

# Trabajo de investigación: Spotify

---

Por Abraham Corta Ramírez y José Calcedo Vázquez

# Índice de la presentación

- Introducción
- Procedimientos para estudiar Spotify
- Extracción de datos
- Generación de la red
- Medidas de centralidad, comunidades y otros resultados
- Conclusiones
- ¿Dudas?

# Introducción

Spotify, quizás la aplicación de música más famosa de todos los tiempos. Dentro de ella se pueden hallar centenares de listas de reproducción e innumerables artistas y canciones, pero...

¿Cómo se relacionan todos ellos?



# Procedimientos para estudiar Spotify

- Spotify tiene una cantidad inabarcable de canciones a estudiar, por no hablar del número de artistas. ¿Por dónde empezar?
- Para estudiarla, tomaremos un **artista como punto de partida** y estudiaremos las **listas de reproducción asociadas**.
- Crearemos un grafo donde los **vértices** serán los **artistas** de las distintas listas de reproducción y las **aristas** representan si tienen **alguna lista de reproducción en común**.
- A partir de ahí, podremos estudiar todo lo que queramos.
- Nos interesa que nuestro candidato sea relativamente famoso pero sin una cantidad abrumadora de listas de reproducción asociadas a él.

# 50 Cent

Él será nuestro punto de partida,  
pues cumple nuestros requisitos  
con solo 4 listas de reproducción  
siendo moderadamente famoso



# Extracción de datos

- Spotify nos ofrece una API REST desde la cual extraer los datos de cualquier artista, canción o lista de reproducción que queramos
- Primeramente tendremos que crearnos una **cuenta de desarrollador Spotify**. Esto nos dará unas credenciales llamadas **Client ID** y un **Client Secret**. Las usaremos más adelante
- El siguiente paso es crear un **script** en Python (puede ser en otro lenguaje de programación si no os gusta) que nos permita recolectar datos de la API y generar **archivos .json** con los que trabajar en a SAGE.
- Usaremos estas librerías:
  - **pandas**,
  - **spotipy**
  - **spotipy.oauth2-SpotifyClientCredentials**

# ¿Qué se puede extraer de la API?


De los artistas:

- Nombre
- N° de seguidores
- Popularidad
- Mejores canciones
- Artistas relacionados
- Playlists

De las canciones:

- Nombre
- Autor/Autores
- Duración
- Favoritos
- Géneros musicales

Esta información nos servirá para generar la red y para otros resultados



```
from spotipy.oauth2 import SpotifyClientCredentials
import pandas as pd
import spotipy

# Conexion con la API
sp = spotipy.Spotify()
cid = "*****"
secret = "*****"
client_credentials_manager = SpotifyClientCredentials(
    client_id=cid, client_secret=secret)
sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)
sp.trace = False
```

Importamos librerías y accedemos con nuestra cuenta





```
# Usuario 50 cent  
# https://open.spotify.com/user/1217753577  
  
# Playlists de 50 cent  
# The Singles Collection: https://open.spotify.com/playlist/1hBQZ8nCtIHrEByxICx8tU  
# Featuring 50cent: https://open.spotify.com/playlist/3pDliBuh3MwiSfj0Ko5mKl  
# 50 cent best of: https://open.spotify.com/playlist/49RoQF55lyRSgSZwRAHh5K  
# 50 my playlist: https://open.spotify.com/playlist/1s7ipxB42mCucqQUWVMP4
```

Identificamos a 50 Cent y anotamos URLs de sus playlists

```

#lista donde tenemos todas las playlists anteriores
pp = ["3pDliBuh3MwiSfj0Ko5mKl", "1hBQZ8nCtIHrEByxICx8tU",
      "49RoQF55lyRSgSZwRAHh5K", "1s7ipxTB42mCucqQUWVMP4"]

for i in pp:
    # obtenemos una playlist
    playlist = sp.user_playlist("1217753577", i, fields="tracks,next")
    tracks = playlist["tracks"]
    songs = tracks["items"]

    ids = []
    song = []
    artist = []

    # obtenemos las canciones
    for k in range(len(songs)):
        s = songs[k]["track"]
        ids.append(s["id"])
        artists = []

        for j in range(len(s["artists"])):
            # dataset de playlists
            artists.append(s["artists"][j]["name"])
            song.append([s["name"], s["popularity"], artists])

            # dataset de artistas
            a = sp.artist(s["artists"][j]["id"])
            artist.append(
                [a["name"], a["genres"], a["popularity"], a["followers"]["total"]]
            )

```

Obtenemos todas las canciones y extraemos la información pertinente

```
# Exportamos a Json los datos recabados
dataArtists = pd.DataFrame(artist)
dataSongs = pd.DataFrame(song)
outputArtists = "api/50cent-dataArtists"+str(pp.index(i)+1)+".json"
output = "api/50cent-dataPlaylist"+str(pp.index(i)+1)+".json"
dataSongs.to_json(output, orient='records')
dataArtists.to_json(outputArtists, orient='records')
```

Generamos los archivos .json para usarlos en SAGE.

# Generando la red (1): Obtención de los datos

- Una vez que tenemos los archivos .json es cuestión de transformar los datos que contienen en una lista.
- Usando `os.path.join` almacenamos el contenido de un archivo json en una variable.
- Con `json.loads` leemos los datos de la variable anterior y conseguimos la lista que deseamos

Haciendo esto procesamos los datos de artistas y de las canciones halladas en las listas de reproducción de 50 Cent.

```
jArt1 = os.path.join(USER_DIR, '50cent-dataArtists1.json')
jArt2 = os.path.join(USER_DIR, '50cent-dataArtists2.json')
jArt3 = os.path.join(USER_DIR, '50cent-dataArtists3.json')
jArt4 = os.path.join(USER_DIR, '50cent-dataArtists4.json')
dataArt1 = json.loads(open(jArt1).read())
dataArt2 = json.loads(open(jArt2).read())
dataArt3 = json.loads(open(jArt3).read())
dataArt4 = json.loads(open(jArt4).read())
dataArtists = [dataArt1, dataArt2, dataArt3, dataArt4]
```

# Generando la red(2): Creación del grafo

Nuestro algoritmo para generar el grafo es el siguiente (previamente le asignamos un índice a cada lista):

Para cada lista de reproducción:

- Obtenemos todos los artistas involucrados, tanto si colaboran en una canción como si no
- Añadimos los vértices en el grafo (en nuestro caso los artistas)
- Añadimos las aristas, que tendrán un peso igual al índice de la lista. Una arista se forma si dos artistas comparten una lista de reproducción.

Esto generará un subgrafo para cada lista de reproducción, que se unirán entre sí ya que compartirán vértices. Este algoritmo resulta en el código en Sage que viene a continuación.

```

G=Graph()
b = len(dataPlayLists)
for i in range(0,b):
    artistas = [] # todos los artistas de todas las listas
    a = dataPlayLists[i]
    z = len(a)
    for j in range(0, z):
        c = a[j]
        artistas.append(c['2'])

l = [val for sublist in artistas for val in sublist] #añade los artistas

l = list(dict.fromkeys(l))
G=anadeVertices(G,l)
G=anadeArista(G,l,i)

d={0: "red", 1: "purple", 2:'orange', 3:'green'}

G.plot(edge_colors=G._color_by_label(d), edge_style='solid').show(figsize=25 ,fontsize=20)    #artistas que aparecen en
una playlist

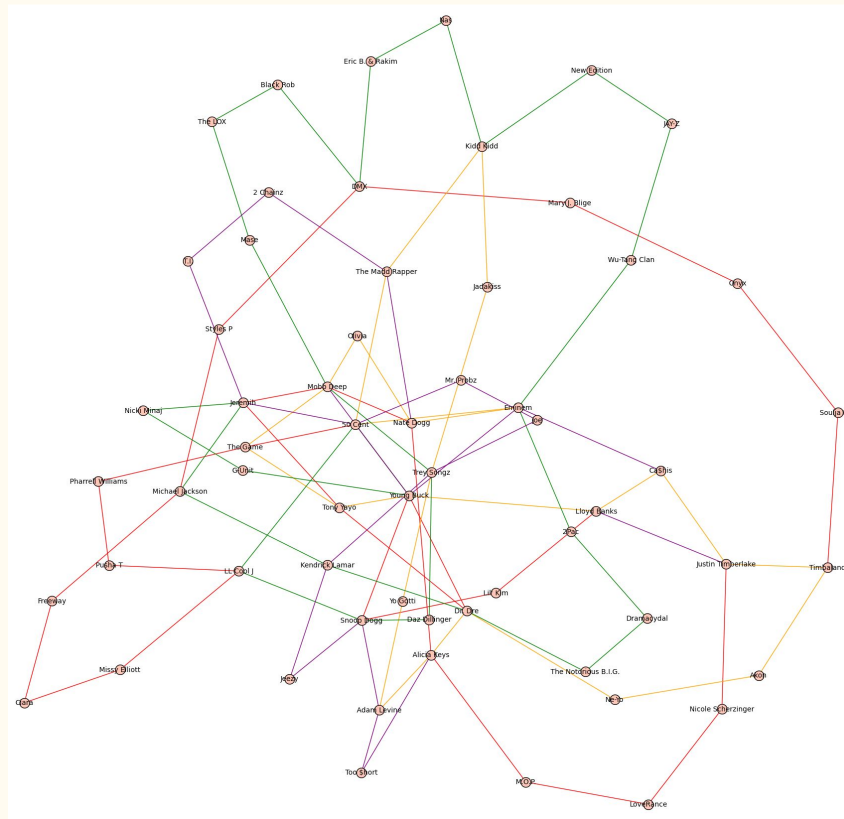
```

Código principal

```
def anadeVertices(G, a): #añade vertices al grafo en funcion de la lista de reproduccion
    G.add_vertices(a)
    return G
def anadeArista(G, a, z): #añade aristas al grafo en funcion de los artistas
    for i in range(0,len(a)):
        if(i==len(a)-1):
            G.add_edge(a[-1],a[0],z)
        else:
            G.add_edge(a[i], a[i+1], z)
    return G
```

Funciones auxiliares





Resultado gráfico con SAGE . Bonito, ¿verdad? Para cada índice le hemos asignado un color distinto.



# Centralidad y otras medidas: popularidad (1)

Primero hemos obtenido de cada lista de reproducción el artista más famoso, resultando que Eminem es el más famoso entre todas las listas.

Un resultado interesante, ya que él no era el artista central del estudio.



# ¿Cómo obtuvimos la popularidad?

```
for i in range(0, len(dataArtists)):  
  
    a = dataArtists[i]  
    popular=dict()  
    popularidadTotal=list()  
    for j in range(0, len(dataArtists[i])):  
        c=a[j]  
        popular.update({c['0']:c['2']})  
  
    p={k: v for k, v in sorted(popular.items(), key=lambda item: item[1])} #sort dict  
  
    print(p, "\n") #Popularidades de esta playliyst  
  
    for key, value in p.items():  
        s=key+": "+str(value)  
        popularidadTotal.append(s)  
  
    print('Los mas populares son:')  
    print(popularidadTotal[20:len(popularidadTotal)])  
    print('-----')
```

```
{'LoveRance': 41, 'Tony Yayo': 52, 'Freeway': 52, 'Ca$his': 54, 'Onyx': 55, 'M.O.P.': 57, 'Young Buck': 63, 'Nicole S  
cherzinger': 63, 'Lloyd Banks': 64, 'Styles P': 64, 'LL Cool J': 65, 'Lil' Kim': 68, 'Mobb Deep': 69, 'Soulja Boy': 6  
9, 'DMX': 72, 'Pusha T': 72, 'Ciara': 73, 'Mary J. Blige': 74, 'Missy Elliott': 74, 'The Game': 75, 'Nate Dogg': 78,  
'Timbaland': 78, 'Dr. Dre': 80, 'Jeremih': 80, 'Pharrell Williams': 80, '2Pac': 81, 'Alicia Keys': 82, 'Justin Timber  
lake': 82, '50 Cent': 84, 'Michael Jackson': 84, 'Snoop Dogg': 85, 'Eminem': 94}
```

Los mas populares son:

```
['Nate Dogg:78', 'Timbaland:78', 'Dr. Dre:80', 'Jeremih:80', 'Pharrell Williams:80', '2Pac:81', 'Alicia Keys:82', 'Ju  
stin Timberlake:82', '50 Cent:84', 'Michael Jackson:84', 'Snoop Dogg:85', 'Eminem:94']
```

```
-----  
{'Joe': 31, 'The Madd Rapper': 32, 'Ca$his': 54, 'Young Buck': 63, 'Lloyd Banks': 64, 'Olivia': 65, 'Too $hort': 67,  
'Mr. Probz': 68, 'Mobb Deep': 69, 'Adam Levine': 73, 'Jeezy': 75, 'Trey Songz': 77, 'T.I.': 78, 'Nate Dogg': 78, 'Tim  
baland': 78, 'Jeremih': 80, 'Ne-Yo': 80, 'Dr. Dre': 80, 'Akon': 81, 'Justin Timberlake': 82, 'Alicia Keys': 82, '2 Ch  
ainz': 83, '50 Cent': 84, 'Snoop Dogg': 85, 'Kendrick Lamar': 87, 'Eminem': 94}
```

Los mas populares son:

```
['Alicia Keys:82', '2 Chainz:83', '50 Cent:84', 'Snoop Dogg:85', 'Kendrick Lamar:87', 'Eminem:94']
```

```
-----  
{'The Madd Rapper': 32, 'Tony Yayo': 52, 'Ca$his': 54, 'Kidd Kidd': 55, 'Young Buck': 63, 'Lloyd Banks': 64, 'Olivia  
' : 65, 'Jadakiss': 67, 'Mobb Deep': 69, 'Adam Levine': 73, 'The Game': 75, 'Yo Gotti': 75, 'Trey Songz': 77, 'Nate Do  
gg': 78, 'Timbaland': 78, 'Ne-Yo': 80, 'Dr. Dre': 80, 'Akon': 81, 'Justin Timberlake': 82, 'Alicia Keys': 82, '50 Cen  
t': 84, 'Eminem': 94}
```

Los mas populares son:

```
['50 Cent:84', 'Eminem:94']
```

```
-----  
{'Dramacydal': 49, 'Black Rob': 54, 'Eric B. & Rakim': 55, 'Kidd Kidd': 55, 'The LOX': 57, 'New Edition': 59, 'Daz Di  
llinger': 62, 'G-Unit': 63, 'Young Buck': 63, 'LL Cool J': 65, 'Mase': 67, 'Mobb Deep': 69, 'Wu-Tang Clan': 69, 'DMX  
' : 72, 'Nas': 75, 'Trey Songz': 77, 'The Notorious B.I.G.': 79, 'Dr. Dre': 80, 'Jeremih': 80, '2Pac': 81, '50 Cent':  
84, 'Michael Jackson': 84, 'Snoop Dogg': 85, 'JAY-Z': 85, 'Kendrick Lamar': 87, 'Nicki Minaj': 89, 'Eminem': 94}
```

Los mas populares son:

```
['50 Cent:84', 'Michael Jackson:84', 'Snoop Dogg:85', 'JAY-Z:85', 'Kendrick Lamar:87', 'Nicki Minaj:89', 'Eminem:94']
```

-----

Eminem aparece como el más popular al final de cada lista

## Centralidad y otras medidas: género más popular (2)

Siguiendo un procedimiento análogo, obtuvimos como era de esperar que los géneros más populares de cada lista eran el hip-hop y el rap.

Así mismo pudimos encontrar géneros que anteriormente nunca habíamos oído como el nashville hip hop o el crunck



## ¿Cómo obtuvimos el género más popular?

```
for i in range(0, len(dataArtists)):
    generos=[]
    a = dataArtists[i]
    genres=dict()
    for j in range(0, len(dataArtists[i])):
        c=a[j]
        generos.append(c['1'])

g = [val for sublist in generos for val in sublist] #añade los generos de esta playlist

#cuenta generos que aparecen
for v in range(0, len(g)):
    if g[v] in genres:
        genres[g[v]]+=1
    else:
        genres[g[v]]=1

genres={k: v for k, v in sorted(genres.items(), key=lambda item: item[1])} #sort dict
print(genres, "\n")
print("El genero mas popular es", list(genres)[-1])
print('-----')
```

```
{'trap queen': 1, 'chicago rap': 1, 'electropop': 1, 'europop': 1, 'atl hip hop': 1, 'soul': 1, 'alternative hip hop': 1, 'philly rap': 1, 'new jack swing': 1, 'old school hip hop': 1, 'underground hip hop': 1, 'crunk': 2, 'nashville hip hop': 2, 'neo soul': 2, 'pop dance': 2, 'post-teen pop': 2, 'battle rap': 2, 'virginia hip hop': 2, 'dirty south rap': 3, 'detroit hip hop': 6, 'g funk': 6, 'west coast rap': 6, 'hip pop': 6, 'r&b': 7, 'urban contemporary': 8, 'pop': 8, 'dance pop': 9, 'hardcore hip hop': 10, 'trap': 12, 'southern hip hop': 13, 'east coast hip hop': 40, 'queens hip hop': 41, 'pop rap': 54, 'gangster rap': 55, 'rap': 64, 'hip hop': 67}
```

El genero mas popular es hip hop

-----

```
{'nashville hip hop': 1, 'hip pop': 1, 'neo soul': 1, 'cali rap': 1, 'hyphy': 1, 'oakland hip hop': 1, 'pop rock': 1, 'conscious hip hop': 1, 'edm': 1, 'tropical house': 1, 'chicago rap': 2, 'crunk': 2, 'pop dance': 2, 'atl hip hop': 3, 'dirty south rap': 3, 'hardcore hip hop': 3, 'detroit hip hop': 3, 'r&b': 5, 'g funk': 5, 'urban contemporary': 6, 'pop': 7, 'southern hip hop': 7, 'trap': 7, 'west coast rap': 7, 'dance pop': 9, 'east coast hip hop': 44, 'queens hip hop': 45, 'gangster rap': 55, 'pop rap': 55, 'rap': 59, 'hip hop': 62}
```

El genero mas popular es hip hop

-----

```
{'pop dance': 1, 'neo soul': 1, 'pop rock': 1, 'memphis hip hop': 1, 'tennessee hip hop': 1, 'battle rap': 1, 'new orleans rap': 1, 'g funk': 2, 'west coast rap': 2, 'crunk': 2, 'nashville hip hop': 2, 'hip pop': 2, 'dirty south rap': 3, 'r&b': 3, 'hardcore hip hop': 4, 'southern hip hop': 4, 'urban contemporary': 4, 'detroit hip hop': 5, 'dance pop': 5, 'pop': 5, 'trap': 7, 'east coast hip hop': 37, 'queens hip hop': 39, 'gangster rap': 45, 'pop rap': 45, 'rap': 50, 'hip hop': 52}
```

El genero mas popular es hip hop

-----

```
{'crunk': 1, 'dirty south rap': 1, 'nashville hip hop': 1, 'west coast trap': 1, 'alternative hip hop': 1, 'bboy': 1, 'electro': 1, 'turntablism': 1, 'new orleans rap': 1, 'boy band': 1, 'funk': 1, 'quiet storm': 1, 'detroit hip hop': 1, 'soul': 1, 'chicago rap': 1, 'pop dance': 1, 'post-teen pop': 1, 'new jack swing': 2, 'old school hip hop': 2, 'harem hip hop': 2, 'conscious hip hop': 2, 'trap': 5, 'g funk': 5, 'dance pop': 5, 'pop': 5, 'urban contemporary': 6, 'west coast rap': 6, 'r&b': 6, 'hip pop': 7, 'southern hip hop': 8, 'queens hip hop': 9, 'hardcore hip hop': 12, 'east coast hip hop': 14, 'pop rap': 16, 'gangster rap': 20, 'rap': 24, 'hip hop': 26}
```

El genero mas popular es hip hop

-----

El Hip Hop y el Rap son los géneros raíz de los demás géneros.

# Ahora sí, centralidad

Hemos aplicado las medidas de centralidad de forma similar a la de las prácticas, obteniendo lo siguiente:

- **Centralidad de grado:** 50 Cent y Mobb Deep son los artistas con más centralidad, con una centralidad de  $7/60$ . Son los artistas con más conexiones de todos.
- **Centralidad de intermediación:** 50 Cent es el más influyente del grafo, con un 0.1767, seguido otra vez de Mobb Deep, con un 0.1491
- **Centralidad de cercanía:** 50 Cent y Mobb Deep vuelven a empatar con un 0.35928.



# Mobb Deep aparece en todos lados

¿Por qué ocurre esto? ¿Han  
colaborado varias veces? ¿Quizás  
una mano invisible los pone  
juntos muy a menudo?



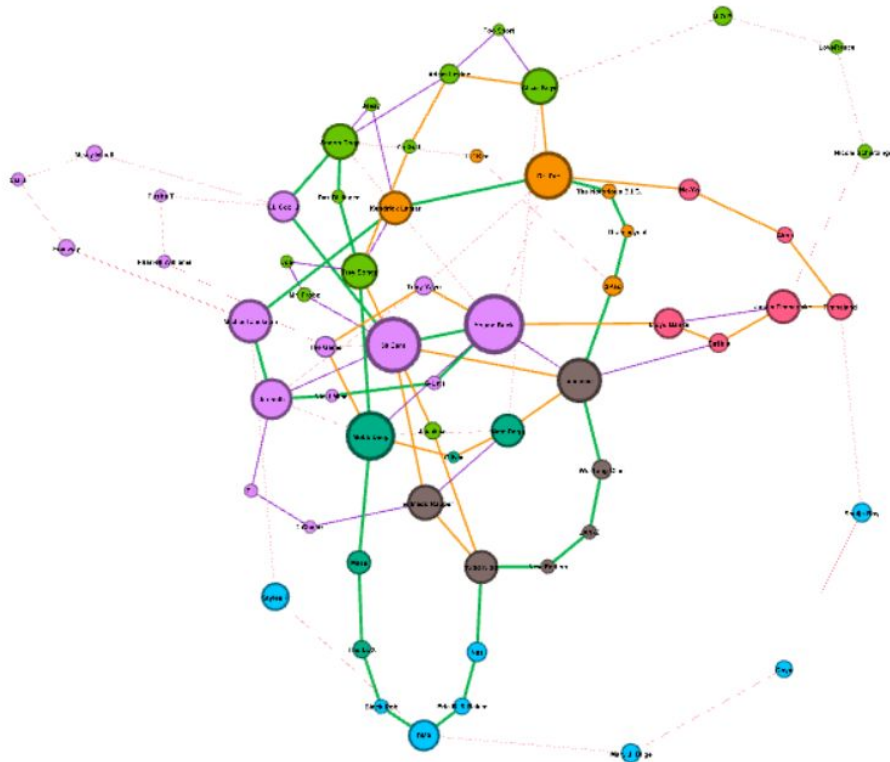
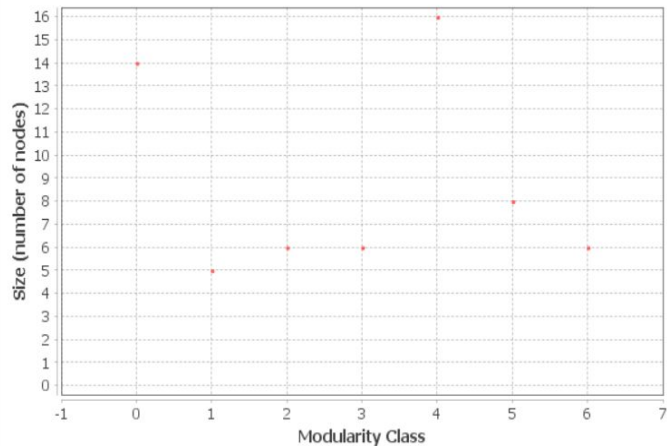


# Modularidad y comunidades con Gephi

## Results:

Modularity: 0,522  
Modularity with resolution: 0,522  
Number of Communities: 7

Size Distribution



# Conclusiones

- Spotify es más difícil de estudiar que otras redes sociales. No es tan trivial como Twitter o Facebook a la hora de generar el grafo; éstas permiten fácilmente montar una red basándose en retweets o amistades entre usuarios, no así con nuestra aplicación de música.
- El artista que se toma como centro de estudio no tiene por qué ser el más popular (véase el caso de Eminem), y salvo alguna que otra excepción podemos suponer que siempre habrá alguien más famoso que un artista cualquiera del grafo.
- Un artista menos conocido puede ser tan influyente como el centro de estudio del grafo, como Mobb Deep.

¿Dudas?