NOT Your Typical Spotify Project

## **Abstract**

Spotify is widely considered the largest music streaming platform, with a substantial variety of artists and genres from all over the world. From classical music like Betohoven and Mozart to heavy metal like Pantera and Metallica. With a wide selection of older artists like the Rolling Stones and Aretha Franklin to up and coming musicians like Balcony Bridge. Given this scope, Spotify’s selection is unique in its size and variety. Moreover, the platform has developed a variety of song features that turn seemingly esoteric qualities into real data that allows us to better understand what makes a “good song”.

However, given Spotify’s popularity and the wide availability of their data, analyses like this are quite common. Typically, analyses have looked at a group of songs and done Exploratory Data Analysis to find trends among popular songs. This can be extrapolated to predict if a song will be popular or not. In our project, we will gather data from our peers in Data 557, and compare the music listened to by the class to mainstream popular music. We intend to and to build a model that attempts to predict a song's popularity using a variety of features such as danceability, energy and others. Then use this model to understand how similar/different Data 557 preferences are from popular music

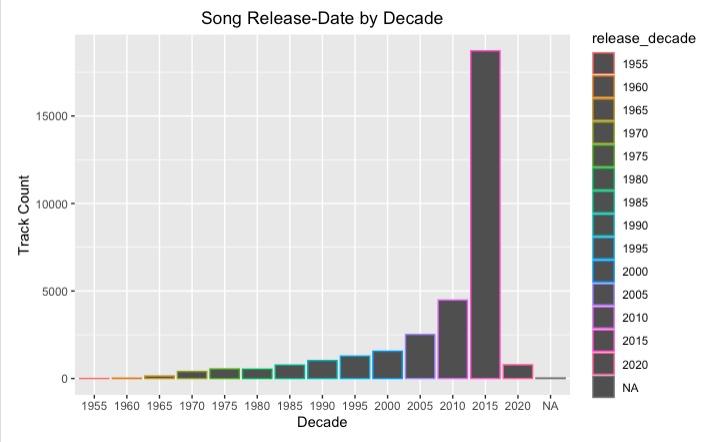
## **Introduction / Project motivation**

Music is widely considered foundational to understanding a society and its cultural trends. However, it is also a collective phenomenon which unites millions across many countries and societies. This dichotomy makes music not only a fun pastime, but an interesting subject to explore as it might hold key-insights into common universal principles, we might all share. While the scope of this project is not large enough to answer this question, we hope to gain some understanding of what components contribute to a song’s popularity. And, perhaps, start understanding the common likes we share.

## **Data description**

### **Data Source 1:**

The first data source was compiled by Tidey Tuesday, a podcast by R4DS Online Learning Community dedicated to help R learners in real world contexts. This dataset includes a selection of 32,833 songs spanning a wide timeframe beginning in the late 1960s and up to today. Each song includes **all** the features included below. The majority of the songs in this dataset are from the past 10 years. However, approximately 5,000 songs are from the previous century. Despite this bias in the data towards recent songs, this part of the analysis is designed to test the effect of song features on the popularity. In addition to Spotify’s variables, we created a “title track length” categorical variable bucketed into 10 letter increments and a release decade categorical variable. One major limitation we face using this data source is the limited generalization this project might have. Songs are predominantly Western, and popularity is based on those countries’ charts.



### **Data Source 2:**

The second data source was compiled by scraping song information from playlists on Spotify using their developer API. One of the playlists from which we scraped data was the Spotify Top 500 playlist, which contained the 500 most popular songs on Spotify according to monthly listeners. This playlist was intended to be a representative sample of mainstream and popular music.

### **Data Source 3:**

This is the most challenging part of our data collection process. We were tasked to collect information about the song tastes of the class. It is impossible to collect the individual data manually, and to make the collection more efficient, we decided to develop a web application that would ask permissions from the volunteers in the class, and then extract their songs from the playlists, liked songs and their most recently played songs.

#### 

#### Challenges Faced in data collection:

Throughout the development of our web application, my team and I encountered several complex technical issues. Our project required working with the Spotify API, which presented several challenges. Firstly, the API's authentication process imposed [restrictions on hobby and individual projects such as ours](https://developer.spotify.com/documentation/web-api/guides/development-extended-quota-modes/), which meant we had to adhere to specific requirements before accessing the API's data. As a result, we had to devise workarounds and manually add users in development mode, as the public mode's requirements and regulations stipulated by Spotify were strict and complicated, taking up more time than we had allocated for the project.

Moreover, we had to set up a streamlined process to obtain data with a single click, which proved to be a herculean task. Our goal was to design a user-friendly interface that facilitated easy access to desired data without causing confusion. To accomplish this, we implemented staged procedures to ensure that each step was comprehensible and uncomplicated. We leveraged the best practices in user experience design to make the interface intuitive, simple, and easy to navigate, ensuring that users could find what they needed with ease.

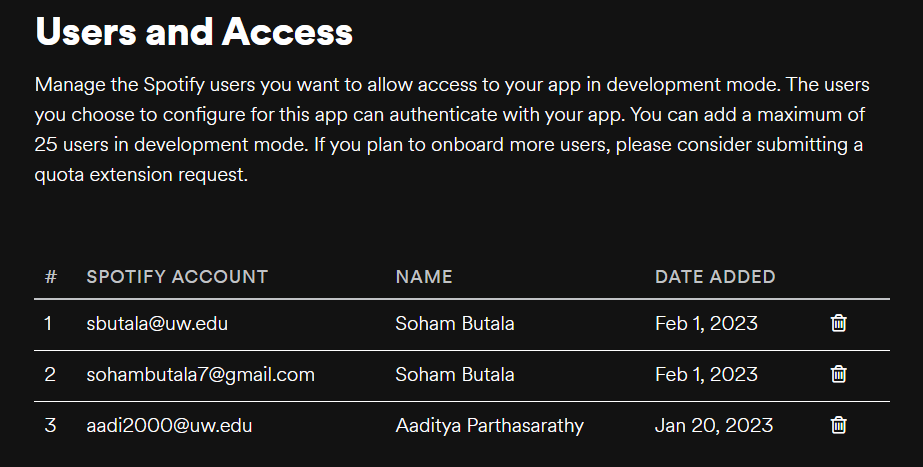
In addition to the authentication and user interface challenges, we had to ensure the security of our users' data. There were significant concerns that our team could access user accounts once they granted access to our application, which could lead to data leaks and other security breaches. To address these concerns, we informed our users that we employed Spotify's OAuth on the backend, which is a robust authentication protocol that ensures secure access to user data. Additionally, we informed our users that their signup process was entirely handled by Spotify, and we could only access user data based on the permissions granted. We made no attempt to attribute a song to any specific person in the volunteer pool. Furthermore, all data was safely destroyed upon completion of the project, eliminating any possibility of identifiable data leakage.

Lastly, we faced limitations on certain areas of the API, such as the inability to extract more than the last 50 recently played songs in history. This restriction impacted our ability to extract the volunteer data, but we accepted it as a necessary measure to comply with the GDPR rules for privacy and data storage/access. As such, we had to devise creative solutions that enabled us to work within the API's limitations while providing us with the data that we needed.

#### Stages of the web application development:

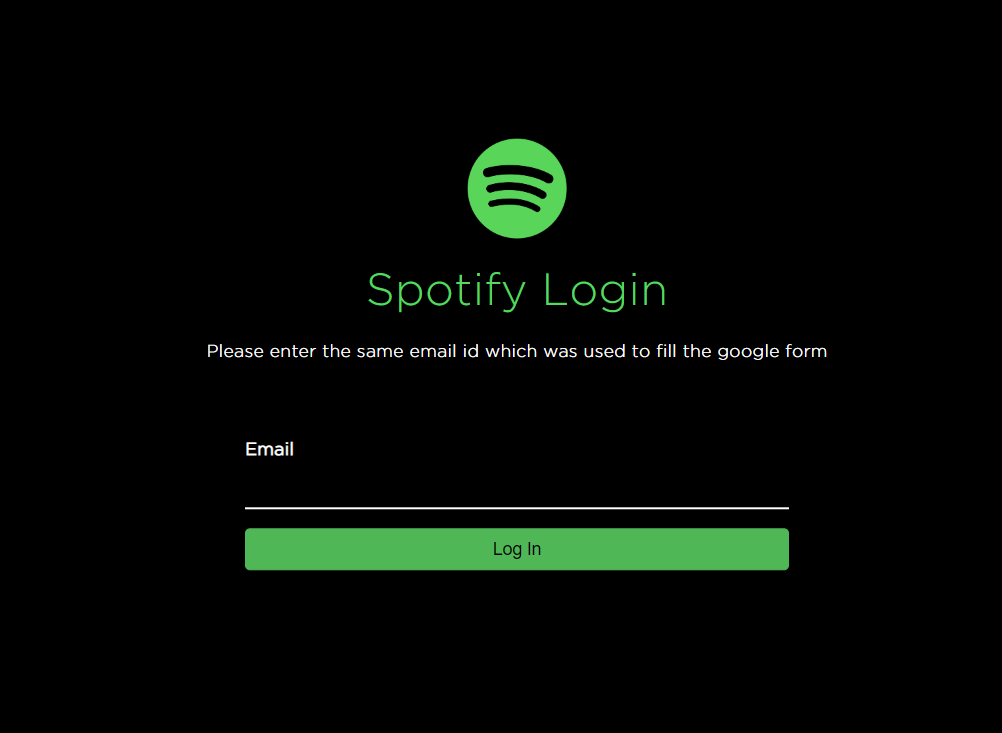
As mentioned in the challenges, we split up the data collection process into 2 parts

1. Asking volunters to sign up for our project though a simple google sheets link. In the end, we received about 10 volunteers who were willing to join the project in the class.
   1. The emails were used to sign up in the spotify dashboard.



*The example console where the names and the emails were entered*

1. Sent out the link for the website to all the volunteers for them to sign up to the project. This redirected them to a website which looked like the one below



*The website frontend*

1. This redirected them to the Spotify’s Oauth and then to our website, which collected the data in the backend and stored it in the Firebase Database, which is our cloud database of choice for the project.

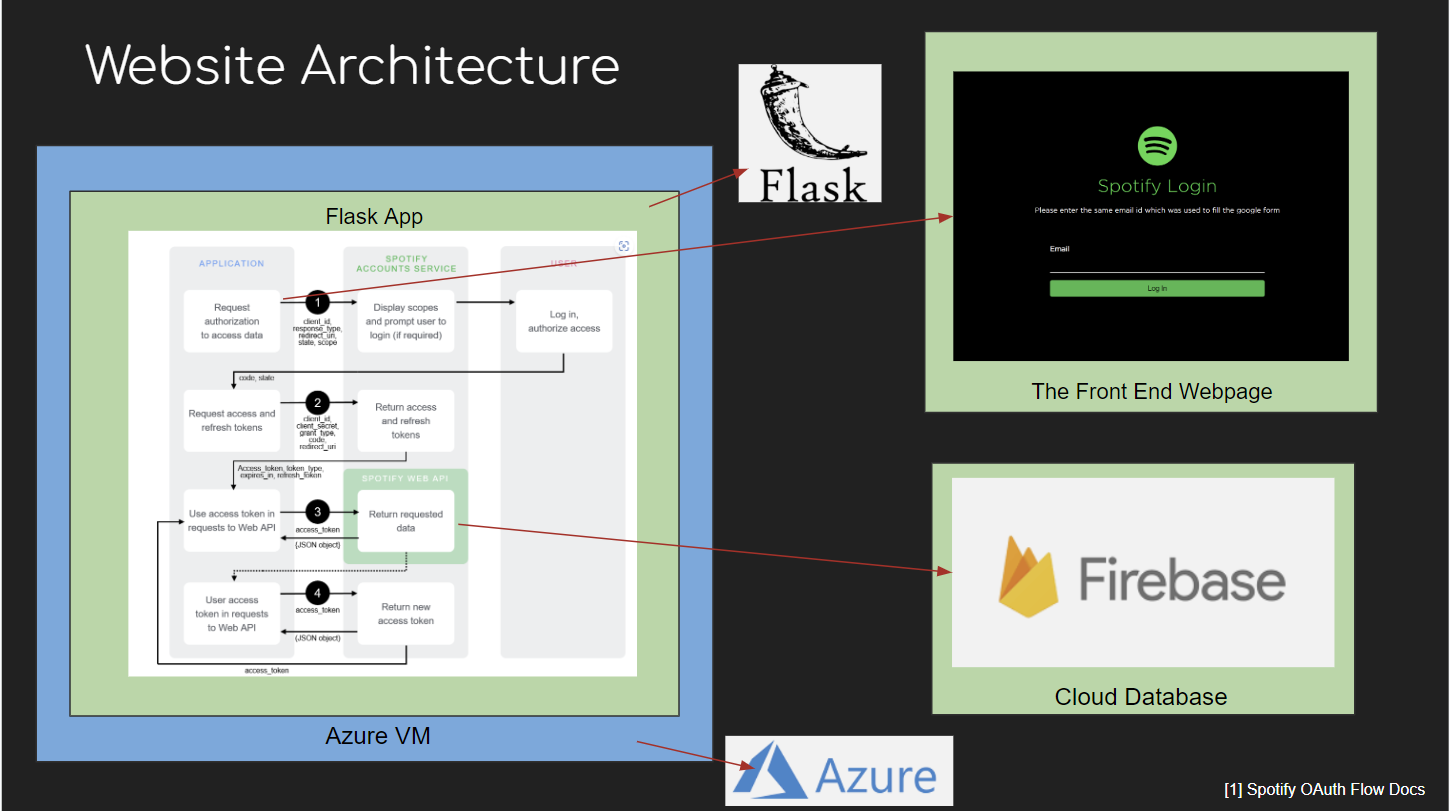
#### The Website Flow:

The website architecture flow is shown in the figure below. Some of the tools and technologies used in this project are:

* Flask for the web application
* HTML/CSS/Javascript for the frontend
* Firebase for the database storage
* Azure for the virtual machines
* Spotify APIs for the data collection

The flow of the website is as follows:

1. **Initialization**: The user is shown a page, where they just have to enter their email address.
2. **Authentication**: When a user accesses the application, they are prompted to grant access to their Spotify account. The user is then redirected to the Spotify login page to enter their login credentials.
3. **Authorization**: After successful authentication, the user is prompted to grant the application specific permissions to access their Spotify data. The permissions requested by the application are determined by the scope parameter in the OAuth request.
4. **Access Token**: Once the user grants permission, the application is issued an access token by Spotify. The access token is a temporary code that allows the application to make requests to the Spotify API on behalf of the user.
5. **Refresh Token**: The access token has an expiration time, after which it becomes invalid. To avoid repeated login requests, the application can use the refresh token to obtain a new access token. The refresh token is a long-lived token that is issued when the user grants the application offline access.
6. **Accessing Data**: The application can use the access token to make requests to the Spotify API and access the user's data based on the permissions granted. The API responses are in JSON format and contain the requested data.
   1. We collect the data from 3 major sources of data using multiple functions.
      1. Top 50 currently played(cannot extract more than that)
      2. Saved Playlists
      3. Liked Songs
   2. The data is turned into a JSON file
   3. It is sent to the firebase database, removing any identifying information.



*The website architecture*

#### Processing the JSON Files:

One of the primary issues we faced now was dealing with the vast amounts of metadata stored in the JSON files. To extract only the unique song IDs from the entire dataset, we implemented a robust data parsing mechanism in Python that allowed us to filter and extract only the relevant song IDs.

After obtaining the song IDs, we utilized the power of Spotify's Python API wrapper, spotipy, to query the Spotify API. This approach enabled us to extract and gather various parameters of the song, such as artist name, song title, duration, and genre, among others. We carefully curated this information, storing them in a tabular format as a CSV file that was easy to import into R Studio for statistical analysis.

By juxtaposing these two sources of song data(sources 2 and 3), we hoped to be able to look for differences in song attributes between mainstream popularity and more niche types of music. In total, the data set had 6,178 rows with 14 columns. We manually created the variable ‘source’ which would act as our response variable and indicate if a song came from the Spotify Top 500 playlist, or from a playlist of a Data 557 participant. There were 500 songs from the Spotify Top 500 playlist and 5,674 songs from Data 557 participants. The other 13 columns contained the same song attribute variables as Data Source 1, but did not contain information about genre or subgenre. Genre is an album level variable, and hence was difficult to collect using the Spotify API.

### **Features**

All of our data sources share the following common features shown in the [table](#_xx9ahql2drsy) below.

## **Statistical methods**

### Research Question 1

Our first goal with this project is to understand what makes a song popular. To answer this, we will first determine which genres are most popular. Then, what features, as defined by Spotify and our team, contribute to a song’s popularity. Finally, are these features correlated in any way?

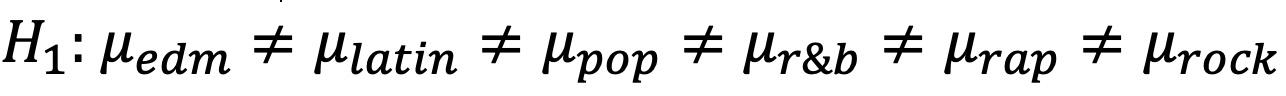
#### Question 1a

Which genres are most popular? And, do we see a statistically significant difference in popularity in the genres? Our data includes six distinct genres: EDM, Latin, Pop, R&B, Rap and Rock. In addition, there are twenty-four different sub-genres. Each genre in our sample includes approximately an even number of songs.

Null hypothesis: The mean song popularity is equal among all genres:

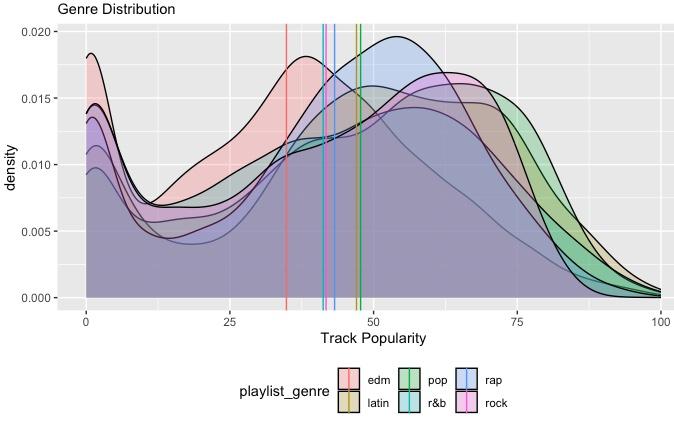


Alternative hypothesis: The mean song popularity is not equal for at least one genre:



To answer this question, we will use a one-way ANOVA test. This test assumes normality, equal variance and independence. Inspecting the density curve we see that the data is approximately bimodal with two distinct normalish regions. In addition, the means and variances are very similar. Finally, given the amount of data and the data’s time-span, we can assume independence. To add robustness to this test, we will also do a Willcox test.

To investigate the popularity difference between genres and playlists we will also perform a t-test on two samples for example "Dance Pop" and "Electro Pop". We used t-test to compare the popularity of these two playlists, and assume null hypothesis that there is no difference between the mean popularity of both playlists. The alternative hypothesis is that there is a significant difference between the mean popularity of both playlists.



#### Question 1b

This section aims to tackle the fundamental question in this project: what features make a song popular. Our assumption coming into this project is that features may not be linearly related to song popularity. As such, we decided to test each one of our numeric predictors and find the highest polynomial degree that still has significance when regressed on song popularity. For example, for energy we created 5 models:

1. track\_popularity ~ energy (p-values: 2e-16, 2e-16)

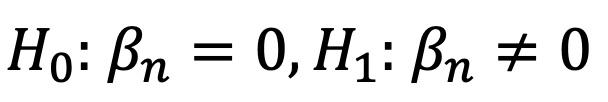
2. track\_popularity ~ energy + energy2 (p-values: 2e-16, 2e-16, 2e-16)

3. track\_popularity ~ energy + energy2 + energy3 (p-values: 2e-16, 0.009,0.0002, 2e-16)

4. track\_popularity ~ energy + energy2 + energy3 + energy4 (p-values: 2e-16, 0.94, 0.86, 0.605, 0.282)

However, given that we lose significance for energy on the fourth degree, our model to predict popularity goes up to energy3. Based on this test, our initial model includes all the categorical variables as well as, energy: 3 degrees, loudness: 1 degree, valence: 3 degrees, tempo: 2 degrees, duration\_ms: 4 degrees, speechiness: 0 degrees, acousticness: 4 degrees, instrumentalness: 4 degrees, liveness: 4 degrees.

To determine significance we perform a hypothesis test for each coefficient at 0.05 threshold:



We will fit several models trying to predict song popularity using several combinations of our predictors. Then use R2 to assess goodness of fit for each model, as well as Akaike’s Information Criteria (AIC) and Bayesian Information Criteria (BIC) to compare the model’s performance. Linear regression model makes four assumptions:

1. Independence

2. Linearity – checked using Fitted values vs Residuals plot

3. Constant variance/Homoscedasticity – checked using Fitted values vs sqrt(Standardized Residuals).

4. Normality – checked using a Theoretical Quantile vs Standardized Residuals plot.

We will confirm that all our models meet these assumptions to be included in the analysis. Given the space limitation, we only include plots showing our model assumptions are met and summary statistics for model comparison. A full report of coefficient values and statistics will be given in the appendix.

#### Question 1c

The Pearson linear correlation coefficient is a measure of the strength of the linear association between two variables. The goal for this section is to understand whether features are correlated. This measure allows us to test our independence assumption for the regression. But, more importantly, it gives us insights to what features might be related so we gain a more holistic understanding of how feature interaction affects song popularity.

We will further examine whether there is a correlation between two musical features for a specific genre, for example danceability and valence, for tracks in the EDM genre. A correlation coefficient and a p-value have been used to answer this question. The null hypothesis assumes that there is no correlation between danceability and valence in the EDM genre. The alternative hypothesis is that there is a significant correlation between danceability and valence in the EDM genre.

## **Research Question 2**

Our second goal with this project was to create a model which could classify a given song as coming from the Spotify Top 500 playlist, or from a playlist of a Data 557 participant. Typically, classifying models attempt to differentiate between two mostly homogeneous groups; however for our analysis, the “group” of Data 557 was not homogeneous. Songs had been scraped from the playlists of 8 individuals in the class, all of whom had unique taste in music. This created a situation where the distribution of song attribute variables for Data 557 songs generally had a much wider spread than the same attribute for Spotify Top 500 songs. We were unsure whether there was substantial difference between the class and mainstream music, but attempted to fit a logistic classification model with these potential challenges in mind.

Linear regression assumes that observations are independent of each other, that there are sufficiently large sample sizes, and that predictor variables are linearly related to the logit of the response variable. Checking on the independence assumption, it at first appears troubling that we have collected data from many individuals, creating groups within our larger data set. The songs listened to by an individual are likely similar to one another, does this not create dependence between observations from a given listener? This would be an issue, if not for the fact that our response variable *is the listener.* Essentially, we can think of the Spotify Top 500 playlist as being a single listener, and we are trying to predict if a given song was listened to by them or by someone else in the class. This way, knowing which listener chose a song gives no additional information about its relationship to the response. Moving onto the sample size assumption, our probability of success (being from the Spotify Top 500) is 500 over 6,178 or about 8.09%. Hence, we want about 5/(0.0809) = 62 observations for each individual hypothesis test we conduct. This will not be an issue given our large sample size. Finally, we need to check that predictors are linearly related to the logit of the response. This will be assessed with model diagnostics once a model has been fit and correctly adjusted.

Moving forward with fitting a logistic regression model, we first conducted exploratory data analysis on all predictor variables, assessing their correlation with ‘source’ visually. We saw small correlations with many predictor variables, with no one predictor obviously standing out. The variable ‘instrumentalness’ was heavily skewed right with a median of 0.000019 but a mean of 0.083. The Data 557 group also seemed to have more songs with high values of ‘instrumentalness’. Because of this heavy skew, we decided to re-code ‘instrumentalness’ as a categorical variable with a threshold of 0.02. Approximately 20% of songs had ‘instrumentalness’ values greater than 0.02 which we thought was a more apt way to categorize the variable. We also decided that the categorical variable ‘key’ which contained 12 categories corresponding to the 12 musical notes (C, Db, D, …) was too complex to use in the model. There did not appear to be a large difference visually in key between the Spotify Top 500 and Data 557, and we thought that 12 individual coefficients might cause overfit in the model.

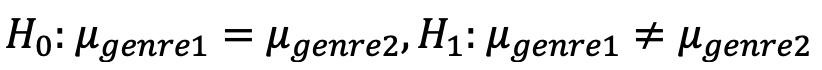
Having completed exploratory analysis, we fit a logistic regression model containing all available predictor variables (using instrumentalness as a categorical variable and excluding key). There were 13 predictor variables used in the model that created 15 unique coefficients. We wanted to perform a hypothesis test to determine if each individual predictor contributed significantly to the model. Hence, we used a significance level of 0.05/15 = 0.003, adjusting for multiple testing using the Bonferroni correction. We used Wald Tests with robust standard errors for each predictor variable, where the full model was the model fit with all significant coefficients and the reduced model was the full model excluding the predictor variable of interest. Once we arrived at the final model, we evaluated binned residual plots to ensure that our model did not over/under-predict and that our logit was linearly related to our predictor variables included in the final model. From there, we split our data into a training set (80% of songs) and a test set (20% of songs) and evaluated accuracy, specificity, and sensitivity using an ROC curve to determine what our threshold probability should be to consider a prediction a success. Using this threshold, we could then make predictions about songs not included in our original datasets to make some generalizations about the effectiveness of the model.

## **Results**

### **Research Question 1**

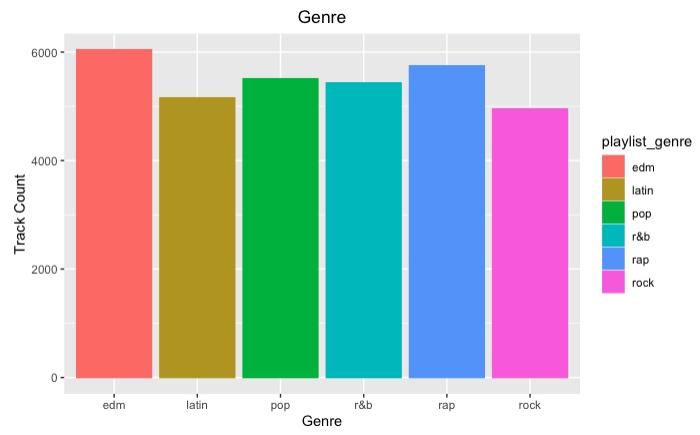
#### Research question 1a results:

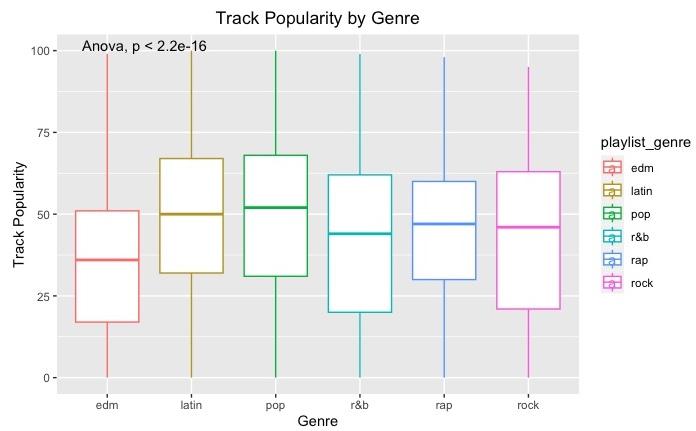
Data source 1 includes six distinct genres with approximately equal number of songs per genre. Based on the p-value produced from the AONVA test, we reject the null hypothesis that the average popularity among all the genres is equal. Although the ANOVA test failed to reject, it does not imply that all genres are equal in average popularity. To test this, we ran Welch Two Sample t-test for each pair of genres where the null and alternative hypotheses are:

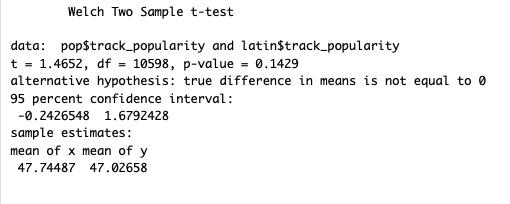


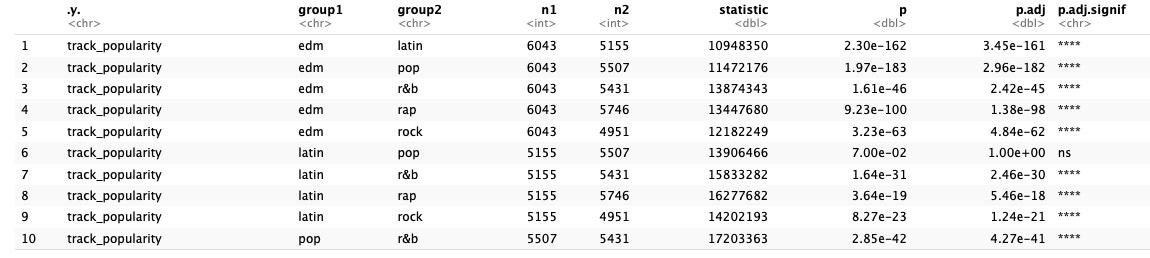
We failed to reject the null hypothesis in all pairs of genres except for Pop and Latin. This implies that the average popularity between these two genres might be equal.

Finally, we ran a Wilcox test that handles non-parametric data to increase results’ robustness.



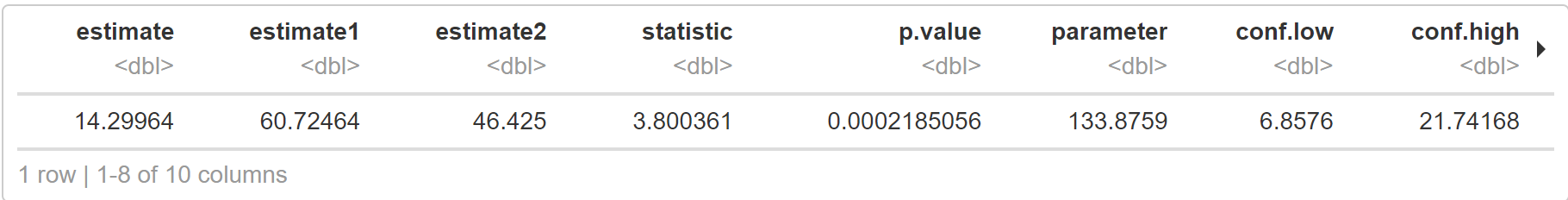


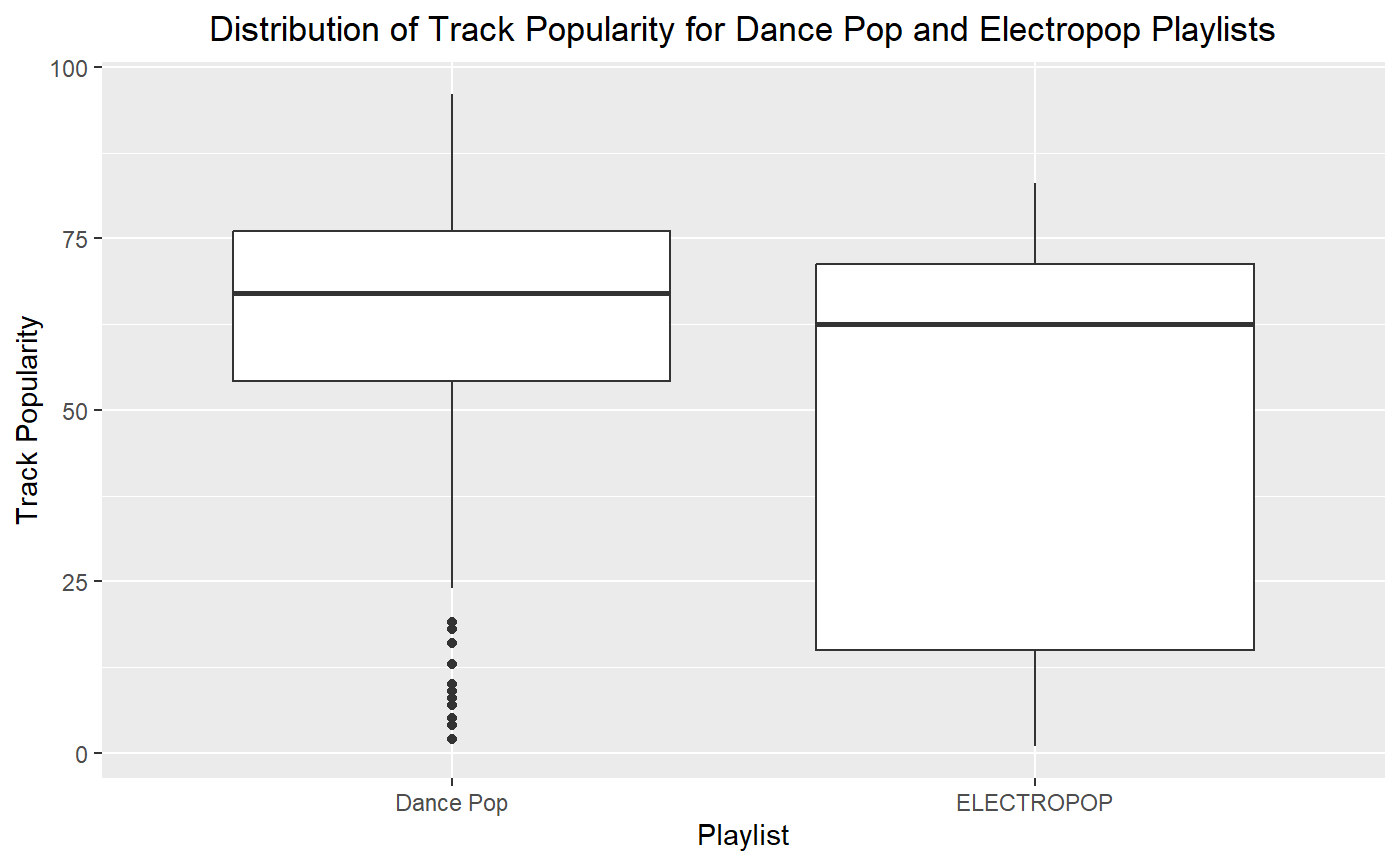




Investigating further into popularity differences between different playlists, we found that the results of the t-test show that there is a significant difference between the popularity of "Dance Pop" and "Electro Pop" playlists. The p-value for the t-test is 0.0002, which is less than the commonly used significance level of 0.05, indicating strong evidence against the null hypothesis. Therefore, we reject the null hypothesis and conclude that there is a significant difference between the mean popularity of both playlists.

The confidence interval for the difference in means of the two playlists is [6.8576, 21.74168], which indicates that we can be 95% confident that the true difference in mean popularity between the two playlists is between 6.8576 and 21.74168. Since the confidence interval does not contain zero, this further supports the conclusion that there is a significant difference in popularity between the two playlists.In conclusion, the analysis indicates a significant difference in popularity between the "Dance Pop" and "Electro Pop" playlists, with the former having higher popularity on average.



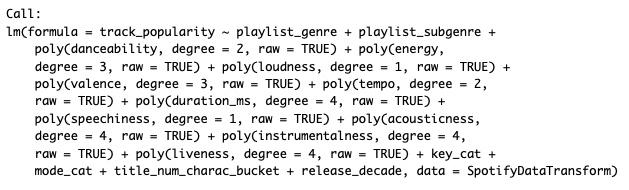


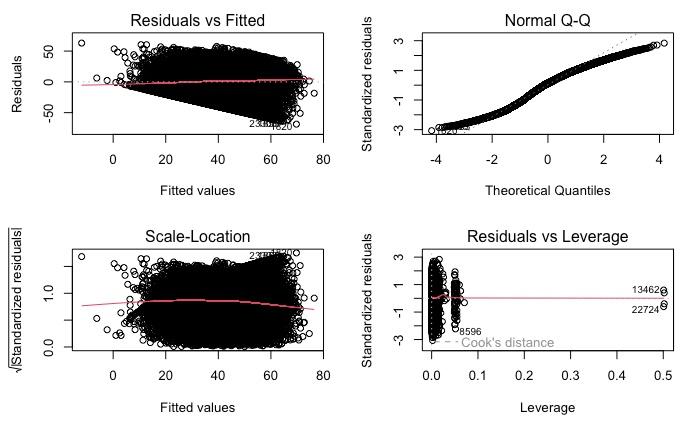
#### Research question 1b results:

##### Model 1:

All variables (including polynomials discussed in Research Question 1b section).

We find that all genres and most subgenres are statically significant when predicting a song’s popularity. We also see that danceability, energy, loudness, duration, instrumentalness and liveness are significant when predicting song popularity. R2 = 0.1972, AIC = 297065.7, BIC = 297065.7.

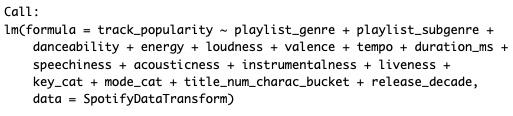


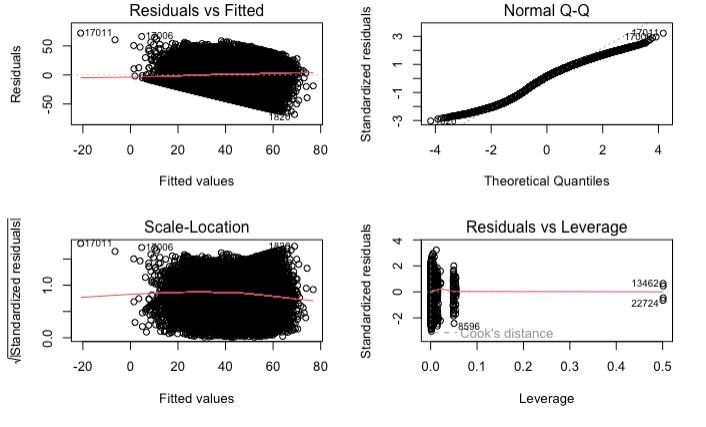


##### Model 2:

All variables with no polynomials.

We find that significant variables are the same as the previous model when predicting song popularity. R2 = 0.193, AIC = 297203.2, BIC = 297791.1. We find that despite our initial explorations, adding polynomial terms did not increase the model’s performance significantly.

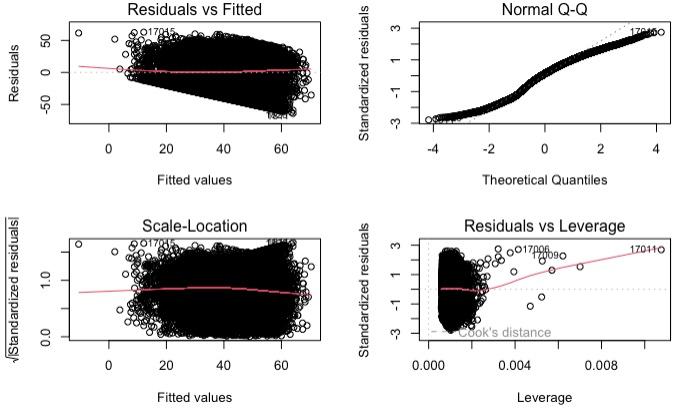
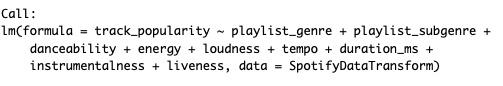




##### Model 3:

Use only significant variables from model 1 and 2. no polynomials.

We find that significant variables are the same as the previous model when predicting song popularity. R2 = 0.193, AIC = 298976.2, BIC = 299244.9. We find that although the variables we removed from this model are not significant, they still help in explaining some of the variability of the model. In addition, our error measures increase.

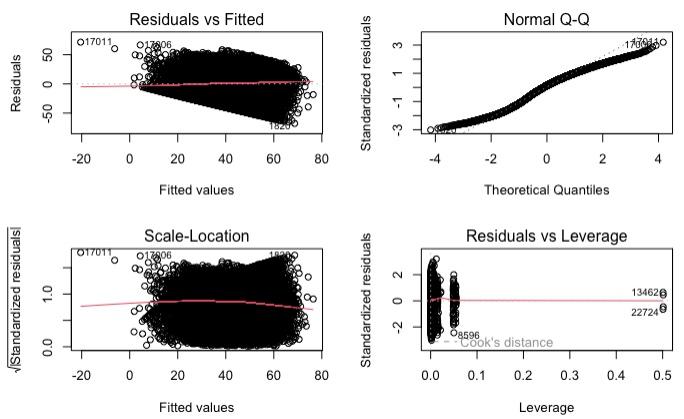
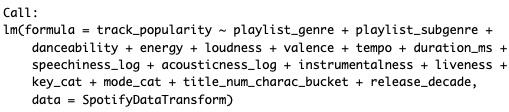


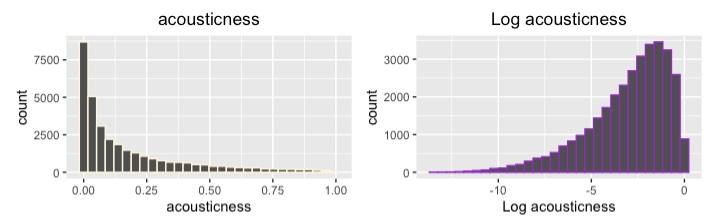
##### Model 4:

All variables. no polynomials. Log transformation on predictors non-normal predictors.

Our initial exploration we found that some variables are not normally distributed. We tried a variety of transformations but found that they mostly had no effect. The exceptions were speechiness and acousticness. We decided to add those as predictors to the model.

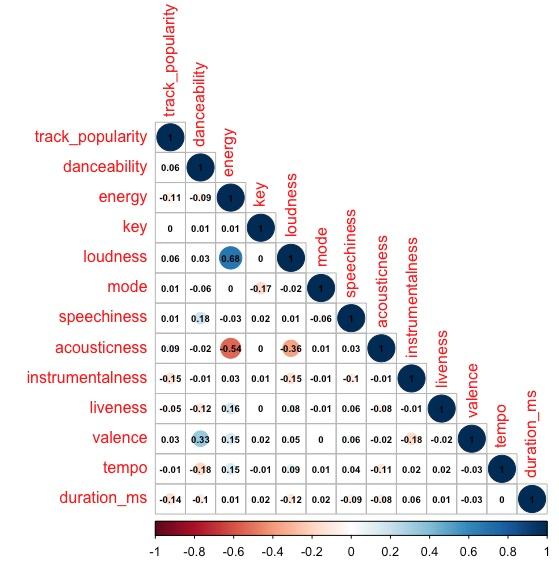
We find that acousticness becomes significant after the transformation. R2 = 0.1931, AIC = 297195.6, BIC = 297783.9. We see a slight improvement.





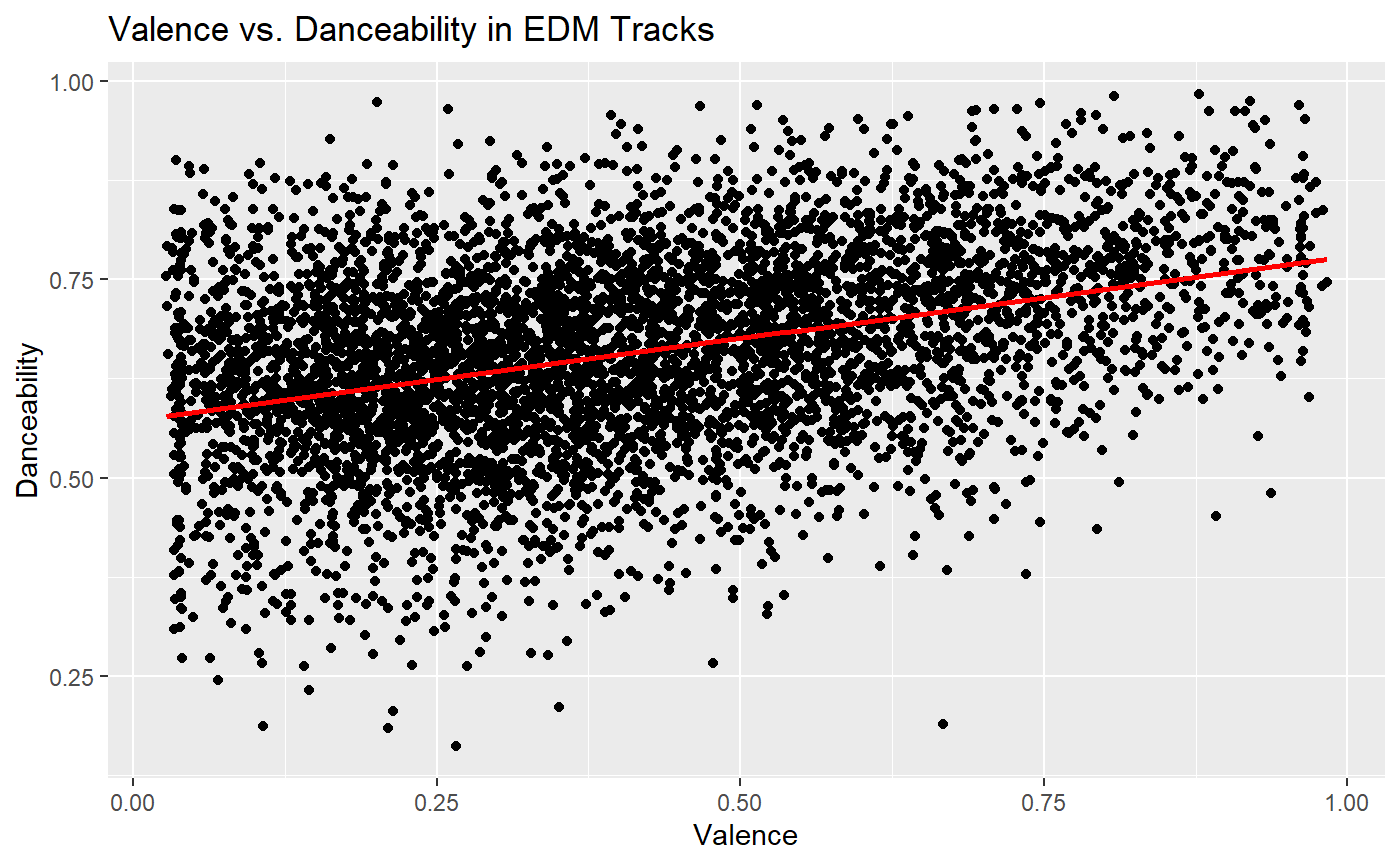
#### Research question 1c results

The correlation matrix shows that most of our numeric variables are uncorrelated. One notable exception is energy, which is positively correlated with loudness and negatively correlated with acousticness. In addition, we see that valence and danceability are somewhat correlated.



The results of the correlation test show that there is a moderate positive correlation between danceability and valence for tracks in the EDM genre. The correlation coefficient is 0.38, indicating that there is a positive relationship between the two features. The p-value for the correlation test is < 0.001, which is much smaller than the commonly used significance level of 0.05, indicating strong evidence against the null hypothesis. Therefore, we reject the null hypothesis and conclude that there is a significant correlation between danceability and valence in the EDM genre.

In conclusion, the analysis indicates that there is a moderate positive correlation between danceability and valence for tracks in the EDM genre. The correlation coefficient is 0.38, and the p-value is < 0.001. These results provide evidence for the alternative hypothesis that there is a significant correlation between danceability and valence in the EDM genre.

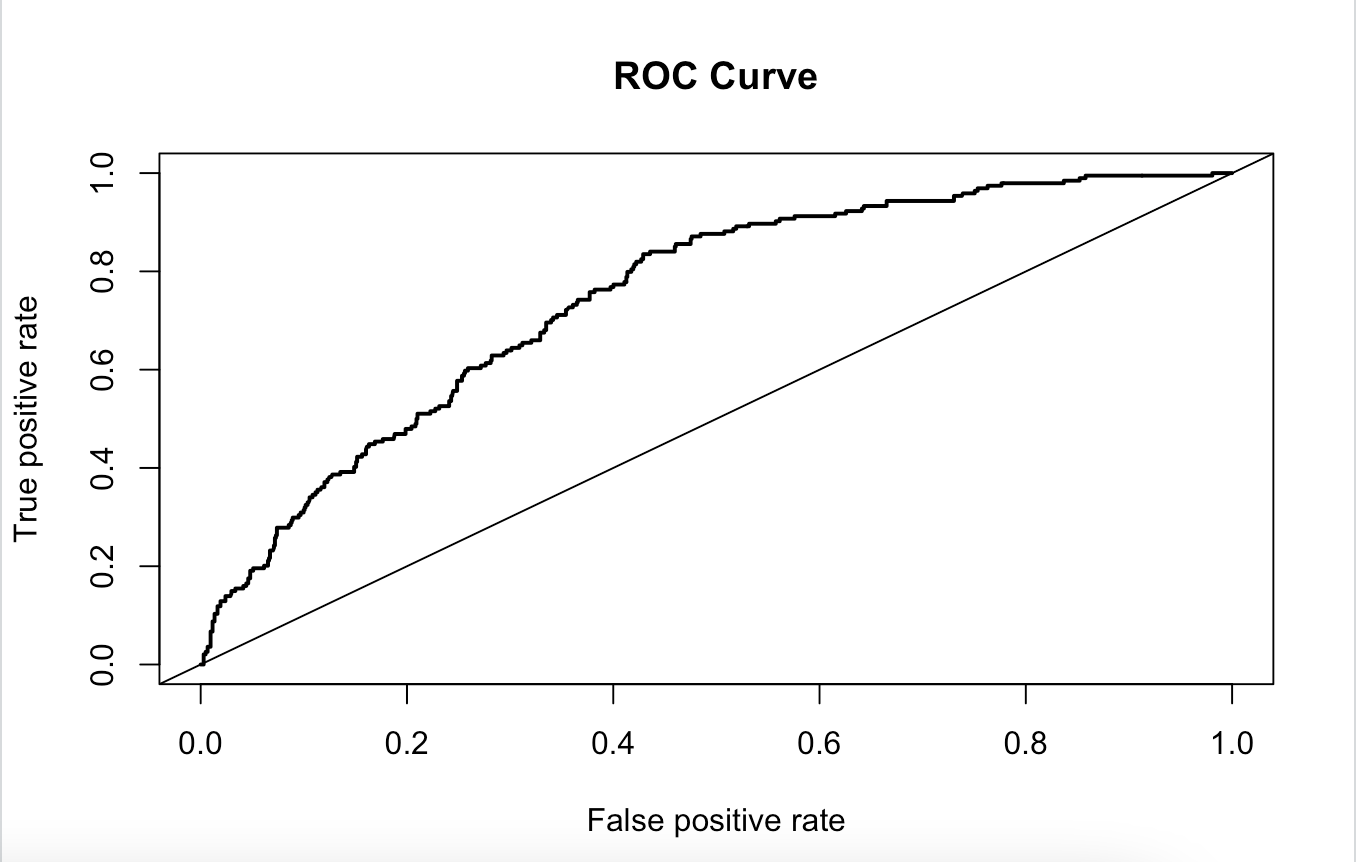


### Research Question 2:

Overall, we were successful in fitting a logistic regression model to predict whether or not a song came from the Spotify Top 500 or Data 557. The following table shows the estimated coefficients for all variables included in our final model, all found significant according to a Wald Test with robust standard errors at significance level alpha = 0.003.

| **Variable** | **Estimate** | **95% Lower** | **95% Upper** | **P-value** |
| --- | --- | --- | --- | --- |
| danceability | 3.584 | 2.999 | 4.178 | 2.2e-16 |
| energy | -3.104 | -3.774 | -2.444 | 2.2e-16 |
| loudness | 0.225 | 0.181 | 0.269 | 2.2e-16 |
| acousticness | -0.853 | -1.220 | -0.493 | 1.998e-06 |
| instrumentalness (factor: >0.02) | -1.813 | -2.155 | -1.497 | 2.2e-16 |
| valence | -1.469 | -1.844 | -1.097 | 8.09e-16 |
| mode (factor: Minor) | 0.335 | 0.186 | 0.483 | 1.122e-05 |

Using the model above, we split our data into a training set and a test set, refit our model on the training set, and then calculated accuracy, specificity, and sensitivity for the test set using a variety of thresholds for classifying a fitted value as a success. Using an ROC curve (shown below), we determined that our best classification threshold was 46.8% resulting in an AUC of 0.748. Using the threshold of 46.8% labeling a prediction as being from the Spotify Top 500, we achieved an accuracy of 84.8%, a specificity of 98.7%, and a sensitivity of 9.8%.



Although we were able to achieve a high accuracy (84.8%) and high AUC (0.748), the data set on which we fit the model contained a much larger number of observations from the Data 557 group than it did the Spotify Top 500 group. If we were to instead predict that an observation was from Data 557 100% of the time, we get an accuracy of 84.3%, nearly identical to the accuracy of our model. Even though the accuracies of these two prediction methods are similar, our model has the advantage of having interpretable terms and giving a more nuanced view as to why a song is either from Data 557 or Spotify Top 500.

For example, according to our model, increasing the average loudness of a song by 3 decibels (the smallest perceivable difference by humans) increases the odds of being from the Top 500 group by a factor of 1.97 (95% CI: 1.72, 2.24). Similarly, a song with an instrumentalness score greater than 0.02 is 0.16 times as likely to be from the Top 500 group compared to a song with an instrumentalness score less than 0.02 (95% CI: 0.11, 0.22). Finally, increasing the danceability score by 0.1 increases the odds of being from the Top 500 group by a factor of 1.43 (95% CI: 1.35, 1.51). We can interpret every term in the model in this manner, which gives much more insight into the nature of the Top 500 group than simply predicting that all songs are from Data 557. To summarize the findings of our model, a song that is most likely to be from the Data 557 group and not from the Spotify Top 500 group is one that has low danceability, high energy, low loudness, high acousticness, high instrumentalness, low valency, and that was written in a minor key.

## **Discussion:**

**ADD App limitations and Question 1 limitations**

This project began as an attempt to understand what factors contribute to a song’s success. While “success” in this context can be a bit vague, we decided to use popularity as a proxy. Both our datasets as well as 557 preferences, include songs that are ranked 0-100 based on how popular an artist is relative to other artists on Spotify.

We found several features that were significant when regressed on track popularity using higher order polynomials. Implying that the relationship between the features and popularity may not be linear. As such, we fit several models in different variations. The polynomial model performed only slightly better than the others. As such, we opted for the simpler model. There were, however, several features that were consistently significant to track popularity in all models: danceability, energy, loudness, tempo, duration and instrumentalness. Of the numeric values, danceability (positive), energy (negative) and instrumentalness (negative) have the highest effect on track popularity. In addition, genres play a large role in how popular a song is, with Pop, R&B and Rap having the largest impact.

Additionally, in fitting our logistic classification model, we ran into the limitation of our data source containing many more songs from Data 557 than it did the Spotify Top 500. We were surprised at the level of engagement we got from the class, and in hindsight, should have found a larger representation of popular music from Spoitfy to pair with the class data. This hindered our ability to assess the prediction power of our model. In future studies, data should be more evenly distributed between Data 557 and Spotify groups.Another area of future study with a song classification model could be creating a model to differentiate between two different genres of music. The musical distinctions between Data 557 and the Spotify Top 500 was at times unclear. There are multiple aspects to a song that make it popular, and there is no one combination of musical attributes that can ensure a popular song. Genre is more closely related to the musical attributes of a song, and thus a classification model would likely be more successful.

*References:*

* *List any books or articles you used. It is not necessary to do a lot of background research, so this could be short. If you use any statistical methods not discussed in class, they should be referenced.*

## **Table And Figures**

### **Spotify Features**

All of our data sources share the following common features shown in the table below.

| Spotify Features | | |
| --- | --- | --- |
| Variable Name | Description | Comment or transformation |
| Track\_name |  |  |
| Track\_artist |  |  |
| track\_popularity | Song popularity from 0 - 100 where the larger number is better | Dependent variable. |
| playlist\_genre | Playlist genre | Categorical |
| playlist\_subgenre | Playlist Subgenre | Categorical |
| danceability | 0 - 1.0 scale of how suitable a track is to dance to. |  |
| energy | 0 - 1.0 scale of a track’s perpetually measured activity and intensity where higher values are more energetic. |  |
| loudness | The average loudness of a track measured in decibels (dB). |  |
| valence | 0 - 1.0 scale that measures the positiveness conveyed from the track where 1 is positive and 0 is negative. |  |
| tempo | Average estimated beats per minute (BPM) for a track. |  |
| duration\_ms | Song duration in milliseconds. |  |
| speechiness | 0 - 1.0 of how much of a track consists of words where the higher value likely is a voice recording. | Log transformation to normalize |
| acousticness | 0 - 1.0 scale that measures how likely a track is to be acoustic. | Log transformation to normalize |
| instrumentalness | 0 - 1.0 scale that measures how likely a track contains any vocals where 1.0 is a track without vocals. |  |
| liveness | Detects likelihood of the track having an audience in the recording. |  |
| key | The average key/pitch of a track. Integers map pitches with pitch class notation. | Categorical |
| mode | Indicates whether a track is a major or minor with major equally 1 and minor equal to 0. | Indicator |
| title\_num\_charac\_bucket | Number of characters in the title |  |
| release\_decade | Release Decade |  |

*Appendices:*

* *If you need to include any more technical information about your analyses, please include it here. This is not necessary to include.*

*Code:*

* *Provide all the code needed to reproduce the analyses. Code does not count to the 20-page limit.*

## **Code**

library(spotifyr)

library(lubridate)

library(tidyr)

library(rjson)

library(readxl)

library(janitor)

library(dplyr)

library(tidyverse)

library(ggplot2)

library(corrplot)

library(reshape2)

library(cowplot)

library(ggpubr)

library(ggcorrplot)

library(ggstatsplot)

library(plyr)

library(rstatix)

# Get Data

SpotifyData <- read.csv("~/Documents/College Docs/Grad School/DATA 557/Project/SpotifyData.csv")

#View(SpotifyData)

SpotifyDataTransform <- SpotifyData

#### Look at numerics

SpotifyDataNumeric <- SpotifyData %>%

select(track\_popularity, danceability, energy,key, loudness, mode,speechiness,acousticness,instrumentalness,liveness, valence,tempo,duration\_ms)

#### Data Transformations

SpotifyDataTransform$track\_popularity\_log <- log(SpotifyDataNumeric$track\_popularity)

SpotifyDataTransform$track\_popularity\_log[is.infinite(SpotifyDataTransform$track\_popularity\_log)] <- 0

SpotifyDataTransform$speechiness\_log <- log(SpotifyDataNumeric$speechiness)

SpotifyDataTransform$speechiness\_log[is.infinite(SpotifyDataTransform$speechiness\_log)] <- 0

SpotifyDataTransform$acousticness\_log <- log(SpotifyDataNumeric$acousticness)

SpotifyDataTransform$acousticness\_log[is.infinite(SpotifyDataTransform$acousticness\_log)] <- 0

SpotifyDataTransform$instrumentalness\_log <- log(SpotifyDataNumeric$instrumentalness)

SpotifyDataTransform$instrumentalness\_log[is.infinite(SpotifyDataTransform$instrumentalness\_log)] <- 0

SpotifyDataTransform$liveness\_log <- log(SpotifyDataNumeric$liveness)

SpotifyDataTransform$liveness\_log[is.infinite(SpotifyDataTransform$liveness\_log)] <- 0

SpotifyDataTransform$key\_cat <- factor(SpotifyDataTransform$key)

SpotifyDataTransform$mode\_cat <- factor(SpotifyDataTransform$mode)

#### Title Track Count

title\_num\_charac <- nchar(SpotifyData$track\_name)

title\_num\_charac\_bucket <- cut(x = title\_num\_charac, breaks = c(0,10, 20,30, 40,50, 60,70, 80,90, 100,110))

SpotifyDataTransform$title\_num\_charac\_bucket <-title\_num\_charac\_bucket

eleven <- ggplot(SpotifyDataTransform, aes(x=title\_num\_charac\_bucket, y=track\_popularity, color = title\_num\_charac\_bucket)) +

geom\_boxplot() +

ggtitle("Track Popularity by Title Track Number Of Characters") +

xlab("Title Track Number Of Characters") + ylab("Track Popularity") +

theme(plot.title = element\_text(hjust = 0.5))

twelve <-ggplot(SpotifyDataTransform, aes(x = factor(title\_num\_charac\_bucket), color = title\_num\_charac\_bucket)) +

geom\_bar() +

ggtitle("Title Track Number Of Characters", ) +

xlab("Title Track Number Of Characters") + ylab("Track Count") +

theme(plot.title = element\_text(hjust = 0.5))

### Research Question 2 Code:

```{r setup, include=FALSE}

knitr::opts\_chunk$set(echo = TRUE)

library(tidyverse)

library(janitor)

library(sandwich)

library(lmtest)

library(ROCR)

library(readxl)

```

```{r}

key\_ref <- read\_xlsx("./Key Mode Ref.xlsx") %>%

clean\_names()

```

```{r}

# Read in data, recode response variable

master <- read.csv("./data\_final\_combined.csv") %>%

mutate(source\_int = ifelse(source == "data\_557", 0, 1)) %>%

mutate(time\_signature\_factor = ifelse(time\_signature == 0, "No Time Signature Detected", str\_c(time\_signature, "/4"))) %>%

mutate(instrumentalness\_bin = ifelse(instrumentalness > 0.02, "> 0.02", "<= 0.02")) %>%

left\_join(key\_ref %>% select(key, key\_factor\_2, key\_mode\_factor\_2))

names(master)[2:22]

ggplot(master) +

geom\_boxplot(aes(x = source, y = danceability, fill = source))

ggplot(master) +

geom\_boxplot(aes(x = source, y = energy, fill = source))

ggplot(master) +

geom\_bar(aes(x = source, fill = key\_mode\_factor\_2), position = "fill", color = "black") +

labs(fill = "key\_mode\_factor")

ggplot(master) +

geom\_bar(aes(x = source, fill = mode\_factor), position = "fill", color = "black")

ggplot(master) +

geom\_boxplot(aes(x = source, y = loudness, fill = source))

ggplot(master) +

geom\_boxplot(aes(x = source, y = instrumentalness, fill = source))

ggplot(master) +

geom\_bar(aes(x = source, fill = instrumentalness\_bin), position = "fill", color = "black")

ggplot(master) +

geom\_boxplot(aes(x = source, y = valence, fill = source))

ggplot(master) +

geom\_bar(aes(x = source, fill = time\_signature\_factor), position = "fill", color = "black")

```

```{r}

0.05/20

fit\_saturated <- glm(source\_int ~ danceability + energy + loudness +

speechiness + acousticness + instrumentalness\_bin + liveness +

valence + mode\_factor + tempo + duration\_ms + time\_signature\_factor,

data = master, family = "binomial")

summary(fit\_saturated)

# Fit full model, based on which coefficients were initally significant at the 0.05 level

fit\_full <- glm(source\_int ~ danceability + energy + loudness +

acousticness + instrumentalness\_bin +

valence + mode\_factor,

data = master, family = "binomial")

summary(fit\_full)

# We have 7 variables in the model, we want to test if each one of them is significant - for each predictor, we will do wald test where reduced model is full model excluding that variable. We use a bonferoni correction and want alpha = 0.5/7.

fit\_red\_danceability <- update(fit\_full, . ~ . - danceability)

fit\_red\_energy <- update(fit\_full, . ~ . - energy)

fit\_red\_loudness <- update(fit\_full, . ~ . - loudness)

fit\_red\_acousticness <- update(fit\_full, . ~ . - acousticness)

fit\_red\_instrumentalness\_bin <- update(fit\_full, . ~ . - instrumentalness\_bin)

fit\_red\_valence <- update(fit\_full, . ~ . - valence)

fit\_red\_mode\_factor <- update(fit\_full, . ~ . - mode\_factor)

# fit\_red\_danceability - pval = 2.2e-16

waldtest(fit\_red\_danceability, fit\_full, test = "Chisq", vcov = vcovHC)

# fit\_red\_energy - pval = 2.2e-16

waldtest(fit\_red\_energy, fit\_full, test = "Chisq", vcov = vcovHC)

# fit\_red\_loudness - pval = 2.2e-16

waldtest(fit\_red\_loudness, fit\_full, test = "Chisq", vcov = vcovHC)

# fit\_red\_acousticness - pval = 1.998e-06

waldtest(fit\_red\_acousticness, fit\_full, test = "Chisq", vcov = vcovHC)

# fit\_red\_instrumentalness - pval = 2.2e-16

waldtest(fit\_red\_instrumentalness\_bin, fit\_full, test = "Chisq", vcov = vcovHC)

# fit\_red\_valence - pval = 8.09e-16

waldtest(fit\_red\_valence, fit\_full, test = "Chisq", vcov = vcovHC)

# fit\_red\_mode\_factor - pval = 1.122e-05

waldtest(fit\_red\_mode\_factor, fit\_full, test = "Chisq", vcov = vcovHC)

# We accept all of the variables into the model

# Look at a residual plot

master <- master %>% mutate(fitted = fit\_full$fitted.values,

residuals = fit\_full$residuals,

fitted.bin = ntile(fitted, n = 30))

master %>% group\_by(fitted.bin) %>%

summarize(mean\_resid = mean(residuals),

mean\_fitted = mean(fitted)) %>%

ggplot() +

geom\_point(aes(x = mean\_fitted, y = mean\_resid))

```

```{r}

set.seed(20230307)

# Splitting data up into train and test sets

index <- sample(1:nrow(master), round(nrow(master)\*.2))

master <- master %>%

mutate(group = ifelse(row\_number() %in% index, 1, 0))

train\_set <- master %>% filter(group == 0)

test\_set <- master %>% filter(group == 1)

fit\_test <- glm(source\_int ~ danceability + energy + loudness +

acousticness + instrumentalness\_bin +

valence + mode\_factor,

data = train\_set, family = "binomial")

summary(fit\_test)

test\_prob <- predict(fit\_test, newdata = test\_set, type = "response") %>% unname()

test\_set <- test\_set %>%

mutate(test\_prob = test\_prob)

# Using ROC curve to determine highest accuracy

pred <- prediction(test\_prob, test\_set$source\_int)

perf <- performance(pred, "acc")

plot(perf, main = "Accuracy ~ Threshold Plot")

abline(v = cutoff, col = "red")

max\_ind <- which.max(slot(perf, "y.values")[[1]] )

acc <- slot(perf, "y.values")[[1]][max\_ind]

cutoff <- slot(perf, "x.values")[[1]][max\_ind]

print(c(accuracy= acc, cutoff = cutoff))

roc <- performance(pred,"tpr","fpr")

plot(roc, colorize = F, lwd = 2, main = "ROC Curve")

abline(a = 0, b = 1)

auc\_ROCR <- performance(pred, measure = "auc")

auc\_ROCR@y.values[[1]]

# How does our model do at classifying songs with the determined cutoff

test\_set <- test\_set %>%

mutate(guess = test\_prob > cutoff, 1, 0)

test\_set %>% filter(source\_int == 1) %>%

summarize(sensitivity = mean(source\_int == guess))

test\_set %>% filter(source\_int == 0) %>%

summarize(specificity = mean(source\_int == guess))

test\_set %>%

summarize(accuracy = mean(source\_int == guess),

always\_577\_accuracy = mean(source\_int == 0))

```

```{r}

summary(fit\_full)

confint(fit\_full, vcov = vcovHC)

# For variables on scale of 0 to 1

exp(coef(fit\_full) \* .1)

exp(confint(fit\_full) \* .1)

# For loudness, measured in dbs

options(scipen=999)

exp(coef(fit\_full) \* 3)

exp(confint(fit\_full) \* 3)

# For factor variables

exp(coef(fit\_full) \* 1)

exp(confint(fit\_full) \* 1)

```

```{r, eval = F}

version <- data.frame(danceability = 0.24,

energy = 0.519,

loudness = -10.2,

acousticness = 0.0112,

instrumentalness\_bin = "> 0.02",

valence = 0.113,

mode\_factor = "Minor")

predict(fit\_full, newdata = version, type = "response", se.fit = TRUE)

pred\_intervals(fit\_full, newdata = version, type = "response")

``

```{R}

# Load the necessary libraries

library(tidyverse)

library(rstatix)

library(dplyr)

library(tidyr)

library(broom)

library(ggplot2)

# Read in the Spotify dataset

spotify <- read\_csv("SpotifyData.csv")

head(spotify)

# create a contingency table

playlist\_table <- spotify %>%

select(playlist\_genre, playlist\_name) %>%

drop\_na() %>%

table()

# conduct chi-square test

chi\_sq\_result <- chisq.test(playlist\_table)

# print the results

chi\_sq\_result

# From the results of the chi-square test, it was found that there is a significant association between the playlist genre and playlist name variables in the Spotify dataset. The null hypothesis of independence was rejected, indicating that there is a relationship between these variables.

# Perform the ANOVA

anova\_result <- spotify %>%

anova\_test(track\_popularity ~ playlist\_genre) # Specify the continuous variable and categorical variable of interest

# View the ANOVA results

anova\_result

#The ANOVA table (type II tests) shows that there is a significant main effect of playlist genre on track popularity (F(5, 32827) = 207.209, p < .05). The effect size, measured by generalized eta squared (ges), is 0.031, indicating a small but statistically significant effect of playlist genre on track popularity. Overall, the results suggest that playlist genre has a significant influence on track popularity in the Spotify dataset.

# Filter the dataset to only include the two playlists of interest

spotify\_dance <- spotify %>% filter(playlist\_name == "Dance Pop")

spotify\_electro <- spotify %>% filter(playlist\_name == "ELECTROPOP")

# Perform t-tests on track popularity for each playlist

edm\_t\_test <- t.test(spotify\_dance$track\_popularity)

vocal\_edm\_t\_test <- t.test(spotify\_electro$track\_popularity)

# Format and print the output using the broom package

tidy(edm\_t\_test)

tidy(vocal\_edm\_t\_test)

# Filter the dataset to only include the two playlists of interest

spotify\_dance <- spotify %>% filter(playlist\_name == "Dance Pop")

spotify\_electro <- spotify %>% filter(playlist\_name == "ELECTROPOP")

# Perform Welch t-tests on track popularity for each playlist

edm\_t\_test <- t.test(spotify\_dance$track\_popularity,

spotify\_electro$track\_popularity)

# Format and print the output using the broom package

tidy(edm\_t\_test)

# Create a data frame with the two playlists

dance\_vs\_electropop <- data.frame(

Playlist = c(rep("Dance Pop", nrow(spotify\_dance)), rep("ELECTROPOP", nrow(spotify\_electro))),

Track\_Popularity = c(spotify\_dance$track\_popularity, spotify\_electro$track\_popularity)

)

# Create a box plot of track popularity for each playlist

ggplot(dance\_vs\_electropop, aes(x = Playlist, y = Track\_Popularity)) +

geom\_boxplot() +

xlab("Playlist") +

ylab("Track Popularity") +

ggtitle("Distribution of Track Popularity for Dance Pop and Electropop Playlists") +

theme(plot.title = element\_text(hjust = 0.5))

#For the "Dance Pop" playlist, the t-test result shows a mean track popularity of 60.72464, a t-statistic of 31.83259, and a very small p-value of 3.625261e-65, indicating a highly significant difference in mean track popularity between this playlist and the overall population. The 95% confidence interval for the mean difference is [56.95244, 64.49683].

#For the "ELECTROPOP" playlist, the t-test result shows a mean track popularity of 46.425, a t-statistic of 14.31418, and a very small p-value of 1.194039e-23, indicating a highly significant difference in mean track popularity between this playlist and the overall population. The 95% confidence interval for the mean difference is [39.9694, 52.8806].

#Overall, the results suggest that both "Dance Pop" and "ELECTROPOP" playlists have significantly different mean track popularity compared to the overall population. However, as these are independent t-tests, they do not provide information about whether there is a significant difference in mean track popularity between the two playlists themselves. Additional analysis, such as a paired t-test or a confidence interval for the difference in means, would be needed to assess this.

# Filter the dataset to include only the two playlists of interest

spotify\_dance <- spotify %>% filter(playlist\_name == "Dance Pop")

spotify\_electro <- spotify %>% filter(playlist\_name == "ELECTROPOP")

# Perform an unpaired t-test on track popularity for the two playlists

unpaired\_t\_test <- t.test(spotify\_dance$track\_popularity, spotify\_electro$track\_popularity, var.equal = FALSE)

# Format and print the output using the broom package

tidy(unpaired\_t\_test)

# Calculate the confidence interval for the difference in means

mean\_diff <- mean(spotify\_dance$track\_popularity) - mean(spotify\_electro$track\_popularity)

se\_diff <- sqrt(var(spotify\_dance$track\_popularity)/nrow(spotify\_dance) + var(spotify\_electro$track\_popularity)/nrow(spotify\_electro))

lower\_ci <- mean\_diff - qt(0.975, unpaired\_t\_test$parameter) \* se\_diff

upper\_ci <- mean\_diff + qt(0.975, unpaired\_t\_test$parameter) \* se\_diff

cat("The 95% confidence interval for the difference in means is [", lower\_ci, ", ", upper\_ci, "].", sep = "")

#The estimate value of 46.425 suggests that, on average, tracks in the "Dance Pop" playlist are more popular than tracks in the "ELECTROPOP" playlist. The p-value of 6.694111e-05 indicates that this difference is statistically significant, meaning that it is unlikely to have occurred by chance alone.

#The confidence interval for the difference in means is [6.883318, 21.71596], indicating that we are 95% confident that the true difference in means between the two playlists falls between these values.

#Overall, these results suggest that there is a significant difference in track popularity between the two playlists, with the "Dance Pop" playlist having higher popularity on average than the "ELECTROPOP" playlist.

# Filter tracks to only include EDM genre

edm\_tracks <- spotify %>%

filter(playlist\_genre == "edm")

# Create scatterplot with trend line

ggplot(data = edm\_tracks, aes(x = valence, y = danceability)) +

geom\_point() +

geom\_smooth(method = "lm", se = FALSE, color = "red") +

labs(x = "Valence", y = "Danceability") +

ggtitle("Valence vs. Danceability in EDM Tracks")

```