Computational Social Science

Observational Studies and Application Programming Interfaces II

Dr. Thomas Davidson

Rutgers University

September 25, 2024

Plan

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

Recap

- 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

Using the Spotify API

Documentation

- 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.

Using the Spotify API

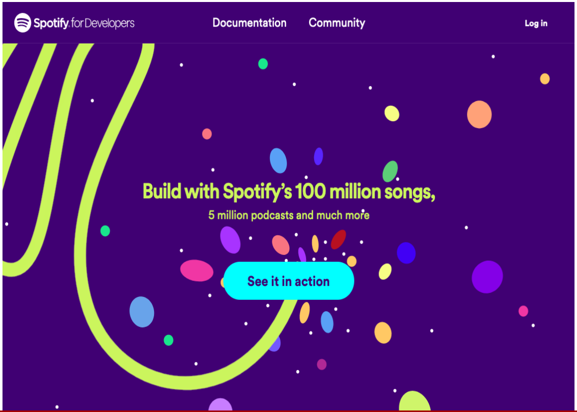
Signing up

- To use the Spotify API you need a Spotify account
- If you don't have one already, please sign up for a free account

Using the Spotify API

Signing up

This API requires authentication. Visit the link below and click "Log in". If you are already logged in, click your username in the top-right then "Dashboard".


Spotify for Developers
Documentation
Community
Log in
Build with Spotify's 100 million songs,
5 million podcasts and much more
See it in action

Using the Spotify API

Creating an app

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

Using the Spotify API

Creating an app

- Add a name and a short description
  - e.g. “Computational Social Science”, “App for class”
- Click on the app in Dashboard
- Click “SHOW CLIENT SECRET”
  - Copy Client ID and Client Secret and store them somewhere safe

APIs

Access credentials

- Often APIs will use credentials to control access
  - A *key* (analogous to a user name)
  - A *secret* (analogous to a password)
  - An *access token* (grants access based on key and password)
  - Generally the access token is provided as part of the call
- Keep credentials private
  - Avoid accidentally sharing them on Github

APIs

JSON

- An API will commonly return data in JSON (JavaScript Object Notation) format
  - JSON files consist of key-value pairs, enclosed in braces as such: `{"key": "value"}`
  - JSON files are structured in a way that makes them relatively easy to parse to retrieve relevant data

Using the Spotify API

Storing credentials

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

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

Using the Spotify API

Loading packages

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. To install it, you must uncomment and run the line below.

```
# devtools::install_github('t-davidson/spotifyr') # uncomment and run t
# install
```

You can read more about the library here.

Using the Spotify API

Authentication

Now let's load the packages, read in the credentials, and create an access token. Run this chunk to proceed.

```
library(spotifyr)
library(tidyverse)
library(jsonlite)
library(lubridate)

creds <- read_json("../credentials/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
```

Using the Spotify API

API functions

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)
```

Using the Spotify API

Querying the 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. Add an artist name to `get_artist_audio_features`.

```
artist1 <- get_artist_audio_features("") %>% as_tibble() # Add artist n
head(artist1)
```

Using the Spotify API

Inspecting the data

```
head(artist1$track_name, n=10)
```

Using the Spotify API

Creating a summary

Let's calculate some statistics using this table. What does this show?

```
results <- artist1 %>%
            group_by(album_release_year) %>%
  summarize(mean.dan = mean(danceability),
            mean.ac = mean(acousticness))
```

Using the Spotify API

Visualizing the data

```
p <- ggplot(artist1, aes(x=album_release_year, y=danceability))
p + geom_smooth() +
  labs(title="Danceability over time",
       caption = "Data from collect from Spotify API") +
  xlab("") + ylab("Mean danceability") + theme_bw()
```

Using the Spotify API

Visualizing the data

```
p <- ggplot(artist1, aes(x=album_release_year, y=acousticness))
p + geom_smooth() +
  labs(title="Acousticness over time",
       caption = "Data from collect from Spotify API")  +
    xlab("") + ylab("Mean acousticness") + theme_bw()
```

Using the Spotify API

Collecting more data

Let's collect the same data for a second artist and combine it. Add an artist name to the `get_artist_audio_features`.

```
artist2 <- get_artist_audio_features("") %>% as_tibble()
both <- bind_rows(artist1, artist2) # adding 2nd artist to the same tib
both %>% sample_n(5) %>% select(artist_name)
```

Using the Spotify API

Creating a new summary

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))
```

Using the Spotify API

Comparing the artists

```
p <- ggplot(both, aes(x=album_release_year, y=danceability,
                      group = artist_name, color = artist_name))
p + geom_point(alpha=0.3) + geom_smooth() +
  labs(title="Comparing danceability",
       caption = "Data from collect from Spotify API") +
  xlab("") + ylab("Mean danceability") + theme_bw()
```

Using the Spotify API

Comparing the artists

```
p <- ggplot(both, aes(x=album_release_year, y=acousticness,
                      group = artist_name, color = artist_name))
p + geom_point(alpha=0.1) + geom_smooth() +
  labs(title="Comparing acousticness",
       caption = "Data from collect from Spotify API") +
  xlab("") + ylab("Mean acousticness") + theme_bw()
```

Using the Spotify API

Collecting more data

Let's try another type of query. Add a genre name to get_genre_artists. Note that not all genres will work.

```
## # A tibble: 10 x 4
##    id                     name                     popularity follow
##    <chr>                  <chr>                         <int>
##  1 5a2EaR3hamoenG9rDuVn8j Prince                           71
##  2 2xiIXseIJcq3nG7C8fHeBj Three Days Grace                 77
##  3 1snhtMLeb2DYoMOcVbb8iB Kenshi Yonezu                    76
##  4 3CkvROUTQ6nRi9yQOcsB50 Genesis                          68
##  5 7r8RF1tN2A4CiGEplkp1oP Ginuwine                         66
##  6 0Sadg1vgvaPqGTOjxu0N6c Girls' Generation                64
##  7 2cy1zPcrFcXAJTP0APWewL Gente De Zona                    70
##  8 3bGXaFVQLASmDMdjjeJr8a Montgomery Gentry                58
##  9 7nzSoJISlVJsn7O0yTeMOB Joe Hisaishi                     71
## 10 0MK8l3nURwwQIjafvXoJJt ASIAN KUNG-FU GENERATION         61
```

Using the Spotify API

Programming complex queries

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

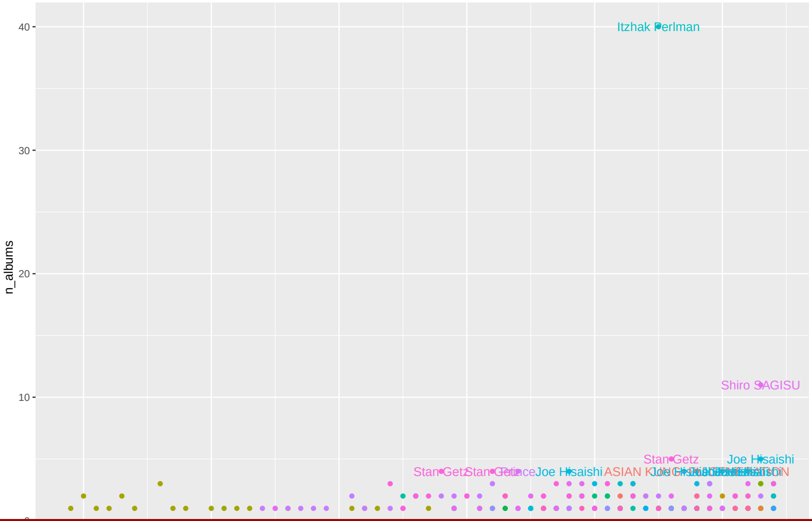
Using the Spotify API

Creating a summary

Let's count the number of albums each artist released each year. Why is `n_distinct` useful here?

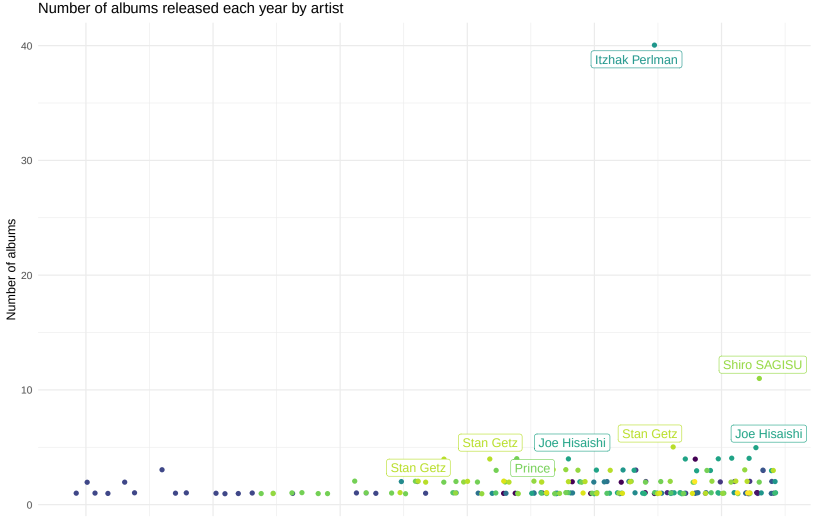
Using the Spotify API

Visualizing the data



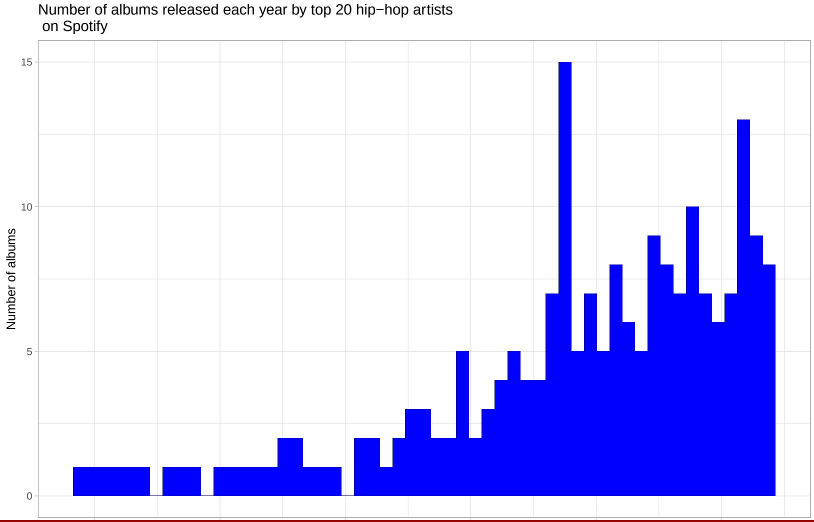
Using the Spotify API

Improving the visualization



Using the Spotify API

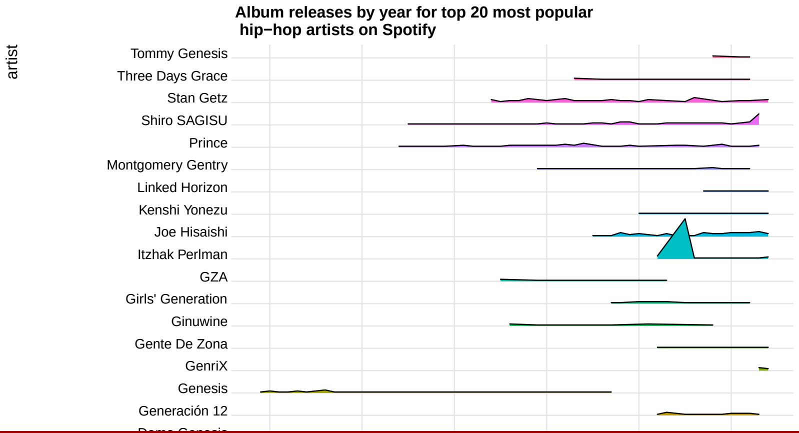
Creating a histogram



Using the Spotify API

More advanced visualizations

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.



Using the Spotify API

Exercise

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

Using the Spotify API

Exercise

Summary

- Application programming interfaces provide programmatic access to data stored on websites and social media platforms, making them an ideal source of digital trace data for social scientific research
- APIs can be queried using web requests or custom R packages, making them relatively easy to use
- But major social media platforms have cut back access to APIs and smaller websites do not have them

Next week

- Collecting data from websites using webscraping