# Final Project

## Michael Whelan 17338833

Packages I will be using throughout the project

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.3.6      ✔ purrr   0.3.4
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.2.1      ✔ stringr 1.4.1
## ✔ readr   2.1.3      ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
```

As you can see the `tidyverse` library contains other libraries such as `tibble` , `ggplot` and `dplyr` which I will be using throughout.

I will introduce more as I go on.

This project is made up of 3 components:

- Analysis
- R Packages
- Functions/Programming

## Analysis

### Context:

The Spotify dataset provides insight into users data about which songs people listen to, and not just the popularity of tracks, but also features of the tracks they have in their library is recorded in their database.

For further reading: https://github.com/rfordatascience/tidytuesday/blob/master/data/2020/2020-01-21/readme.md (https://github.com/rfordatascience/tidytuesday/blob/master/data/2020/2020-01-21/readme.md)

```
#reading my csv file from local machine
df <- read.csv("spotify_songs.csv", header = T)

#A glimpse at the dataset
df[1:5,1:19]
```

```
##                  track_id                                            track_name
## 1 6f807x0ima9a1j3VPbc7VN I Don't Care (with Justin Bieber) - Loud Luxury Remix
## 2 0r7CVbZTWZgbTCYdfa2P31                       Memories - Dillon Francis Remix
## 3 1z1Hg7Vb0AhHDiEmnDE79l                       All the Time - Don Diablo Remix
## 4 75FpbthrwQmzHlBJLuGdC7                     Call You Mine - Keanu Silva Remix
## 5 1e8PAfcKUYoKkxPhrHqw4x               Someone You Loved - Future Humans Remix
##       track_artist track_popularity         track_album_id
## 1       Ed Sheeran               66 2oCs0DGTsRO98Gh5ZSl2Cx
## 2         Maroon 5               67 63rPSO264uRjW1X5E6cWv6
## 3     Zara Larsson               70 1HoSmj2eLcsrR0vE9gThr4
## 4 The Chainsmokers               60 1nqYsOef1yKKuGOVchbsk6
## 5    Lewis Capaldi               69 7m7vv9wlQ4i0LFuJiE2zsQ
##                                         track_album_name
## 1 I Don't Care (with Justin Bieber) [Loud Luxury Remix]
## 2                       Memories (Dillon Francis Remix)
## 3                       All the Time (Don Diablo Remix)
## 4                           Call You Mine - The Remixes
## 5               Someone You Loved (Future Humans Remix)
##   track_album_release_date playlist_name            playlist_id playlist_genre
## 1               14/06/2019     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
## 2               13/12/2019     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
## 3               05/07/2019     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
## 4               19/07/2019     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
## 5               05/03/2019     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
##   playlist_subgenre danceability energy key loudness mode speechiness
## 1         dance pop        0.748  0.916   6   -2.634    1      0.0583
## 2         dance pop        0.726  0.815  11   -4.969    1      0.0373
## 3         dance pop        0.675  0.931   1   -3.432    0      0.0742
## 4         dance pop        0.718  0.930   7   -3.778    1      0.1020
## 5         dance pop        0.650  0.833   1   -4.672    1      0.0359
##   acousticness instrumentalness
## 1       0.1020         0.00e+00
## 2       0.0724         4.21e-03
## 3       0.0794         2.33e-05
## 4       0.0287         9.43e-06
## 5       0.0803         0.00e+00
```

Now we will look at the structure of the data set. We will see it is a data frame with 32,828 observations and 23 variables

```
str(df)
```

```
## 'data.frame':    32833 obs. of  23 variables:
##  $ track_id                : chr  "6f807x0ima9a1j3VPbc7VN" "0r7CVbZTWZgbTCYdfa2P3
1" "1z1Hg7Vb0AhHDiEmnDE79l" "75FpbthrwQmzHlBJLuGdC7" ...
##  $ track_name              : chr  "I Don't Care (with Justin Bieber) - Loud Luxury
Remix" "Memories - Dillon Francis Remix" "All the Time - Don Diablo Remix" "Call You
Mine - Keanu Silva Remix" ...
##  $ track_artist            : chr  "Ed Sheeran" "Maroon 5" "Zara Larsson" "The Chai
nsmokers" ...
##  $ track_popularity        : int  66 67 70 60 69 67 62 69 68 67 ...
##  $ track_album_id          : chr  "2oCs0DGTsRO98Gh5ZSl2Cx" "63rPSO264uRjW1X5E6cWv
6" "1HoSmj2eLcsrR0vE9gThr4" "1nqYsOef1yKKuGOVchbsk6" ...
##  $ track_album_name        : chr  "I Don't Care (with Justin Bieber) [Loud Luxury
Remix]" "Memories (Dillon Francis Remix)" "All the Time (Don Diablo Remix)" "Call You
Mine - The Remixes" ...
##  $ track_album_release_date: chr  "14/06/2019" "13/12/2019" "05/07/2019" "19/07/20
19" ...
##  $ playlist_name           : chr  "Pop Remix" "Pop Remix" "Pop Remix" "Pop Remix"
...
##  $ playlist_id             : chr  "37i9dQZF1DXcZDD7cfEKhW" "37i9dQZF1DXcZDD7cfEKh
W" "37i9dQZF1DXcZDD7cfEKhW" "37i9dQZF1DXcZDD7cfEKhW" ...
##  $ playlist_genre          : chr  "pop" "pop" "pop" "pop" ...
##  $ playlist_subgenre       : chr  "dance pop" "dance pop" "dance pop" "dance pop"
...
##  $ danceability            : num  0.748 0.726 0.675 0.718 0.65 0.675 0.449 0.542
0.594 0.642 ...
##  $ energy                  : num  0.916 0.815 0.931 0.93 0.833 0.919 0.856 0.903
0.935 0.818 ...
##  $ key                     : int  6 11 1 7 1 8 5 4 8 2 ...
##  $ loudness                : num  -2.63 -4.97 -3.43 -3.78 -4.67 ...
##  $ mode                    : int  1 1 0 1 1 1 0 0 1 1 ...
##  $ speechiness             : num  0.0583 0.0373 0.0742 0.102 0.0359 0.127 0.0623
0.0434 0.0565 0.032 ...
##  $ acousticness            : num  0.102 0.0724 0.0794 0.0287 0.0803 0.0799 0.187
0.0335 0.0249 0.0567 ...
##  $ instrumentalness        : num  0.00 4.21e-03 2.33e-05 9.43e-06 0.00 0.00 0.00
4.83e-06 3.97e-06 0.00 ...
##  $ liveness                : num  0.0653 0.357 0.11 0.204 0.0833 0.143 0.176 0.111
0.637 0.0919 ...
##  $ valence                 : num  0.518 0.693 0.613 0.277 0.725 0.585 0.152 0.367
0.366 0.59 ...
##  $ tempo                   : num  122 100 124 122 124 ...
##  $ duration_ms             : int  194754 162600 176616 169093 189052 163049 187675
207619 193187 253040 ...
```

## Attributes:

I will explain the variables for this dataset to give some context

- **track_id**: Song ID
- **track_name**: Song Namw
- **track_artist**: Song Artist
- **track_popularity**: Song popularity (rating 0-100)

- **track_album_id**: Album ID
- **track_album_name**: Song album name
- **track_album_name**: Song album name
- **track_album_release_date**: Date the album was released
- **playlist_name**: Name of playlist
- **playlist_id**: Playlist ID
- **playlist_genre**: Playlist genre
- **playlist_subgenre**: Playlist subgenre
- **danceability**: Danceability describes how suitable a track is for dancing. 0.0 being least danceable 1.0 being most danceable
- **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.
- **key**: The estimated overall key of the track.
- **loudness**: The overall loudness of a track in decibels (dB). Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.
- **mode**: Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
- **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. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
- **acousticness**: A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
- **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. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0. The distribution of values for this feature look like this:
- **liveness**: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
- **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.
- **duration_ms**: The duration of the track in milliseconds.

Some questions I asked myself before this:

- What kind of distributions are the musical variables?
- Who is the most popular artist?
- Is there more positvity in major modes?
- What year was the most popular between 2010 - 2020?
- Can Post Malone be considered a rapper under these variables?

## Data Cleaning:

Before I answer some of these questions the data needs to be cleaned. I will lay out each variable that I will be changing anything I don't mention I have left as is.

```
#Changing my dataset into a tibble for easier manipulation and make full use of dplyr
and ggplots

df <- as_tibble(df)
```

## id

The original data set never had an `id` variable so I decided to add one in to give a number to each row

```
# Create a column with numbers 1:32833 (no. of rows)
# and append it to the data set before the track column
# which is originally the first column



df %>%
  mutate(id = c(1:32833),.before = track_id) -> df

#Taking the first 4 rows and 3 columns to check if id was appended
df[1:4,1:3]
```

```
## # A tibble: 4 × 3
##      id track_id               track_name
##   <int> <chr>                  <chr>
## 1     1 6f807x0ima9a1j3VPbc7VN I Don't Care (with Justin Bieber) - Loud Luxury …
## 2     2 0r7CVbZTWZgbTCYdfa2P31 Memories - Dillon Francis Remix
## 3     3 1z1Hg7Vb0AhHDiEmnDE79l All the Time - Don Diablo Remix
## 4     4 75FpbthrwQmzHlBJLuGdC7 Call You Mine - Keanu Silva Remix
```

## track_id

We won't need the `track_id` for each track because we now have the `id` variable so we will drop this variable

```
#selecting the data frame and minusing the variable uri
df <- select(df,-track_id)
```

## track_album_id

Same as above we won't need `track_album_id`

```
#selecting the data frame and minusing the variable uri
df <- select(df,-track_album_id)
```

## track_album_release year

The release year is in YYYY-MM-DD. To make the analysis easier I will turn this variable into YYYY

```
#Set of functions to deal with dates in an easier way
library(lubridate)
```

```
## Loading required package: timechange
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
# use a for loop to go through each value
for (i in 1:length(df$track_album_release_date)){
  # check if the value is a date in the DD/MM/YYYY format
  if (grepl("^[0-9]{4}$", df$track_album_release_date[i])) {
  #Adding a 01-01 to release years that just have YYYY so I can use the lubridate fun
ctions without getting NA's
    df$track_album_release_date[i] <- paste("01","01",df$track_album_release_date[i],
sep = "/")
  }
}


# Convert the dates to Date objects
dates <- as.Date(df$track_album_release_date, format = "%d/%m/%Y")

# Extract the year from the Date objects
df$track_album_release_date <- format(dates, "%Y")

#Turning years into factor variables
df$track_album_release_date <- factor(df$track_album_release_date)

#Analysing structure
str(df$track_album_release_date)
```

```
##  Factor w/ 63 levels "1957","1958",..: 62 62 62 62 62 62 62 62 62 62 ...
```

## playlist_id

```
#selecting the data frame and minusing the variable uri
df <- select(df,-playlist_id)
```

## mode

mode is a categorical variable so we will let `0 = Minor` and `1 = Major` how ever when we look at the structure again R will set `0` to `1` and `1` to `2` with `1 = Minor` and `2 = Major`

```
#Setting the variable mode as a factor with categories major and minor
df$mode <- factor(df$mode, labels = c("Minor", "Major"))
str(df$mode)
```

```
##  Factor w/ 2 levels "Minor","Major": 2 2 1 2 2 2 1 1 2 2 ...
```

## playlist_genre

`playlist_genre` can be categorised into factors which I will do

```
#Used to eliminate or delete the duplicate values or the rows present in the vector
unique(df$playlist_genre)
```

```
## [1] "pop"   "rap"   "rock"  "latin" "r&b"   "edm"
```

```
#Categorising playlist_genre variable
df$playlist_genre <- factor(df$playlist_genre, labels = c("pop","rap","rock","latin",
"r&b","edm"))
str(df$playlist_genre)
```

```
##  Factor w/ 6 levels "pop","rap","rock",..: 3 3 3 3 3 3 3 3 3 3 ...
```

## playlist_subgenre

`playlist_subgenre` can be categorised into factors which I will do

```
unique(df$playlist_subgenre)
```

```
##  [1] "dance pop"                   "post-teen pop"
##  [3] "electropop"                  "indie poptimism"
##  [5] "hip hop"                     "southern hip hop"
##  [7] "gangster rap"                "trap"
##  [9] "album rock"                  "classic rock"
## [11] "permanent wave"              "hard rock"
## [13] "tropical"                    "latin pop"
## [15] "reggaeton"                   "latin hip hop"
## [17] "urban contemporary"          "hip pop"
## [19] "new jack swing"              "neo soul"
## [21] "electro house"               "big room"
## [23] "pop edm"                     "progressive electro house"
```

```
#Categorising playlist_subgenre variable
df$playlist_subgenre <- factor(df$playlist_subgenre)
str(df$playlist_genre)
```

```
##  Factor w/ 6 levels "pop","rap","rock",..: 3 3 3 3 3 3 3 3 3 3 ...
```

## key

I will remove key as I am not educated enough in music do perform any analysis in terms of keys and scales etc…

```
#minusing the key variable
df <- select(df,-key)
```

## tempo

We will change the tempo name to BPM as it is measured in ‘Beats per Minute’ (BPM)

```
df %>%
  #Appending the name 'BPM' to the variable 'tempo'
  rename("BPM" = "tempo") -> df
```

## duration_ms

The song duration is in milliseconds we will change this to minutes and seconds, it is easier to read and that is the usual convention in music apps. We will also change the variable name from `duration_ms` to `duration(secs)`

```
#Convert input in any one of character, integer, numeric, factor, or ordered type int
o 'POSIXct' (or 'Date') objects
library(anytime)

# as. POSIXct stores both a date and time with an associated time zone. The default t
ime zone selected, is the time zone that my computer is set to
as.POSIXct(Sys.Date()) + df$duration_ms/ 1000
df$duration_ms  <- format( as.POSIXct(Sys.Date()) +
                            df$`duration_ms`  /1000,"%M:%S")

#Changing the name of the column
df <- rename(df, "duration(secs)" = "duration_ms")
```

The times are now in MM:SS

```
#Glimpse of variable in MM:SS and mod 60
head(df$`duration(secs)`,10)
```

```
##  [1] "03:14" "02:42" "02:56" "02:49" "03:09" "02:43" "03:07" "03:27" "03:13"
## [10] "04:13"
```

Now lets look at our data again

```
str(df)
```

```
## tibble [32,833 × 20] (S3: tbl_df/tbl/data.frame)
##  $ id                      : int [1:32833] 1 2 3 4 5 6 7 8 9 10 ...
##  $ track_name              : chr [1:32833] "I Don't Care (with Justin Bieber) - Lo
ud Luxury Remix" "Memories - Dillon Francis Remix" "All the Time - Don Diablo Remix"
"Call You Mine - Keanu Silva Remix" ...
##  $ track_artist            : chr [1:32833] "Ed Sheeran" "Maroon 5" "Zara Larsson"
"The Chainsmokers" ...
##  $ track_popularity        : int [1:32833] 66 67 70 60 69 67 62 69 68 67 ...
##  $ track_album_name        : chr [1:32833] "I Don't Care (with Justin Bieber) [Lou
d Luxury Remix]" "Memories (Dillon Francis Remix)" "All the Time (Don Diablo Remix)"
"Call You Mine - The Remixes" ...
##  $ track_album_release_date: Factor w/ 63 levels "1957","1958",..: 62 62 62 62 62
62 62 62 62 62 ...
##  $ playlist_name           : chr [1:32833] "Pop Remix" "Pop Remix" "Pop Remix" "Po
p Remix" ...
##  $ playlist_genre          : Factor w/ 6 levels "pop","rap","rock",..: 3 3 3 3 3 3
3 3 3 3 ...
##  $ playlist_subgenre       : Factor w/ 24 levels "album rock","big room",..: 4 4 4
4 4 4 4 4 4 4 ...
##  $ danceability            : num [1:32833] 0.748 0.726 0.675 0.718 0.65 0.675 0.44
9 0.542 0.594 0.642 ...
##  $ energy                  : num [1:32833] 0.916 0.815 0.931 0.93 0.833 0.919 0.85
6 0.903 0.935 0.818 ...
##  $ loudness                : num [1:32833] -2.63 -4.97 -3.43 -3.78 -4.67 ...
##  $ mode                    : Factor w/ 2 levels "Minor","Major": 2 2 1 2 2 2 1 1 2
2 ...
##  $ speechiness             : num [1:32833] 0.0583 0.0373 0.0742 0.102 0.0359 0.127
0.0623 0.0434 0.0565 0.032 ...
##  $ acousticness            : num [1:32833] 0.102 0.0724 0.0794 0.0287 0.0803 0.079
9 0.187 0.0335 0.0249 0.0567 ...
##  $ instrumentalness        : num [1:32833] 0.00 4.21e-03 2.33e-05 9.43e-06 0.00 0.
00 0.00 4.83e-06 3.97e-06 0.00 ...
##  $ liveness                : num [1:32833] 0.0653 0.357 0.11 0.204 0.0833 0.143 0.
176 0.111 0.637 0.0919 ...
##  $ valence                 : num [1:32833] 0.518 0.693 0.613 0.277 0.725 0.585 0.1
52 0.367 0.366 0.59 ...
##  $ BPM                     : num [1:32833] 122 100 124 122 124 ...
##  $ duration(secs)          : chr [1:32833] "03:14" "02:42" "02:56" "02:49" ...
```

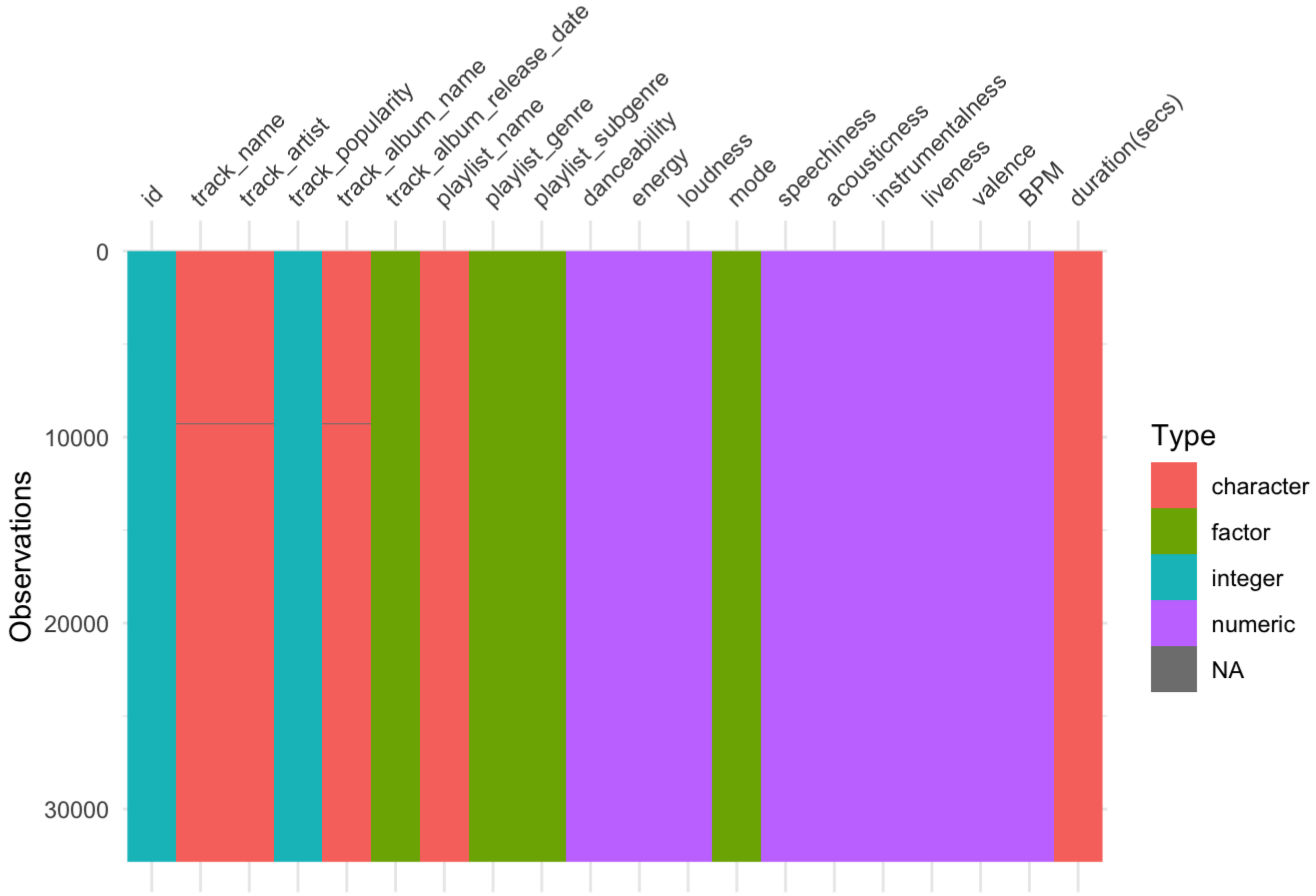
We now have a tibble with 32,828 observations and this time only 20 variables

We can graph this structure too

```
#The visdat package provides visualisations of an entire dataframe at once
library(visdat)
```

```
visdat::vis_dat(df,sort_type = FALSE)
```

```
## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## Please use `gather()` instead.
```



## NA Values

It would make sense to remove rows with any `NA` values in them

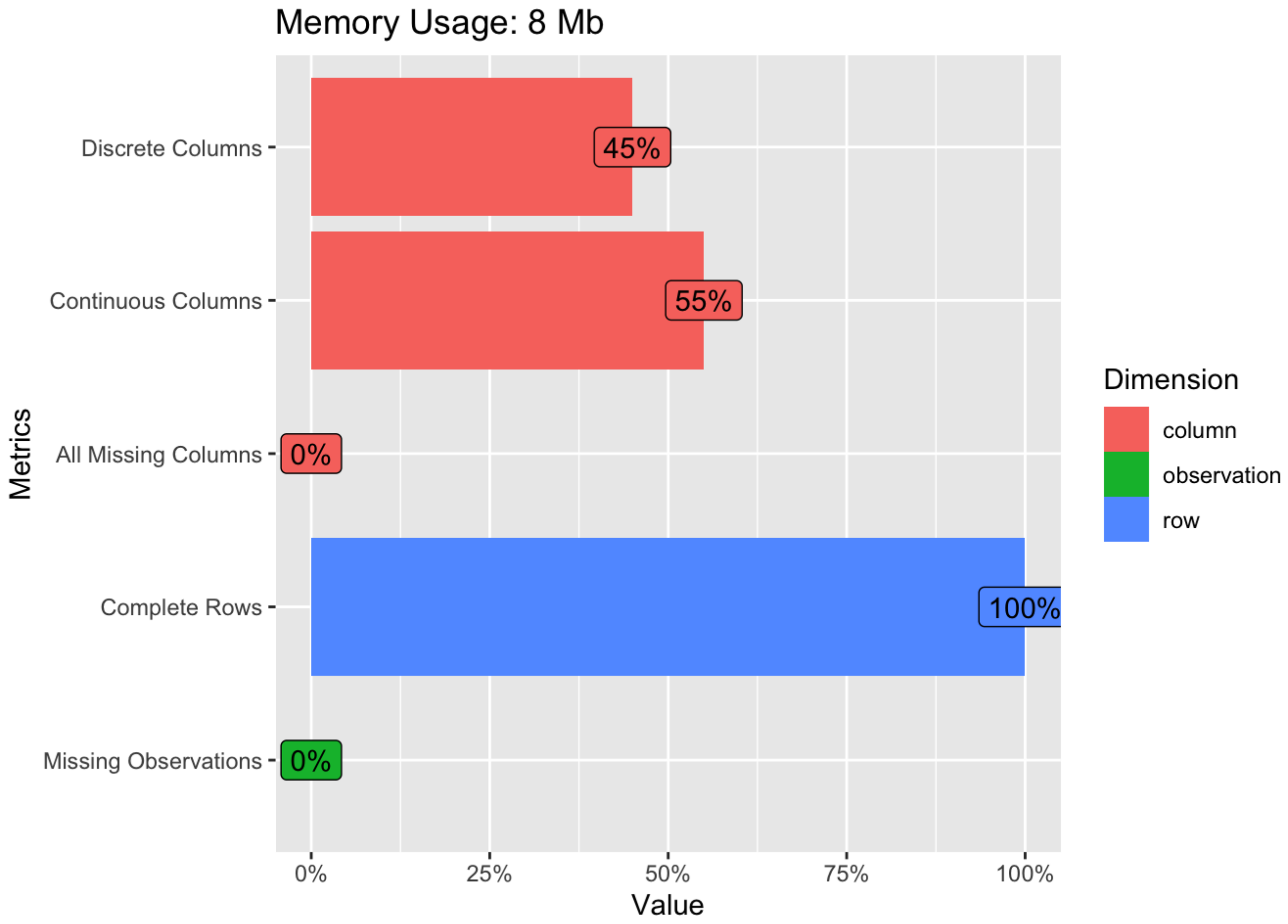
```
df <- print(df[complete.cases(df), ] )
```

```
## # A tibble: 32,797 × 20
##       id track…¹ track…² track…³ track…⁴ track…⁵ playl…⁶ playl…⁷ playl…⁸ dance…⁹
##    <int> <chr>   <chr>     <int> <chr>   <fct>   <chr>   <fct>   <fct>     <dbl>
##  1     1 I Don'… Ed She…      66 I Don'… 2019    Pop Re… rock    dance …   0.748
##  2     2 Memori… Maroon…      67 Memori… 2019    Pop Re… rock    dance …   0.726
##  3     3 All th… Zara L…      70 All th… 2019    Pop Re… rock    dance …   0.675
##  4     4 Call Y… The Ch…      60 Call Y… 2019    Pop Re… rock    dance …   0.718
##  5     5 Someon… Lewis …      69 Someon… 2019    Pop Re… rock    dance …   0.65 
##  6     6 Beauti… Ed She…      67 Beauti… 2019    Pop Re… rock    dance …   0.675
##  7     7 Never … Katy P…      62 Never … 2019    Pop Re… rock    dance …   0.449
##  8     8 Post M… Sam Fe…      69 Post M… 2019    Pop Re… rock    dance …   0.542
##  9     9 Tough … Avicii       68 Tough … 2019    Pop Re… rock    dance …   0.594
## 10    10 If I C… Shawn …      67 If I C… 2019    Pop Re… rock    dance …   0.642
## # … with 32,787 more rows, 10 more variables: energy <dbl>, loudness <dbl>,
## #   mode <fct>, speechiness <dbl>, acousticness <dbl>, instrumentalness <dbl>,
## #   liveness <dbl>, valence <dbl>, BPM <dbl>, `duration(secs)` <chr>, and
## #   abbreviated variable names ¹track_name, ²track_artist, ³track_popularity,
## #   ⁴track_album_name, ⁵track_album_release_date, ⁶playlist_name,
## #   ⁷playlist_genre, ⁸playlist_subgenre, ⁹danceability
```

## Data Exploration:

A useful thing to look at which we didn't talk too much about in the course was memory consumption. This is quite a big data set so memory consumption would be something worth keeping an eye on

```
#automate data exploration and treatment
library(DataExplorer)
plot_intro(df)
```



The total memory usage is 8Mb with the rows taking up the most amount of memory.

This can also show if you have any missing data. If we did completed rows would not be 100% and discrete and continuous columns both add to give 100%. Missing columns and observations is also 0%

Now we will look at answering some of my questions previously.

I wanted to find mean of song duration

```
#Converting times into seconds for easier calculation
toSeconds <- function(x){

  #stop if the input is not a string in H:M:S
  if (!is.character(x)) stop("x must be a character string of the form H:M:S")
  #If x is <= 0 we return the input as you can have negative or 0 time
  if (length(x)<=0)return(x)

  #The function knows that if you put 1 digit in it is seconds
  # 2 digits is minutes:seconds
  # 3 digits is houra:minutes:seconds
  unlist(lapply(x, function(i){
    i <- as.numeric(strsplit(i,':',fixed=TRUE)[[1]])

    if (length(i) == 3) #Hours
      i[1]*3600 + i[2]*60 + i[3]

    else if (length(i) == 2) #Mins
      i[1]*60 + i[2]

    else if (length(i) == 1) #Seconds
      i[1]
  }
  )
  )
}


#dividing and rounding seconds by 60 to get minutes and seconds
mean(toSeconds(df$`duration(secs)`)) / 60
```

```
## [1] 3.755009
```

As we can see the mean for song duration is 3 minutes and 76 seconds. We can run this in a function called `timecon` which will convert the duration to minutes and seconds if the seconds are greater than 60

```
timecon <- function(x){

  # seperating the whole number and decimal to get the decimal part
  l <- x - floor(x)

  #If the decimal part is <= 60 we return the number as is because the seconds are st
ill between 1 and 60 so no conversion needed
  if (l <= 0.6){return(x)}

  #Otherwise we multiply the decimal part by 100 and divide by 60 adding on the while
number
  else {(round((x - floor(x)) * 100,0) / 60) + floor(x)}

}


round(timecon(3.76),2)
```

```
## [1] 4.27
```

The actual mean is 4 minutes and 27 seconds.

I wanted to look at some information about the dataset's genres

```
df %>%
  count(genre = playlist_genre) -> gencount

gencount %>%
   arrange(desc(n), gencount) %>%
    rename(count = n) -> gencount
```

```
knitr::kable(gencount, col.names = gsub("", "", names(gencount)))
```

| genre | count |
|:---|---:|
| pop | 6043 |
| r&b | 5742 |
| rock | 5505 |
| latin | 5427 |
| rap | 5153 |
| edm | 4927 |

Unsurprisingly, pop has the most songs in the dataset followed by r&b, rock, latin, rap and finally edm. How about subgenres?

Let's pick the top 10

```
df %>%
  count(playlist_subgenre) -> subcount

subcount %>%
   arrange(desc(n), subcount) %>%
  rename(count = n) -> subcount
```

```
knitr::kable(head(subcount, 10), col.names = gsub("", "", names(subcount)))
```

| playlist_subgenre | count |
|:---|---:|
| progressive electro house | 1809 |
| southern hip hop | 1673 |
| indie poptimism | 1672 |
| latin hip hop | 1655 |
| neo soul | 1636 |

| playlist_subgenre | count |
|---|---:|
| pop edm | 1517 |
| electro house | 1511 |
| hard rock | 1482 |
| gangster rap | 1456 |
| electropop | 1406 |

It is interesting that edm is the lowest of the genres but progressive electro house is the highest of the subgenres. Which indicates most of edm’s subgenres fall under progressive electro house.

Personally, I like live music even listening to it on recording. I want to see what percentage of is above 0.8 (In the music attributes it is stated anything greater than 0.8 is normally live)

```
df %>%
  count(liveness >= 0.8) %>%
  rename(count = n) -> livecount

livecount$count[2] / length(df$liveness)
```

```
## [1] 0.0100619
```

Only 1% of tracks have 0.8 and over for the amount of liveness. In other words only 1% of tracks are live.

Who has the highest BPM?

```
df %>%
  filter(BPM == max(BPM)) -> highbpm

highbpm %>%
  select(track_name, track_artist, BPM) -> highbpm

knitr::kable(head(highbpm, 10), col.names = gsub("", "", names(highbpm)))
```

| track_name | track_artist | BPM |
|---|---|---:|
| Dope’s Gotta Hold On Me (feat. Ese Rich Roc) | Spanish F.L.Y. | 239.44 |

“Dope’s gotta hold on me by Spanish F.L.Y” has the highest BPM with a result of 239.44

Who has the lowest danceability?

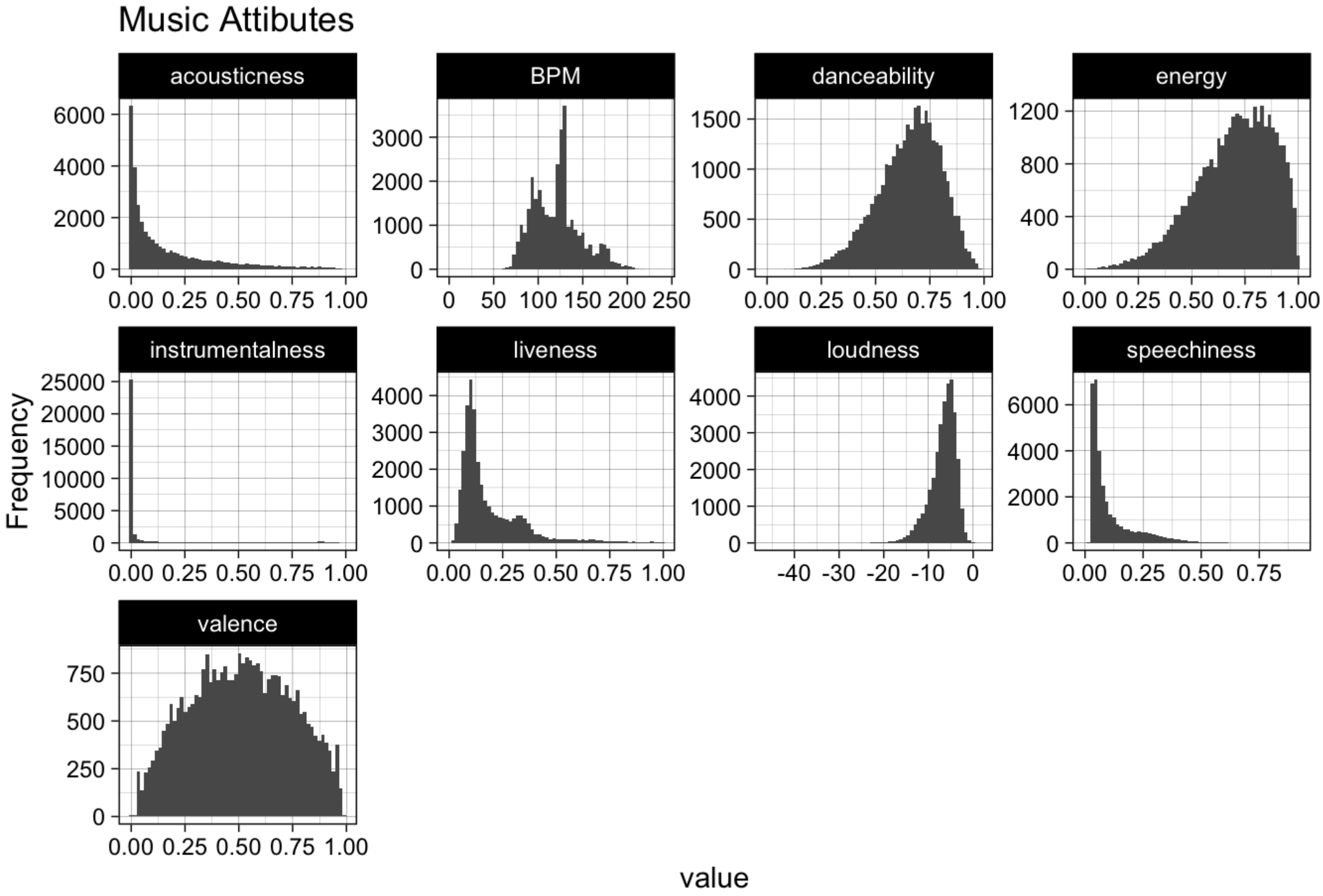
```
df %>%
  filter(danceability == min(danceability)) -> lowdnce

lowdnce %>%
  select(track_name, track_artist, danceability) -> lowdnce

knitr::kable(head(lowdnce, 10), col.names = gsub("", "", names(lowdnce)))
```

| track_name | track_artist | danceability |
|---|---|---:|
| Hi, How’re You Doin’? | DREAMS COME TRUE | 0 |

It is “Hi, How’re You Doin” by “DREAMS COME TRUE” with 0

Next, we can graph all the music attributes and look at them at a glimpse

```
library(ggthemes)
#I am also using the Data Explorer library here too but I have called it previously
plot_histogram(
  df[,10:20], #Only variables 10 - 20
  geom_histogram_args = list(bins = 60L),
  scale_x = "continuous",
  title = 'Music Attibutes',
  ggtheme = theme_linedraw(),
  theme_config = list(),
  nrow = 4L,
  ncol = 4L,
  parallel = T
)
```

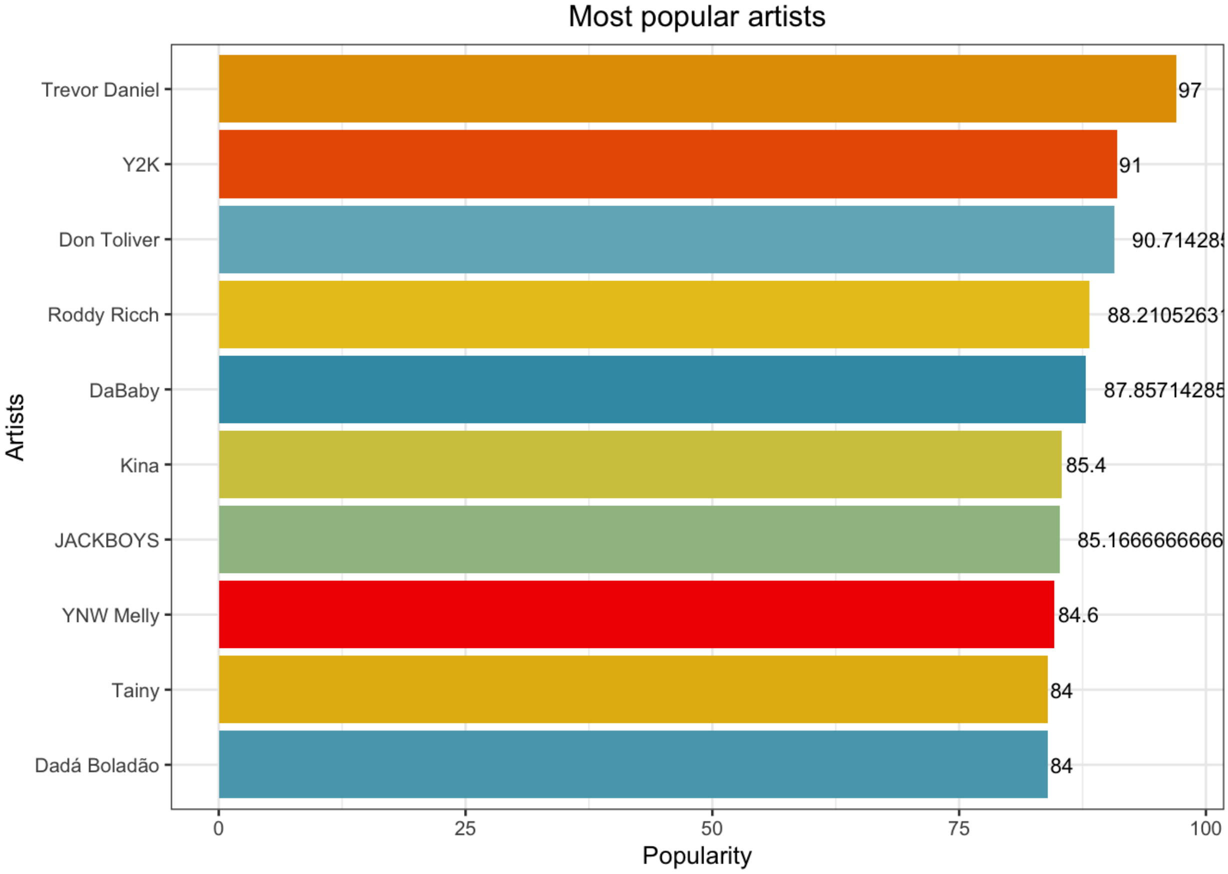


From the graphs we can say:

- Acousticness is rightly skewed
- Valence is normally distributed
- Speechiness in songs are not as popular, people prefer less words
- High energy songs are popular
- Most songs have a loudness between 6-10 dBs

Next I thought it would be interesting to look at the most popular artists in terms of average popularity

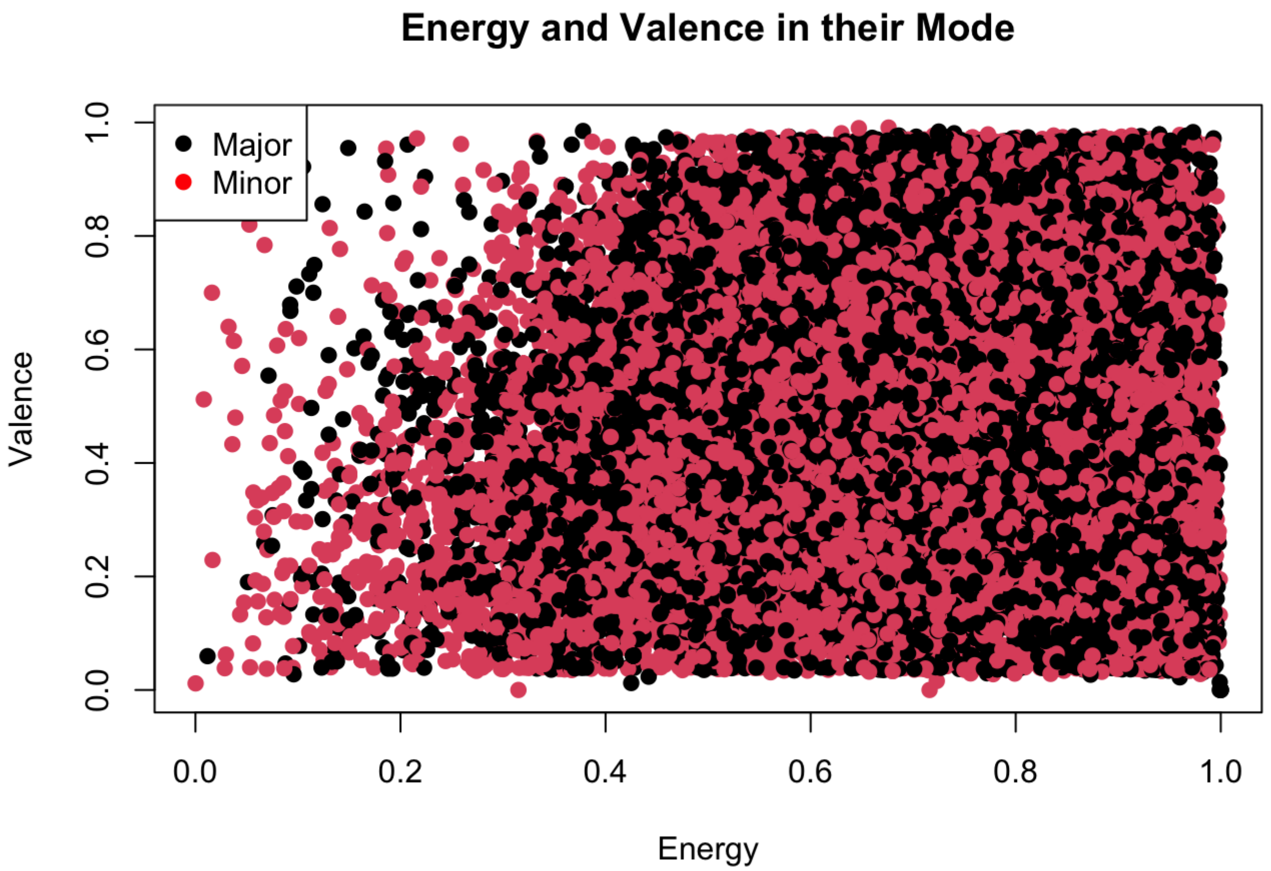
```
#Subset of artists with the amount of songs they have in the dataset
df %>%
  count(track_artist) -> artcount

#Grouping by artist and adding up their total popularity in a new column
df %>%
  group_by(track_artist) %>%
  summarise(Total_Popularity = sum(track_popularity)) -> pop

# Adding the number of tracks we original got to our new subset
pop %>%
  mutate(No_of_tracks = artcount$n) -> pop

# a new column averaging the popularity dividing total popularity by amount of songs
pop %>%
  mutate(average_popularity = Total_Popularity / No_of_tracks) -> popavg



library(wesanderson) #Color pallette package
col2 = c(wes_palette('Zissou1', 10, type = "continuous")) #Setting up a gradual color
pallette

popavg %>%
  top_n(10, average_popularity) %>% #Top 10 in average popularity
  ggplot(aes(x = reorder(track_artist,+ average_popularity), y = average_popularity))
+ #setting up axes
  geom_bar(stat = 'identity', fill = col2) + #Using color pallette
  coord_flip() + #Flipping the axes so the names of artists arent at the bottom
  theme_bw(base_size=10) +
  labs(y="Popularity", x="Artists") +
  ggtitle("Most popular artists") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_text(aes(label = average_popularity), hjust=-0.1, size=3) #Adjusting position
```



On average the top artists are quite close with a range from 84 - 97 but Trevor Daniels takes the lead by a good bit.

I thought I would look at major and minor modes. In music people say major modes sound more positive while minor modes tend to be more sad. I decided to take the modes and plot them against valence which is how positive a song is and energy.

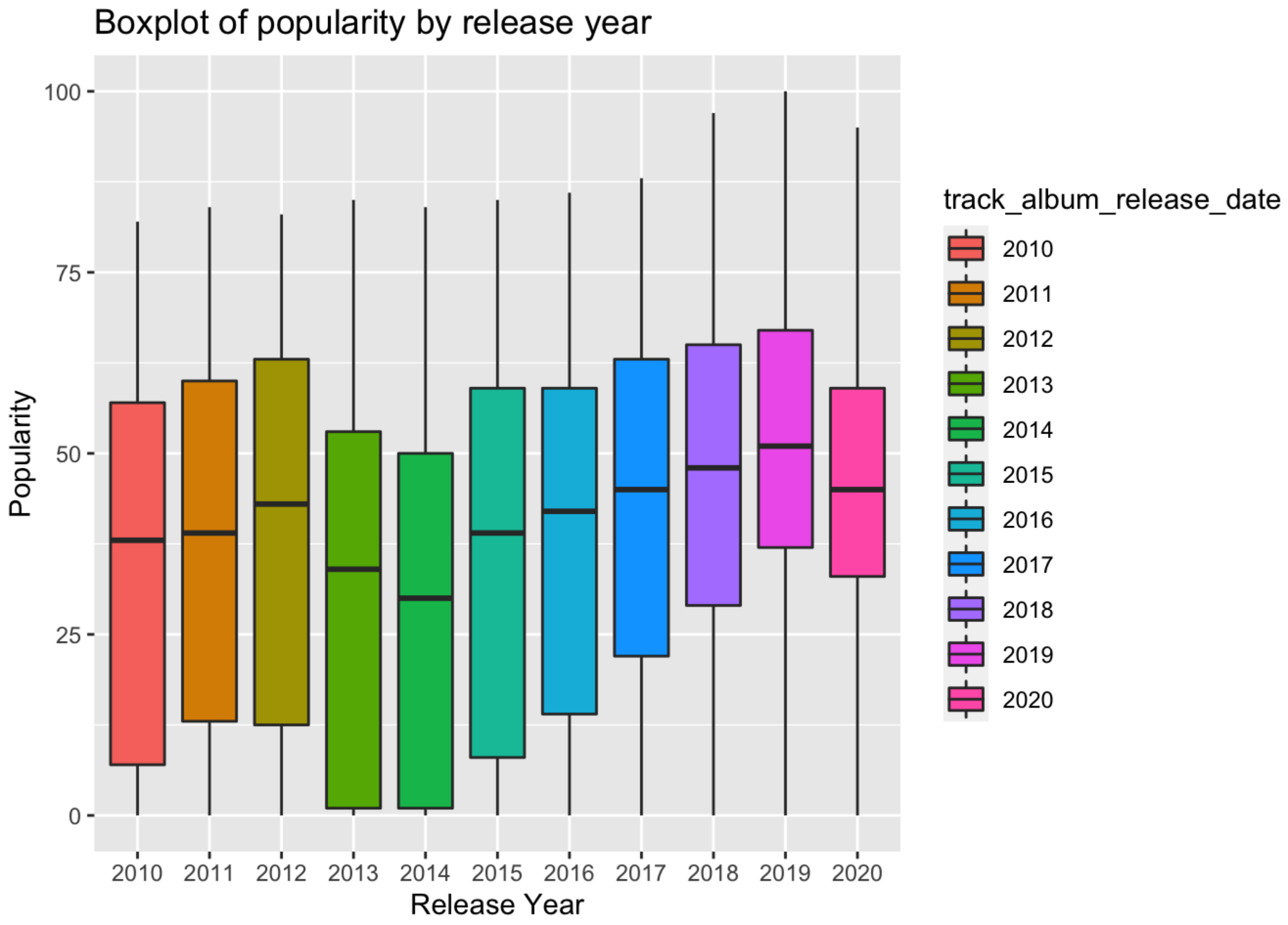
```
plot(x = df$energy, y = df$valence,
     pch = 19,
     col = df$mode,
     xlab = 'Energy',
     ylab = 'Valence',
     main = 'Energy and Valence in their Mode')
legend("topleft",
       legend = c('Major', 'Minor'),
       pch = 19,
       col = c('black', 'red'))
```



Generally it is quite evenly spread I would argue that major does not mean happy and minor does not mean sad. On the left there are some outliers, they are mostly minor with a high valence but just a low energy. This does not necessarily mean the modes determine the positivity of the songs.

I wanted to look at the track popularity with each year between 2010 - 2020, for this I did a boxplot

```
#Picking out release dates with years between 2010 - 2020
OOs <- df %>%
  filter(track_album_release_date %in% c("2010", "2011", "2012", "2013", "2014", "201
5", "2016", "2017", "2018", "2019", "2020"))


#Boxplot
OOs %>%
  ggplot(aes(x = track_album_release_date, y = track_popularity, fill = track_album_r
elease_date)) +
  geom_boxplot() +
  labs(y="Popularity", x="Release Year") +
  ggtitle("Boxplot of popularity by release year")
```



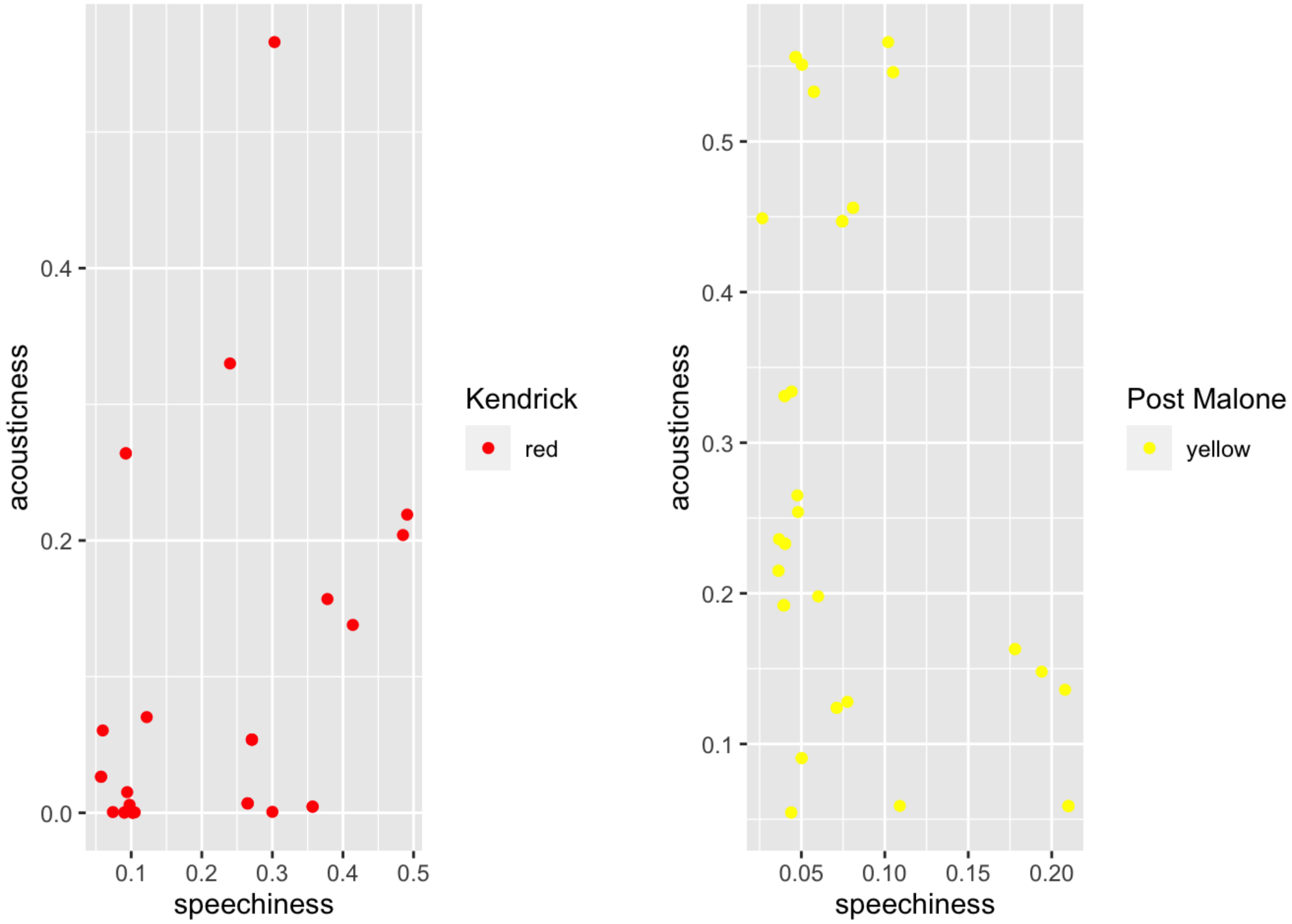
It is close between 2019 and 2012 but judging from the graph 2019 seems to be the most popular year with one of the smallest variances.

In the Rap/Hip-Hop world there are people that dominate. Two of my all time favorites are Kendrick Lamar and Post Malone. Kendrick is considered to be a lyrical genius constantly creating music and winning many awards. Post Malone is similar. Post Malone gets grouped into the Rap/Hip Hop genre all the time however there is a big debate whether he belongs there or not. I thought it would be interesting to compare acousticness, speechiness and BPM between the 'King of Compton' and Post Malone.

```
#Gathering data with the artist being Kendrick
df %>%
  filter(track_artist == 'Kendrick Lamar') -> kendrick

#Gathering data with the artist being Post Malone
df %>%
  filter(track_artist == 'Post Malone') -> posty
```

Now we have each artist's data lets look at their graphs

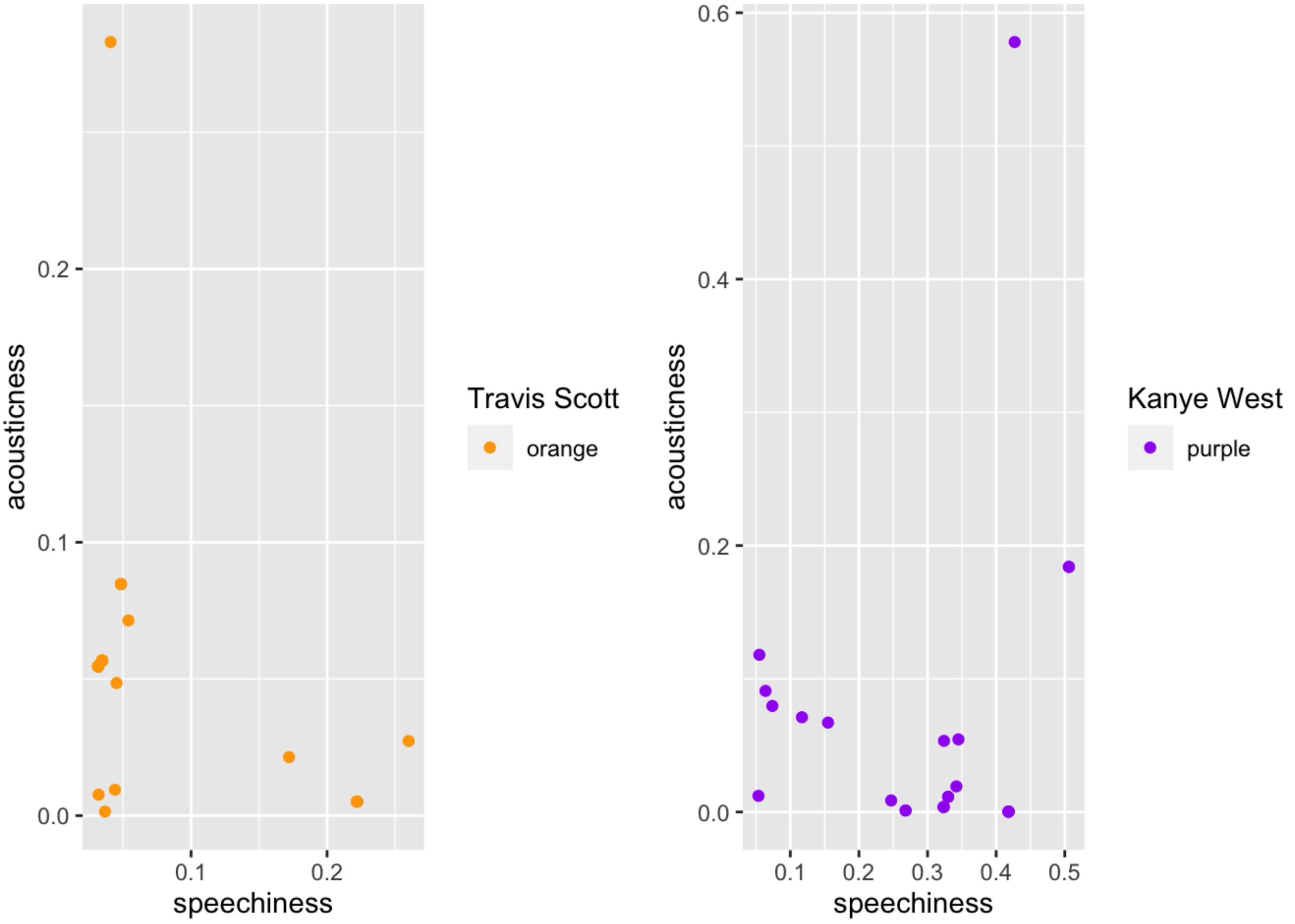
```
#Plotting speechiness against acousticness for Kendrick
kendrick %>%
  ggplot(aes(x = speechiness, y = acousticness)) +
  geom_point(aes(color = 'red')) +
  scale_color_manual(name = "Kendrick", values = 'red') -> kenplot


#Plotting speechiness against acousticness for Post
posty %>%
  ggplot(aes(x = speechiness, y = acousticness)) +
  geom_point(aes(color = "yellow")) +
  scale_color_manual(name = "Post Malone", values = 'yellow') -> postplot




#Displaying the 2 plots against each other
plot_grid(kenplot, postplot, ncol=2,   align = "v", axis = "lr")
```



So from the graphs, we can see Kendrick is more lyrical and Post Malone is more accoustic. Lyrical songs are a characteristic of rap and although rappers can use accoustics in their tracks however it is not as common as rock or pop.

We are only comparing Post Malone to one rapper. Let us look at some more rappers with these characteristics. We will look at Kanye West and Travis Scott, two massive names in rap

```
#Gathering data with the artist being Drake
df %>%
  filter(track_artist == 'Travis Scott') -> travis


travis %>%
  ggplot(aes(x = speechiness, y = acousticness)) +
  geom_point(aes(color = 'orange')) +
  scale_color_manual(name = "Travis Scott", values = 'orange') -> traplot


#Gathering data with the artist being Kanye
df %>%
  filter(track_artist == 'Kanye West') -> kanye

kanye %>%
  ggplot(aes(x = speechiness, y = acousticness)) +
  geom_point(aes(color = 'purple')) +
  scale_color_manual(name = "Kanye West", values = 'purple') -> kanyeplot


plot_grid(traplot, kanyeplot,
          ncol=2,   align = "v", axis = "lr")
```



As you can see these are similar to Kendrick. There is a common trend here one that Post Malone does not follow.

I think it is fair to say based on these graphs Post Malone on paper at least shouldn’t be considered a rapper. I think he is more in another genre. I think his accousticness is too much for the rap genre. On a personal note I do agree, I feel Post Malone is very melodic compared to rappers in industry today.

This concludes my analysis of the Spotify dataset. I picked this as I love music and wanted to look at some statistics beyond just popularity or who has the most songs. I think we answered our questions at the beginning plus a few extra in between.

# R Packages

For the R Packages section of the project I thought it would be interesting to look into packages that enable R to access a database and retrieve data.

In this section I will look at 2 packages `DBI` and `sqldf`

## DBI:

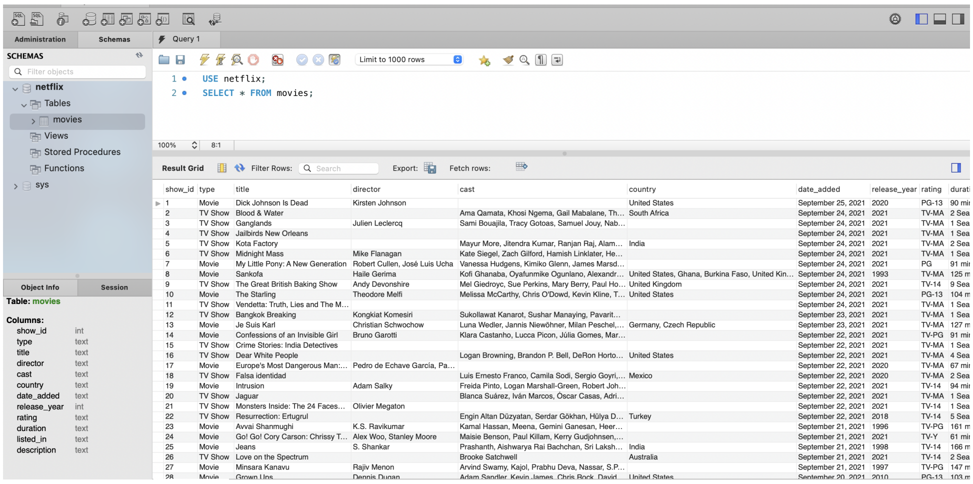
In the world today there is so much data and Excel and CSV is not a sufficient way to store massive amounts of data anymore so data is stored in databases. Data analysts retrieve this data from databases to begin their analysis. In order to do this in R we use the `DBI` package.

The `DBI` package is a database interface definition for communication between R and relational database management systems

```
library(DBI) #Calling DBI
```

I will connect my SQL database. My database is called “netflix” and I have a table called “movies” within that database. Within that table I imported a dataset called netflix_dataset.csv (https://app.datacamp.com/workspace/w/3ae668c3-7d26-4c7b-852e-60f39a69a2d7/edit?file=notebook.ipynb) which I imported in my database prior.

Below is a screenshot of my database in the SQL Workbench environment



```
USE netflix;
SELECT * FROM movies;
```

Table: movies

Columns:

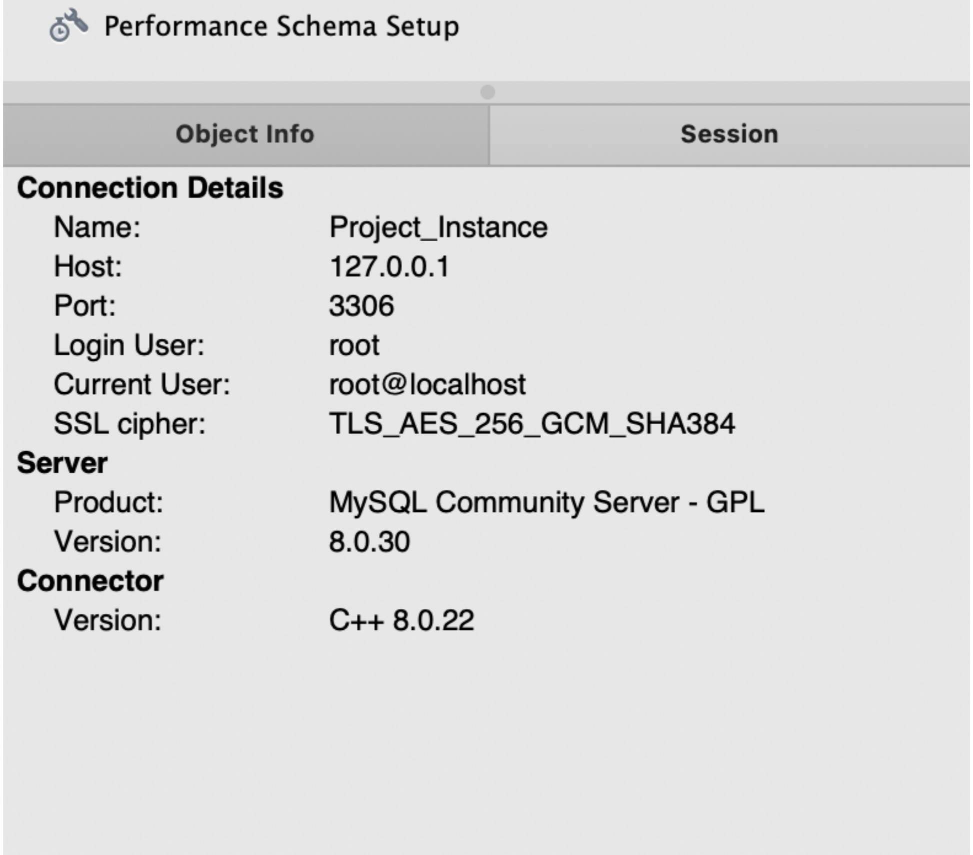
| | |
|---|---|
| show_id | int |
| type | text |
| title | text |
| director | text |
| cast | text |
| country | text |
| date_added | text |
| release_year | int |
| rating | text |
| duration | text |
| listed_in | text |
| description | text |

| show_id | type | title | director | cast | country | date_added | release_year | rating | durati |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Movie | Dick Johnson Is Dead | Kirsten Johnson | | United States | September 25, 2021 | 2020 | PG-13 | 90 mi |
| 2 | TV Show | Blood & Water | | Ama Qamata, Khosi Ngema, Gail Mabalane, Th… | South Africa | September 24, 2021 | 2021 | TV-MA | 2 Sea |
| 3 | TV Show | Ganglands | Julien Leclercq | Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nab… | | September 24, 2021 | 2021 | TV-MA | 1 Sea |
| 4 | TV Show | Jailbirds New Orleans | | | | September 24, 2021 | 2021 | TV-MA | 1 Sea |
| 5 | TV Show | Kota Factory | | Mayur More, Jitendra Kumar, Ranjan Raj, Alam… | India | September 24, 2021 | 2021 | TV-MA | 2 Sea |
| 6 | TV Show | Midnight Mass | Mike Flanagan | Kate Siegel, Zach Gilford, Hamish Linklater, He… | | September 24, 2021 | 2021 | TV-MA | 1 Sea |
| 7 | Movie | My Little Pony: A New Generation | Robert Cullen, José Luis Ucha | Vanessa Hudgens, Kimiko Glenn, James Marsd… | | September 24, 2021 | 2021 | PG | 91 mi |
| 8 | Movie | Sankofa | Haile Gerima | Kofi Ghanaba, Oyafunmike Ogunlano, Alexandr… | United States, Ghana, Burkina Faso, United Kin… | September 24, 2021 | 1993 | TV-MA | 125 m |
| 9 | TV Show | The Great British Baking Show | Andy Devonshire | Mel Giedroyc, Sue Perkins, Mary Berry, Paul Ho… | United Kingdom | September 24, 2021 | 2021 | TV-14 | 9 Sea |
| 10 | Movie | The Starling | Theodore Melfi | Melissa McCarthy, Chris O'Dowd, Kevin Kline, T… | United States | September 24, 2021 | 2021 | PG-13 | 104 m |
| 11 | TV Show | Vendetta: Truth, Lies and The M… | | | | September 24, 2021 | 2021 | TV-MA | 1 Sea |
| 12 | TV Show | Bangkok Breaking | Kongkiat Komesiri | Sukollawat Kanarot, Sushar Manaying, Pavarit… | | September 23, 2021 | 2021 | TV-MA | 1 Sea |
| 13 | Movie | Je Suis Karl | Christian Schwochow | Luna Wedler, Jannis Niewöhner, Milan Peschel,… | Germany, Czech Republic | September 23, 2021 | 2021 | TV-MA | 127 m |
| 14 | Movie | Confessions of an Invisible Girl | Bruno Garotti | Klara Castanho, Lucca Picon, Júlia Gomes, Mar… | | September 22, 2021 | 2021 | TV-PG | 91 mi |
| 15 | TV Show | Crime Stories: India Detectives | | | | September 22, 2021 | 2021 | TV-MA | 1 Sea |
| 16 | TV Show | Dear White People | | Logan Browning, Brandon P. Bell, DeRon Horto… | United States | September 22, 2021 | 2021 | TV-MA | 4 Sea |
| 17 | Movie | Europe's Most Dangerous Man:… | Pedro de Echave García, Pa… | | | September 22, 2021 | 2020 | TV-MA | 67 mi |
| 18 | TV Show | Falsa identidad | | Luis Ernesto Franco, Camila Sodi, Sergio Goyri,… | Mexico | September 22, 2021 | 2020 | TV-MA | 2 Sea |
| 19 | Movie | Intrusion | Adam Salky | Freida Pinto, Logan Marshall-Green, Robert Joh… | | September 22, 2021 | 2021 | TV-14 | 94 mi |
| 20 | TV Show | Jaguar | | Blanca Suárez, Iván Marcos, Óscar Casas, Adri… | | September 22, 2021 | 2021 | TV-MA | 1 Sea |
| 21 | TV Show | Monsters Inside: The 24 Faces… | Olivier Megaton | | | September 22, 2021 | 2021 | TV-14 | 1 Sea |
| 22 | TV Show | Resurrection: Ertugrul | | Engin Altan Düzyatan, Serdar Gökhan, Hülya D… | Turkey | September 22, 2021 | 2018 | TV-14 | 5 Sea |
| 23 | Movie | Avvai Shanmughi | K.S. Ravikumar | Kamal Hassan, Meena, Gemini Ganesan, Heer… | | September 21, 2021 | 1996 | TV-PG | 161 m |
| 24 | Movie | Go! Go! Cory Carson: Chrissy T… | Alex Woo, Stanley Moore | Maisie Benson, Paul Killam, Kerry Gudjohnsen,… | | September 21, 2021 | 2021 | TV-Y | 61 mi |
| 25 | Movie | Jeans | S. Shankar | Prashanth, Aishwarya Rai Bachchan, Sri Laksh… | India | September 21, 2021 | 1998 | TV-14 | 166 m |
| 26 | TV Show | Love on the Spectrum | | Brooke Satchwell | Australia | September 21, 2021 | 2021 | TV-14 | 2 Sea |
| 27 | Movie | Minsara Kanavu | Rajiv Menon | Arvind Swamy, Kajol, Prabhu Deva, Nassar, S.P… | | September 21, 2021 | 1997 | TV-PG | 147 m |
| 28 | Movie | Grown Ups | Dennis Dugan | Adam Sandler, Kevin James, Chris Rock, David | United States | September 20, 2021 | 2010 | PG-13 | 103 m |

In order to do anything R and the SQL database need to communicate in some way in order for me to interact with the database through R. Below is how we do it

```
# Connecting to the MySQL database:
con <- dbConnect(RMySQL::MySQL(), #con will be the variable name I will use to access
my db
                 dbname = "netflix",
                 host = "localhost",
                 port = 3306,
                 user = "root",
                 password = "Pa88word@1")
```

I found these details in SQL Workbench here:



Performance Schema Setup

Object Info | Session

**Connection Details**

| | |
|---|---|
| Name: | Project_Instance |
| Host: | 127.0.0.1 |
| Port: | 3306 |
| Login User: | root |
| Current User: | root@localhost |
| SSL cipher: | TLS_AES_256_GCM_SHA384 |

**Server**

| | |
|---|---|
| Product: | MySQL Community Server - GPL |
| Version: | 8.0.30 |

**Connector**

| | |
|---|---|
| Version: | C++ 8.0.22 |

Now my database is connected to R I can use these packages to interact with my database and start writing queries.

First I will call all the tables in my database in this case we only have 1 which is “movies”

```
# Getting a list of tables from the database using object 'con'
dbListTables(con)
```

```
## [1] "movies"
```

Next, I will read the first 3 rows of data from the table “movies”

```
#Reading table query associating database, table
head(dbReadTable(con, "movies"), 3)  #Only reading the first 3 rows
```

```
##   show_id    type                title        director
## 1       1   Movie Dick Johnson Is Dead Kirsten Johnson
## 2       2 TV Show        Blood & Water
## 3       3 TV Show            Ganglands Julien Leclercq
##
cast
## 1
## 2 Ama Qamata, Khosi Ngema, Gail Mabalane, Thabang Molaba, Dillon Windvogel, Natash
a Thahane, Arno Greeff, Xolile Tshabalala, Getmore Sithole, Cindy Mahlangu, Ryle De M
orny, Greteli Fincham, Sello Maake Ka-Ncube, Odwa Gwanya, Mekaila Mathys, Sandi Schul
tz, Duane Williams, Shamilla Miller, Patrick Mofokeng
## 3
Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabiha Akkari, Sofia Lesaffre, Salim Kechio
uche, Noureddine Farihi, Geert Van Rampelberg, Bakary Diombera
##         country         date_added release_year rating  duration
## 1 United States September 25, 2021         2020  PG-13    90 min
## 2  South Africa September 24, 2021         2021  TV-MA 2 Seasons
## 3               September 24, 2021         2021  TV-MA  1 Season
##                                                           listed_in
## 1                                                     Documentaries
## 2               International TV Shows, TV Dramas, TV Mysteries
## 3 Crime TV Shows, International TV Shows, TV Action & Adventure
##
description
## 1 As her father nears the end of his life, filmmaker Kirsten Johnson stages his de
ath in inventive and comical ways to help them both face the inevitable.
## 2      After crossing paths at a party, a Cape Town teen sets out to prove whether
a private-school swimming star is her sister who was abducted at birth.
## 3       To protect his family from a powerful drug lord, skilled thief Mehdi and h
is expert team of robbers are pulled into a violent and deadly turf war.
```

We will write our first query. In this query we will select the amount of rows from “movies”

```
#Writing a SQL query to the database
dbGetQuery(con,'SELECT COUNT(*) FROM movies;') #This query counts every row from movi
es
```

```
##   COUNT(*)
## 1     8788
```

A very useful feature is after running a query you can then save this query into a variable take for example:

```
#saving the query into an object
tsmovies <- dbGetQuery(con,'SELECT * FROM movies WHERE director = "Toshiya Shinohar
a";')
```

I have ran a query to select all movies where the director is Toshiya Shinohara and saved it in a object called `tsmovies` . I can now use this object as we normally would in R.

```
str(tsmovies) #Looking at the structure of tsmovies
```

```
## 'data.frame':	7 obs. of  12 variables:
##  $ show_id     : int  52 53 54 55 7089 7090 7091
##  $ type        : chr  "Movie" "Movie" "Movie" "Movie" ...
##  $ title       : chr  "InuYasha the Movie 2: The Castle Beyond the Looking Glass"
"InuYasha the Movie 3: Swords of an Honorable Ruler" "InuYasha the Movie 4: Fire on t
he Mystic Island" "InuYasha the Movie: Affections Touching Across Time" ...
##  $ director    : chr  "Toshiya Shinohara" "Toshiya Shinohara" "Toshiya Shinohara"
"Toshiya Shinohara" ...
##  $ cast        : chr  "Kappei Yamaguchi, Satsuki Yukino, Mieko Harada, Koji Tsujit
ani, Houko Kuwashima, Kumiko Watanabe, Noriko Hidaka"| __truncated__ "Kappei Yamaguch
i, Satsuki Yukino, Koji Tsujitani, Houko Kuwashima, Kumiko Watanabe, Ken Narita, Akio
Otsuka, Kikuko Inoue" "Kappei Yamaguchi, Satsuki Yukino, Koji Tsujitani, Houko Kuwash
ima, Kumiko Watanabe, Noriko Hidaka, Ken Narita, "| __truncated__ "Kappei Yamaguchi,
Satsuki Yukino, Koji Tsujitani, Houko Kuwashima, Kumiko Watanabe, Kenichi Ogata, Nori
ko Hidak"| __truncated__ ...
##  $ country     : chr  "Japan" "Japan" "Japan" "Japan" ...
##  $ date_added  : chr  "September 15, 2021" "September 15, 2021" "September 15, 202
1" "September 15, 2021" ...
##  $ release_year: int  2002 2003 2004 2001 2004 2003 2002
##  $ rating      : chr  "TV-14" "TV-14" "TV-PG" "TV-PG" ...
##  $ duration    : chr  "99 min" "99 min" "88 min" "100 min" ...
##  $ listed_in   : chr  "Action & Adventure, Anime Features, International Movies"
"Action & Adventure, Anime Features, International Movies" "Action & Adventure, Anime
Features, International Movies" "Action & Adventure, Anime Features, International Mo
vies" ...
##  $ description : chr  "With their biggest foe seemingly defeated, InuYasha and his
friends return to everyday life. But the peace is s"| __truncated__ "The Great Dog De
mon beaqueathed one of the Three Swords of the Fang to each of his two sons. Now the
evil power"| __truncated__ "Ai, a young half-demon who has escaped from Horai Island
to try to help her people, returns with potential savi"| __truncated__ "A powerful de
mon has been sealed away for 200 years. But when the demon's son is awakened, the fat
e of the worl"| __truncated__ ...
```

Now looking at the data

```
tsmovies
```

```
##   show_id  type                                                      title
## 1      52 Movie  InuYasha the Movie 2: The Castle Beyond the Looking Glass
## 2      53 Movie         InuYasha the Movie 3: Swords of an Honorable Ruler
## 3      54 Movie            InuYasha the Movie 4: Fire on the Mystic Island
## 4      55 Movie        InuYasha the Movie: Affections Touching Across Time
## 5    7089 Movie           Inuyasha the Movie - L'isola del fuoco scarlatto
## 6    7090 Movie     Inuyasha the Movie - La spada del dominatore del mondo
## 7    7091 Movie InuYasha: The Movie 2: The Castle Beyond the Looking Glass
##            director
## 1 Toshiya Shinohara
## 2 Toshiya Shinohara
## 3 Toshiya Shinohara
## 4 Toshiya Shinohara
## 5 Toshiya Shinohara
## 6 Toshiya Shinohara
## 7 Toshiya Shinohara
##
cast
## 1                                 Kappei Yamaguchi, Satsuki Yukino, Mieko Harada, 
Koji Tsujitani, Houko Kuwashima, Kumiko Watanabe, Noriko Hidaka, Kenichi Ogata, Toshi
yuki Morikawa, Izumi Ogami
## 2                                                                       Kappei Yam
aguchi, Satsuki Yukino, Koji Tsujitani, Houko Kuwashima, Kumiko Watanabe, Ken Narita,
Akio Otsuka, Kikuko Inoue
## 3                                                Kappei Yamaguchi, Satsuki Yukino,
Koji Tsujitani, Houko Kuwashima, Kumiko Watanabe, Noriko Hidaka, Ken Narita, Cho, Mam
iko Noto, Nobutoshi Canna
## 4                                       Kappei Yamaguchi, Satsuki Yukino, Koji Tsu
jitani, Houko Kuwashima, Kumiko Watanabe, Kenichi Ogata, Noriko Hidaka, Hisako Kyoda,
Ken Narita, Tomokazu Seki
## 5 Kappei Yamaguchi, Satsuki Yukino, Koji Tsujitani, Houko Kuwashima, Kumiko Watana
be, Noriko Hidaka, Ken Narita, Cho, Mamiko Noto, Nobutoshi Canna, Fumiko Orikasa, Ric
hard Ian Cox, Moneca Stori
## 6                                                                       Kappei Yam
aguchi, Satsuki Yukino, Koji Tsujitani, Houko Kuwashima, Kumiko Watanabe, Ken Narita,
Akio Otsuka, Kikuko Inoue
## 7                                              Kappei Yamaguchi, Satsuki Yukino, K
oji Tsujitani, Houko Kuwashima, Kumiko Watanabe, Noriko Hidaka, Kenichi Ogata, Toshiy
uki Morikawa, Mieko Harada
##   country         date_added release_year rating duration
## 1   Japan September 15, 2021         2002  TV-14   99 min
## 2   Japan September 15, 2021         2003  TV-14   99 min
## 3   Japan September 15, 2021         2004  TV-PG   88 min
## 4   Japan September 15, 2021         2001  TV-PG  100 min
## 5   Japan  September 1, 2017         2004  TV-PG   88 min
## 6   Japan  September 1, 2017         2003  TV-14   99 min
## 7   Japan  September 1, 2017         2002  TV-14   99 min
##                                                   listed_in
## 1 Action & Adventure, Anime Features, International Movies
## 2 Action & Adventure, Anime Features, International Movies
## 3 Action & Adventure, Anime Features, International Movies
## 4 Action & Adventure, Anime Features, International Movies
## 5 Action & Adventure, Anime Features, International Movies
## 6 Action & Adventure, Anime Features, International Movies
## 7 Action & Adventure, Anime Features, International Movies
```

```
##
description
## 1 With their biggest foe seemingly defeated, InuYasha and his friends return to ev
eryday life. But the peace is soon shattered by an emerging new enemy.
## 2  The Great Dog Demon beaqueathed one of the Three Swords of the Fang to each of
his two sons. Now the evil power of the third sword has been awakened.
## 3     Ai, a young half-demon who has escaped from Horai Island to try to help her
people, returns with potential saviors InuYasha, Sesshomaru and Kikyo.
## 4                       A powerful demon has been sealed away for 200 years. But w
hen the demon's son is awakened, the fate of the world is in jeopardy.
## 5     Ai, a young half-demon who has escaped from Horai Island to try to help his
people, returns with potential saviors InuYasha, Sesshomaru and Kikyo.
## 6  The Great Dog Demon beaqueathed one of the Three Swords of the Fang to each of
his two sons. Now the evil power of the third sword has been awakened.
## 7 With their biggest foe seemingly defeated, InuYasha and his friends return to ev
eryday life. But the peace is soon shattered by an emerging new enemy.
```

`tsmovies` is a data frame in R we can manipulate as normal. For example I will change the variable `show_id` to `ID`

```
tsmovies %>%
  rename("ID" = "show_id") #Proving we can manipulate the table
```

```
colnames(tsmovies)
```

```
##  [1] "show_id"      "type"         "title"        "director"     "cast"
##  [6] "country"      "date_added"   "release_year" "rating"       "duration"
## [11] "listed_in"    "description"
```

We can also query using `dplyr`

```
#using dplyr which is already installed from the beginning
movies1 <- tbl(con, "movies")
head(movies1)
```

```
## # Source:   SQL [6 x 12]
## # Database: mysql 8.0.30 [@localhost:/netflix]
##   show_id type    title     direc…¹ cast  country date_…² relea…³ rating durat…⁴
##     <int> <chr>   <chr>     <chr>   <chr> <chr>   <chr>     <int> <chr>  <chr>
## 1       1 Movie   Dick Joh… "Kirst… ""    "Unite… Septem…    2020 PG-13  90 min
## 2       2 TV Show Blood & … ""      "Ama… "South… Septem…    2021 TV-MA  2 Seas…
## 3       3 TV Show Ganglands "Julie… "Sam… ""      Septem…    2021 TV-MA  1 Seas…
## 4       4 TV Show Jailbird… ""      ""    ""      Septem…    2021 TV-MA  1 Seas…
## 5       5 TV Show Kota Fac… ""      "May… "India" Septem…    2021 TV-MA  2 Seas…
## 6       6 TV Show Midnight… "Mike … "Kat… ""      Septem…    2021 TV-MA  1 Seas…
## # … with 2 more variables: listed_in <chr>, description <chr>, and abbreviated
## #   variable names ¹director, ²date_added, ³release_year, ⁴duration
```

```
summary <- movies1 %>% #saving movies1 into summary object
  group_by(type) %>% #grouping by type (movie or tv show)
  arrange(desc(release_year)) #sorting in descending order by release year
```

Finally we `collect` the query

```
summary %>% collect() #displaying the results
```

```
## # A tibble: 8,788 × 12
## # Groups:   type [2]
##    show_id type    title    direc…¹ cast  country date_…² relea…³ rating durat…⁴
##      <int> <chr>   <chr>    <chr>   <chr> <chr>   <chr>     <int> <chr>  <chr>
##  1    8438 TV Show The Net… ""      "Dav… "Unite… Januar…    2021 TV-MA  1 Seas…
##  2    1261 Movie   Crazy A… "Dani … "\xc… "Spain" Februa…    2021 TV-MA  107 min
##  3    1263 Movie   Ginny &… ""      "Dav… ""      Februa…    2021 TV-MA  35 min
##  4    1266 Movie   The Gir… "Ribhu… "Par… "India" Februa…    2021 TV-MA  120 min
##  5    1270 Movie   Geez & … "Rizki… "Han… "Indon… Februa…    2021 TV-PG  106 min
##  6    1273 TV Show Age of … ""      "Mas… "Unite… Februa…    2021 TV-MA  1 Seas…
##  7    1275 TV Show Canine … ""      "Jas… "Unite… Februa…    2021 TV-PG  1 Seas…
##  8    1276 TV Show Ginny &… ""      "Bri… "Unite… Februa…    2021 TV-14  1 Seas…
##  9    1278 TV Show Two Sen… ""      "Nic… "Unite… Februa…    2021 TV-14  2 Seas…
## 10    1279 Movie   Brian R… "Troy … "Bri… "Unite… Februa…    2021 TV-G   58 min
## # … with 8,778 more rows, 2 more variables: listed_in <chr>, description <chr>,
## #   and abbreviated variable names ¹director, ²date_added, ³release_year,
## #   ⁴duration
```

You are also able to translate the R query back into SQL queries. This isn't carrying out any tasks, it is just to display the SQL query.

```
summary %>% show_query() #showing the query in SQL
```

```
## <SQL>
## SELECT *
## FROM `movies`
## ORDER BY `release_year` DESC
```

## sqldf:

Another extremely useful package we can use is `sqldf`. This allows you to make SQL queries to a dataset imported in R.

```
library(sqldf)
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
## Warning in doTryCatch(return(expr), name, parentenv, handler): unable to load shar
ed object '/Library/Frameworks/R.framework/Resources/modules//R_X11.so':
##   dlopen(/Library/Frameworks/R.framework/Resources/modules//R_X11.so, 0x0006): Lib
rary not loaded: /opt/X11/lib/libSM.6.dylib
##   Referenced from: <FF564E7B-F7DD-3BAE-972C-DE65F8735FC9> /Library/Frameworks/R.fr
amework/Versions/4.2/Resources/modules/R_X11.so
##   Reason: tried: '/opt/X11/lib/libSM.6.dylib' (no such file), '/System/Volumes/Pre
boot/Cryptexes/OS/opt/X11/lib/libSM.6.dylib' (no such file), '/opt/X11/lib/libSM.6.dy
lib' (no such file), '/Library/Frameworks/R.framework/Resources/lib/libSM.6.dylib' (n
o such file), '/Library/Internet Plug-Ins/JavaAppletPlugin.plugin/Contents/Home/lib/s
erver/libSM.6.dylib' (no such file)
```

```
## tcltk DLL is linked to '/opt/X11/lib/libX11.6.dylib'
```

```
## Could not load tcltk.  Will use slower R code instead.
```

```
## Loading required package: RSQLite
```

For example if we take the spotify dataset from the first chapter of this project. Let us say we want to write a query to find the average `track_popularity`

```
options(sqldf.driver = "SQLite") #This tells sqldf which DB driver to use

#Picking the average track popularity
sqldf("SELECT AVG(track_popularity) FROM df")
```

```
##   AVG(track_popularity)
## 1              42.48157
```

```
#picking out the amount tracks that were releaed in 2010
sqldf("SELECT COUNT(track_album_release_date) FROM df WHERE track_album_release_date
 = 2020")
```

```
##   COUNT(track_album_release_date)
## 1                             785
```

The average `track_popularity` is 42.47708 and we did this strictly using SQL and from a dataset already in R not extracted from a database.

This is how versataile R is and how smoothly it works with SQL which is a big advantage.

This concludes my R - Packages section and i will now move on to Functions/Programming.

# Functions/Programming

For this section I will continue using the spotify dataset from the analysis section

```
#setting the class of my dataset to 'music'
class(df) <- 'music'
```

## Summary

For the summary function I decided to write a function that would give me back the mean of all the musical attributes

```
#Using the class music and assigning it to a function
summary.music <- function(dataset){ newdf <- data.frame(
#New data frame where its extracting each musical attribute from the dataset thats be
en
#inserted into the function
    danceability = dataset$danceability,
    energy = dataset$energy,
    loudness = dataset$loudness,
    speechiness = dataset$speechiness,
    acousticness = dataset$acousticness,
    instrumentalness = dataset$instrumentalness,
    liveness = dataset$liveness,
    valence = dataset$valence,
    bpm = dataset$BPM)
#Printing out the results getting the mean of each attribute and appending it to
#some string to make it presentable
  cat('Mean Danceability =', mean(newdf$danceability), '\n')
  cat('Mean Energy =', mean(newdf$energy), '\n')
  cat('Mean Loudness =', mean(newdf$loudness), '\n')
  cat('Mean Speechiness =', mean(newdf$speechiness), '\n')
  cat('Mean Acousticness =', mean(newdf$acousticness), '\n')
  cat('Mean Instrumentalness =',mean(newdf$instrumentalness), '\n')
  cat('Mean Liveness =', mean(newdf$liveness), '\n')
  cat('Mean Valence =', mean(newdf$valence), '\n')
  cat('Mean BPM =', mean(newdf$bpm), '\n')
}

#Calling the function
summary(df)
```

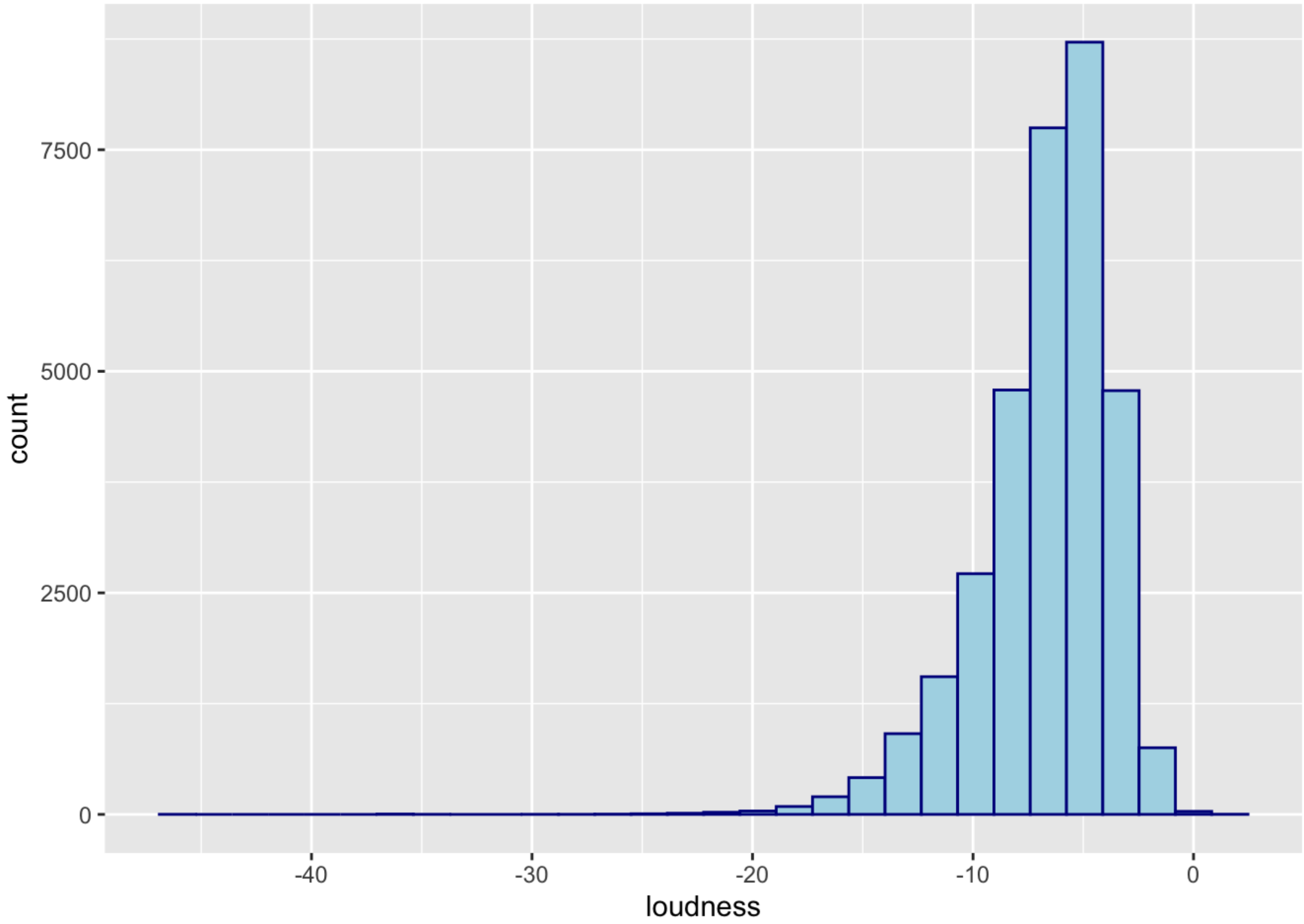
```
## Mean Danceability = 0.6549629
## Mean Energy = 0.6986913
## Mean Loudness = -6.716466
## Mean Speechiness = 0.1071129
## Mean Acousticness = 0.1752232
## Mean Instrumentalness = 0.08477423
## Mean Liveness = 0.1900711
## Mean Valence = 0.5104031
## Mean BPM = 120.8798
```

## Plot

For the plot function I have decided to plot loudness for any dataset that has the class ‘music’ and the variable loudness

```
plot.music <- function(dataset){
  new <- data.frame(
    loudness = df$loudness
    )
  ggplot(new, aes(loudness), binwidth = 20) +
    geom_histogram(col = 'darkblue',fill = 'lightblue')
}

plot(df)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



## Print

Finally for the print function I have written a function that prints how many ongs are in each genre with a dataset of class ‘music’ and the variable `playlist_genre` with the same genres.

```
print.music <- function(dataset)
  { newdf2 <- data.frame(playlist_genre = dataset$playlist_genre)
  cat('Pop =', sum(newdf2$playlist_genre == 'pop', na.rm = TRUE), '\n')
  cat('R&B =', sum(newdf2$playlist_genre == 'r&b', na.rm = TRUE), '\n')
  cat('Rock =', sum(newdf2$playlist_genre == 'rock', na.rm = TRUE), '\n')
  cat('Latin =', sum(newdf2$playlist_genre == 'latin', na.rm = TRUE),'\n')
  cat('Rap =', sum(newdf2$playlist_genre == 'rap', na.rm = TRUE),'\n')
  cat('EDM =', sum(newdf2$playlist_genre == 'edm', na.rm = TRUE), '\n')
}
print(df)
```

```
## Pop = 6043
## R&B = 5742
## Rock = 5505
## Latin = 5427
## Rap = 5153
## EDM = 4927
```

This concludes the final section of this project.

# Citations

## Analysis

```
citation('tidyverse')
```

```
##
## To cite package 'tidyverse' in publications use:
##
##   Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R,
##   Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller
##   E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V,
##   Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to
##   the tidyverse." _Journal of Open Source Software_, *4*(43), 1686.
##   doi:10.21105/joss.01686 <https://doi.org/10.21105/joss.01686>.
##
## A BibTeX entry for LaTeX users is
##
##   @Article{,
##     title = {Welcome to the {tidyverse}},
##     author = {Hadley Wickham and Mara Averick and Jennifer Bryan and Winston Chang
and Lucy D'Agostino McGowan and Romain François and Garrett Grolemund and Alex Hayes
and Lionel Henry and Jim Hester and Max Kuhn and Thomas Lin Pedersen and Evan Miller
and Stephan Milton Bache and Kirill Müller and Jeroen Ooms and David Robinson and Dan
a Paige Seidel and Vitalie Spinu and Kohske Takahashi and Davis Vaughan and Claus Wil
ke and Kara Woo and Hiroaki Yutani},
##     year = {2019},
##     journal = {Journal of Open Source Software},
##     volume = {4},
##     number = {43},
##     pages = {1686},
##     doi = {10.21105/joss.01686},
##   }
```

```
citation('ggplot2')
```

```
##
## To cite ggplot2 in publications, please use:
##
##   H. Wickham. ggplot2: Elegant Graphics for Data Analysis.
##   Springer-Verlag New York, 2016.
##
## A BibTeX entry for LaTeX users is
##
##   @Book{,
##     author = {Hadley Wickham},
##     title = {ggplot2: Elegant Graphics for Data Analysis},
##     publisher = {Springer-Verlag New York},
##     year = {2016},
##     isbn = {978-3-319-24277-4},
##     url = {https://ggplot2.tidyverse.org},
##   }
```

```
citation('lubridate')
```

```
##
## To cite lubridate in publications use:
##
##   Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy
##   with lubridate. Journal of Statistical Software, 40(3), 1-25. URL
##   https://www.jstatsoft.org/v40/i03/.
##
## A BibTeX entry for LaTeX users is
##
##   @Article{,
##     title = {Dates and Times Made Easy with {lubridate}},
##     author = {Garrett Grolemund and Hadley Wickham},
##     journal = {Journal of Statistical Software},
##     year = {2011},
##     volume = {40},
##     number = {3},
##     pages = {1--25},
##     url = {https://www.jstatsoft.org/v40/i03/},
##   }
```

```
citation('visdat')
```

```
##
## To cite package 'visdat' in publications use:
##
##   Tierney N (2017). "visdat: Visualising Whole Data Frames." _JOSS_,
##   *2*(16), 355. doi:10.21105/joss.00355
##   <https://doi.org/10.21105/joss.00355>,
##   <http://dx.doi.org/10.21105/joss.00355>.
##
## A BibTeX entry for LaTeX users is
##
##   @Article{,
##     title = {visdat: Visualising Whole Data Frames},
##     author = {Nicholas Tierney},
##     doi = {10.21105/joss.00355},
##     url = {http://dx.doi.org/10.21105/joss.00355},
##     year = {2017},
##     publisher = {Journal of Open Source Software},
##     volume = {2},
##     number = {16},
##     pages = {355},
##     journal = {JOSS},
##   }
```

```
citation('anytime')
```

```
##
## To cite package 'anytime' in publications use:
##
##   Eddelbuettel D (2020). _anytime: Anything to 'POSIXct' or 'Date'
##   Converter_. R package version 0.3.9,
##   <https://CRAN.R-project.org/package=anytime>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {anytime: Anything to 'POSIXct' or 'Date' Converter},
##     author = {Dirk Eddelbuettel},
##     year = {2020},
##     note = {R package version 0.3.9},
##     url = {https://CRAN.R-project.org/package=anytime},
##   }
##
## ATTENTION: This citation information has been auto-generated from the
## package DESCRIPTION file and may need manual editing, see
## 'help("citation")'.
```

```
citation('DataExplorer')
```

```
##
## To cite package 'DataExplorer' in publications use:
##
##   Cui B (2020). _DataExplorer: Automate Data Exploration and
##   Treatment_. R package version 0.8.2,
##   <https://CRAN.R-project.org/package=DataExplorer>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {DataExplorer: Automate Data Exploration and Treatment},
##     author = {Boxuan Cui},
##     year = {2020},
##     note = {R package version 0.8.2},
##     url = {https://CRAN.R-project.org/package=DataExplorer},
##   }
```

```
citation('ggthemes')
```

```
##
## To cite package 'ggthemes' in publications use:
##
##   Arnold J (2021). _ggthemes: Extra Themes, Scales and Geoms for
##   'ggplot2'_. R package version 4.2.4,
##   <https://CRAN.R-project.org/package=ggthemes>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {ggthemes: Extra Themes, Scales and Geoms for 'ggplot2'},
##     author = {Jeffrey B. Arnold},
##     year = {2021},
##     note = {R package version 4.2.4},
##     url = {https://CRAN.R-project.org/package=ggthemes},
##   }
```

```
citation('wesanderson')
```

```
##
## To cite package 'wesanderson' in publications use:
##
##   Ram K, Wickham H (2018). _wesanderson: A Wes Anderson Palette
##   Generator_. R package version 0.3.6,
##   <https://CRAN.R-project.org/package=wesanderson>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {wesanderson: A Wes Anderson Palette Generator},
##     author = {Karthik Ram and Hadley Wickham},
##     year = {2018},
##     note = {R package version 0.3.6},
##     url = {https://CRAN.R-project.org/package=wesanderson},
##   }
```

```
citation('cowplot')
```

```
##
## To cite package 'cowplot' in publications use:
##
##   Wilke C (2020). _cowplot: Streamlined Plot Theme and Plot Annotations
##   for 'ggplot2'_. R package version 1.1.1,
##   <https://CRAN.R-project.org/package=cowplot>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'},
##     author = {Claus O. Wilke},
##     year = {2020},
##     note = {R package version 1.1.1},
##     url = {https://CRAN.R-project.org/package=cowplot},
##   }
```

## R - Packages

```
citation("DBI")
```

```
##
## To cite package 'DBI' in publications use:
##
##   R Special Interest Group on Databases (R-SIG-DB), Wickham H, Müller K
##   (2022). _DBI: R Database Interface_. R package version 1.1.3,
##   <https://CRAN.R-project.org/package=DBI>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {DBI: R Database Interface},
##     author = {{R Special Interest Group on Databases (R-SIG-DB)} and Hadley Wickha
m and Kirill Müller},
##     year = {2022},
##     note = {R package version 1.1.3},
##     url = {https://CRAN.R-project.org/package=DBI},
##   }
```

```
citation("sqldf")
```

```
##
## To cite package 'sqldf' in publications use:
##
##   Grothendieck G (2017). _sqldf: Manipulate R Data Frames Using SQL_. R
##   package version 0.4-11, <https://CRAN.R-project.org/package=sqldf>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {sqldf: Manipulate R Data Frames Using SQL},
##     author = {G. Grothendieck},
##     year = {2017},
##     note = {R package version 0.4-11},
##     url = {https://CRAN.R-project.org/package=sqldf},
##   }
##
## ATTENTION: This citation information has been auto-generated from the
## package DESCRIPTION file and may need manual editing, see
## 'help("citation")'.
```