Tidyverse Extended Assignment

Warner Alexis

2023-11-12

##1. Loading the Tidyverse packages and the data into R: This is Souleymane Doumbia Assignment

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
library(readr)

spotify_data <- read_csv("https://raw.githubusercontent.com/Doumgit/Sentiment-Analysis-Project/main/spo
```

```
## Rows: 32833 Columns: 23
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (10): track_id, track_name, track_artist, track_album_id, track_album_na...
## dbl (13): track_popularity, danceability, energy, key, loudness, mode, speec...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(spotify_data)
```

```
## # A tibble: 6 x 23
##   track_id               track_name track_artist track_popularity track_album_id
##   <chr>                  <chr>      <chr>                  <dbl> <chr>
## 1 6f807x0ima9a1j3VPbc7VN I Don't C~ Ed Sheeran                66 2oCs0DGTsRO98~
## 2 0r7CVbZTWZgbTCYdfa2P31 Memories ~ Maroon 5                  67 63rPSO264uRjW~
## 3 1z1Hg7Vb0AhHDiEmnDE79l All the T~ Zara Larsson              70 1HoSmj2eLcsrR~
## 4 75FpbthrwQmzHlBJLuGdC7 Call You ~ The Chainsm~              60 1nqYsOef1yKKu~
## 5 1e8PAfcKUYoKkxPhrHqw4x Someone Y~ Lewis Capal~              69 7m7vv9wlQ4i0L~
## 6 7fvUMiyapMsRRxr07cU8Ef Beautiful~ Ed Sheeran                67 2yiy9cd2QktrN~
## # i 18 more variables: track_album_name <chr>, track_album_release_date <chr>,
## #   playlist_name <chr>, playlist_id <chr>, playlist_genre <chr>,
## #   playlist_subgenre <chr>, danceability <dbl>, energy <dbl>, key <dbl>,
## #   loudness <dbl>, mode <dbl>, speechiness <dbl>, acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
```

```
## #   duration_ms <dbl>
```

##2. Data Manipulation:

The 'dplyr' package is a powerful tool for data transformation and summarization within the Tidyverse collection. It allows for clear and concise data manipulation, enabling a variety of operations such as filtering rows, selecting specific columns, mutating the dataset to include new variables, summarizing data, and arranging rows based on certain criteria. For example:

```
track_pop_above_60 <- spotify_data %>%
  filter(track_popularity > 60) %>%
  select(track_name, track_artist, danceability, energy, tempo)
head(track_pop_above_60)
```

```
## # A tibble: 6 x 5
##   track_name                                track_artist danceability energy tempo
##   <chr>                                     <chr>               <dbl>  <dbl> <dbl>
## 1 I Don't Care (with Justin Bieber) - Lo~ Ed Sheeran          0.748  0.916  122.
## 2 Memories - Dillon Francis Remix         Maroon 5            0.726  0.815  100.
## 3 All the Time - Don Diablo Remix         Zara Larsson        0.675  0.931  124.
## 4 Someone You Loved - Future Humans Remix Lewis Capal~        0.65   0.833  124.
## 5 Beautiful People (feat. Khalid) - Jack~ Ed Sheeran          0.675  0.919  125.
## 6 Never Really Over - R3HAB Remix         Katy Perry          0.449  0.856  113.
```

In the example above, we demonstrated how to use `dplyr` to refine the spotify_data dataset to focus on tracks with a popularity greater than 60. We achieve this by employing the `filter()` function. Subsequently, we pare down the dataset to include only relevant columns that we are interested in analyzing: track name, artist, danceability, energy, and tempo by using the `select()` function. This streamlined dataset is then outputted, with `head()` used to display just the first few entries for a quick preview of the transformed data.

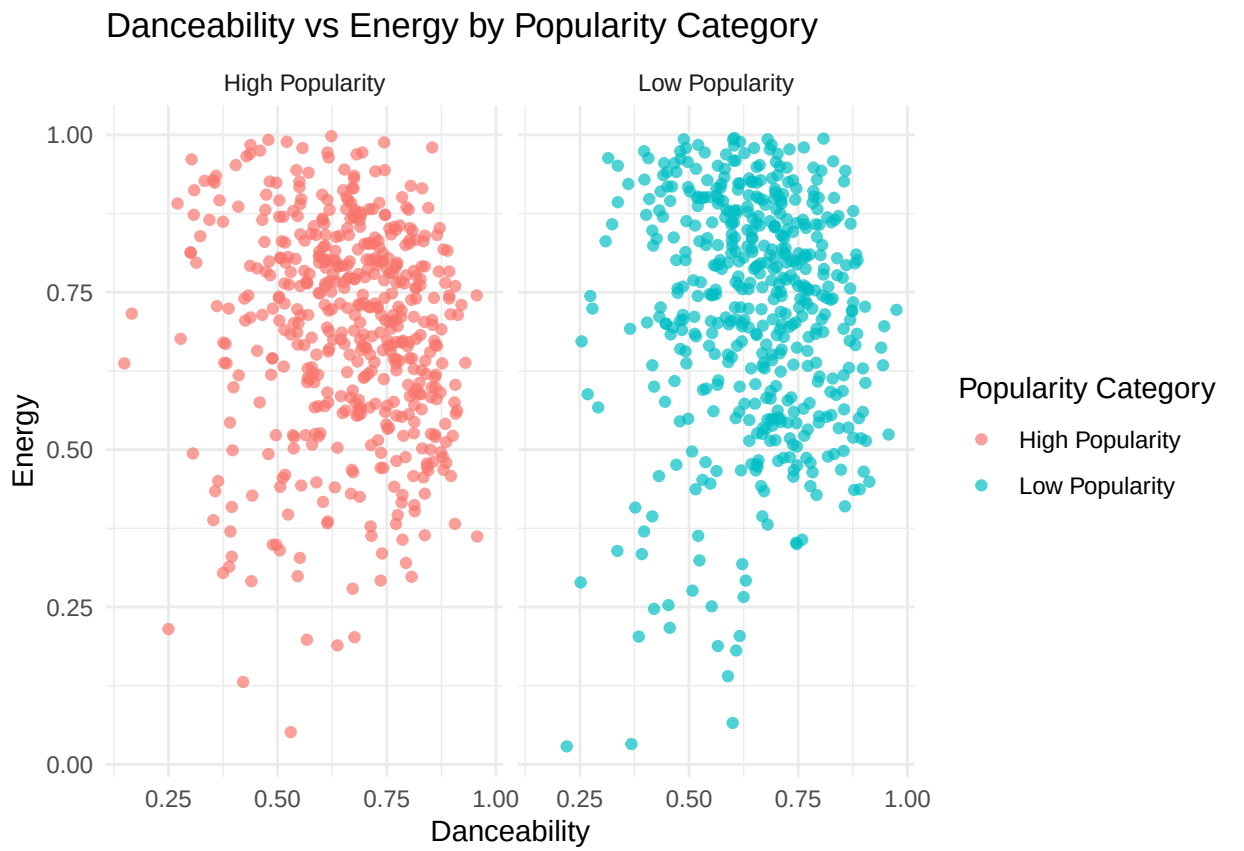
##3. Data Visualization:

With `dplyr` and `ggplot2` together, you can create a variety of visualizations. For instance, a scatter plot to see the relationship between 'danceability' and 'energy' could be made like so:

```
# Data manipulation with dplyr: let's categorize tracks as 'High popularity' or 'Low popularity'
# assuming the median of the 'track_popularity' can be a good threshold
spotify_data_mutated <- spotify_data %>%
  mutate(popularity_category = if_else(track_popularity > median(track_popularity, na.rm = TRUE),
                                       "High Popularity",
                                       "Low Popularity"))

# For a large dataset like spotify_data, you might want to take a sample to make plotting faster
sampled_data <- sample_n(spotify_data_mutated, 1000)

ggplot(sampled_data, aes(x = danceability, y = energy, color = popularity_category)) +
  geom_point(alpha = 0.7) +
  facet_wrap(~popularity_category) +
  labs(title = "Danceability vs Energy by Popularity Category",
       x = "Danceability",
       y = "Energy",
       color = "Popularity Category") +
  theme_minimal()
```




```
# Saving the plot as png
ggsave("Danceability_vs_Energy_by_Popularity_Category.png", width = 10, height = 8)
```

In this vignette, we leverage the capabilities of the Tidyverse, specifically `dplyr` for data manipulation and `ggplot2` for data visualization. First, we use `dplyr` to enhance our dataset by creating a new column that categorizes tracks based on their popularity. This categorization allows us to explore nuances in the data, such as differences in danceability and energy between tracks with high and low popularity. Due to the potential size of the dataset, we use `dplyr` to sample the data, making our subsequent visualization more efficient and manageable.

Once our data is prepared, we transition to visualizing it with `ggplot2`. We construct a scatter plot that illustrates the relationship between `danceability` and `energy`, utilizing the newly created popularity categories to color-code the points. This not only adds a layer of information to our plot but also enhances readability. To further refine our visualization, we employ `facet_wrap` to generate separate plots for each popularity category, providing a clearer comparison between the groups. Finally, we add appropriate labels and titles for context and clarity and save the resulting plot as a PNG file. This process from data manipulation to visualization exemplifies a seamless workflow within the Tidyverse ecosystem, yielding insightful and aesthetically pleasing representations of our data.

##4. Data Summarization:

Summarization is a crucial step in data analysis, allowing us to extract meaningful statistics from larger datasets. The `dplyr` package simplifies this process by providing intuitive functions such as `group_by()` and `summarize()`. For example, to calculate the average `loudness` by `playlist_genre`:

```
summarising_data <- spotify_data %>%
  group_by(playlist_genre) %>%
  summarize(avg_loudness = mean(loudness, na.rm = TRUE))
```

```
summarising_data
```

```
## # A tibble: 6 x 2
##   playlist_genre avg_loudness
##   <chr>                 <dbl>
## 1 edm                   -5.43
## 2 latin                 -6.26
## 3 pop                   -6.32
## 4 r&b                   -7.86
## 5 rap                   -7.04
## 6 rock                  -7.59
```

In the example above, we use these functions to calculate the average 'loudness' for each 'playlist_genre' within the 'spotify_data' dataset. The 'group_by()' function clusters the data by each unique genre, setting the stage for the calculation of summary statistics within each group. Then, 'summarize()' is applied to compute the mean 'loudness' across these groups, while 'na.rm = TRUE' ensures that missing values do not affect the calculation. The resulting object, 'summarising_data', contains the average loudness values neatly organized by genre, providing an immediate snapshot of this particular attribute across different genres.

Extended Tidyverse Assignment

Spotify data set has a lot information about songs, albums and artists. Our analysis is to find relevant information about the data set. We are going to read the data from Github and omit all **NA** values. It is important that we create new variables. We create a subset of the data taking every songs and albums released in 2001 to the most recent date.

```
spotify_data <- read_csv("https://raw.githubusercontent.com/Doumgit/Sentiment-Analysis-Project/main/spo
```

```
## Rows: 32833 Columns: 23
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (10): track_id, track_name, track_artist, track_album_id, track_album_na...
## dbl (13): track_popularity, danceability, energy, key, loudness, mode, speec...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
spotify_data <- na.omit(spotify_data)
head(spotify_data)
```

```
## # A tibble: 6 x 23
##   track_id               track_name track_artist track_popularity track_album_id
##   <chr>                  <chr>      <chr>                   <dbl> <chr>
## 1 6f807x0ima9a1j3VPbc7VN I Don't C~ Ed Sheeran                 66 2oCs0DGTsRO98~
## 2 0r7CVbZTWZgbTCYdfa2P31 Memories ~ Maroon 5                   67 63rPSO264uRjW~
## 3 1z1Hg7Vb0AhHDiEmnDE79l All the T~ Zara Larsson                70 1HoSmj2eLcsrR~
## 4 75FpbthrwQmzHlBJLuGdC7 Call You ~ The Chainsm~               60 1nqYsOef1yKKu~
## 5 1e8PAfcKUYoKkxPhrHqw4x Someone Y~ Lewis Capal~               69 7m7vv9wlQ4i0L~
## 6 7fvUMiyapMsRRxr07cU8Ef Beautiful~ Ed Sheeran                 67 2yiy9cd2QktrN~
## # i 18 more variables: track_album_name <chr>, track_album_release_date <chr>,
## #   playlist_name <chr>, playlist_id <chr>, playlist_genre <chr>,
## #   playlist_subgenre <chr>, danceability <dbl>, energy <dbl>, key <dbl>,
## #   loudness <dbl>, mode <dbl>, speechiness <dbl>, acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
```

```
## #   duration_ms <dbl>
```

```
# Create Yearly version of realease date
spotify_data$track_album_release_date <-    as.Date(spotify_data$track_album_release_date)
spotify_data$album_release_year <-   format(spotify_data$track_album_release_date,format = "%Y")



# Createa a subset of this data set
# spotify
album_data <- spotify_data %>% arrange(desc(duration_ms)) %>%
  select(track_album_name,track_name,track_artist,speechiness,loudness,tempo, danceability,speechiness
  filter( album_release_year >= 2001 & album_release_year <= 2023)
head(album_data)
```

```
## # A tibble: 6 x 9
##   track_album_name            track_name track_artist speechiness loudness tempo
##   <chr>                       <chr>      <chr>              <dbl>    <dbl> <dbl>
## 1 47 (Remix)                  47 - Remix Anuel AA          0.193     -4.89  146.
## 2 Jam On It (Re-Recorded Ver~ Jam On It~ Newcleus          0.0485    -5.93  116.
## 3 Solitude                    Come as Y~ V-Sag             0.0401    -7.82  121.
## 4 Joey Negro presents It's A~ Summer Gr~ The Joneses       0.0568    -9.07  127.
## 5 Greatest Hits               Won't Get~ The Who           0.0416    -9.39  135.
## 6 English Electric (Part One) The First~ Big Big Tra~      0.0305    -7.62  145.
## # i 3 more variables: danceability <dbl>, duration_ms <dbl>,
## #   album_release_year <chr>
```

With the subset created, we are going to retrieve all the 10 top songs with actual words spoken on them. This is how Spotify describes their feature metadata:

1- Speechiness: "Speechiness detects the presence of spoken words in a track". If the speechiness of a song is above 0.66, it is probably made of spoken words, a score between 0.33 and 0.66 is a song that may contain both music and words, and a score below 0.33 means the song does not have any speech.

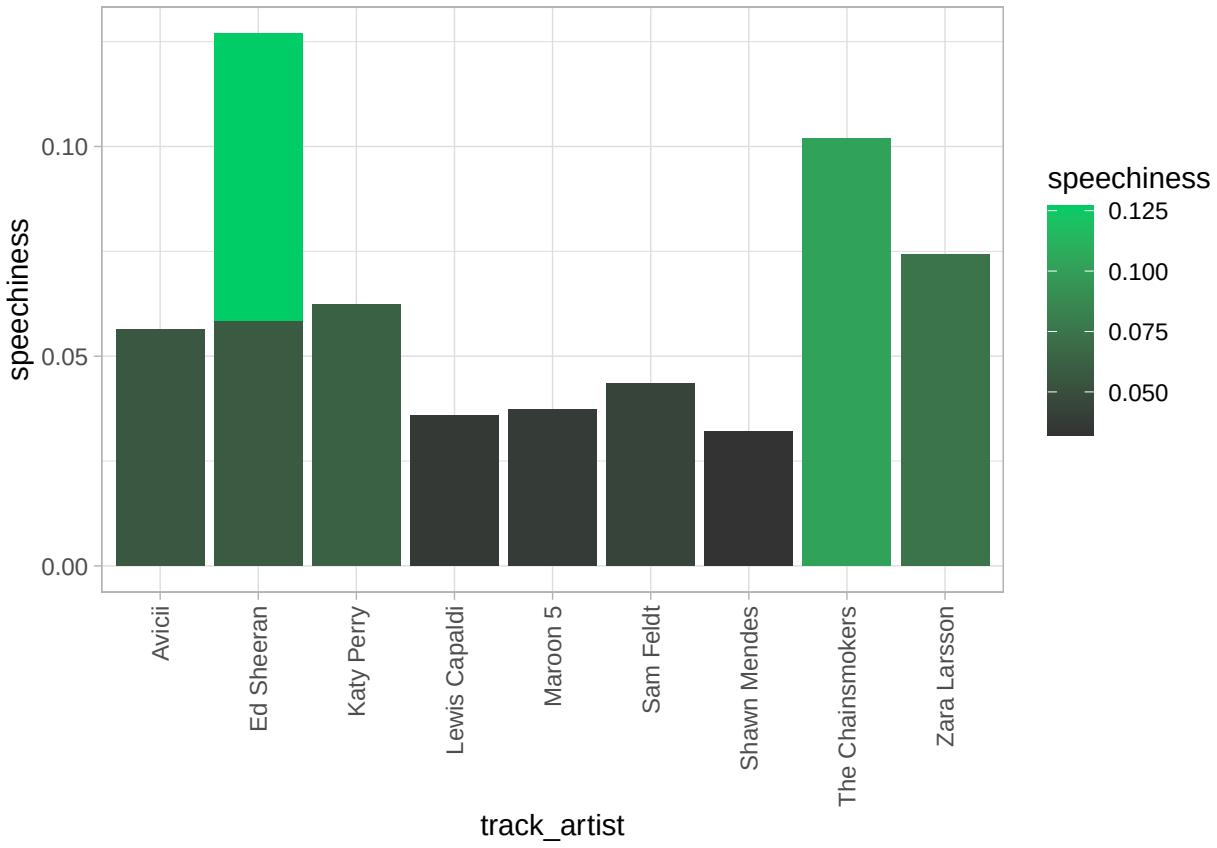
2- Energy: "(energy) represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy".

3- 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".

4- 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)".

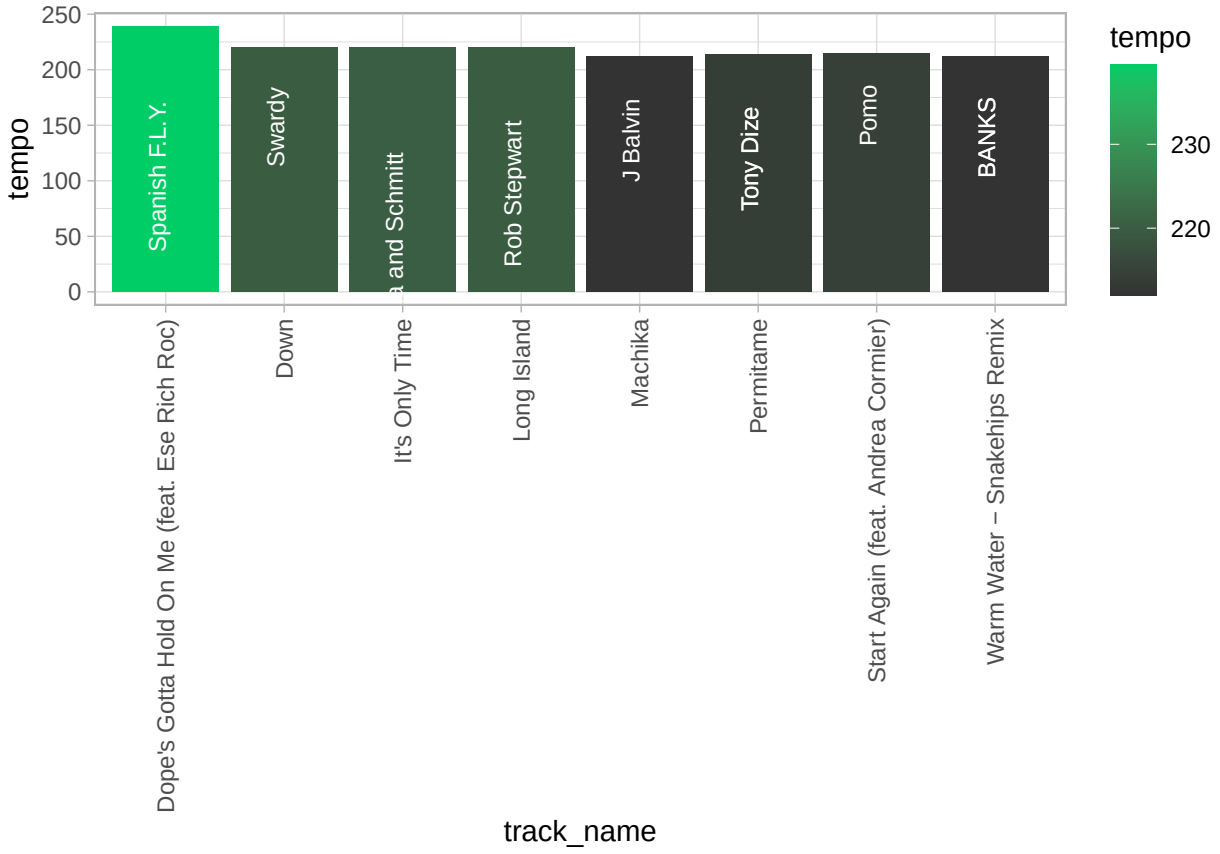
5 - Beats Per Minute (BPM) - The tempo of the song.

```
spotify_data %>% head(10)  %>% arrange(desc(speechiness)) %>% select(track_artist, speechiness) %>%
  ggplot(t, mapping = aes(x=track_artist, y=speechiness, fill=speechiness))+
  geom_bar(position = "dodge", stat="identity")+
  scale_fill_gradient(low = "grey20", high = "springgreen3")+
  theme_light()+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



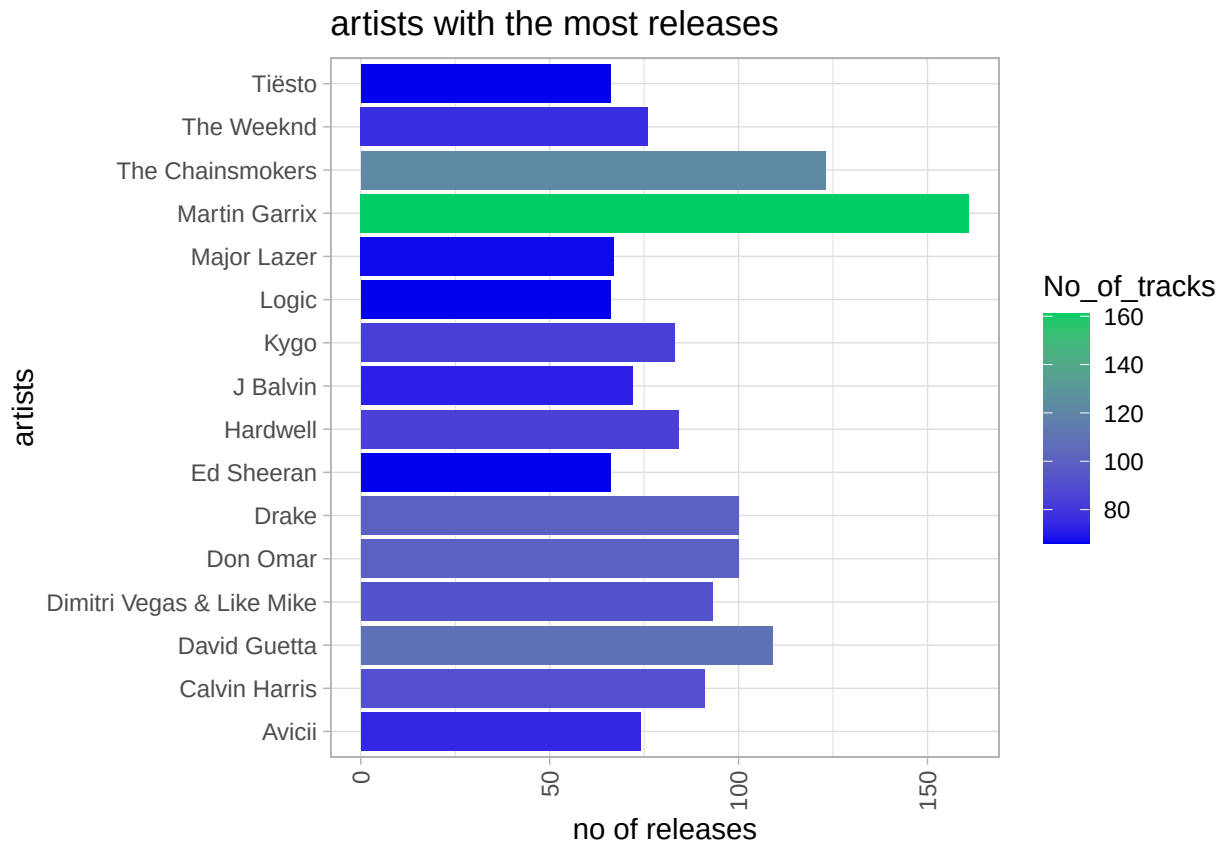
we have revealed the top 10 artists with the highest rate of spoken words in Spottily data set. There were a lot of good songs in 2019 and profound lyrics. Ed Sheeran, Katy Perry, Maroon 5 were incredible. Now lets look about songs that have the highest tempo since 2001.

```
album_data %>% select(track_name,track_artist,tempo) %>%
  arrange(desc(tempo)) %>% head(10) %>%
  ggplot( mapping = aes(x=track_name, y=tempo, fill=tempo))+
  geom_bar(position = "dodge", stat="identity")+
  scale_fill_gradient(low = "grey20", high = "springgreen3")+
  theme_light()+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))+labs(height=10, width=5)+
  geom_text(aes(label = track_artist),color="white",hjust= 1.5, vjust = 0, size = 3, angle = 90, positi
```



The top artists with the highest number of releases since 2001.

```
# artists with most releases
album_data %>% group_by(Artist = track_artist) %>%
  summarise(No_of_tracks = n()) %>%
  arrange(desc(No_of_tracks)) %>%
  top_n(15, wt = No_of_tracks) %>%
  ggplot(aes(x = Artist, y = No_of_tracks, fill = No_of_tracks)) +
  geom_bar(position = "dodge", stat="identity")+
  scale_fill_gradient(low = "blue2", high = "springgreen3")+
  theme_light()+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))+labs(height=10, width=5)+
  coord_flip() + labs(title = "artists with the most releases", x = "artists", y = "no of releases")
```



Conclusion

There are still a lot information to retrieve from the data set. we seen the top artists who made songs with the highest tempo and had profound lyrics on their songs. from 2001 to now, there have been few artist that released a lot of songs but, Martin Garrix and the Chainsmokers have released more songs since 2001 than any other artists or bands.