# Spotify Music
# Playlist generation

MLDM - M1 2020-2021

Pablo Javier Sánchez Díaz

## 1 Problem understanding

I have had a Spotify account for over four years now and over this time I have collected 2526 songs in my library, with a range of 817 different genres (artists can belong to more than one music genre). Also interesting is that the genre with more songs (rock) has as much as 20% of the library, but it's not the only one. As a result, listening to the library as is usually results in a mix of rhythms, genres and styles that are not very coherent between each other. Spotify does provide daily mixes that are aimed to create coherent mixes, but this are limited to only 6 per day and more importantly, we are not provided with any way to control what type of playlists get done.

The objective of this project is to generate good playlists from my music library, grouping songs that are both similar in music genre and the mood that they set. All of this in an automated way, without having to manually go through the entire library. This is to be done from the information that spotify provides through their web api.

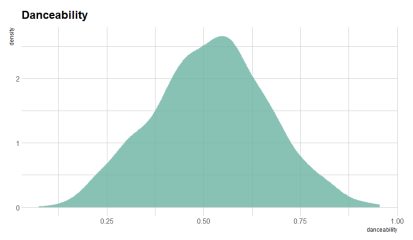
## 2 Data understanding

Spotify provides an API from which one can interact with their services and this includes getting access to their musics database as well as their users (provided they have given permission) libraries[5]. The intended purpose of this API is to build services on top of Spotify, but we can use it to obtain the dataset to be analyzed. Using the API we get the list of songs to analyze, their artists and the genre they are label as, and lastly audio features for each of the songs to be working with. These audio features are:

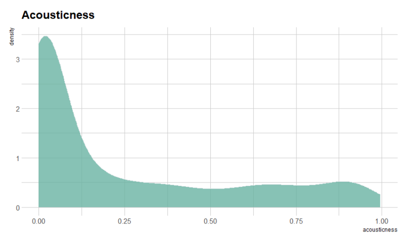
- **Danceability:** Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
- **Acousticness:** A measure from 0.0 to 1.0 of whether the track is acoustic.
- **Energy:** Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.
- **Instrumentalness:** Predicts whether a track contains no vocals. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content.
- **Speechiness:** Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value.

- **Valence:** A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
- **Tempo:** The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.

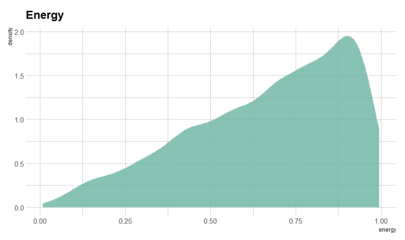
The descriptions have been copied verbatim from their website. There are more features, but these are the one selected for this project.



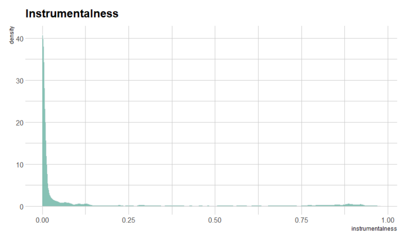
(a) Danceability distribution



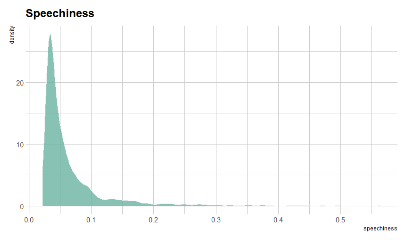
(b) Acousticness distribution



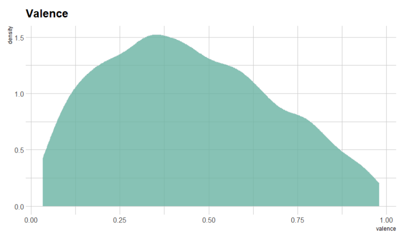
(c) Energy distribution



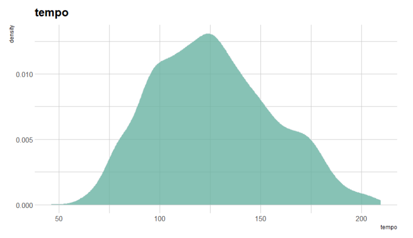
(d) Instrumentalness distribution



(e) Speechiness distribution



(f) Valence distribution



(g) Tempo distribution

Figure 1: Audio features distribution in music library

There are 3 features that behave similar to a normal distribution (danceability, valence and tempo). Accousticness, instrumentalness and seechiness have a peak on the lower side of the metric. Lastly, the energy metric has a peak on the higher side of the metric, with a more stable decrease to the other extreme of the metric.

The top music genre is rock, followed with multiple subgenres of rock where pop starts to show. It's very intesting that a fifth of the songs have the rock label. Is important to mention that the percentages in table 1 do not add to 100, since artists usually have assigned more than one genre.

Because an artist can have more than one genre assigned, if two genres are assigned to the same artist, this can be seen as a connection between the two genres. Repeating this process with all of the listed genres, we end with a graph where each node is a genre and the edges when the two genres appear on the same artist. Furthermore, we can weight the nodes by the amount of songs from that genre and the

| Genre | Songs | % labeled |
|---|---|---|
| rock | 557 | 22% |
| modern rock | 487 | 19% |
| indie rock | 328 | 13% |
| indie pop | 318 | 13% |
| modern alternative rock | 189 | 7% |
| metropopolis | 180 | 7% |
| shimmer pop | 171 | 7% |
| pop | 168 | 7% |
| permanent wave | 166 | 7% |
| stomp and holler | 159 | 6% |
| classic rock | 157 | 6% |
| electropop | 153 | 6% |
| indie poptimism | 147 | 6% |
| latin alternative | 141 | 6% |
| pop rock | 140 | 6% |

Table 1: Top 15 genres

edges by the amount of connections found. Sadly, since there are 817 different genres, the visualization of the resulting graph results complicated.

## 3 Data preparation

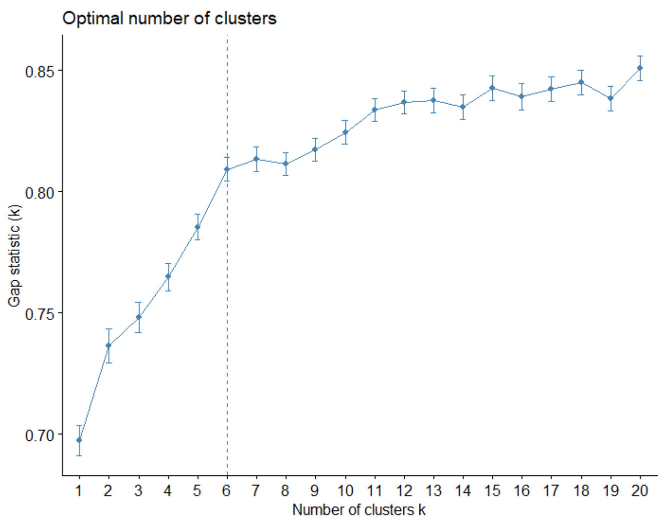
The preparation for the data had the following steps:

1. Obtain the list of songs in the music library from the Spotify API.
2. Obtain the music genres of the artists from the Spotify API.
3. Obtain the audio features of the songs in the music library from the Spotify API.
4. Combine the 3 previous dataset into one with the audio features and the music genres for each song.
5. Filter the songs according the desired music genre.
6. Perform a PCA analysis for the remaining songs to simply the creation of clusters.

The first three steps where performed with a python script in a jupyter notebook. The data was extracted with python because there exists a third-party library in it that allows for an easier connection with the Spotify API [2]. The obtained data was then placed in a local mongo database (selected because of how natural it handles json like data). From the mongo documents the data was transformed into a tabular structure that was unified in the forth step. The rest of the steps where performed in R, where the actual modeling was done. After following the previous steps the data is ready to be splitted into clusters where each correspond to a different playlist.

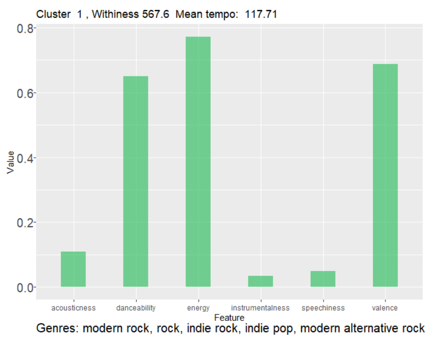
## 4 Modeling

To generate the playlists two important steps are performed to the data. The first one is filtering the music library based on the genre. For this step a base genre needs to be selected. This is to make the desired playlists consistent in terms of music style. To make the playlist more varied, the data is not filtered only by the music genre, but also the music genres that represent at least 80% of the songs in the music genres that are connected the first music genre. This is to not include loosely connected genres and take into account more than one genre to have variety.
The second steps consists in creating clusters of remaining songs. Initially a gap statistic analysis is performer on the data, from which the optimal number of clusters is obtained[3]. Example of this can be seen on figure 2 Then, a k-means clustering model with the desired amount of clusters is run, which results in that amount of generated playlists[4].

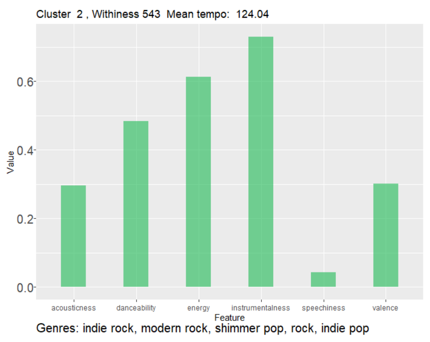


[h]

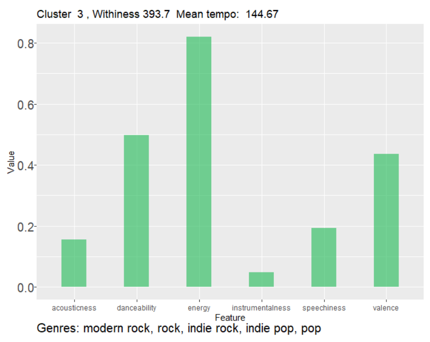
Figure 2: Gap statistic analysis for genre metropopolis



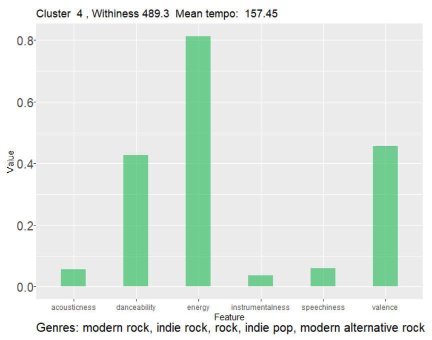
(a) Features of cluster 1



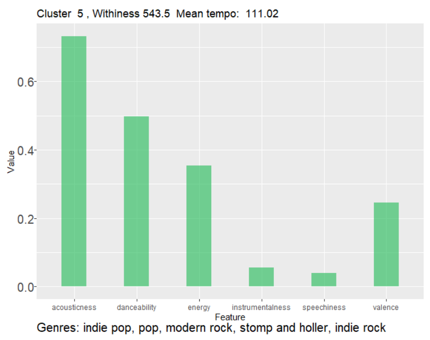
(b) Features of cluster 2



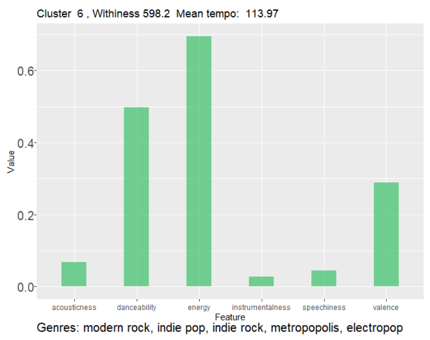
(c) Features of cluster 3



(d) Features of cluster 4



(e) Features of cluster 5



(f) Features of cluster 6

Figure 3: Features of clusters generated from genre metropolis

Analysing the features of the resulting clusters in figure 3 we can see that the values are considerably different between each cluster having dramatic differences between the acousticness in cluster 5 vs the rest, or the valence in cluster 1 and the rest.

## 5 Other attempted approaches

Besides the final approach described above, other approaches were attempted without much success. Two of this approaches are mentioned here

### 5.1 Clustering without filtering music genres

The first attempt to create clusters was made with k-means but without filtering the songs by their music genre. The main difference here is taking into account all of the songs, and as result the suggested amount of clusters resulted in a higher number, 14 clusters to be precise.
When analysing the distinct generated playlists, while the features where different between each other, when closely inspecting each playlist, songs that were not related where categorized in the same cluster. For example finnish power metal in the same playlist as a brass band or slow electronic music with folk music. While some of the features for these songs where similar, they did not belong on the same playlist because of how different the music genres are [1].

### 5.2 Community finding in music genres

An attempt to separate the music genres into different groups was made. This based on community detection with the Newman-Girvan algorithm, which attempts to split a graph into sub-graphs that are densely connected with itself and at the same time have few connections with the rest of the nodes. This approach resulted in 114 different clusters of genres. While there was one community that seem to make a lot of sense with 50 different metal genres, at the same time there was one community with 161 genres which corresponds to about a fifth of the total genres. Most of the generated communities where of 5 or less genres. It's worth mentioning that this clustering was considerably slower than the clustering, taking upwards of five minutes to run, compared to less than three seconds for the clustering.

## 6 Evaluation

### 6.1 Quality of the source data

During the exploration of the data certain details may appear that give light of the quality of the source data. For example, in the top 10 songs with the highest instrumentalness (no vocal), all the songs have a value above 0.95, yet the song coffe by beabadoobee contains vocals through all of the song even if with an altered tone.
On a different note, the existing music genres may be too specific in some cases, represent different aspects of the artists or maybe poorly labeled, maybe making it necessary to clean the labels first, which can be time consuming. And example of an specific genre is 'beatlesque' for the Beatles and a couple of other bands. An example of a poorly labeled one is 'rock nacional', national rock, which comes with the question: which nation? while it was found mostly in argentinian bands, an uruguayan band also had it. An example of a genre that represents something else that the music style is 'singer-songwriter' which describes the author more than its music style. Labeling artists requires a lot of work and it is expected that the data is not perfect, but more than that I would guess that the objective of labeling the artists with a music genre is not for playlist generation but maybe just for visualization in some way.

### 6.2 Quality of generated playlists

While not perfect, the generated playlists seem to have thematic consistency in terms of music genre, have enough variety to not just have a single artist, but also without mixing to many strange artists. It also succeeds to maintain similarity in the songs in terms of the feeling. However, it still mixes some songs do not belong together, like is the case for mellow indie music in english and some of Shakira's early work in spanish.

## 7 Deployment

While no final product was designed for this project, the ideas and even some part of the code could be transformed in a web application that could automatically generate a playlist for a given user from its music library, a selection of a music genre and lastly selecting which of the resulting clusters is desired as a playlist. In the scope of this project, a jupyter notebook that reads the csv output of the clustering creates a new playlist and populates it with the given songs.

## 8 Resources used in this project

The source code can be accessed in the public repository
`https://github.com/pablo8421/spotify-playlist-generation`
The files and their contents are described here and in the readme of the repository.

- **clean_dataset.r:** Initial visualization of the data and cleaning it to just the used data. Output is tracks_simplified.
- **clustering_songs.r:** The attempt to create clusters of songs without taking into account their music genre. Output is tracks_clusterized.
- **clustering_songs_genre.r:** The final model used, where the clusters take into account the music genre (example with metropopolis). Output is tracks_clusterized_metropopolis.
- **genres_graph.r:** The attempt to create clusters based only on the music genres. The outputs are the files inside the communities folder.
- **Datasetgeneration.ipynb:** The generation of the dataset from the spotify web api. Requires a key/secret pair.
- **Playlist-creation.ipynb:** The creation a filling of the generated playlist. Requires a key/secret pair.
- **tracks_info.csv:** Source info of all the tracks.
- **tracks_simplified.csv:** Simplified version of the tracks.
- **tracks_clusterized.csv:** Clusterized tracks without considering music genre.
- **tracks_clusterized_metropopolis.csv:** Clusters from the metropopolis music genre.

## References

[1] IGRAPH TEAM. igraph manual pages: Cluster$_{e}dge_{b}etweeness$.

[2] LAMERE, P. plamere/spotipy.

[3] MUNDT, F. Dertermining and visualizing the optimal number of clusters.

[4] MUNDT, F. hkmeans: Hierarchical k-means clustering.

[5] SPOTIFY TEAM. Webapi reference — spotify for developers.