Spotify Personal Music Data Analysis
SQL Project

Madeline Lin

July 31st, 2022

1. Introduction

Music, an essential element of my mundane life, brings me joys and delights my day. I am an active subscriber of Spotify application and a devoted fan following the news of Spotify company. My career goal is to join the band as a data scientist to improve their music recommendation system. In terms of the final project, I am eager to apply the knowledge I've learned in statistical computing course to analyze my personal Spotify dataset. I hope to gain a deeper understanding of my music preference, including my top artists, playlists, tracks and genres as well as explore some potential music that match my taste.

2. Data

2.1 Dataset

Spotify was founded by a small group of engineers in Sweden in 2008 ("Spotify, About Spotify", 2022). The data driven culture makes Spotify focus on research and development heavily. From my profile – account – privacy settings, I requested my personal music dataset on June 29, 2022 and received a group of files (i.e. Follow, Identifiers, Identity, Inferences, Marquee, Payments, Playlist, Search Queries, Streaming History 1, Streaming History 2, Streaming History 3, User Data and Your Library) by July 3, 2022. The information of each file can be accessed here[1]. I downloaded the dataset of global weekly (Week of 07-21-2022) top 100 songs from Spotify Charts, which can be accessed here[2].

Based on my review, four datasets (i.e. Your Library, Streaming History, Playlist and Global Weekly Top 100 Songs) are used for my analysis of this project.

2.2 Data Preparation

Data preparation is the process of cleaning and transforming raw data prior to processing and analysis. Data preparation is an important phase in the data analysis process as we are supposed to ensure the efficient analysis, limit errors and inaccuracies that can occur to data during processing. The steps of my data preparation are as follows.

- Your Library
  - Removed N/A values
  - Only kept the category Tracks of the Name column
  - Removed the unnecessary columns
  - Renamed the columns (i.e. trackName → Track, artistName → Artist, etc.)
- Streaming History
  - Merged three Streaming History into one
  - Renamed the columns
  - Removed the rows of Miliseconds_Played column with zero value
  - Removed the unnecessary columns
  - Added Playlist column and input information based on Vlookup from Playlist file
- Playlist
  - Renamed the columns
- Global Weekly Top 100 Songs

[1] https://support.spotify.com/us/article/understanding-my-data/
[2] https://spotifycharts.com

- Renamed the columns
- Added Predicted Milliseconds Played column and input information based on the optimal prediction model (Note: the explanation will be elaborated in the following)

3. Program Interface

- Program Environment
  - R Studio
- Program Libraries
  - dplyr
  - ggplot2
  - class
  - MASS
  - gamlr
  - tree

4. Analysis

Note: The Program Execution Section and the Input and Output Section have been embedded within the Analysis throughout.

4.1 Visualization

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, tables and maps, it provides an accessible way to see and understand trends, outliers and patterns in data (Tableau, 2022). An effective and accurate visualization can make the data come to life and convey the message in a powerful way. Data visualization has become a faster, more effective communication and motivation tool. By using basic R command and ggplot2 library, I created some visualizations as indicated below.

4.1.1. My Favorite Artist

From my perspective, more songs should be collected in my library when it comes to an artist I prefer; In the meantime, I probably spend more time listening to the songs of the artist I love. Therefore, I used two criteria to determine my favorite artist.
First, I would like to show artists with the top 5 number of songs collected in my library. As each song corresponds to the corresponding artist, I simply used the count function built in R to count the number of artists in the Your Library dataset.

Input

```
My_Fav_Artist = count(Your_Library, Artist, sort=TRUE)
head(My_Fav_Artist, n=5)
```

Output

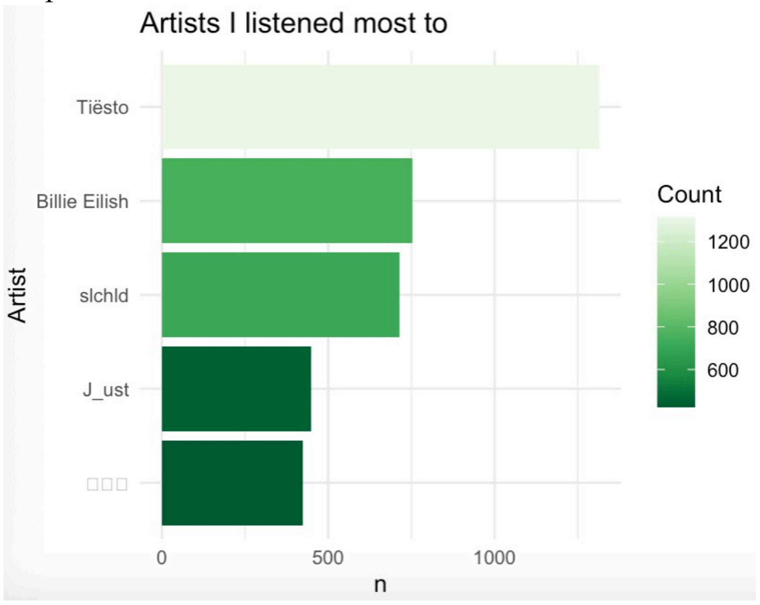
```
        Artist n
1 Billie Eilish 9
2         J_ust 8
3        Tiësto 8
4        slchld 6
5      Jay Chou 5
```

Next, I used ggplot2 library to visualize my favorite artist based on the streaming time of songs.

Input

```
Streaming_History %>%
  count(Artist, sort = TRUE) %>%
  top_n(5) %>%
  mutate(Artist = reorder(Artist, n)) %>%
  ggplot(aes(x = Artist, y = n)) +
  geom_bar(aes(fill=n),
           stat="identity") +
  scale_fill_distiller(palette="ggplot default") +
  xlab(NULL) +
  coord_flip() +
  labs(x = "Artist",
       title = "Artists I listened most to",
       fill = "Count") +
  theme_minimal()
```

Output



Analysis
As demonstrated in above two outputs, I have 9 songs of Billie Eilish in my library and have listened to her each piece of music around 750 times during my streaming period (i.e. from 06-15-2021 to 07-02-2022). I have 8 songs of Tiësto in my library and have listened to his each piece of music over 1250 times during my streaming period. Slchld, a Korean Indie artist, can be regarded as my top artists as I have 6 songs of him and his streaming times are almost the same as Billie Eilish. J_ust, another Korean Indie artist, can also be considered as my top artists as I have 8 songs of him which are the same as Tiësto though the streaming times may not be as high as Tiësto, Billie Eilish and Slchld. Additionally, Jay Chou is a Chinese artist I love. In the second bar plot, the artist with the 5th streaming times does not show properly due to the fact that R is not able to recognize the foreign language. I suppose the name of that artist is in Korean or Chinese.

The results showed above are pretty much what I expected. Starting from last September from electronic zoo musical festival, I have dived into Tiësto and listened to an ocean of his music repeatedly including Secrets, The business, Don't be shy, Red lights and so forth. Apart from electronic dance music (EDM), I spent a ton of time listening to K-Indie. J_ust is a Korean Indie singer who I started loving back to 2019. I mostly love his song Soft. Say what's on your mind from Slchld, is also a song I binge on.

4.1.2. My Favorite Playlist

I usually do not listen to the songs from my playlist but directly from my library (i.e. liked songs). I created a few playlists when I organize a home party or just spontaneously. Still, I guess it is interesting for me to get a sense of which playlist I listen to most.

Input
```
table(Playlist$Playlist)
```

Output
```
madeline is fucking fabulous                              mm1                              mm2
                          29                               17                               38
                         mm3                           mmpshh
                          12                                4
```

The above shows an overview of my playlist. I overall have 5 playlists. They are madeline is f*** fabulous (bear me about this playlist name), mm1, mm2, mm3 and mmpshh with 29, 17, 38, 12 and 4 songs within respectively.

Input
```
aggregate(x=Streaming_History$Miliseconds_Played,
          by = list(Streaming_History$Playlist),
          FUN = sum
          )
```
Output
```
                       Group.1          x
1                         #N/A 3131001176
2 madeline is fucking fabulous   88467476
3                          mm1  179982616
4                          mm2  507587260
5                          mm3  324018464
6                       mmpshh   15230174
```

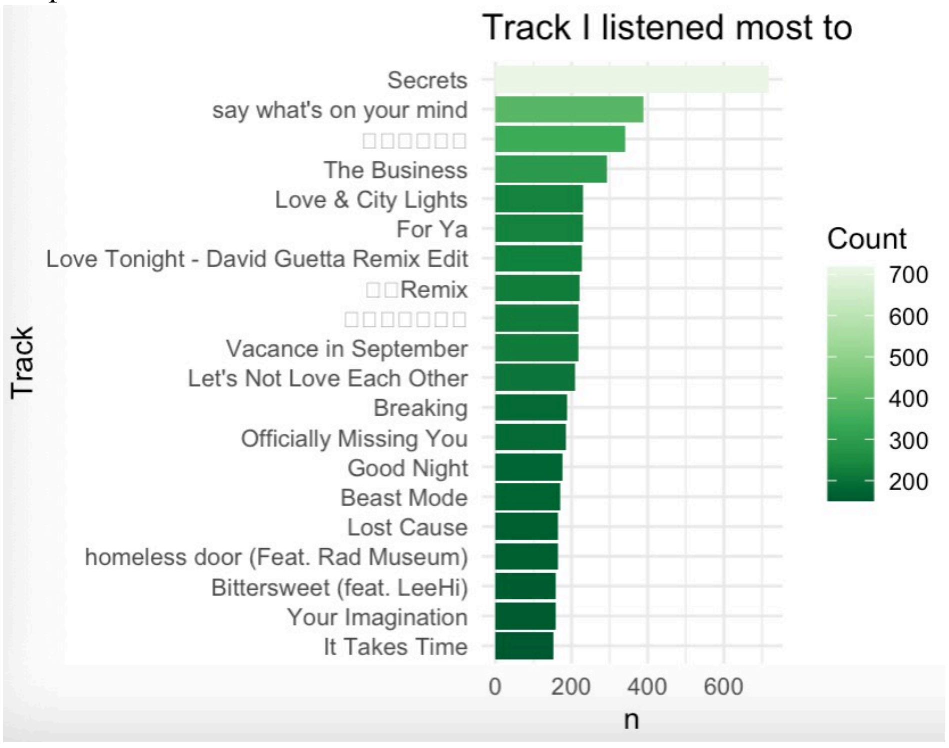
Analysis
Based on the streaming time, mm2 playlist has the longest streaming time compared to other playlists. As I've mentioned that I usually listen to the music from my library, so it makes sense that #N/A has the longest streaming time as I did not spend efforts to category my library songs into different playlists and listen directly from the playlist.

4.1.3. My Favorite Track
Tracks that I listen to repeatedly are considered to be my favorites. Therefore, I created a bar plot using ggplot2 library similar to the second method I've used for my favorite artist section.

Input

```
Streaming_History %>%
  count(Track, sort = TRUE) %>%
  top_n(20) %>%
  mutate(Track = reorder(Track, n)) %>%
  ggplot(aes(x = Track, y = n)) +
  geom_bar(aes(fill=n),
           stat="identity") +
  scale_fill_distiller(palette="ggplot default") +
  xlab(NULL) +
  coord_flip() +
  labs(x = "Track",
       title = "Track I listened most to",
       fill = "Count") +
  theme_minimal()
```

Output



Analysis

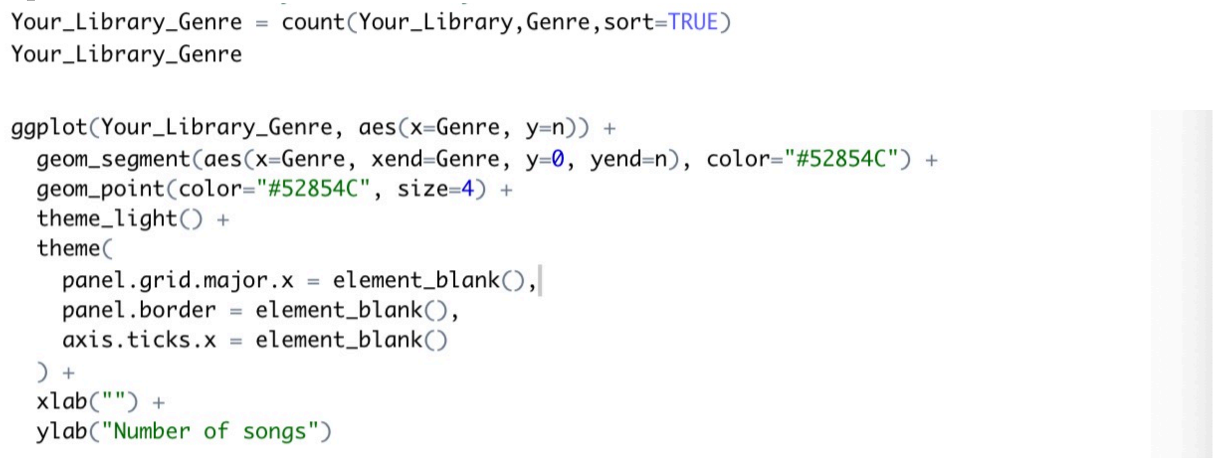
As illustrated above, Secrets, Say what's on your mind, followed by The Business are my top tracks that I binge on. Other tracks including Love & city lights, For ya, Love tonight – David Guetta Remix Edit, Vacance in September, Let's not love each other, Breaking, Officially missing you have the approximate the same streaming time. Some tracks are not able to show properly due to the fact that R is not able to recognize the foreign languages.

The results showed above overall are within my expectation. Moreover, they validate my statements in the ma favorite artist section.

4.1.4. Genre Overview

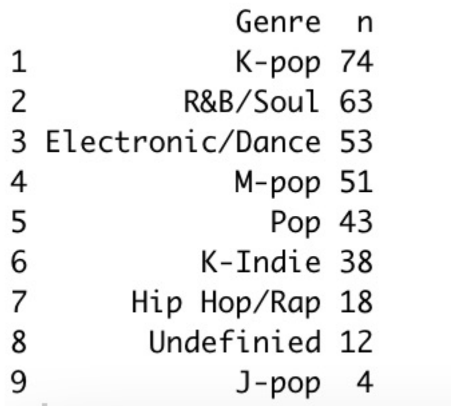
It would also be nice to see my music preference based on the genre. I used count function to create a table of the overview of genre based on the dataset of my library and also used ggplot library to create a lollipop chart.

Input

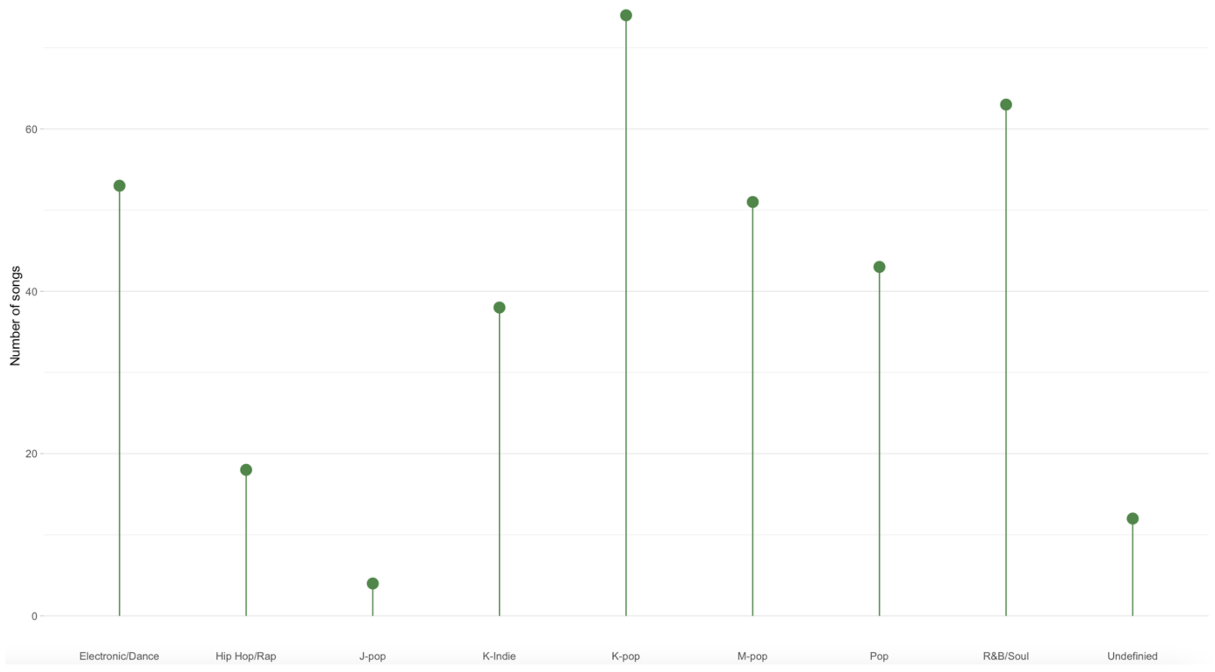


```
Your_Library_Genre = count(Your_Library,Genre,sort=TRUE)
Your_Library_Genre


ggplot(Your_Library_Genre, aes(x=Genre, y=n)) +
  geom_segment(aes(x=Genre, xend=Genre, y=0, yend=n), color="#52854C") +
  geom_point(color="#52854C", size=4) +
  theme_light() +
  theme(
    panel.grid.major.x = element_blank(),
    panel.border = element_blank(),
    axis.ticks.x = element_blank()
  ) +
  xlab("") +
  ylab("Number of songs")
```

Output



```
             Genre  n
1            K-pop 74
2         R&B/Soul 63
3 Electronic/Dance 53
4            M-pop 51
5              Pop 43
6          K-Indie 38
7      Hip Hop/Rap 18
8       Undefinied 12
9            J-pop  4
```



Analysis
As we can see above, genres from my library are K-pop, R&B/Soul, Electronic/Dance, M-pop, Pop, K-Indie, Hip Hop/Rap, J-pop and undefined ones. Among these, K-pop accounts for the majority of the songs in my library, followed by R&B/Soul, Electronic/dance, and M-pop. Apart from J-pop and Hip-Hop/Rap, I think all other genres have a pretty flat distribution.

The result sounds okay to me. I previously expected that K-Indie and R&B/Soul music would take the first and second place of my library. Maybe I just listen to these types of music more but the number of songs of these two types are not as many as K-pop.

4.2 Prediction

By now, I have finished my analysis of Visualization part. I am glad that I have understand better about my music preference and listening habits. Now, I am excited to turn to the part of exploring some potential music that match my taste.

My steps for how to explore the potential music are as follows.

First, based on the Your Library dataset, I would like to try different prediction regression models and calculate the root-mean-square error (RMSE) of each model. The optimal model should be the one with least RMSE.

Second, after determining the optimal model, I will interpret the coefficients of this model and understand my music taste.

Third, based on the Global Weekly Top 100 Songs dataset, I will apply the optimal model and calculated the Predicted Miliseconds Played as a new column.

Fourth, songs with the top 20 predicted miliseconds played can be considered as my potential liked songs as they match my music taste.

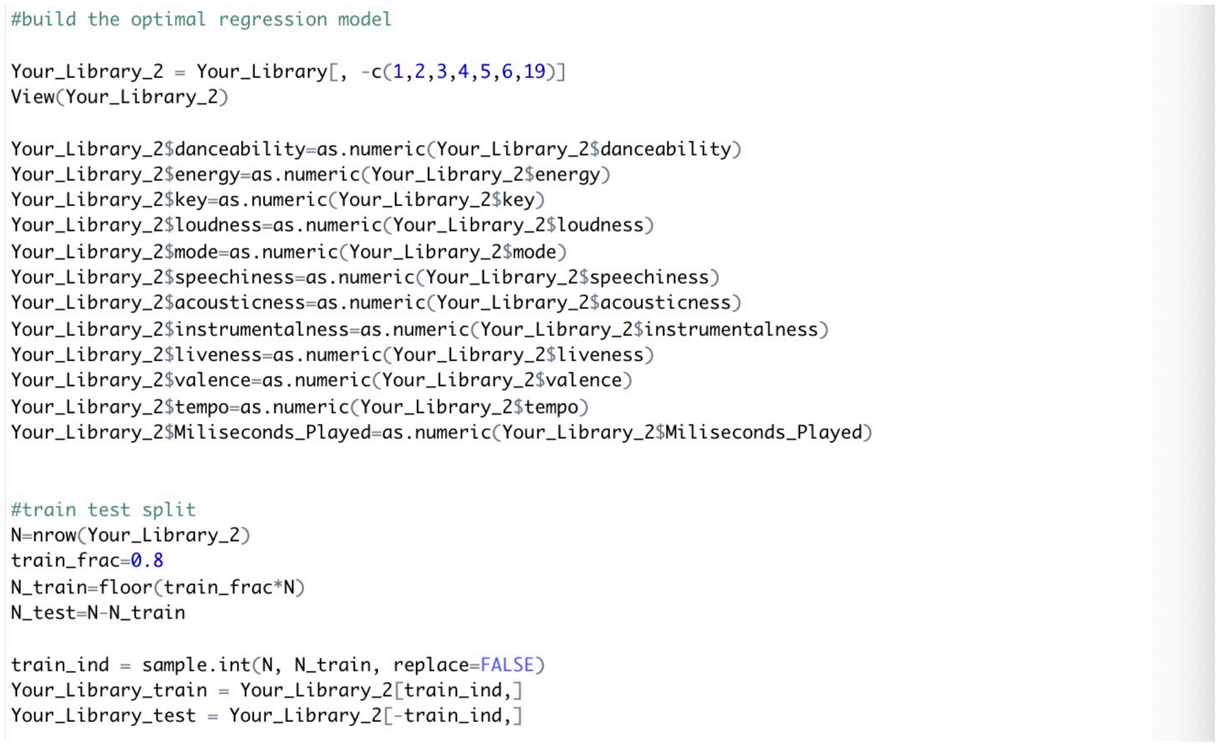
Fifth, I will try to listen to these 20 songs and determine how many songs that I want to add to my current library. It would be great if there are a bunch, as it means I am able to explore new songs based on my successful optimal model.

4.2.1 Before Prediction

Spotify Web API[3] has determined calculated audio feature of tracks to learn about its danceability, energy, valence, tempo, acousticness, and more. A more detailed description of each audio feature can be found in the Appendices Section.

I used these audio features as my x variables and miliseconds played as my y variable. Before jumping into the following model sections, I did the below steps.

First, drop the columns not regarding to our analysis.
Second, switch the data type to be numeric.
Third, conduct a train and test split of my dataset.



```
#build the optimal regression model

Your_Library_2 = Your_Library[, -c(1,2,3,4,5,6,19)]
View(Your_Library_2)

Your_Library_2$danceability=as.numeric(Your_Library_2$danceability)
Your_Library_2$energy=as.numeric(Your_Library_2$energy)
Your_Library_2$key=as.numeric(Your_Library_2$key)
Your_Library_2$loudness=as.numeric(Your_Library_2$loudness)
Your_Library_2$mode=as.numeric(Your_Library_2$mode)
Your_Library_2$speechiness=as.numeric(Your_Library_2$speechiness)
Your_Library_2$acousticness=as.numeric(Your_Library_2$acousticness)
Your_Library_2$instrumentalness=as.numeric(Your_Library_2$instrumentalness)
Your_Library_2$liveness=as.numeric(Your_Library_2$liveness)
Your_Library_2$valence=as.numeric(Your_Library_2$valence)
Your_Library_2$tempo=as.numeric(Your_Library_2$tempo)
Your_Library_2$Miliseconds_Played=as.numeric(Your_Library_2$Miliseconds_Played)


#train test split
N=nrow(Your_Library_2)
train_frac=0.8
N_train=floor(train_frac*N)
N_test=N-N_train

train_ind = sample.int(N, N_train, replace=FALSE)
Your_Library_train = Your_Library_2[train_ind,]
Your_Library_test = Your_Library_2[-train_ind,]
```

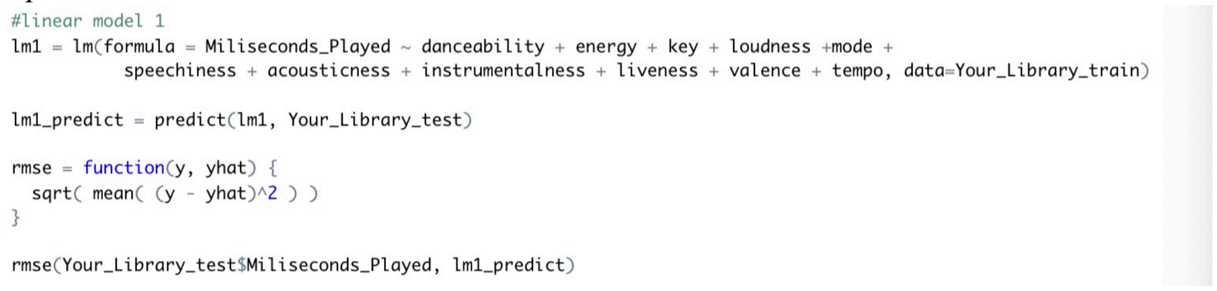
4.2.2 Model Selection

4.2.2.1 Model 1: Simple linear model

First, I used the simple linear model including all x variables. Generally speaking, each audio feature should play a role in contributing the streaming time of the songs.

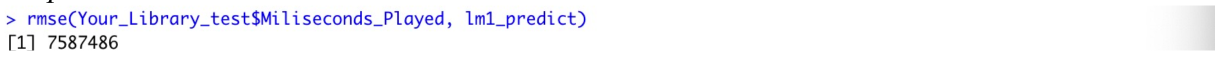
[3] https://developer.spotify.com/documentation/web-api/reference/#/operations/get-track

Input



```
#linear model 1
lm1 = lm(formula = Miliseconds_Played ~ danceability + energy + key + loudness +mode +
           speechiness + acousticness + instrumentalness + liveness + valence + tempo, data=Your_Library_train)

lm1_predict = predict(lm1, Your_Library_test)

rmse = function(y, yhat) {
  sqrt( mean( (y - yhat)^2 ) )
}

rmse(Your_Library_test$Miliseconds_Played, lm1_predict)
```

Output



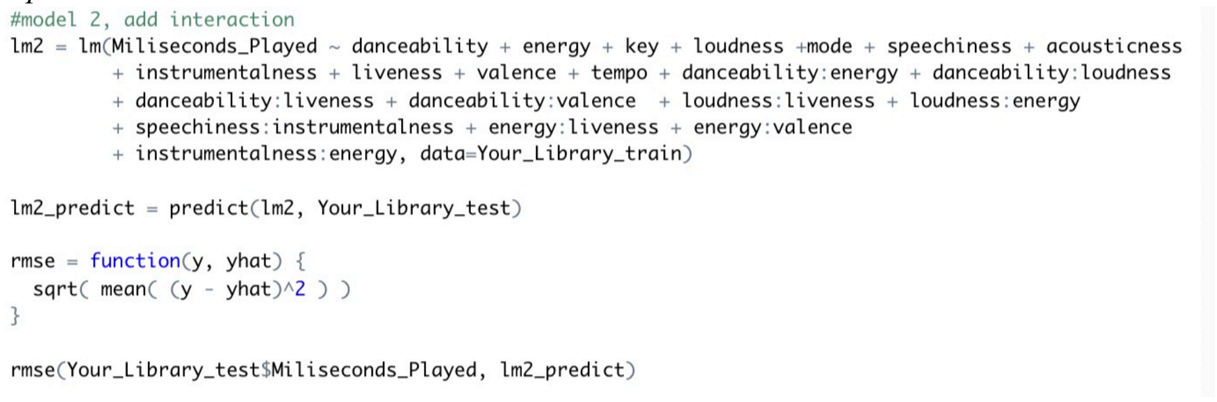
```
> rmse(Your_Library_test$Miliseconds_Played, lm1_predict)
[1] 7587486
```

Based on the result, it shows that the RMSE is approximately 7,587,486.

4.2.2.2 Model 2: Linear model with interaction

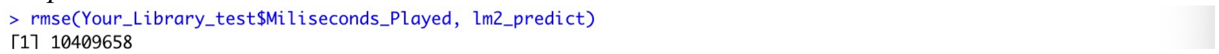
Next, I built a linear model with interactions as I think lots of variables have correlation with one another. For instance, if a song has a high danceability, it probably will have more energy, high loudness and high liveness. I added the interactions based on my intuition as shown in the model 2 in the screenshot and calculated the RMSE.

Input



```
#model 2, add interaction
lm2 = lm(Miliseconds_Played ~ danceability + energy + key + loudness +mode + speechiness + acousticness
         + instrumentalness + liveness + valence + tempo + danceability:energy + danceability:loudness
         + danceability:liveness + danceability:valence  + loudness:liveness + loudness:energy
         + speechiness:instrumentalness + energy:liveness + energy:valence
         + instrumentalness:energy, data=Your_Library_train)

lm2_predict = predict(lm2, Your_Library_test)

rmse = function(y, yhat) {
  sqrt( mean( (y - yhat)^2 ) )
}

rmse(Your_Library_test$Miliseconds_Played, lm2_predict)
```

Output



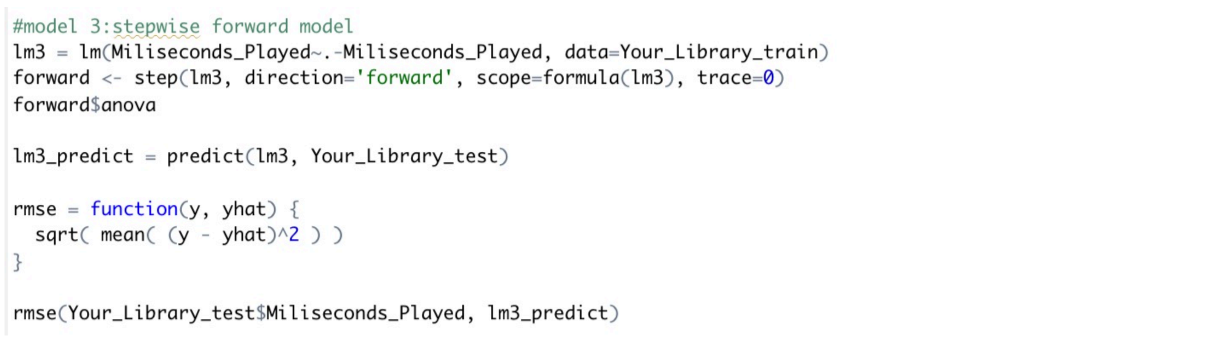
```
> rmse(Your_Library_test$Miliseconds_Played, lm2_predict)
[1] 10409658
```

Based on the result, it shows that the RMSE is approximately 10,409,658.

4.2.2.3 Model 3: Stepwise forward model

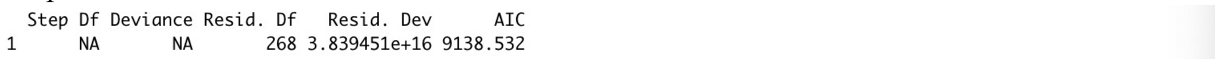
I used the stepwise forward model including all x variables.

Input

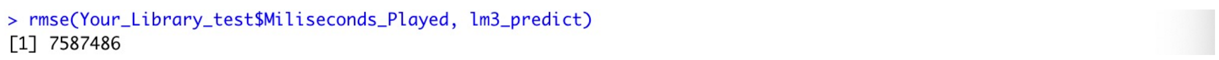


```
#model 3:stepwise forward model
lm3 = lm(Miliseconds_Played~.-Miliseconds_Played, data=Your_Library_train)
forward <- step(lm3, direction='forward', scope=formula(lm3), trace=0)
forward$anova

lm3_predict = predict(lm3, Your_Library_test)

rmse = function(y, yhat) {
  sqrt( mean( (y - yhat)^2 ) )
}

rmse(Your_Library_test$Miliseconds_Played, lm3_predict)
```

Output



```
  Step Df Deviance Resid. Df   Resid. Dev      AIC
1      NA       NA       268 3.839451e+16 9138.532
```



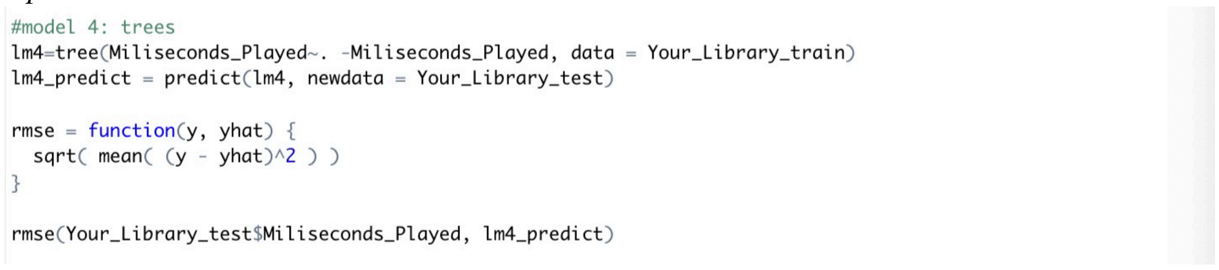
```
> rmse(Your_Library_test$Miliseconds_Played, lm3_predict)
[1] 7587486
```

Based on the result, the model with the least error is when AIC reaches to around 9139 and the corresponding RMSE is approximately 7,587,486.

4.2.2.4 Model 4: Tree model

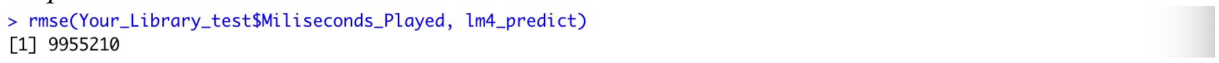
I used the regression decision tree model including all x variables.

Input



```
#model 4: trees
lm4=tree(Miliseconds_Played~. -Miliseconds_Played, data = Your_Library_train)
lm4_predict = predict(lm4, newdata = Your_Library_test)

rmse = function(y, yhat) {
  sqrt( mean( (y - yhat)^2 ) )
}

rmse(Your_Library_test$Miliseconds_Played, lm4_predict)
```

Output

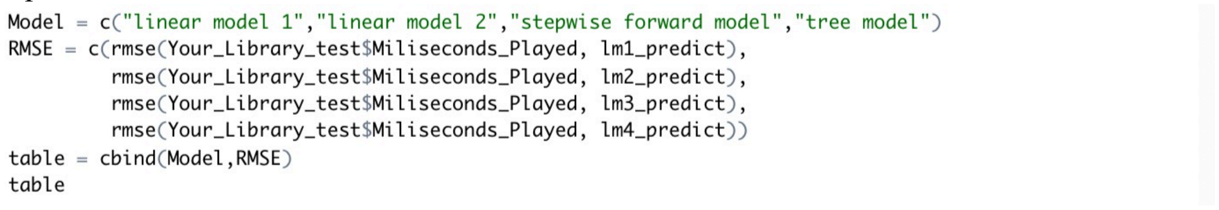


```
> rmse(Your_Library_test$Miliseconds_Played, lm4_predict)
[1] 9955210
```

Based on the result, it shows that the RMSE is approximately 9,955,210.

4.2.3 Model Summary

Input



```
Model = c("linear model 1","linear model 2","stepwise forward model","tree model")
RMSE = c(rmse(Your_Library_test$Miliseconds_Played, lm1_predict),
         rmse(Your_Library_test$Miliseconds_Played, lm2_predict),
         rmse(Your_Library_test$Miliseconds_Played, lm3_predict),
         rmse(Your_Library_test$Miliseconds_Played, lm4_predict))
table = cbind(Model,RMSE)
table
```

Output

```
     Model                   RMSE
[1,] "linear model 1"        "7587485.8404385"
[2,] "linear model 2"        "10409657.8469714"
[3,] "stepwise forward model" "7587485.8404385"
[4,] "tree model"            "9955210.13234925"
```

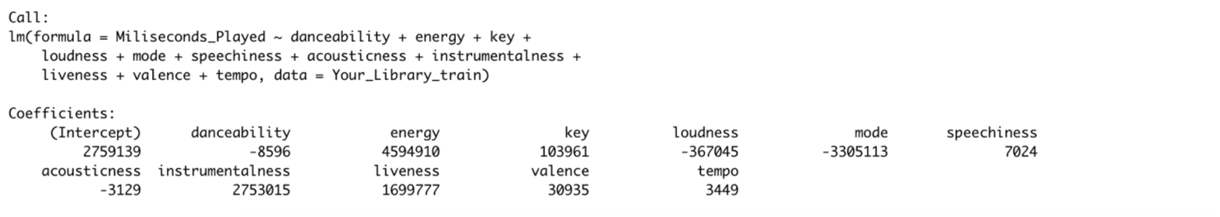
Based on the above table, the linear model 1 and stepwise forward model perform the best with the relatively least RMSE. Thus, I chose linear model 1 (or stepwise forward model) as the most optimal model for further analysis.

4.2.4 Interpret The Optimal Model

Input

```
#Choose the optimal linear model 2
lm2
```

Output



```
Call:
lm(formula = Miliseconds_Played ~ danceability + energy + key +
    loudness + mode + speechiness + acousticness + instrumentalness +
    liveness + valence + tempo, data = Your_Library_train)

Coefficients:
     (Intercept)      danceability            energy               key          loudness              mode       speechiness
         2759139             -8596           4594910            103961           -367045          -3305113              7024
    acousticness  instrumentalness          liveness           valence             tempo
           -3129           2753015           1699777             30935              3449
```

The Simple Linear model proves to have the best performance in predicting my streaming time. Based on the results of the coefficients, my findings are as follows.

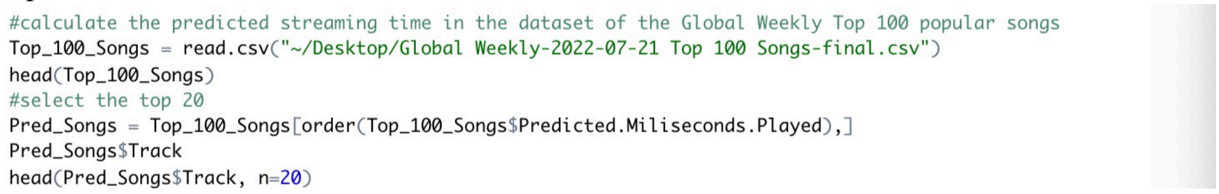
Energy, key, liveness, speechiness, instrumentalness, valence, tempo have contributed positively for my streaming time. I like cheerful music that have high energy and liveness. Compared to danceability, energy and liveness, speechiness, valence and tempo contribute positively with lower coefficient as it plays less significant role of determining my preference of listening to the music.

Danceability, mode and acousticness play a negative role in determining my streaming time, as I do not like music but am more of a fan about enjoying the melodies instead of the lyrics.

However, I am not very sure about why the coefficient of danceability and loudness are negative since I listen to lots of electronic dance music as I've mentioned that I listen to Tiësto a lot. My guess is that probably the overall streaming time of my other songs exceed the electronic dance music as I also listen to K-Indie and R&B/Soul a lot.

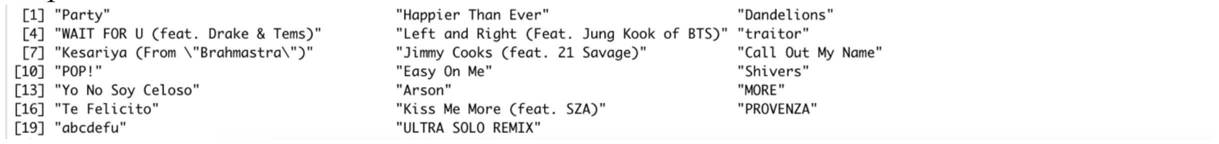
4.2.5 Explore potential liked songs

Input



```
#calculate the predicted streaming time in the dataset of the Global Weekly Top 100 popular songs
Top_100_Songs = read.csv("~/Desktop/Global Weekly-2022-07-21 Top 100 Songs-final.csv")
head(Top_100_Songs)
#select the top 20
Pred_Songs = Top_100_Songs[order(Top_100_Songs$Predicted.Miliseconds.Played),]
Pred_Songs$Track
head(Pred_Songs$Track, n=20)
```

Output



```
 [1] "Party"                             "Happier Than Ever"                          "Dandelions"
 [4] "WAIT FOR U (feat. Drake & Tems)"   "Left and Right (Feat. Jung Kook of BTS)"    "traitor"
 [7] "Kesariya (From \"Brahmastra\")"    "Jimmy Cooks (feat. 21 Savage)"              "Call Out My Name"
[10] "POP!"                              "Easy On Me"                                 "Shivers"
[13] "Yo No Soy Celoso"                  "Arson"                                      "MORE"
[16] "Te Felicito"                       "Kiss Me More (feat. SZA)"                   "PROVENZA"
[19] "abcdefu"                           "ULTRA SOLO REMIX"
```

Based on the Global Weekly Top 100 Songs dataset, I added a new column of predicted milliseconds played of the dataset of global weekly top 100 songs and used the results of these coefficients of the simple linear model to calculate the predicted milliseconds played.

I selected the songs with the top 20 highest miliseconds played. As indicated in the screenshot, I found out that there are two songs that have already existed in my current library. They are Shivers and Kiss Me More. This gives me a favorable sign the performance of my optimal model as I do like a few songs in the list of my potential liked songs. I tried listening the rest of the songs and I found out that I like 6 out of 18. The songs that I should add to my current library are Party, Dandelions, traitor, MORE, POP! and Easy On Me. I am glad that I developed a model with the accuracy of around 33% to provides me a solution to discover my potential liked songs.

5. Program Structure

In terms of the Program Structure Section, I would like to give a simple introduction of each program and explain the statements, algorithmic logic to help the reader understand better. Below is a technical manual of the program that I've used.

5.1 ggplot2

Ggplot allows you to create graphs for univariate and multivariate numerical and categorical data in a straightforward manner. It also allows for easy grouping and conditioning. It can generate complex plots create high quality graphics for publication (Kabacoff, 2020).

I used ggplot2 to create a few data visualizations. I would like to illustrate how the program works based on the case of My favorite artist.

I used the Streaming History as the database and count the number of artist variable using descending order. I select the top 5 by using n=5 command. Then, I created a new variable as I re-ordered artist variable. I used gglot by using artist as the x-axis and streaming times as y-axis. In my opinion, bar plot can convey the information best so I used geom_bar function. I filled the color with ggplot default (i.e. green). I used coord_flip function to fip the bar plot. Finally, I put the label of the x-axis, y-axis and title.

5.2 aggregate()

Aggregate() Function in R splits the data into subsets, computes summary statistics for each subsets and returns the result in a group by form. Aggregate() function is useful in performing all the aggregate operations like sum, count, mean, minimum and maximum (DataScience Made Simple, retrieved 2022).

I would like to illustrate how the program works based on the case of My favorite playlist. Basically, I put sum of the miliseconds_played as the x variable and group it by the playlist. Then, the command gives me a summarized table of the playlist based on the streaming time.

5.3 RMSE
The root-mean-square error is a frequently used measure of the differences between values predicted by a model or an estimator and the values observed (Wikipedia, 2022).

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

I use the above formula of the RMSE which is my indicator to determine the optimal prediction model of streaming time.

5.4 Linear Regression Model

Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine: 1) whether a set of predictor variables do a good job in predicting an outcome variable; 2) which variables in particular are significant predictors of the outcome variable and in what way they indicated by the magnitude and sign of the beta estimates impact the outcome variable (Statistical Analyses, 2022).

The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + b*x, where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable. (Statistical Analyses, 2022).

I built two linear regression models in total.
Firstly, I used the lm() and input the y variable in the left hand of the model and all the x variables in the right hand of the model.
Secondly, I still used the lm() as the base but added some interactions of x variables in the right hand of the model.

5.5 Stepwise Regression Model

Stepwise regression is the step-by-step iterative construction of a regression model that involves the selection of independent variables to be used in a final model. It involves adding or removing potential explanatory variables in succession and testing for statistical significance after each iteration (Hayes et al., 2022).

There are three types of stepwise regression models. They are forward selection, backward selection and both selection. Forward selection starts from zero variable and adds up the variables until the results are optimal. Backward selection starts from adding all variables and deletes each variable until the results are optimal. Both selection is a combination of the forward selection and backward selection as it starts somewhere in the middle.

One thing I should mention here is that instead of RMSE, AIC is the indicator to estimate of in-sample prediction error for stepwise regression model.

I firstly used the simple linear model as a base. Then, I built a forward stepwise regression model by using the step function, putting "forward" as the direction with the scope of the simple linear model with trace equals to zero.

5.6 Decision Tree Model

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data. (Decision Tree – Regression, 2022).

Decision tree learning employs a divide and conquer strategy by conducting a greedy search to identify the optimal split points within a tree. This process of splitting is then repeated in a top-down, recursive manner until all, or the majority of records have been classified under specific class labels (Decision Trees, 2022).

I used the regression decision tree to solve the numerical data. I used the tree function and put all x variables in the right hand of the equation and y variable in the left hand.

6. Improvements and Extensions

I wish to build more types of models including the KNN, lasso and random forest and compare their performance.

I also want to produce more visualizations which helps me gain deeper understanding about my music preference. For example, I would be interested to see during what time period I usually listen to the music in a day. I also want to see my streaming time across from the months and years. Moreover, I am excited to do some visualizations of audio features of each genre in my library. For example, I believe the K-indie genre will have much lower danceability and loudness compared to electronic dance music.

7. Difficulties Encountered

There are a bunch of challenges I have encountered.
Firstly, data format. The data that I received from Spotify is Java script format. At first, I could not figure it out how to transform JavaScript into csv. Based on some research, I found out that we can perform the transformation through Excel - Get Data - Import from JSON.

Secondly, languages. As I have addressed within the body part of the report, my dataset includes the foreign languages (i.e. Korean, Mandarin, and some Japanese). It works out fine when importing the dataset but unfortunately, in terms of the analysis, the program is not powerful enough to read.

Thirdly, some library issue. Though I have updated my R studio to the up-to-date version, I have some trouble installing some libraries. It seems that the random forest package is not able to be installed in my current version. Therefore, unfortunately I could not build the random forest model. I used the tree model instead.

Fourth, RMSE issue. I was not sure why the simple linear model that I built has such a high RMSE. I tried to test by playing around with the x variables but did not figure it out. However, I personally think it should be fine as all four models have large RMSEs. The model with the relatively lowest should be the most optimal.

8. Conclusion

My final project consists of two parts: visualization and prediction. The visualization part gives a demonstration about my music preference and listening habits. The prediction part builds a streaming prediction model and provides a solution of music recommendation system. Both, from my perspective, produce favorable results.

9. References

Aggregate( ) Function in R (n.d.). DataScience Made Simple. Retrieved July 31, 2022, from https://www.datasciencemadesimple.com/aggregate-function-in-r/

Data Visualization. (n.d.). What is data visualization? Definition, examples, and learning resources. *Tableau.* https://www.tableau.com/learn/articles/data-visualization

Decision Tree – Regression (n.d.). Decision Tree. Retrieved July 31, 2022, from https://www.saedsayad.com/decision_tree_reg.htm

Decision Tree (n.d.). What is a decision tree. Retrieved July 31, 2022, from https://www.ibm.com/topics/decision-trees

Hayes, A., Khartit, K. & Kvilhaug S. (2022). Stepwise Regression. *Investopedia.* https://www.investopedia.com/terms/s/stepwise-regression.asp

Root-mean-square error. (n.d.). Wikipedia. Retried July 31, 2022 from https://en.wikipedia.org/wiki/Root-mean-square_deviation

Spotify. (n.d.). About Spotify. Retrieved July 30, 2022, from https://newsroom.spotify.com/company-info/

Spotify. (n.d.). Spotify Charts. Retrieved July 20, 2022, from

https://spotifycharts.com
Spotify. (n.d.). Understand My Data. Retrieved July 3, 2022, from https://support.spotify.com/us/article/understanding-my-data/
Spotify. (n.d.). Spotify Developers. Retrieved July 28, 2022, from https://developer.spotify.com/documentation/web-api/reference/#/operations/get-track
Statistical Analyses (2022). What is linear regression. Retrieved July 31, 2022, from https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/what-is-linear-regression/
Kabacoff, R. (2020). *Data Visualization with R.* Quantitatie Analysis Center. Wesleyan University. https://cengel.github.io/R-data-viz/data-visualization-with-ggplot2.html

9. Appendices

Appendix: Audio Feature Description

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.
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. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.
Key	The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C/D, 2 = D, and so on. If no key was detected, the value is -1.
Loudness	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. 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. “Ooh and “aah” sounds are treated as instrumental in this context. Rap or spoken word tracks

	are clearly “vocal”. 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.
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.