Star Wars Music Analysis Using Spotify API

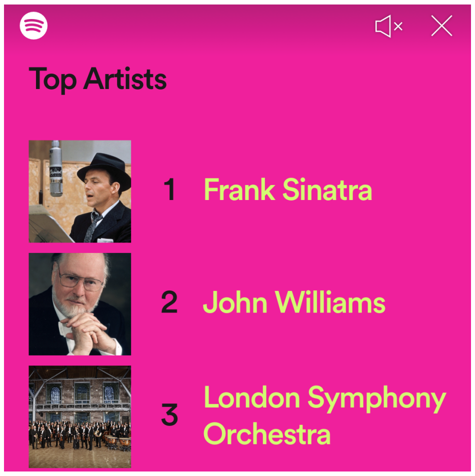
Ben Borovinsky

2020-11-02

It's no secret that *Star Wars* has a massive following. What began as a space opera in 1977 has evolved into a franchise with universal recognition and fandom. *Star Wars* has expanded from a cult following to being a significant part of culture and media.

There are many elements that make *Star Wars* what it is. The iconic characters, cringe dialogue, distinct planets, and, of course, the lightsabers, are all significant pieces that make up the successful franchise. In recent years, the movies themselves have been subject to criticism, but one thing has remained consistently positive since the original trilogy: the music. John Williams has scored many iconic films, winning five academy awards and being nominated for 52 (second all-time behind Walt Disney). Williams is as synonymous with *Star Wars* as George Lucas, for he has scored every film in the Skywalker Saga (Episode I-IX). Like Vin Scully in the world of sports broadcasting, John Williams is the greatest film composer of all-time.

I, like many other *Star Wars* fans, enjoy listening to John Williams outside of the movies. The music streaming application Spotify has been my medium of choice to listen to Williams' catalogue of *Star Wars* music, and I've listened to a lot of his music. At the end of 2019, Spotify released a personalized report of each user's listening habits, and two of the top three I listened to the most were John Williams and the London Symphony Orchestra (the other being Frank Sinatra).



My top artists of 2019.

Spotify's analysis of my listening habits have encouraged me to do some analytics of my own. More specifically, I want to further analyze the music of *Star Wars*. While I do not have an academic background in music and I consider myself at best an amateur at data analytics, I figured this would help me learn more about two things I am very passionate about: *Star Wars* music and data analytics.

Setup

The first step is to load in the libraries needed for this analysis.

```r
library(spotifyr)
library(dplyr)
library(shiny)
library(shinyWidgets)
library(knitr)
library(tidyr)
library(ggplot2)
```

Each library includes its own set of tools that are required for our analysis.

- `spotifyr` : gain access to the Spotify API
- `dplyr` : grammar and data manipulation
- `shiny` : output R code as an interactive web app
- `shinyWidgets` : more options for customizing web app
- `knitr` : structural elements to R code output
- `tidyr` : modify data for analysis
- `ggplot2` : data visualization

To finish setting everything up, I like to make sure that I'm not using any variables from a previous script or looking at a graph that is no longer relevant. I have also found it helpful to set the directory path to that of the current file so that I don't have to remember to do it when I open RStudio.

```r
rm(list=ls())
graphics.off()
dirpath <- dirname(rstudioapi::getSourceEditorContext()$path)
setwd(dirpath)
```

Getting the Data

Note that this section of code is not run to save time on data retrieval from Spotify (and to keep the credentials hidden).

In order to do an analysis, I need to get data. Thankfully, Spotify gives users access to its API, which makes the data acquisition relatively painless. Along with the information you would expect, the data from Spotify also includes metrics for each song to help give numerical values to the attributes of a song (this is probably how Spotify comes up with similar songs based on your listening habits or when you create a playlist); these metrics are danceability, energy, speechiness, acousticness, instrumentalness, liveness, and valence.

Accessing the Spotify API is simple. All that is needed is to register with the Spotify developers website (https://developer.spotify.com/documentation/web-api/quick-start/) and obtain user credentials.

```
client_id <- "###"
client_secret <- "###"

Sys.setenv(SPOTIFY_CLIENT_ID = client_id)
Sys.setenv(SPOTIFY_CLIENT_SECRET = client_secret)

tkn <- get_spotify_access_token(client_id, client_secret)
oauth <- get_spotify_authorization_code(scope = "playlist-read-private user-top-read")
```

Now that we have gained authorization to access the API, we can get all of the music from John Williams.

```
jw_complete <- get_artist_audio_features("john williams")
```

That's all we need from Spotify; the rest is up to us on what we want to do with the data. For starters, I want to analyze the John Williams music available to the United States.

```
jw_us <- jw_complete[grepl("US", jw_complete$available_markets),]
```

John Williams is a very accomplished composer who has conducted beautiful and iconic soundtracks for many films. However, this analysis focuses solely on his work for *Star Wars*.

```
jw_sw <- jw_us[grepl("Star Wars:.*Original Motion Picture Soundtrack", jw_us$album_name),]
```

Everything you could possibly imagine tied to Spotify track information is included in the data, and we don't need all of it. To make things a bit less overwhelming, I'm going to remove a good amount of columns that have no impact on the analysis.

```
jw_df <- jw_sw %>%
  select(!c(
    artist_id,
    album_id,
    album_type,
    album_images,
    album_release_date_precision,
    track_id,
    analysis_url,
    time_signature,
    artists,
    available_markets,
    disc_number,
    explicit,
    track_href,
    is_local,
    track_preview_url,
    type,
    track_uri,
    external_urls.spotify
  ))
```

At this point, the data is in a good spot to export to csv. Having a local copy of the Spotify data means that you can start your analysis offline using only the data you want (and you don't have to access the API every time you run the script).

```
write.csv(jw_df, "./data/john_williams_star_wars_data.csv", row.names=FALSE)
```

Data Cleaning

In the previous section, we retrieved John Williams' *Star Wars* music data from Spotify and removed extraneous columns. This section will show how to clean the data as well as add more columns that describe the data better and more succinctly.

First, let's import the csv file from the previous section and take a look at the columns in the data.

```
song_data <- read.csv("./data/john_williams_star_wars_data.csv", stringsAsFactors=FALSE)
colnames(song_data)
```

```
##  [1] "artist_name"        "album_release_date" "album_release_year"
##  [4] "danceability"       "energy"             "key"
##  [7] "loudness"           "mode"               "speechiness"
## [10] "acousticness"       "instrumentalness"   "liveness"
## [13] "valence"            "tempo"              "duration_ms"
## [16] "track_name"         "track_number"       "album_name"
## [19] "key_name"           "mode_name"          "key_mode"
```

One column I see missing from this collection of *Star Wars* music is the movie each song is associated with. While I can get that information from the `album_name` column, I would prefer to see an `episode` column that tells me the movie with just a number.

```
song_data <- song_data %>%
  mutate(episode = with(., case_when(
    (album_release_year == 1977) ~ 4,
    (album_release_year == 1980) ~ 5,
    (album_release_year == 1983) ~ 6,
    (album_release_year == 1999) ~ 1,
    (album_release_year == 2002) ~ 2,
    (album_release_year == 2005) ~ 3,
    (album_release_year == 2015) ~ 7,
    (album_release_year == 2017) ~ 8,
    (album_release_year == 2019) ~ 9
  )))
```

Notice that I'm using the `album_release_year` instead of `album_name` ; this is because I am lazy and didn't want to have to use regex if I don't have to. But it would be nice to also have the `movie_title` along with the `episode` . Thankfully, each `album_name` contiains the `movie_title` and follows the same format of *Star Wars: [Movie Title] (Original Motion Picture Soundtrack)* which will make the regex really easy.

```
song_data$movie_title <- song_data$album_name %>%
  gsub(".*[:]\\s(.*)\\s\\(.*","\\1",.)
```

Another piece of data we can add is the `trilogy` . Because we already created the `episode` column, this is even easier to create.

```
ELSE <- TRUE
song_data <- song_data %>%
  mutate(trilogy = with(., case_when(
    (episode <= 3) ~ "Prequel",
    (episode <= 6) ~ "Original",
    ELSE ~ "Sequel"
  )))
```

Now that we have all of the columns we need, we can reorganize the data to make more sense and omit the columns that seem redundant or don't add much to the data (i.e. `album_release_date` , `key` , `mode`).

```
song_data <- song_data[order(song_data$album_release_year, song_data$track_number),]
rownames(song_data) <- NULL

song_data <- song_data %>%
  select(
    episode:trilogy,
    track_number,
    track_name,
    artist_name,
    album_name,
    duration_ms,
    album_release_year,
    danceability:energy,
    speechiness:tempo,
    loudness,
    key_name:key_mode
  )
```

There are still a lot of columns that certainly won't be used in this analysis, so let's get only the columns we know we'll need.

```
song_data <- song_data %>%
    select(
        track_name,
        movie_title,
        trilogy,
        danceability:valence
    )
```

The data type in each column may not seem important now, but it will matter soon when outputing the data to a table or graph. More specifically, the character data needs to be converted to factor to preserve its natural order.

```
song_data$movie_title <- song_data$movie_title %>%
  {factor(., levels=unique(.))}
```

Quick Analysis

Before we start showing some tables, it'll be much easier going forward to save the names of the different audio features to a variable.

```
categories <- colnames(song_data)[-c(1:3)]
```

The average value for each audio feature per movie can be easily determined by grouping the tracks by their `movie_title`.

```
song_data %>%
  group_by(movie_title) %>%
  summarise(count = n(),
            across(all_of(categories), ~ mean(.x)),
            .groups = "keep") %>%
  kable(format = "markdown", digits = 3, row.names = FALSE)
```

movie_title	count	danceability	energy	speechiness	acousticness	instrumentalness	liveness	valence
A New Hope	16	0.268	0.198	0.045	0.796	0.726	0.126	0.125
The Empire Strikes Back	17	0.231	0.218	0.042	0.645	0.841	0.170	0.104
Return of the Jedi	11	0.325	0.288	0.049	0.769	0.735	0.119	0.196
The Phantom Menace	17	0.220	0.236	0.054	0.701	0.821	0.162	0.095
Attack of the Clones	13	0.173	0.148	0.041	0.802	0.902	0.118	0.059
Revenge of the Sith	15	0.163	0.214	0.040	0.661	0.882	0.132	0.086
The Force Awakens	23	0.223	0.193	0.039	0.895	0.870	0.155	0.088
The Last Jedi	20	0.185	0.208	0.041	0.810	0.825	0.160	0.081
The Rise of Skywalker	19	0.169	0.193	0.039	0.814	0.900	0.151	0.072

Similarly, we can do the same operation but by `trilogy`.

```
song_data %>%
  group_by(trilogy) %>%
  summarise(count = n(),
            across(all_of(categories), ~ mean(.x)),
            .groups = "keep") %>%
  kable(format = "markdown", digits = 3, row.names = FALSE)
```

trilogy	count	danceability	energy	speechiness	acousticness	instrumentalness	liveness	valence
Original	44	0.268	0.228	0.045	0.731	0.773	0.141	0.135
Prequel	45	0.187	0.203	0.045	0.717	0.864	0.139	0.082
Sequel	62	0.194	0.198	0.040	0.843	0.865	0.156	0.081

Being the *Star Wars* fan I am, I know that there are some songs in the soundtracks that sound a bit more upbeat and different from the rest of the collection. I am able to find these songs based on their `valence` audio feature.

```
song_data[(song_data$valence > 0.5), c("track_name", "movie_title", "trilogy", "valence")] %>%
  kable(format = "markdown", digits = 3, row.names = FALSE)
```

track_name	movie_title	trilogy	valence
Cantina Band	A New Hope	Original	0.822
Lapti Nek (Jabba's Palace Band)	Return of the Jedi	Original	0.730

The iconic Cantina Band (https://www.youtube.com/watch?v=EsvfptdFXf4) and the strange Lapti Nek (https://www.youtube.com/watch?v=wINM8zsdw9s) are certainly outliers when it comes to the typical *Star Wars* track (for the *Star Wars* music conosseuiers, note that Cantina Band #2 (https://www.youtube.com/watch?v=9FVADOLRf54) and Jedi Rocks (https://www.youtube.com/watch?v=2gnHu1M7jxs) are unfortunately not included in their respective movie soundtracks). Needless to say, if you are looking to book a band that will play more upbeat music from the *Star Wars* universe, look no further than Figrin D'an and the Modal Nodes or the Max Rebo Band.

Interactive Application

The tables above are good if you can read and picture data really well. However, being able to display the data effectively is very important in order to visualize and understand the data better. The problem with this data is that there are 150+ tracks that should be displayed, so making the graph interactive is desirable.

In order to appropriately graph the data, the data frame should be in a tidy (long and skinny) format.

```
plot_data <- song_data %>%
  pivot_longer(cols = all_of(categories), "name", "value") %>%
  select("movie_title", "trilogy", "track_name", "name", "value")
```

Some tracks have a really long title, such as *Across the Stars (Love Theme from "Star Wars, Episode II")*. This can be problematic when the legend takes up more space than the graph itself. To resolve this issue, we can truncate the longer titles by using an ellipsis.

```
char_lim <- 35
plot_data$track_name <- plot_data$track_name %>%
  {ifelse(nchar(.) > char_lim, paste0(trimws(substr(., 1, char_lim-5)),"..."), .)}
```

Finally, we need to convert all `character` columns to `factor` columns to preserve order. For example, this means that *A New Hope* will be followed by *The Empire Strikes Back*, not *Attack of the Clones*.

```
plot_data[names(plot_data) != "value"] <- plot_data[names(plot_data) != "value"] %>%
  {lapply(., function(x) {factor(x, levels=unique(x))})}
```

Now the data is ready to be plotted. This graph displays every track from *Star Wars* composed by John Williams which can be filtered by trilogy, movie, and even the tracks themselves. The graph and table respond to the filters applied, so the reader has many ways to digest the data.

Select Trilogy

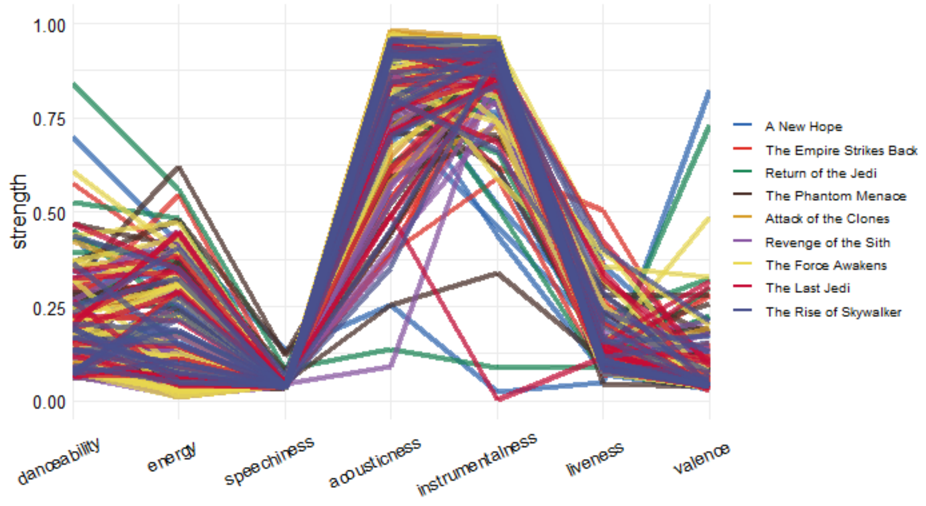
select

Select Movie

select

Select Track

select



track_name	movie_title	trilogy	danceability	energy	speechiness	acousticness	instrumentalness	liveness	valence
Main Title	A New Hope	Original	0.24	0.42	0.04	0.68	0.91	0.13	0.17
Imperial Attack	A New Hope	Original	0.17	0.18	0.04	0.89	0.89	0.08	0.04
Princess Leia's Theme	A New Hope	Original	0.08	0.06	0.04	0.86	0.95	0.07	0.03
The Desert and the Robot Auction	A New Hope	Original	0.27	0.03	0.04	0.82	0.87	0.08	0.05
Ben's Death and TIE Fighter Attack	A New Hope	Original	0.35	0.32	0.04	0.76	0.93	0.09	0.10
The Little People Work	A New Hope	Original	0.45	0.05	0.03	0.90	0.52	0.22	0.06

Final Notes

I hope you learned something new from this blog—whether it's a new technique in programming with R, something new about the music of *Star Wars*, or even just the basics of using the Spotify API. If you found this interesting or even so far as enjoyed it, feel free to check out more of my work at my website (http://borovinsky.com/).