# Computational Social Science

## Observational Studies and Application Programming Interfaces II

Dr. Thomas Davidson

Rutgers University

February 9, 2022

# Plan

- Course updates
- Recap on APIs
- Using the Spotify API in R
- Exercise

# Course updates

## Homework

- Homework due Friday at 5pm ET.
  - Please push your final version to Github with the appropriate commit message
  - Office hours today 5-6pm, 109 Davison Hall or Zoom (link in Canvas announcement)

# Recap

## APIs

- Online data sources for social science
  - Big data, observational data, digital trace data
- Application Programming Interfaces allow us to easily collect these kinds of data
  - API queries
  - JSON data
  - Rate-limiting
- Interacting with the Github API in R

# APIs

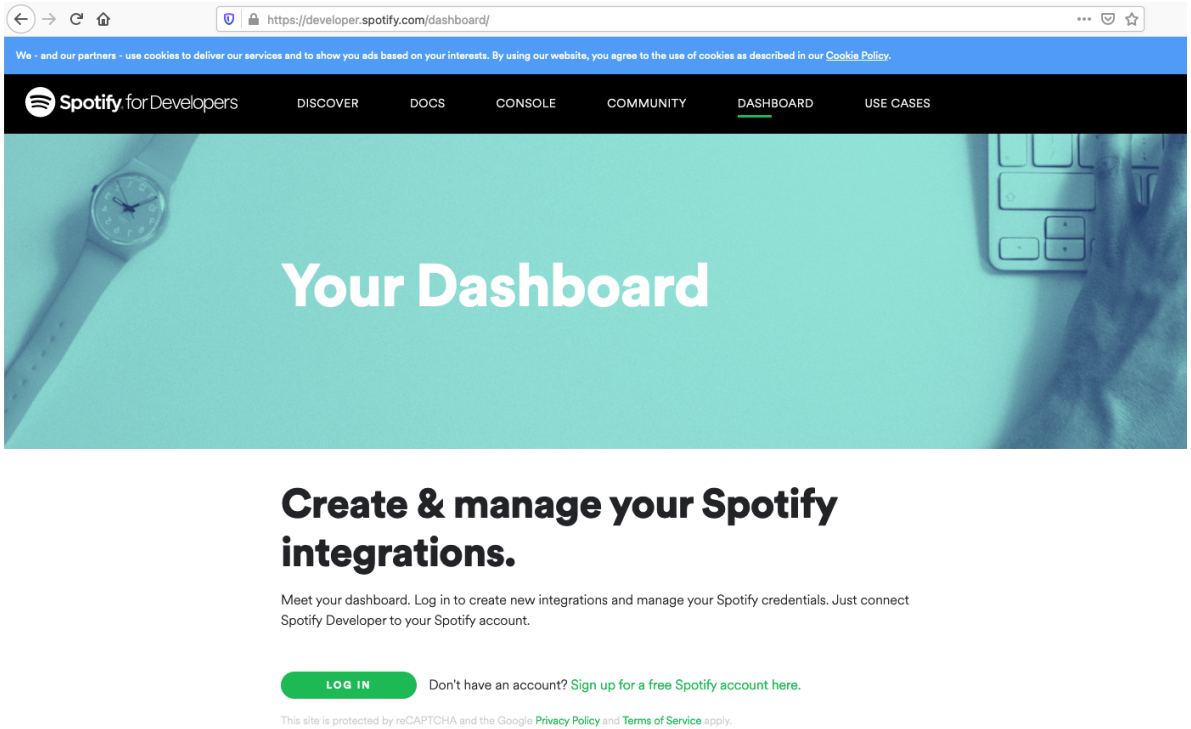
## Using the Spotify API

- It's always good to start by reading the documentation
  - https://developer.spotify.com/documentation/web-api/
- This provides information on the API, endpoints, rate-limits, etc.

# APIs

## Using the Spotify API

This API requires authentication. Let's log in to use the API.




https://developer.spotify.com/dashboard/

# APIs

## Using the Spotify API

Accept the terms of service then click on this button to create a new app.

# APIs

## Using the Spotify API

- Add a name and a short description
  - e.g. “Computational Social Science”, “App for class”
- Click on the app in Dashboard
- Click “Edit Settings”
  - Add http://localhost:1410/ to the Redirect URIs and click Save
- Click “SHOW CLIENT SECRET”
  - Copy Client ID and Client Secret

# APIs

## Using the Spotify API

- Open `creds.json` (located in the `slides` folder) and paste the ID and secret into the relevant fields.
  - Storing credentials in a separate file helps to prevent them from getting committed to Github accidentally
- The file should look ike this:

```
{"id": "328248djkejf298382189du329323c",
"secret": "jw7329889d37f7798383e8d29ew2d"}
```

# APIs

## Using the Spotify API

We're going to be using `spotifyr`, a *wrapper* around the spotify API. This allows us to make use of the functionality without needing to write the API calls, make requests, or convert the results to JSON/tabular format.

```
# install.packages('spotifyr') # uncomment and run to install
library(spotifyr)
library(tidyverse)
library(jsonlite)
library(lubridate)
```

You can read more about the library here.

# APIs

## Using the Spotify API

Now let's read in the credentials and create a token.

```
creds <- read_json("creds.json")  # read creds

Sys.setenv(SPOTIFY_CLIENT_ID = creds$id)  # set creds
Sys.setenv(SPOTIFY_CLIENT_SECRET = creds$secret)

access_token <- get_spotify_access_token()  # retrieve access token
```

# APIs

## Using the Spotify API

Now we're authorized, we can use the package to retrieve information from the API. Let's take a look at one of the functions. Rather than writing all the query code ourselves, we can just pass query parameters to the function.

```
`?`(get_artist_audio_features)
print(get_artist_audio_features)
```

# APIs

## Using the Spotify API

Now we're authorized, we can use the package to retrieve information from the API. Let's take a look at one of the functions.

```
kanye <- get_artist_audio_features("Kanye West") %>% as_tibble()
head(kanye)
```

```
## # A tibble: 6 x 39
##   artist_name artist_id          album_id album_type album_images albu
##   <chr>       <chr>              <chr>    <chr>      <list>        <chr
## 1 Kanye West  5K4W6rqBFWDnAN6~ 2Wiyo7L~ album      <df [3 x 3]> 2021
## 2 Kanye West  5K4W6rqBFWDnAN6~ 2Wiyo7L~ album      <df [3 x 3]> 2021
## 3 Kanye West  5K4W6rqBFWDnAN6~ 2Wiyo7L~ album      <df [3 x 3]> 2021
## 4 Kanye West  5K4W6rqBFWDnAN6~ 2Wiyo7L~ album      <df [3 x 3]> 2021
## 5 Kanye West  5K4W6rqBFWDnAN6~ 2Wiyo7L~ album      <df [3 x 3]> 2021
## 6 Kanye West  5K4W6rqBFWDnAN6~ 2Wiyo7L~ album      <df [3 x 3]> 2021
## # ... with 33 more variables: album_release_year <dbl>,
## #   album_release_date_precision <chr>, danceability <dbl>, energy <
## #   key <int>, loudness <dbl>, mode <int>, speechiness <dbl>,
## #   acousticness <dbl>, instrumentalness <dbl>, liveness <dbl>, vale
```

# APIs

## Using the Spotify API

```
head(kanye$track_name, n=10)
```

```
##  [1] "Donda Chant"                    "Hurricane"
##  [3] "Moon"                           "Life Of The Party (with
##  [5] "Off The Grid"                   "Jail"
##  [7] "Praise God"                     "Come to Life"
##  [9] "Believe What I Say"             "No Child Left Behind"
```

# APIs

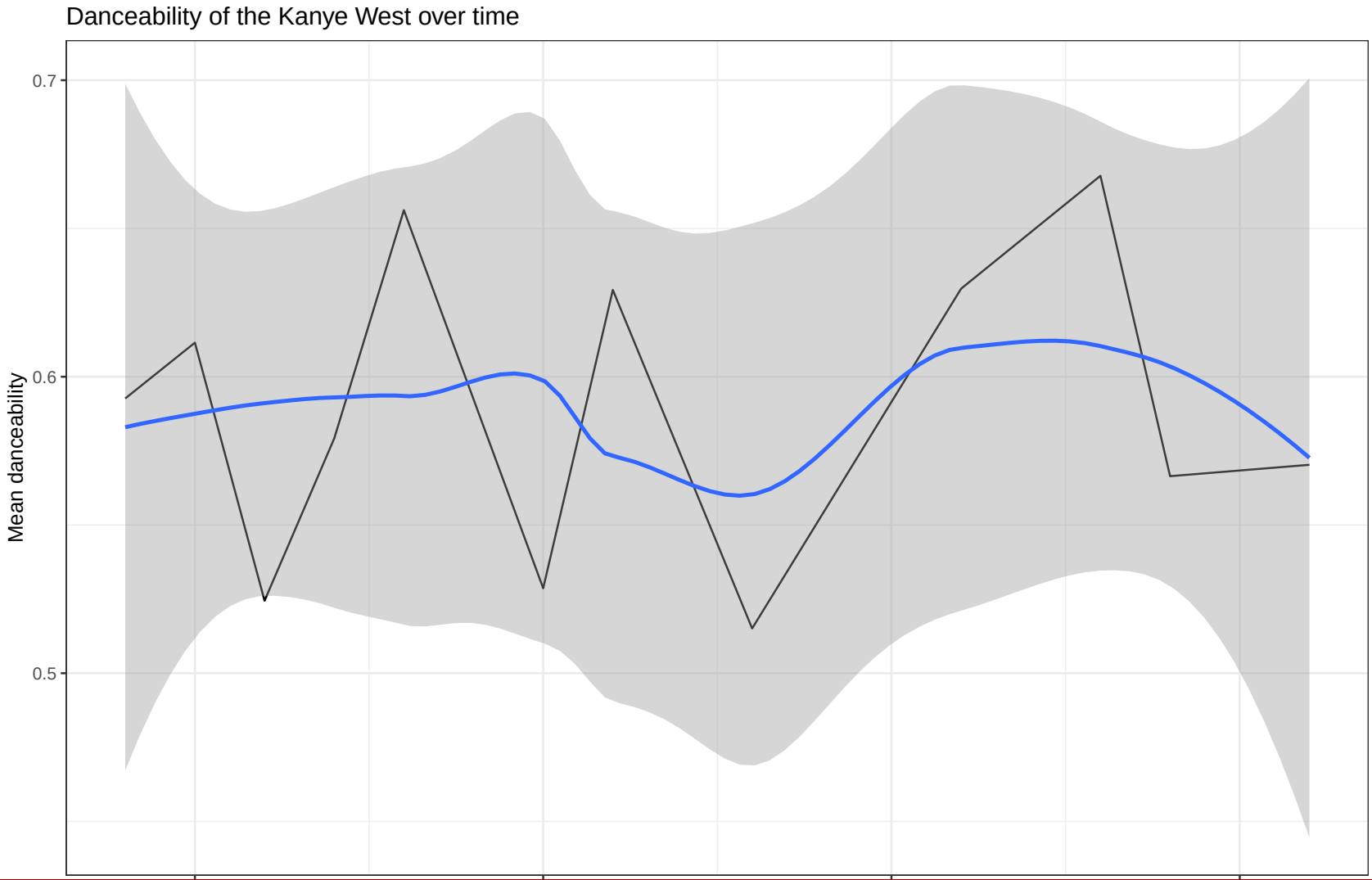
## Using the Spotify API

Let's calculate some statistics using this table.

```
results <- kanye %>%
             group_by(album_release_year) %>%
  summarize(mean.dan = mean(danceability),
            mean.ac = mean(acousticness))
```

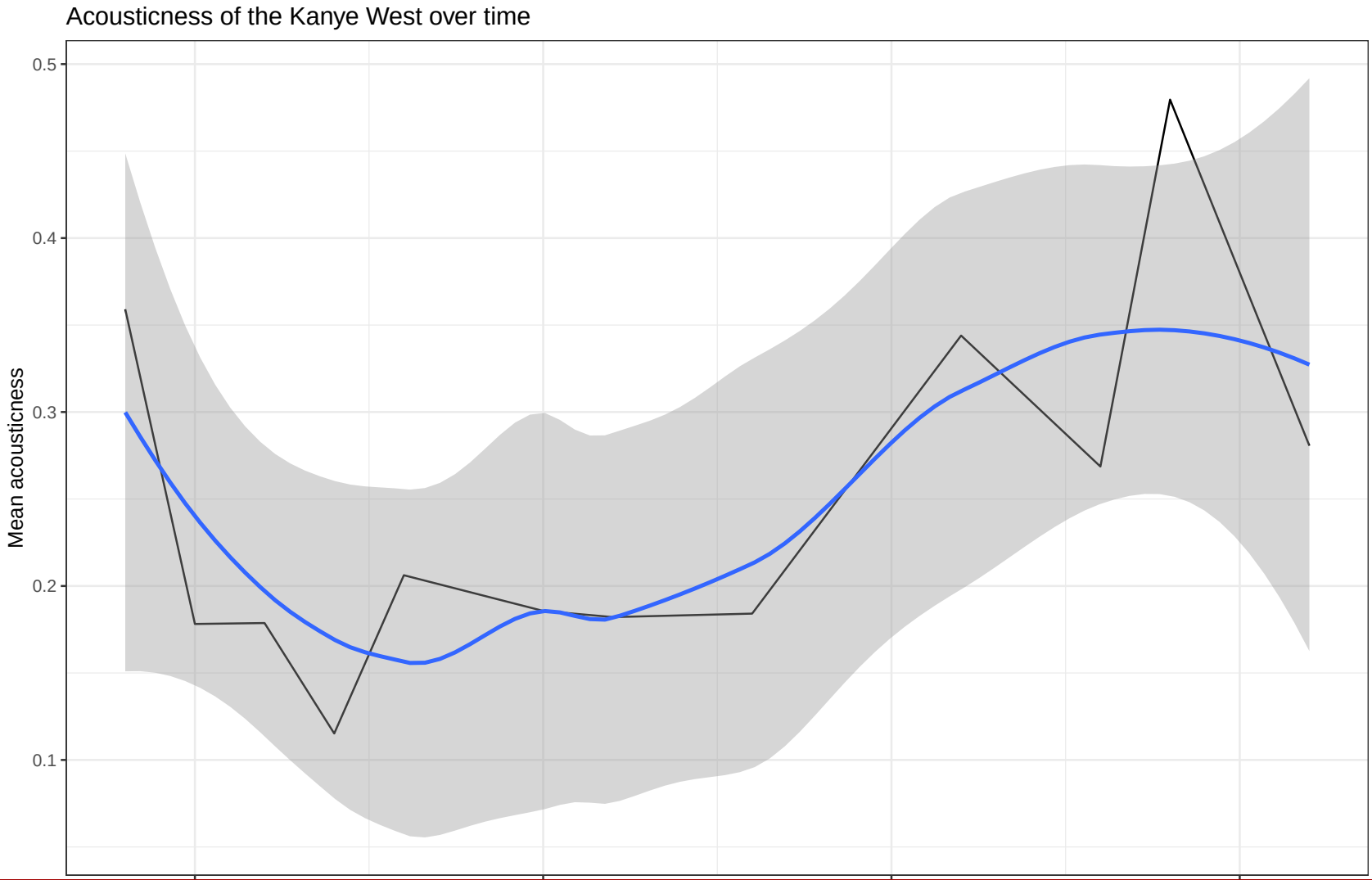
# APIs

## Using the Spotify API



# APIs

## Using the Spotify API



# APIs

## Using the Spotify API

Let's collect the same data for Taylor Swift and combine it.

```
taylor <- get_artist_audio_features("Taylor Swift") %>% as_tibble()
both <- bind_rows(kanye, taylor) # adding TS to the same tibble

head(both)
```

```
## # A tibble: 6 x 39
##   artist_name artist_id          album_id album_type album_images albu
##   <chr>       <chr>              <chr>    <chr>      <list>       <chr
## 1 Kanye West  5K4W6rqBFWDnAN6~ 2Wiyo7L~ album      <df [3 x 3]> 2021
## 2 Kanye West  5K4W6rqBFWDnAN6~ 2Wiyo7L~ album      <df [3 x 3]> 2021
## 3 Kanye West  5K4W6rqBFWDnAN6~ 2Wiyo7L~ album      <df [3 x 3]> 2021
## 4 Kanye West  5K4W6rqBFWDnAN6~ 2Wiyo7L~ album      <df [3 x 3]> 2021
## 5 Kanye West  5K4W6rqBFWDnAN6~ 2Wiyo7L~ album      <df [3 x 3]> 2021
## 6 Kanye West  5K4W6rqBFWDnAN6~ 2Wiyo7L~ album      <df [3 x 3]> 2021
## # ... with 33 more variables: album_release_year <dbl>,
## #   album_release_date_precision <chr>, danceability <dbl>, energy <
## #   key <int>, loudness <dbl>, mode <int>, speechiness <dbl>,
```

# APIs

## Using the Spotify API

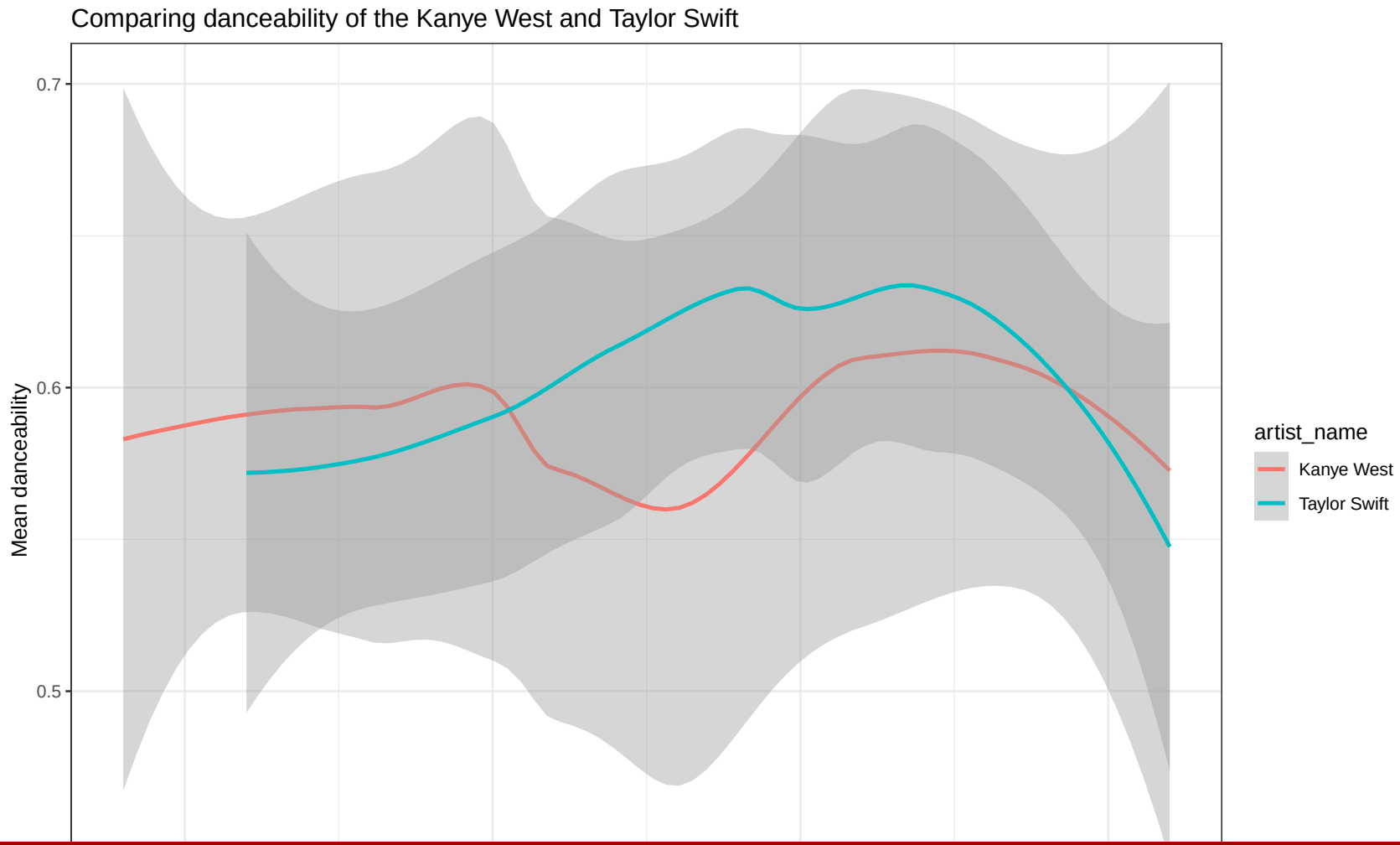
Repeating the summary operation for both artists. Note how we now group by `artist_name` in addition to `album_release_year`.

```
r <- both %>%
  group_by(album_release_year, artist_name) %>%
  summarize(mean.dan = mean(danceability),
            mean.ac = mean(acousticness))
```

# APIs

## Using the Spotify API

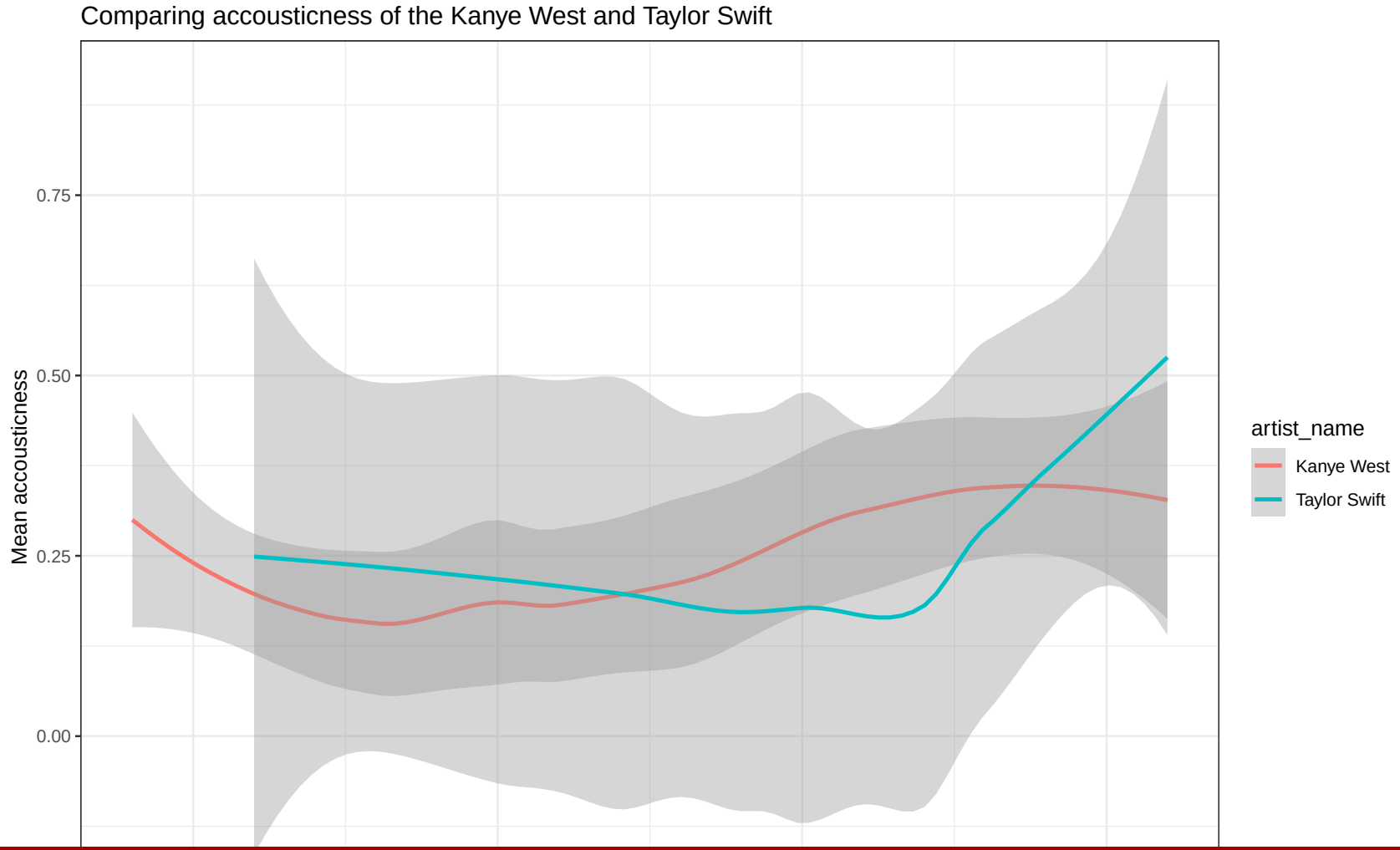
Let's compare their danceability.



# APIs

## Using the Spotify API

We can do the same for acousticness.



# APIs

## Using the Spotify API

Let's try another type of query.

```
## # A tibble: 10 x 4
##    id                     name          popularity followers.total
##    <chr>                  <chr>              <int>           <int>
##  1 3TVXtAsR1Inumwj472S9r4 Drake                 98        60806853
##  2 7dGJo4pcD2V6oG8kP0tJRR Eminem                94        52098320
##  3 15UsOTVnJzReFVN1VCnxy4 XXXTENTACION          92        34488770
##  4 0hCNtLu0JehylgoiP8L4Gh Nicki Minaj           90        24026744
##  5 2YZyLoL8N0Wb9xBt1NhZWg Kendrick Lamar        90        19263564
##  6 6l3HvQ5sa6mXTsMTB19rO5 J. Cole               90        15780811
##  7 55Aa2cqylxrFIXC767Z865 Lil Wayne             91        11539843
##  8 1RyvyyTE3xzB2ZywiAwp0i Future               92        11122090
##  9 1URnnhqYAYcrqrcwql10ft 21 Savage             90        10753381
## 10 5f7VJjfbwm532GiveGC0ZK Lil Baby              93         9525530
```

# APIs

## Using the Spotify API

Now we have a list of artists, let's use this information as input for another query.

# APIs

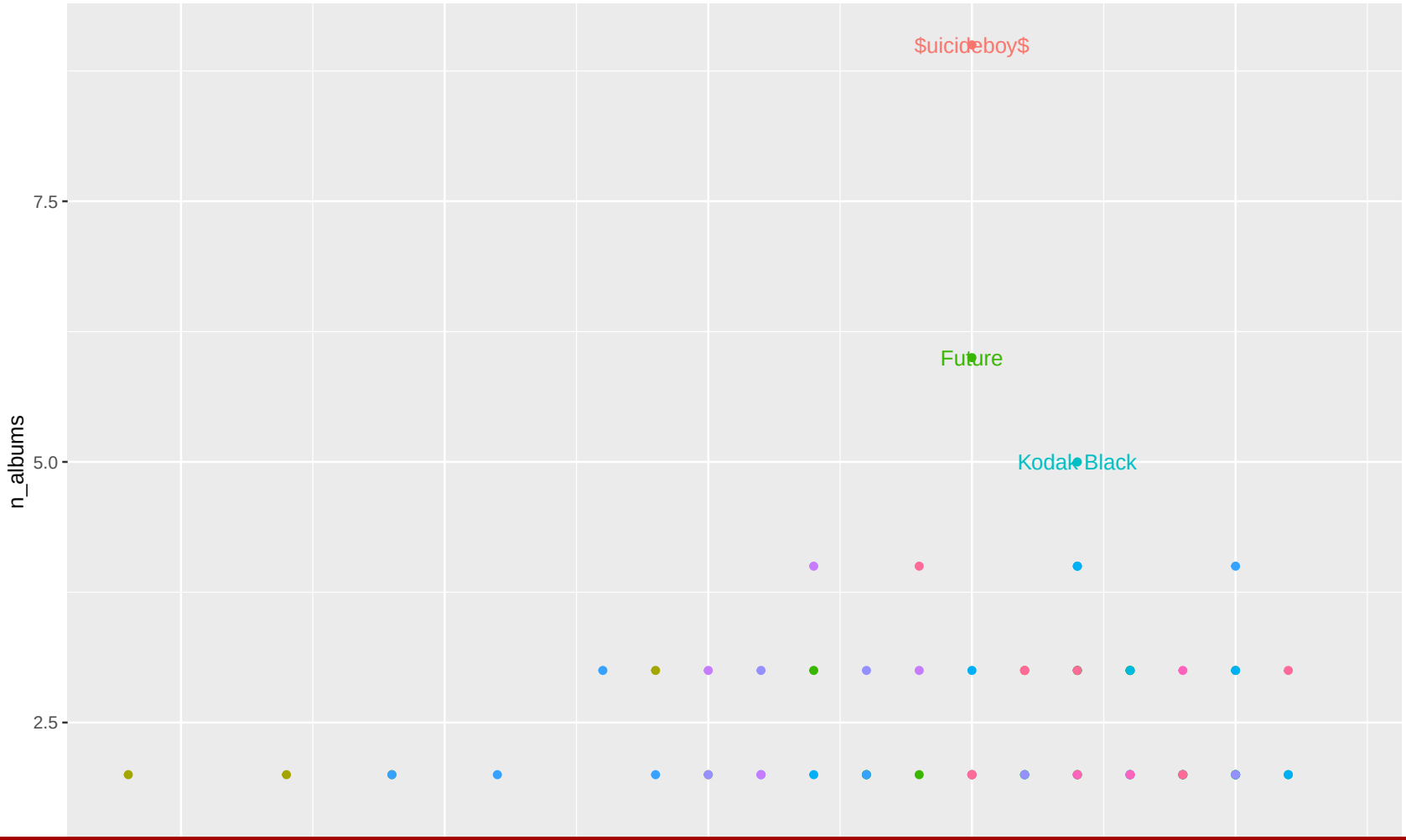
## Using the Spotify API

Let's create a summary of the data. In this case, let's count the number of albums each artist released each year. Why is `n_distinct` useful here?

# APIs

## Using the Spotify API

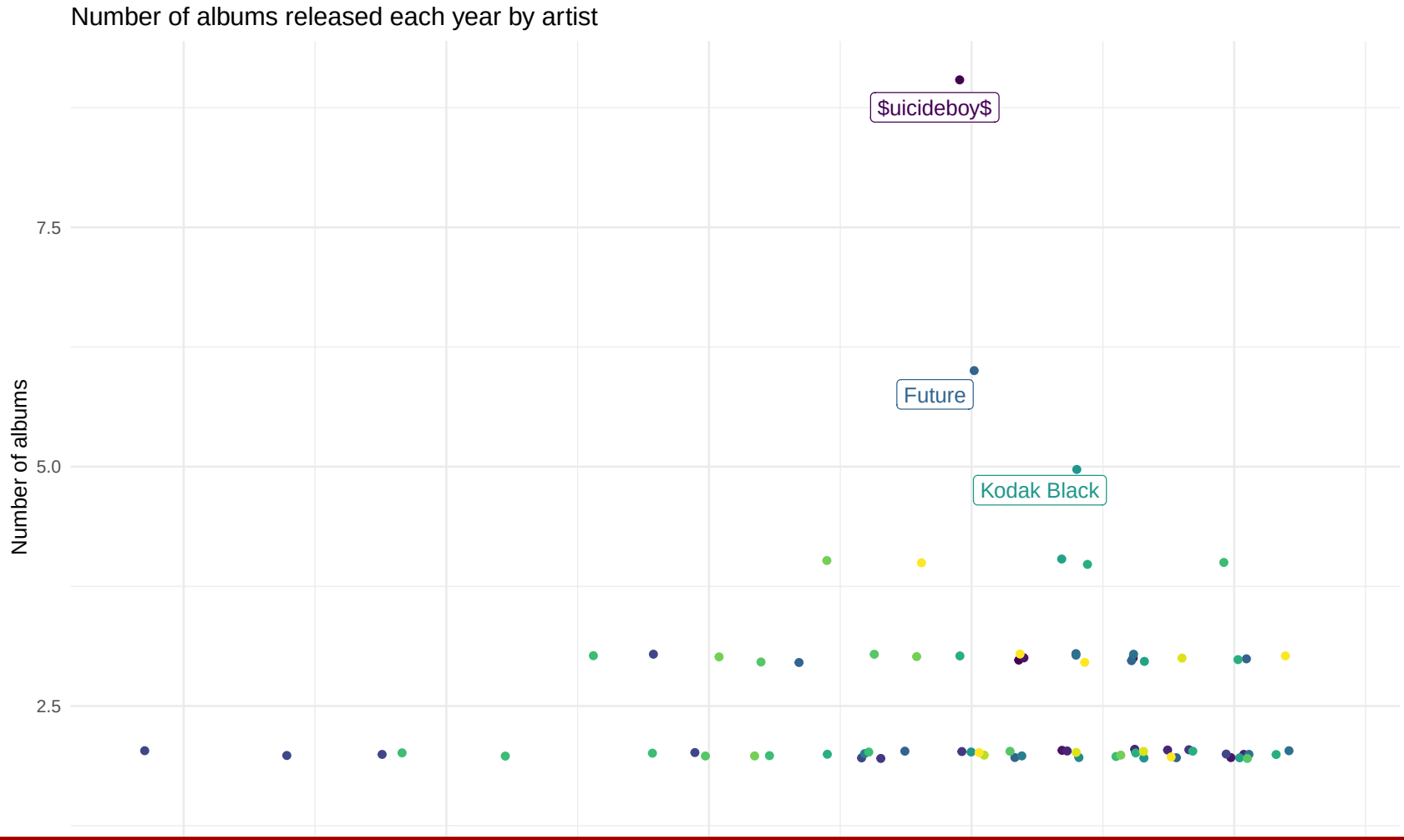
We can represent these data using a scatterplot.



# APIs

## Using the Spotify API

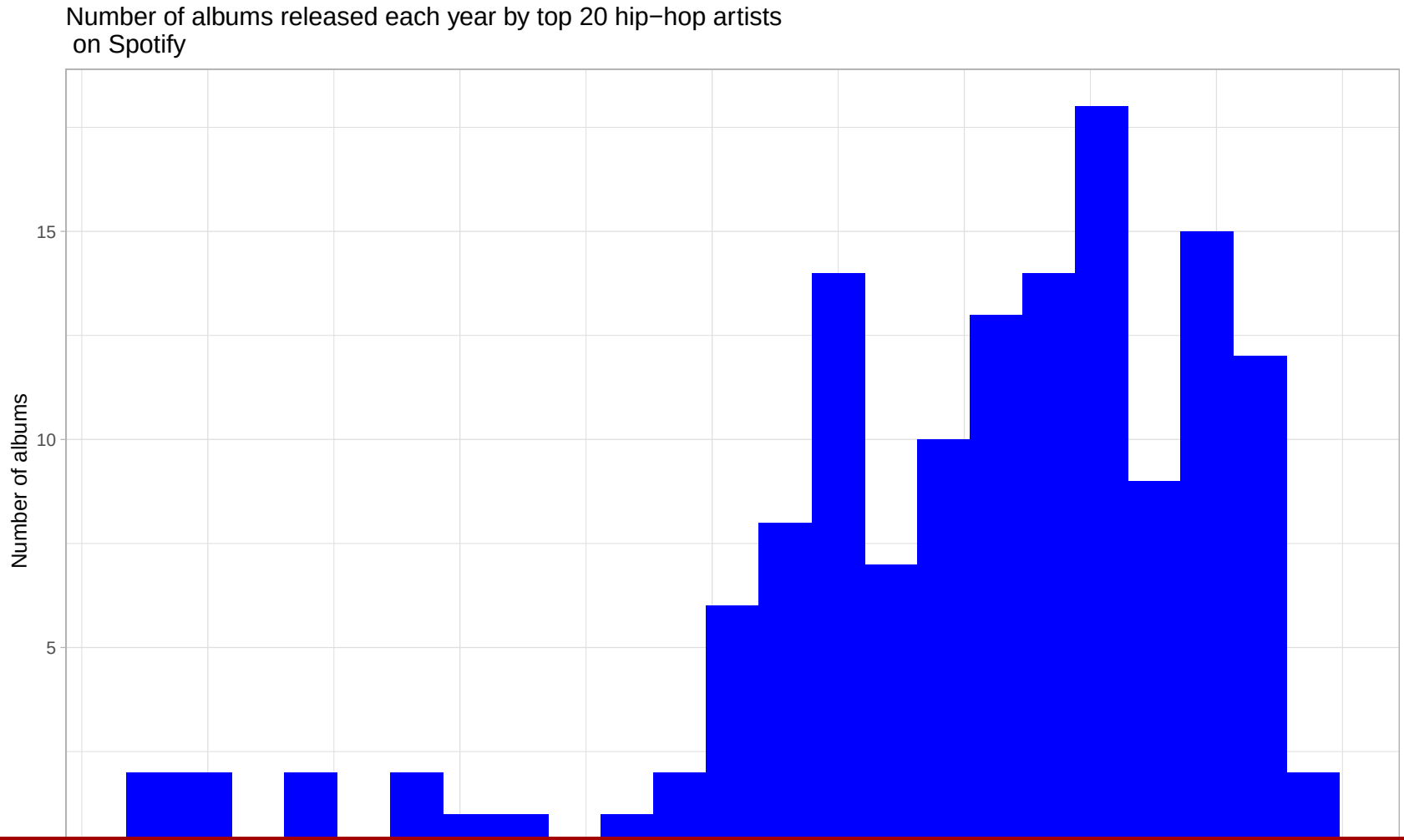
Let's try to make this plot look a little better.



# APIs

## Using the Spotify API

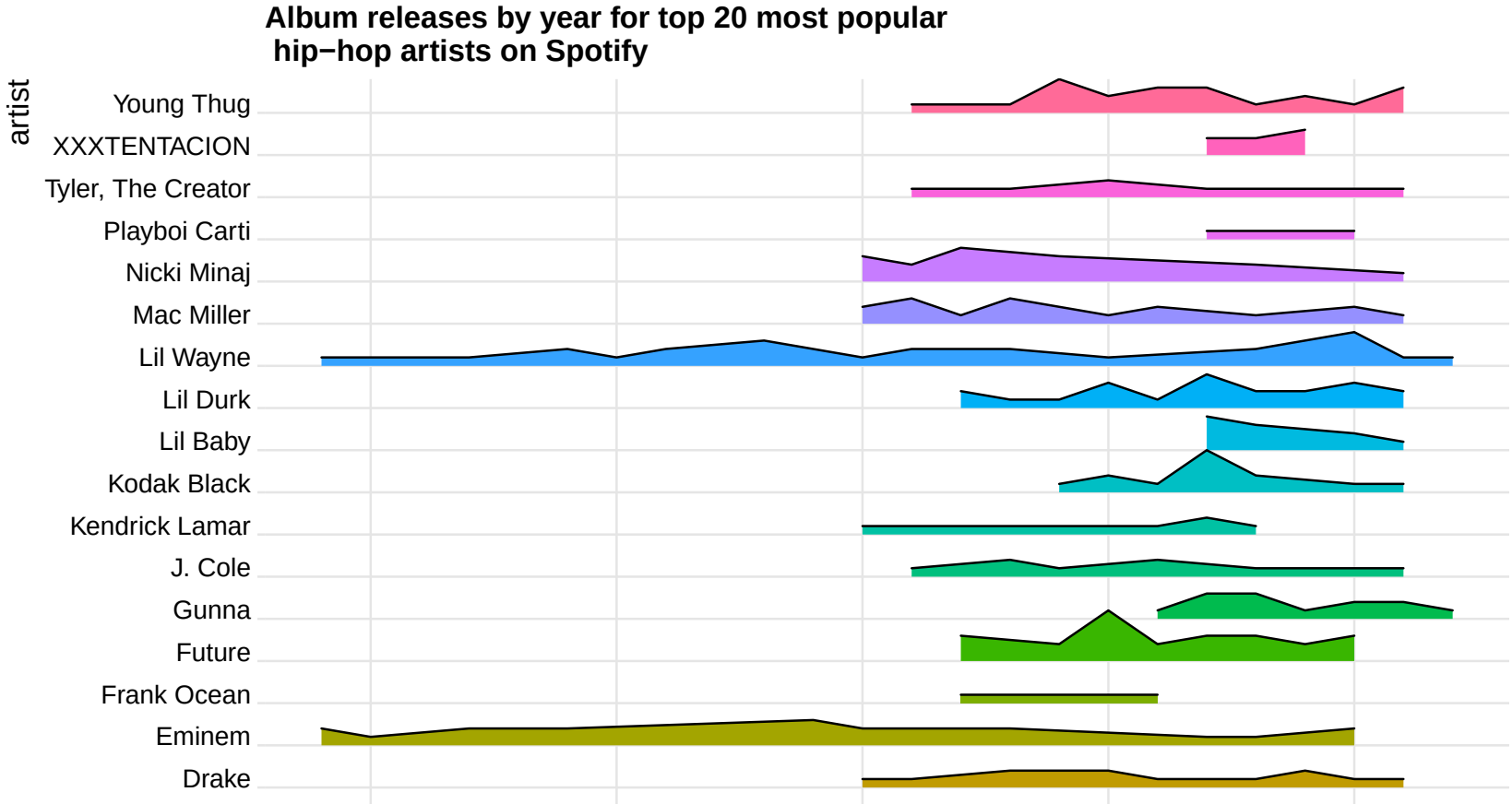
We could also plot the overall values using a histogram.



# APIs

## Using the Spotify API

There are other extensions of `ggplot` that can create even more sophisticated plots. The `ggridges` package allows us to represent multiple artists' trends as overlaid histograms.



# APIs

**Exercise**

1. Use the Spotify API to collect your own data.
2. Use tidyverse functions to select relevant columns and summarize (as necessary)
3. Product a plot using ggplot.
4. Share the plot in this Google Doc: https://bit.ly/3rAG7Uk

# APIs

## Exercise

# APIs

## Accessing your Spotify information

- Some features require a Spotify login
  - You can only use these features if you have set `http://localhost:1410/` in Redirect URIs and authorized your app
  - This tells the API to open up authentication on port 1410 of your computer
  - Note: You may need to install the package `httpuv` for this to work

# APIs

## Accessing your Spotify information

To access your personal data, you can run this code to look at your most recently played tracks. There are many other functions you can use to get and even modify your own data (so use these carefully!). You will have to type 1 into the console after running the chunk.

```
get_my_recently_played(limit = 5) %>%
    mutate(artist.name = map_chr(track.artists, function(x) x$name[1]),
           played_at = as_datetime(played_at)) %>%
    select(track.name, artist.name, track.album.name, played_at) %>% as
```

Example from the `spotifyr` documentation.

# Next week

- Webscraping