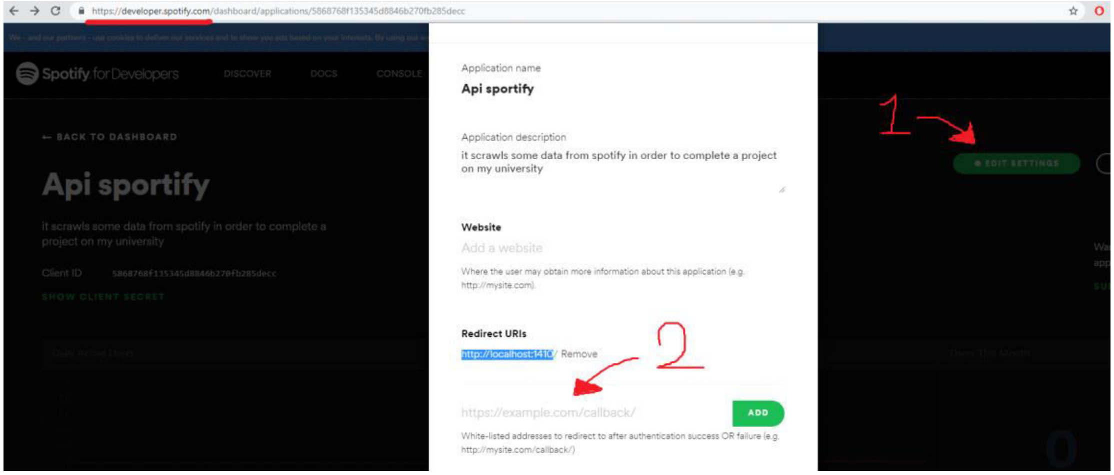
```
# 1) Initialization
```

```
install.packages('spotifyr')
install.packages("tidyverse")
install.packages('knitr', dependencies = TRUE)
install.packages('ggjoy')
install.packages("ggplot2", dependencies = TRUE)
```



https://developer.spotify.com/dashboard/applications/5868768f135345d8846b270fb285decc

Application name
Api sportify

Application description
it scrawls some data from spotify in order to complete a project on my university

Website
Add a website
Where the user may obtain more information about this application (e.g. http://mysite.com).

Redirect URIs
http://localhost:1410/ Remove

https://example.com/callback/ ADD
White-listed addresses to redirect to after authentication success OR failure (e.g. http://mysite.com/callback/)

Api sportify
it scrawls some data from spotify in order to complete a project on my university
Client ID 5868768f135345d8846b270fb285decc

```
library(spotifyr)
library(ggjoy)       # useful for plot
library(ggplot2)     # useful for plot
library(tidyverse)  # makes possible the use of %>%
library(knitr)        # library to appear data results in a better way
library(lubridate)   # useful for date functions
```

```
# 2) Connect
Sys.setenv(SPOTIFY_CLIENT_ID = '5868768f135345d8846b270fb285decc')
Sys.setenv(SPOTIFY_CLIENT_SECRET = 'e6e3c1c764dd41a3a8ba0b16fafc0a7e')
access_token <- get_spotify_access_token()
```

```
# 3) Get your 5 most recently played tracks from your account
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) %>%
   head()
```

```
# 4) Find your 5 all time favorite artists from your account
get_my_top_artists_or_tracks(type = 'artists', limit = 5) %>%
   select(name, genres) %>%
   rowwise %>%
   mutate(genres = paste(genres, collapse = ', ')) %>%
   kable()


# 5) Find your 6 favorite tracks at the moment from your account
get_my_top_artists_or_tracks(type = 'tracks', time_range = 'short_term', limit = 6) %>%
   mutate(artist.name = map_chr(artists, function(x) x$name[1])) %>%
   select(name, artist.name, album.name) %>%
   kable()


# Ask what artist to take data from
artist <- readline(prompt='Search an artist: ')


# 6) Get the data from the artist
artist.audio.features <- get_artist_audio_features(artist)


#Do you want a part of it? For example the audio features from a specific album?
sample <- get_artist_audio_features(artist) %>%
   filter(album_name == 'Brave Enough')


# 7) Plot joy distribution per album
ggplot(artist.audio.features, aes(x = valence, y = album_name)) +
   geom_joy() +
   theme_joy() +
   ggtitle(paste0("Joyplot of ", artist, "'s joy distributions"), subtitle = "Based on valence pulled
from Spotify's Web API with spotifyr")

#or

ggplot(artist.audio.features, aes(x = valence, y = album_name)) +
   geom_density_ridges_gradient(scale = 0.9) +
   scale_fill_gradient(low = "white", high = "maroon3") +
   theme(panel.background = element_rect(fill = "white")) +
   theme(plot.background = element_rect(fill = "white")) +
   xlim(0,1) +
   theme(legend.position = "none")
```

```
# 8) Select tracks from a specific album
artist.audio.features %>%
   select (track_name, valence) %>%
   filter(artist.audio.features$album_name == 'Shatter Me')
```

```
# 9) Select tracks that are longer than 3 min (180.000 milliseconds)
artist.audio.features %>%
   select (track_name, valence) %>%
   filter(artist.audio.features$duration_ms >180000)
```

```
# 10) The 5 most joyful songs of your artist
artist.audio.features%>%
   arrange(-valence) %>%
   select(track_name, valence) %>%
   head(5) %>%
   kable()
```

```
# 11) The 5 most danceable songs of your artist
artist.audio.features%>%
   arrange(-danceability) %>%
   select(track_name, danceability) %>%
   head(5) %>%
   kable()
```

```
# 12) The 5 fastest songs of your artist
artist.audio.features%>%
   arrange(-tempo) %>%
   select(track_name, tempo) %>%
   head(5) %>%
   kable()
```

```
# 13) The 6 loudest songs of your artist
artist.audio.features %>%
   arrange(-loudness) %>%
   select(track_name, loudness) %>%
   head(6) %>%
   kable()
```

```
# 14) The 6 most energetic songs of your artist
artist.audio.features %>%
    arrange(-energy) %>%
    mutate(duration_min = duration_ms /1000) %>%
    select(track_name, energy) %>%
    head(6) %>%
    kable()
```

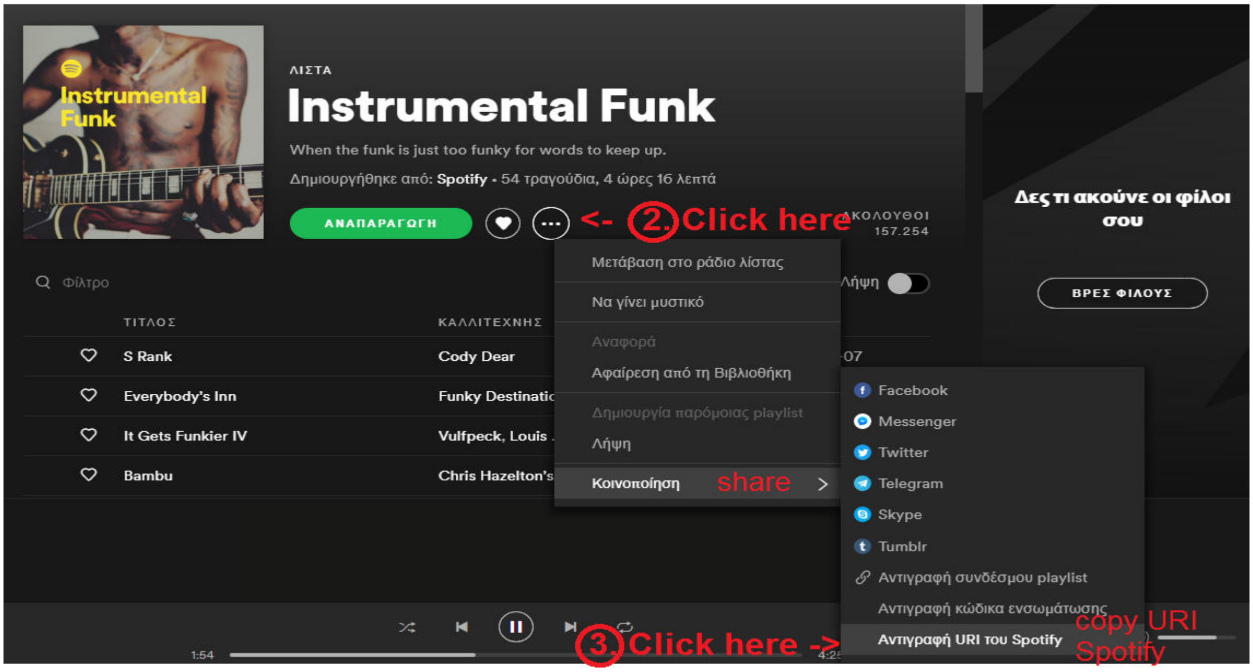
```
# 15) The 6 longest songs of your artist in seconds
artist.audio.features %>%
    mutate(duration_min = duration_ms /1000) %>%
    arrange(-duration_min) %>%
    select(track_name, duration_min) %>%
    head(6) %>%
    kable()
```

```
# 16) What’s the most used key per track of your artist?
most.used.key <- artist.audio.features%>%
    select(track_name, key_name)%>%
    group_by(track_name, key_name)%>%
    mutate(n=n())%>%
    unique()%>%
    group_by(key_name)%>%
    mutate(total=sum(n))%>%
    mutate(percent=round((n/total)*100))
head(most.used.key, 15)
```

```
# 17) Get playlist tracks and audio features from a playlist
```

How to get a playlist id?

1. Go to spotify (desktop app or web)



This is what you copied

spotify:user:spotify:playlist:37i9dQZF1DX8f5qTGj8FYI

This is the playlist id

```
playlist <- get_playlist_tracks('37i9dQZF1DX8f5qTGj8FYI')
```

```
#Do you want a part of it? For example the playlist tracks from 2018?
playlist <- get_playlist_tracks('37i9dQZF1DX8f5qTGj8FYI') %>%
   filter (track.album.release_date > 2018 )
```

```
# Or what about from a specific date?
playlist <- get_playlist_tracks('37i9dQZF1DX8f5qTGj8FYI') %>%
   filter (track.album.release_date > '2018-09')
```

```
# 18) Plot popularity per song of a playlist
ggplot(playlist, aes(track.popularity, reorder(track.name, track.popularity))) +
   labs(x = "Popularity", y = "Songs") +
   geom_point(size = 2, shape = 21)
```

```
# 19) Mean variables per album into a variable
Var1 <- artist.audio.features %>%
    group_by(album_name) %>%
    summarise(year = mean(album_release_year),
        tracks = n(),
        mean_duration = mean(duration_ms/1000),
        mean_tempo = mean(tempo),
        mean_danceability = mean(danceability),
        mean_energy = mean(energy),
        mean_valence = mean(valence),
        mean_instrumentalness = mean(instrumentalness)) %>%
   arrange(year)
Var1
```

```
# 20) Plot mean duration (of songs per album) in sec per album
ggplot(Var1) +
  geom_line(aes(year, mean_duration), size = 1, color = "darkgrey") +
  geom_point(aes(year, mean_duration, fill = reorder(album_name, year)), size = 4, shape = 21)
+
  labs(x = "Year", y = " Duration in sec", fill = "Album") +
    theme(text = element_text(size = 14))
```

```
# 21) Plot mean danceability (of songs per album) in sec per album
ggplot(Var1) +
    geom_line(aes(year, mean_danceability), size = 1, color = "darkgrey") +
    geom_point(aes(year, mean_danceability, fill = reorder(album_name, year)), size = 4, shape
= 21) +
    labs(x = "Year", y = " mean_danceability", fill = "Album") +
    theme(text = element_text(size = 14))
```

```
# 22) Plot mean danceability (of songs per album) per mean valence (of songs per album)
ggplot(Var1) +
    geom_line(aes(mean_valence, mean_danceability), size = 1, color = "darkgrey") +
    geom_point(aes(mean_valence, mean_danceability, fill = reorder(album_name,
mean_valence)), size = 4, shape = 21) +
    labs(x = "mean_valence", y = " mean_danceability", fill = "Album") +
    theme(text = element_text(size = 14))
```

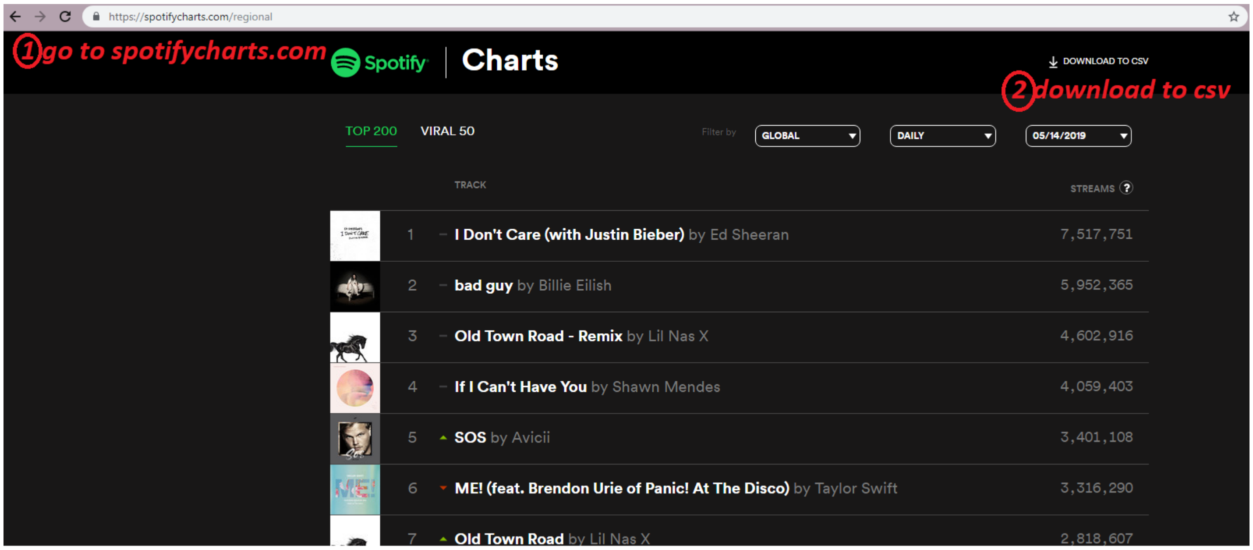
```
# 23) Plot all audio features Values per album (may occur a warning, if more than 6 albums)
Var2 <- Var1 %>%
    select (album_name, year, mean_danceability, mean_energy, mean_valence,
mean_instrumentalness) %>%
    gather(variable, value, -c(album_name, year))

ggplot(Var2) +
    geom_line(aes(year, value, color = variable), size = 1.25) +
    theme(text = element_text(size = 14)) +
    geom_point(aes(year, value, shape = reorder(album_name, year)), size = 2) +
    labs(x = "Year",
        y = "Mean",
        color = "Audio Feature",
        shape = "Album")


# 24) Save the result that appears in console a file
# syntax
capture.output(script , file = "Name_of_file ")

# example:
capture.output(artist.audio.features %>%
                   select (track_name, valence) %>%
                   filter(artist.audio.features$album_name == 'Shatter Me')
               , file = "BestFileEver.txt")
```

25) Take audio features from the artists of “Spotify Charts”

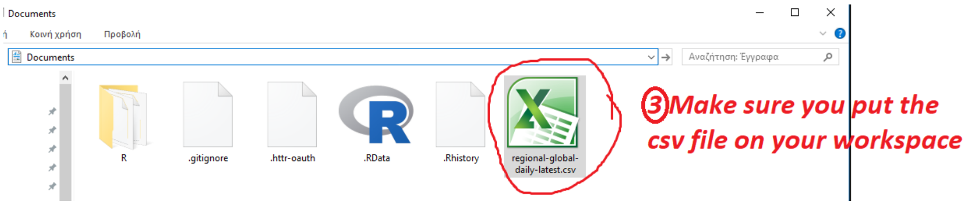


https://spotifycharts.com/regional

①go to spotifycharts.com

②download to csv

	TRACK	STREAMS
1	**I Don't Care (with Justin Bieber)** by Ed Sheeran	7,517,751
2	**bad guy** by Billie Eilish	5,952,365
3	**Old Town Road - Remix** by Lil Nas X	4,602,916
4	**If I Can't Have You** by Shawn Mendes	4,059,403
5	**SOS** by Avicii	3,401,108
6	**ME! (feat. Brendon Urie of Panic! At The Disco)** by Taylor Swift	3,316,290
7	**Old Town Road** by Lil Nas X	2,818,607



③Make sure you put the csv file on your workspace

```
top200 <- read.csv('regional-global-daily-latest.csv')

remaining_artists <- c()
tracks <- c()
for (i in 2:length(top200$X.2)) {
   print(str_c("Processing Artist: ", top200$X.2[i], " ", i, " of ", length(top200$X.2), " pass: "))
   succeed = FALSE
   tryCatch({
      tracks <- get_artist_audio_features(toString(top200$X.2[i])) %>%
         rbind(tracks)
      succeed = TRUE
      print("Succeed")
   }, error = function(e) { print(e) })
   if (!succeed) {
      remaining_artists <- c(remaining_artists, toString(top200$X.2[i]))
   }
}
```

# 26) Which songs from Spotify Chart artists are more like rap or hip-hop?

# Probably the songs with a lot of speechiness are rap or hip-hop. Therefore, for a threshold of 0.5 speechiness, the rap or hip-hop songs are:

```
tracks %>%
   select (track_name, artist_name, track_preview_url) %>%
   filter(tracks $speechiness>0.5) %>%
   kable()
```

# The rest is up to you! Good luck!