 6                  Candy Necklace (feat. Jon Batiste)
```

```
##                                           album_name
## 1 Did you know that there's a tunnel under Ocean Blvd
## 2 Did you know that there's a tunnel under Ocean Blvd
## 3 Did you know that there's a tunnel under Ocean Blvd
## 4 Did you know that there's a tunnel under Ocean Blvd
## 5 Did you know that there's a tunnel under Ocean Blvd
## 6 Did you know that there's a tunnel under Ocean Blvd
```

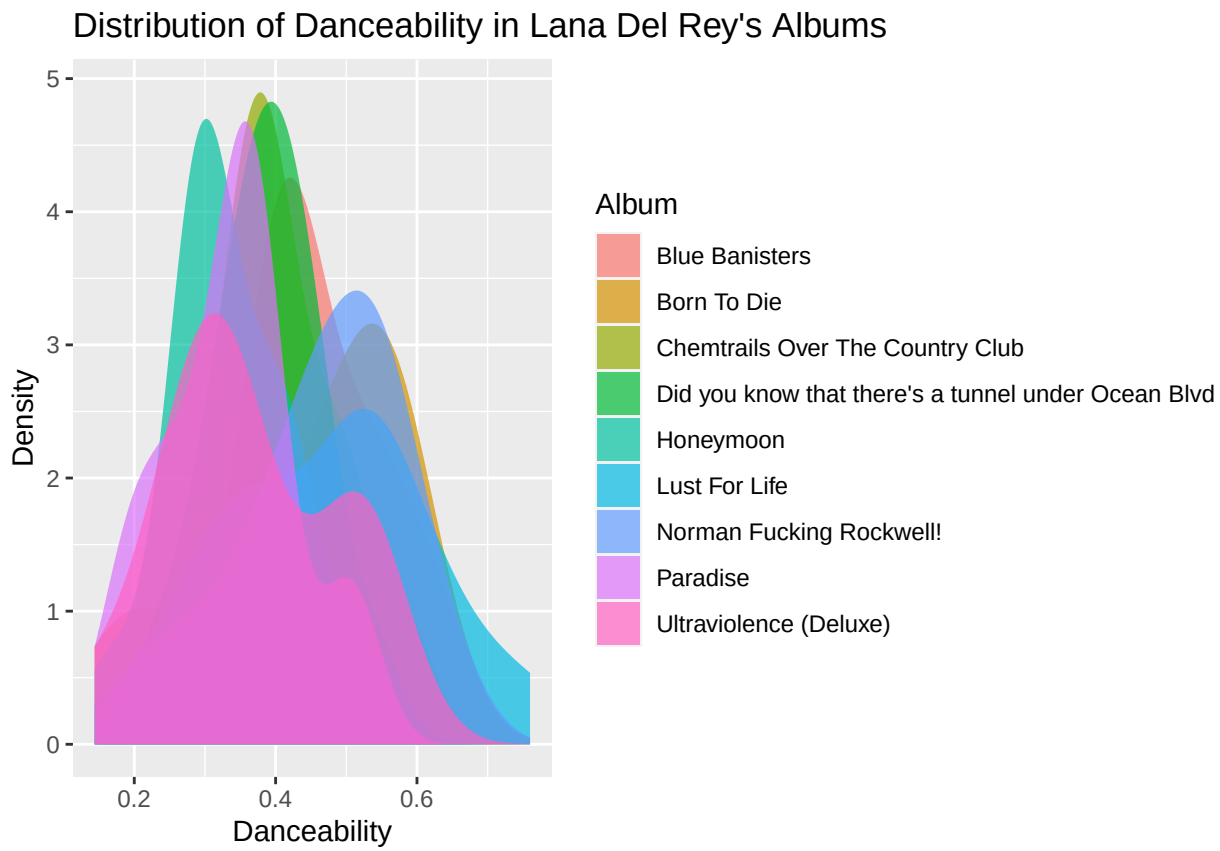
We can start to plot our data to get an idea of the distribution of our variables across albums. Using functions from the `tidyverse` and `ggjoy` packages, we can make density graphs. The first plot shows us the distribution of the `valence` variable in each album across all songs. Having the graph split up by album lets us easily see the distribution of each album while also comparing all the albums to each other. The second plot shows us the distribution of the `danceability` variable in each album across all songs layered against each other.

```
ggplot(LDR, aes(x = valence, y = album_name)) +
  geom_density_ridges() +
  labs(x = "Valence", y = "Album Name", title = "Distribution of Happiness in Lana Del Rey's albums")
```

```
## Picking joint bandwidth of 0.0521
```




```
ggplot(LDR, aes(x = danceability, fill = album_name)) +
  geom_density(alpha = 0.7, color = NA) +
  labs(x = "Danceability", y = "Density", title = "Distribution of Danceability in Lana Del Rey's Albums
  guides(fill = guide_legend(title = "Album"))
```



Discussion From these plots, we can see how these Spotify metrics are distributed across each album. While we are visually comparing the albums to each other, Spotify algorithms are analytically comparing albums to each other in order to later strengthen their recommendation algorithm. In this project, we are only looking at and doing comparisons with one artist's data but the algorithms look at thousands in order to curate a playlist or give a song that is similar to a user's listening history based on these metrics. A further analysis we can do with this data is to see what other Lana Del Rey album we could recommend based on knowing a person's favorite Lana Del Rey album.

References

https://www.music-tomorrow.com/blog/how-spotify-recommendation-system-works-a-complete-guide-2022

https://www.rcharlie.com/spotifyr/

https://msmith7161.github.io/what-is-speechiness/

https://www.cnbc.com/2022/11/10/how-spotify-stayed-no-1-in-streaming-music-vs-apple-youtube-amazon.html