Testing Automated Reporting

Table of contents

Info about the dataset

Data has been retrieved from kaggle.

The dataset description is reported below:

- **Track Name**: Name of the song.
- **Album Name**: Name of the album the song belongs to.
- **Artist**: Name of the artist(s) of the song.
- **Release Date**: Date when the song was released.
- **ISRC**: International Standard Recording Code for the song.
- **All Time Rank**: Ranking of the song based on its all-time popularity.
- **Track Score**: Score assigned to the track based on various factors.
- **Spotify Streams**: Total number of streams on Spotify.
- **Spotify Playlist Count**: Number of Spotify playlists the song is included in.

- **Spotify Playlist Reach**: Reach of the song across Spotify playlists.
- **Spotify Popularity**: Popularity score of the song on Spotify.
- **YouTube Views**: Total views of the song's official video on YouTube.
- **YouTube Likes**: Total likes on the song's official video on YouTube.
- **TikTok Posts**: Number of TikTok posts featuring the song.
- **TikTok Likes**: Total likes on TikTok posts featuring the song.
- **TikTok Views**: Total views on TikTok posts featuring the song.
- **YouTube Playlist Reach**: Reach of the song across YouTube playlists.
- **Apple Music Playlist Count**: Number of Apple Music playlists the song is included in.
- **AirPlay Spins**: Number of times the song has been played on radio stations.
- **SiriusXM Spins**: Number of times the song has been played on SiriusXM.
- **Deezer Playlist Count**: Number of Deezer playlists the song is included in.
- **Deezer Playlist Reach**: Reach of the song across Deezer playlists.
- **Amazon Playlist Count**: Number of Amazon Music playlists the song is included in.
- **Pandora Streams**: Total number of streams on Pandora.
- **Pandora Track Stations**: Number of Pandora stations featuring the song.
- **Soundcloud Streams**: Total number of streams on Soundcloud.
- **Shazam Counts**: Total number of times the song has been Shazamed.
- **TIDAL Popularity**: Popularity score of the song on TIDAL.
- **Explicit Track**: Indicates whether the song contains explicit content.

This is a test to check some of the functionality integrated to be integrated in a possible version of an automated report for **Spotify**!

Creating the R and Python environments using `renv` and `venv`

The environment is created before loading any package and is stored in the file *build-env.R*.

Similarly, python dependencies have been "frozen" into a `requirements.txt` file.

Using R to create a parquet file

> **Note**
>
> This section should be executed only the first time to generate the parquet file, then we can just query that file using the opened connection to the database to extract and load only the necessary data in memory.

```
df <- read_csv("data/raw/2024_spotify-most-streamed.csv")
```

```
Rows: 4600 Columns: 29
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (5): Track, Album Name, Artist, Release Date, ISRC
dbl  (6): Track Score, Spotify Popularity, Apple Music Playlist Count, Deeze...
num (17): All Time Rank, Spotify Streams, Spotify Playlist Count, Spotify Pl...
lgl  (1): TIDAL Popularity

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.
```

```
df_cleaned <- janitor::clean_names(df)
```

```
dir_out <- "data/parquet/"
```

```
arrow::write_dataset(df_cleaned, dir_out, partitioning = "spotify_popularity")
```

Accessing and querying the parquet files using SQL and DuckDB

```
dir_out <- "data/parquet/"
```

```
ds <- open_dataset(dir_out, partitioning = "spotify_popularity")

con <- dbConnect(duckdb())

duckdb_register_arrow(con, "spotify_artists", ds)

df_very_popular <- dbGetQuery(con,
           "SELECT *
           FROM spotify_artists
           WHERE spotify_popularity >= 70")
```

```
df_not_popular <- dbGetQuery(con,
           "SELECT *
           FROM spotify_artists
           WHERE spotify_popularity < 70")
```

```
duckdb_unregister(con, "spotify_artists")
dbDisconnect(con)
```

This ensures that after querying we disconnect and close the connection to the database.

Visualizing the data

We can first check the structure of the dataset to make sure that the type of every variable is consistent with what they are supposed to represent.

```
str(df_very_popular)
```

```
'data.frame':   1465 obs. of  29 variables:
 $ track                     : chr  "Lollipop" "Boy's a Liar Pt. 2" "INDUSTRY BABY (feat. Ja
 $ album_name                : chr  "EVERYBODY GO TO THE DISCOTEK" "Boy's a liar Pt. 2" "IND
 $ artist                    : chr  "Darell" "PinkPantheress" "Lil Nas X" "DaBaby" ...
 $ release_date              : chr  "7/28/2023" "2/3/2023" "7/23/2021" "4/17/2020" ...
 $ isrc                      : chr  "USSD12300307" "GBAYE2300015" "USSM12104539" "USUM720079
 $ all_time_rank             : num  94 96 110 167 174 206 238 245 247 302 ...
 $ track_score               : num  158 154 145 117 114 ...
 $ spotify_streams           : num  2.75e+08 8.51e+08 2.01e+09 1.64e+09 4.40e+08 ...
 $ spotify_playlist_count    : num  37806 129047 354831 272933 26107 ...
 $ spotify_playlist_reach    : num  22826643 50996176 87942438 50425049 15314737 ...
 $ you_tube_views            : num  3.48e+08 2.12e+08 1.00e+09 1.17e+09 5.25e+08 ...
 $ you_tube_likes            : num  1553295 2461166 16856186 13667470 10820829 ...
 $ tik_tok_posts             : num  4349700 3502547 1750366 5241838 3411209 ...
 $ tik_tok_likes             : num  3.21e+08 9.55e+08 9.16e+08 3.65e+08 5.51e+08 ...
 $ tik_tok_views             : num  4.77e+09 8.78e+09 9.56e+09 3.87e+09 6.72e+09 ...
 $ you_tube_playlist_reach   : num  2.68e+09 8.06e+08 3.19e+09 2.02e+08 6.89e+07 ...
 $ apple_music_playlist_count: num  19 162 360 216 33 209 310 63 60 221 ...
 $ air_play_spins            : num  23875 223173 245228 298661 11830 ...
 $ sirius_xm_spins           : num  96 6243 925 62 92 ...
 $ deezer_playlist_count     : num  37 65 109 98 6 79 112 43 39 43 ...
 $ deezer_playlist_reach     : num  3780737 8655684 3242122 1699879 26844 ...
 $ amazon_playlist_count     : num  29 130 74 47 35 45 80 43 63 68 ...
 $ pandora_streams           : num  5.10e+06 1.02e+07 7.52e+07 2.46e+08 1.34e+06 ...
 $ pandora_track_stations    : num  5276 36847 96430 272002 3263 ...
 $ soundcloud_streams        : num  NA 13896129 NA NA NA ...
 $ shazam_counts             : num  1245874 2580691 12693846 7646337 1058579 ...
```

```
 $ tidal_popularity          : logi  NA NA NA NA NA NA ...
 $ explicit_track            : num  1 1 1 1 0 1 1 1 1 0 ...
 $ spotify_popularity        : int  70 70 70 70 70 70 70 70 70 70 ...
```

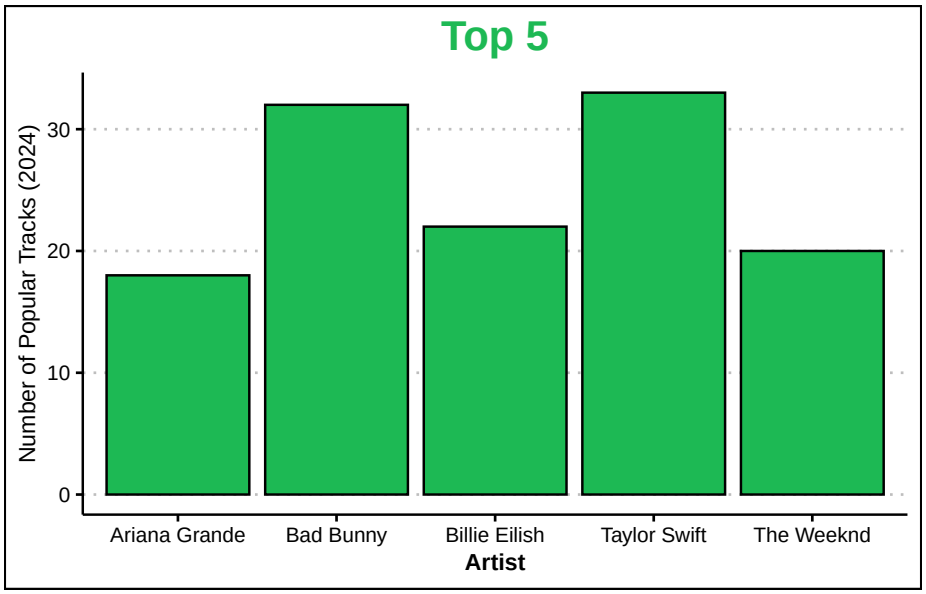
We learn that the column `tidal_popularity` is full of empty values, so we should drop it from the dataset. Also, `release_date` is not properly initialized as a `date` object, so it would be useful to convert it into one.

```
df_cleaned_pop <- df_very_popular %>%
  select(-tidal_popularity) %>%
  mutate(release_date = mdy(release_date))
```

Let's say we want to try to understand the popularity of a track on Spotify based on the number of likes it got. First let's visualize the most popular artists

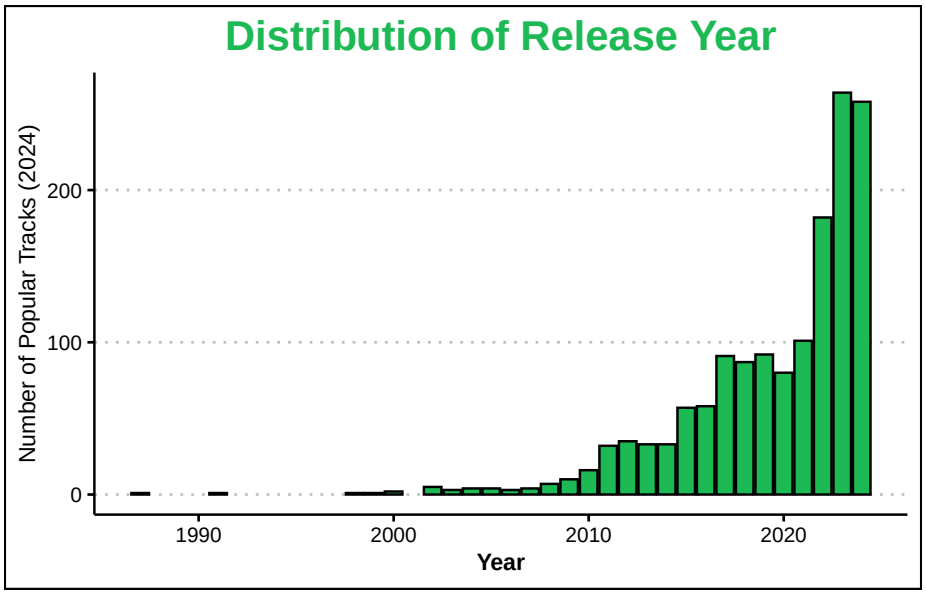
```
spotify_green <- "#1DB954"
```

```
df_cleaned_pop %>%
  group_by(artist) %>%
  summarise(count = dplyr::n()) %>%
  arrange(desc(count)) %>%
  head(5) %>%
  ggplot(aes(x = artist)) +
    geom_col(aes(y = count), color = "black", fill = spotify_green) +
    #stat_summary(aes(y = stat_identity(count))) +
    labs(x = "Artist", y = "Number of Popular Tracks (2024)",
         title = "Top 5") +
    theme_clean() +
    theme(plot.title = element_text(hjust = .5, size = 18, color = spotify_green),
          axis.title.x = element_text(face = "bold"))
```



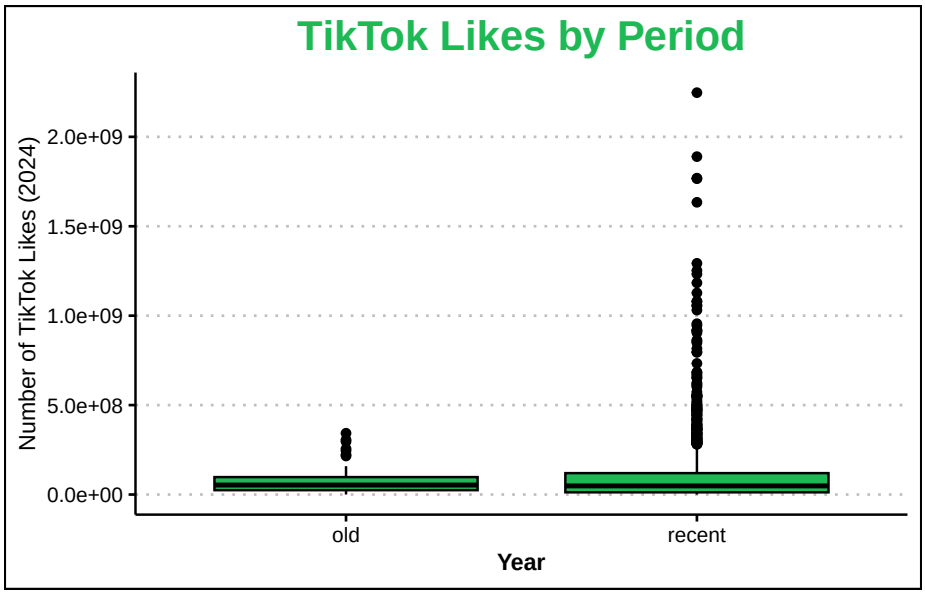
Now we might be interested in knowing from which year are the tracks contained in the dataset. We can observe the distribution using another barplot

```
df_cleaned_pop %>%
  mutate(release_year = year(release_date)) %>%
  group_by(release_year) %>%
  summarize(count = n()) %>%
  ggplot(aes(x = release_year)) +
    geom_col(aes(y = count), color = "black", fill = spotify_green) +
    #stat_summary(aes(y = stat_identity(count))) +
    labs(x = "Year", y = "Number of Popular Tracks (2024)",
         title = "Distribution of Release Year") +
    theme_clean() +
    theme(plot.title = element_text(hjust = .5, size = 18, color = spotify_green),
          axis.title.x = element_text(face = "bold"))
```

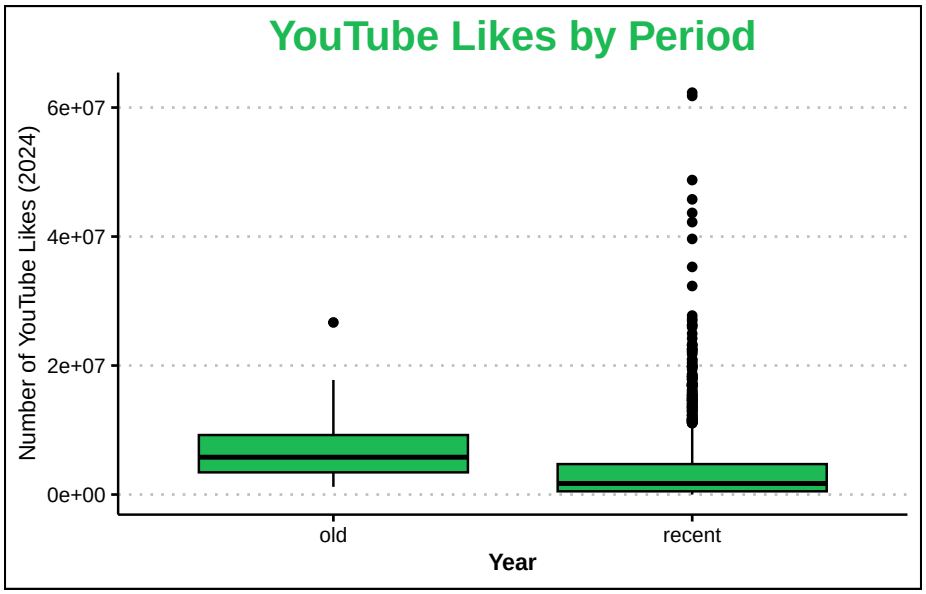


Most of the popular songs are from 2023 and 2024, although there are a few outliers coming from the late 80s and early 90s. At this point one might be interested in understanding if the appearance of these songs on the chart is associated to the number of likes they received on social media platforms (e.g., TikTok & Instagram). Luckily, we also have this kind of data, so we can check how these two groups differ on the number of likes they received on social media.

```
df_cleaned_pop %>%
  mutate(period = ifelse(year(release_date) <= 2010, "old", "recent")) %>%
  ggplot(aes(x = period, y = tik_tok_likes)) +
  geom_boxplot(color = "black", fill = spotify_green) +
  labs(x = "Year", y = "Number of TikTok Likes (2024)",
         title = "TikTok Likes by Period") +
    theme_clean() +
    theme(plot.title = element_text(hjust = .5, size = 18, color = spotify_green),
          axis.title.x = element_text(face = "bold"))
```

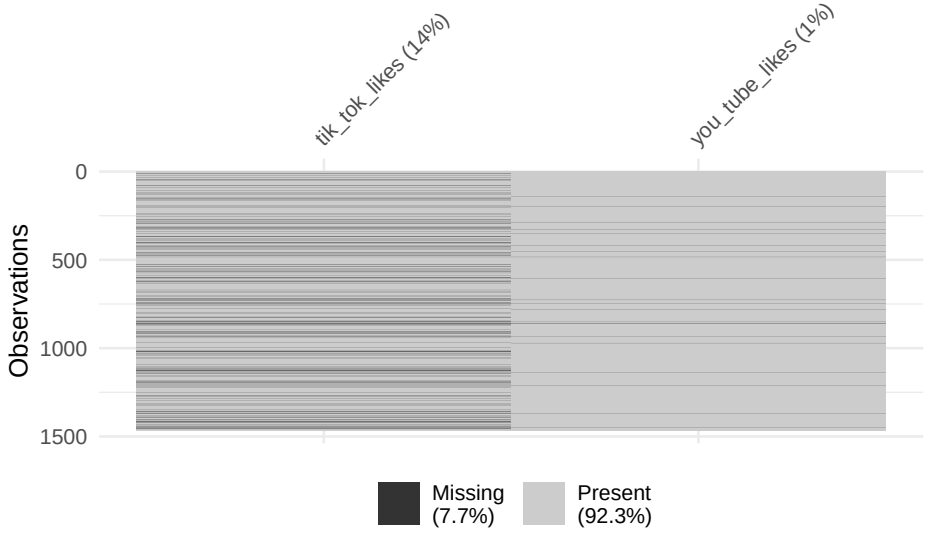



```
df_cleaned_pop %>%
  mutate(period = ifelse(year(release_date) <= 2010, "old", "recent")) %>%
  ggplot(aes(x = period, y = you_tube_likes)) +
  geom_boxplot(color = "black", fill = spotify_green) +
  labs(x = "Year", y = "Number of YouTube Likes (2024)",
       title = "YouTube Likes by Period") +
    theme_clean() +
    theme(plot.title = element_text(hjust = .5, size = 18, color = spotify_green),
          axis.title.x = element_text(face = "bold"))
```



However, there are many missing values for the TikTok category and so it is difficult to make a fair comparison. Also, YouTube has been around for much longer compared to TikTok so it makes sense that older songs gathered more likes throughout the years. Interestingly, among the old songs there is just one outlier in the YouTube category compared to the many in the TikTok group.

```
df_cleaned_pop %>%
  mutate(period = ifelse(year(release_date) <= 2010, "old", "recent")) %>%
  select(tik_tok_likes, you_tube_likes) %>%
  vis_miss()
```



Data Modeling

We now delve deeper into the statistical associations between the different variables in the dataset by fitting a linear model.

```
yt_lm <- lm(you_tube_likes ~ you_tube_views, data = df_cleaned)
summary(yt_lm)
```

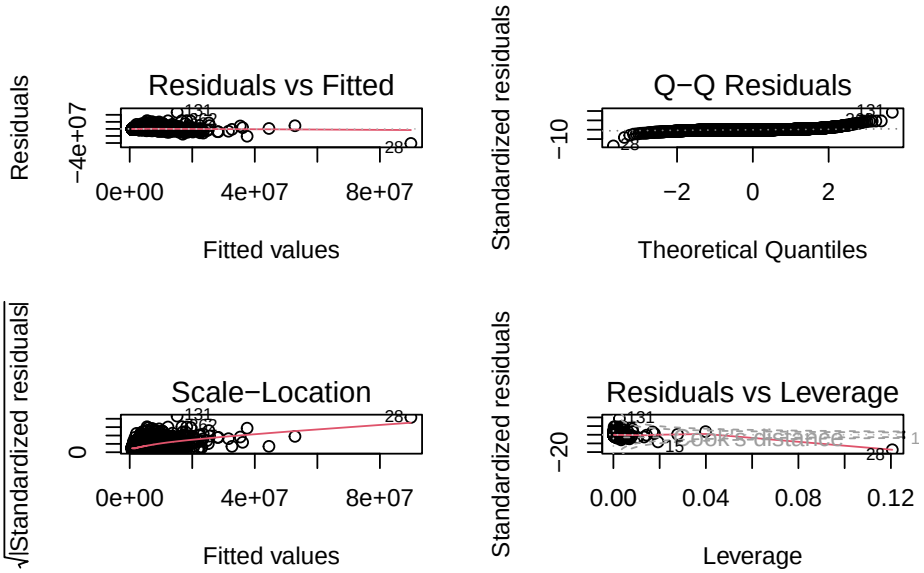
```
Call:
lm(formula = you_tube_likes ~ you_tube_views, data = df_cleaned)

Residuals:
      Min        1Q    Median        3Q       Max
-41159451   -714175   -484235    229979  47158209

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    7.345e+05  4.465e+04   16.45   <2e-16 ***
you_tube_views 5.464e-03  5.526e-05   98.88   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2536000 on 4283 degrees of freedom
  (315 observations deleted due to missingness)
Multiple R-squared:  0.6954,    Adjusted R-squared:  0.6953
F-statistic:  9777 on 1 and 4283 DF,  p-value: < 2.2e-16
```

```
par(mfrow = c(2,2))
plot(yt_lm)
```



This time we learn much more from the diagnostics of the residuals of the model. Importantly, the errors do not have a mean of zero and show significant departures from the assumption of normality (as can be seen in the top-right plot). More importantly, also the assumption of equal error variance is violated. In the bottom-right plot, we also see that there are some outliers whose presence is greatly affecting the estimated regression coefficients.

Conclusion

This document is a streamlined example of how it is possible to produce automatic reports in different formats starting from a single and connected quarto document file. The main features of this automatic report are:

1. connection and retrieval of data from parquet files or any other instances of SQL databases.
2. data cleaning and pre-processing.
3. up-to-date data visualization processes that ensure that the most relevant plots are generated from the most recent data.
4. up-to-date data modeling processes that automatically update their parameters based on the new data they receive.

> **!** Important
>
> This document should not be viewed as a final and finished product as some aspects of the data management and engineering process were not discussed (e.g., data integrity, patterns in data missingness, advanced han-

dling of outliers, etc.)
View this document as an example of the impact that an automatic report can have on the reporting process in a corporate setting.

Resources

Apache Software Foundation, "Tabular Datasets — Apache Arrow v19.0.0." Accessed: Feb. 01, 2025. [Online]. Available: https://arrow.apache.org/docs/python/dataset.html#dataset

Nicholas Tierney, *Exploring patterns with UpSetR*. Accessed: Feb. 01, 2025. [Online]. Available: https://cran.r-project.org/web/packages/naniar/vignettes/naniar-visualisation.html

Nicholas Tierney, Di Cook, Miles McBain, and Colin Fay, "Plot the pattern of missingness using an upset plot. — gg_miss_upset." Accessed: Feb. 01, 2025. [Online]. Available: https://naniar.njtierney.com/reference/gg_miss_upset.html

NIDULA ELGIRIYEWITHANA, "Most Streamed Spotify Songs 2024." Accessed: Feb. 01, 2025. [Online]. Available: https://www.kaggle.com/datasets/nelgiriyewithana/most-streamed-spotify-songs-2024

Rich Pauloo, "Parquet, SQL, DuckDB, arrow, dbplyr and R," Rich Pauloo. Accessed: Feb. 01, 2025. [Online]. Available: https://www.richpauloo.com/blog/parquet/

P. Team, "Building a reporting infrastructure with Quarto," Posit. Accessed: Feb. 01, 2025. [Online]. Available: https://posit.co/blog/building-a-reporting-infrastructure-with-quarto/