What Makes Songs Danceable: An Analysis of Popular Tracks from 2000-2019

Haram Yoon and Shane Foster Smith

```r
# Load tidyverse
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## v readr     2.1.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
# Read in song data
song_data <- read_csv("songs_normalize.csv")
```

```
## Rows: 2000 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): artist, song, genre
## dbl (14): duration_ms, year, popularity, danceability, energy, key, loudness...
## lgl  (1): explicit
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
#split data into training and testing data
set.seed(4774)
spec(song_data)
```

```
## cols(
##   artist = col_character(),
##   song = col_character(),
##   duration_ms = col_double(),
##   explicit = col_logical(),
##   year = col_double(),
##   popularity = col_double(),
##   danceability = col_double(),
##   energy = col_double(),
##   key = col_double(),
```

```
##   loudness = col_double(),
##   mode = col_double(),
##   speechiness = col_double(),
##   acousticness = col_double(),
##   instrumentalness = col_double(),
##   liveness = col_double(),
##   valence = col_double(),
##   tempo = col_double(),
##   genre = col_character()
## )
```

Introduction

Dataset: Paradisejoy. (n.d.). Top Hits Spotify from 2000-2019. Kaggle. Retrieved from https://www.kaggle.com/datasets/paradisejoy/top-hits-spotify-from-20002019 The Spotify songs dataset contains various audio features and metadata on over 2000 popular tracks on the platform from 2000 to 2019. The primary goal of our study is to investigate the factors that influence the "danceability" of a song. Danceability is the measure of how suitable a track is for dancing based on a combination of musical elements.

Our motivation behind analyzing danceability specifically from the fact that highly danceable songs tend to be more widely appreciated, particularly in certain genres like pop, electronic dance music (EDM), and hip-hop. We first thought of predicting popularity of a songs, but that soon turned out to be less feasible and valuable compared to danceability. Furthermore, danceable tracks tend to be more enjoyable for listeners due to their higher energy level, a strong rhythm, and a tempo that aligns with typical dancing speeds. These danceable tracks are espeically popular in nightclubs, parties, and other social settings.

From a commercial perspective, creating danceable hits can be highly beneficial for artists and music labels. Tracks with high danceability tend to perform well on music streaming platforms which make them more likely to be added to playlists. Furthermore, danceable songs often stay popular for longer, as they remain relevant for dance-related events and activities over an extended period.

By analyzing the factors that contribute to danceability, our study aims to provide insights that can guide music producers and artists in crafting more danceable tracks tailored to the preferences of their target audience. The findings could also be valuable for music streaming platforms, radio stations, and other music curators in identifying and promoting highly danceable tracks, thereby tailoring a song to a particular social setting.

In this dataset, the response variable of interest is the "danceability" score, which ranges from 0 to 1, with higher values indicating more danceable tracks. The predictor variables include both numerical features related to the audio characteristics of the songs (e.g., energy, valence, tempo, acousticness, speechiness) and categorical variables like genre, mode and explicit content.

The original data was collected by Spotify, a popular music streaming service, and likely consists of tracks that were popular or frequently streamed on the platform during the given time period. However, the specific details on how the data was collected and processed are not provided.

The dataset includes 2000 observations (songs) and 18 variables, including the artist name, song title, duration, year of release, popularity score, and various audio features computed by Spotify's algorithms. These audio features, such as danceability, energy, valence, and acousticness, are numerical values ranging from 0 to 1 and are designed to capture different aspects of the audio content.

It's worth noting that some variables in the dataset may require further processing or transformation before being included in the statistical models. For instance, the "genre" variable contains multiple genres for some songs, and the "explicit" variable is binary, indicating whether the song contains explicit content. Variables

like “acousticness” and “speechiness” may also need to be explored further to understand their potential impact on danceability.

Moreover, interactions between variables could be important in predicting danceability. For example, the effect of tempo on danceability might vary across different genres, or the influence of energy on danceability could be moderated by the explicit content of a song.

By carefully analyzing and preprocessing these variables, as well as considering potential interactions and transformations, this analysis aims to develop effective statistical models that can accurately predict the danceability of a song based on its audio features and metadata.

By analyzing this dataset, we aim to identify the key factors that contribute to a song’s danceability and develop statistical models that can potentially predict this measure based on the available audio features and metadata. The findings of this analysis could provide valuable insights for music producers, artists, and streaming platforms to create and promote more danceable tracks, catering to the preferences of their audience.

Feature Engineering

To preprocess the data, the first step is to deal with any missing or NA data. Thankfully, there was none. Our next step was to change some of the categorical variables into usable datatypes in our models (binary representations or integers rather than booleans or characters). For example, the ‘Genre’ feature of a song was a string that could contain one or more genres out of the 14 possible genres(such as, “pop, hip hop”). To make this feature usable in our model, we performed two operations; first, we coalesced the nine least frequency genres into a single “Other” genre, which important to ensure dimensionality reduction and because these genres were very infrequent compared to the top five most frequent genres. After, we performed a one-hot encoding of the ‘Genre’ feature (into 6 binary features), which indicates if a song is part of one or more genres.
Second, most of the original features have values in the range [0,1], but a few features had values much larger or in negative ranges. We normalized some of these features’ values, while also keeping into consideration the interpretability of our models’ units. For example, two of the variables we normalized where ‘duration_ms’ and ‘loudness’. ‘Duration_ms’ original feature values where in the approximate range of [100000,400000] milliseconds. So, in order to both keep the original units and to do length normalization, we divided every duration by a constant 100,000. ‘Loudness’, on the other hand, had feature values in the approximate range of [-20,0] decibels. To normalize these values, we first determined that the original units weren’t particularly informative (what really is a one dB increase of loudness?). Instead, we decided to use the scale() function so that new units are standard deviations from the mean. How loud a song is relative to other songs in our dataset is more informative and help us normalize these values.

#1. Preprocessing and Handling Important Categorical Variables

```r
# Look for Na Values -> found none so we don't need to worry about those
na_counts <- colSums(is.na(song_data))
na_counts
```

```
##           artist             song      duration_ms         explicit
##                0                0                0                0
##             year       popularity     danceability           energy
##                0                0                0                0
##              key         loudness             mode      speechiness
##                0                0                0                0
##     acousticness instrumentalness         liveness          valence
##                0                0                0                0
##            tempo            genre
##                0                0
```

```r
#Scale the duration, so it is closer with other feature values
song_data$duration_scale <- song_data$duration_ms / 100000

#Scale loudness, interpret now in std-deviations
song_data$loudness_scale <- scale(song_data$loudness)[,1]

str(song_data$loudness_scale)
```

```
##  num [1:2000] 0.0354 0.3074 -1.8074 0.7496 0.3654 ...
```

```r
# Turn explicit into numeric type
song_data$explicit_numeric <- as.numeric(song_data$explicit)

#Change keys into category: either major or minor key
song_data <- song_data %>%
  mutate(key_numeric = case_when(
    key == -1 ~ 0,          # unknown key
    key %% 2 == 0 ~ 1,      # even key (C, D, ...)
    key %% 2 == 1 ~ -1      # odd keys (C#, D#, ...)
  ))

genres_to_replace <- c("metal", "latin", "country", "Folk/Acoustic",
                       "World/Traditional", "easy listening", "blues",
                       "jazz", "classical")

# Replace low frequency genres with 'Other' and handle multiple genres per song
song_data <- song_data %>%
  mutate(genre = str_split(genre, ",\\s*")) %>%
  mutate(genre = map(genre, ~if_else(. %in% genres_to_replace, "Other", .))) %>%
  mutate(genre = map(genre, ~unique(.))) %>%
  mutate(genre = map_chr(genre, paste, collapse=", "))

song_data_clean <- song_data %>%
  mutate(genre = str_split(genre, ",\\s*")) %>%  # Split genres into a list
  filter(!map_lgl(genre, ~"set()" %in% .)) %>%  # Filter out rows containing 'set()'
  mutate(genre = map_chr(genre, paste, collapse=", "))  # Collapse back to a single string


original_song_data <- song_data_clean

encoded_data <- original_song_data %>%
  mutate(row_id = row_number()) %>%
  separate_rows(genre, sep = ",\\s*") %>%  # Split genres into separate rows
  mutate(presence = 1) %>%  # Binary indicator for presence
  pivot_wider(names_from = genre, values_from = presence, values_fill = list(presence = 0)) %>%
  dplyr::select(-row_id)
```

```r
# Dancebility on Genre
original_song_data %>%
  separate_rows(genre, sep = ",\\s*") %>%  # Split genres into separate rows based on comma
  ggplot(aes(x = danceability)) +           # Use popularity as the x-axis
  geom_histogram(bins = 30, fill = "steelblue", color = "black") +  # Adjust bin count as needed
  facet_wrap(~genre, scales = "free_y") +  # Create separate plots for each genre
```

```
  labs(title = "Danceability Distribution by Genre",
       x = "Danceability",
       y = "Count") +
  theme_minimal()
```

By observation of the histograms, it appears that the frequency of danceability does slightly vary depending on the genre. For example, 'rock' and 'Other' peaks (or means) are slightly lower in the histograms than they are for 'Dance/Electronic' and 'Hip Hop'. Also, for 'Pop' and 'R&B', we can see that their songs are more frequenty in the [0,0.5] danceability range than songs in ''Dance/Electronic' and 'Hip Hop'. These results indicate that dance/electronic and hop hip songs may be considered more danceable, and perhaps deserve special consideration in our models.

Genre Counts

```
original_song_data %>%
  separate_rows(genre, sep = ",\\s*") %>%
  filter(genre != "set()") %>%
  count(genre) %>%
  slice_max(n, n = 6) %>%
  ggplot(aes(x = fct_reorder(genre, n), y = n)) +
  geom_col() +
  labs(
    title = "Top Genres",
```

```
    x = "Genre",
    y = "Count"
  )
```

The bar plot shows 'pop' as the most represented genre in the dataset, followed closely by 'hip hop'. Given the prominence of these highly danceable genres, the dataset is well-suited for analyzing the factors contributing to danceability across different musical styles.

Correlation HeatMap

```
# Create correlation matrix

library(viridis)
```

```
## Loading required package: viridisLite
```

```
cor_mat <- cor(encoded_data %>%
                select_if(is.numeric))

# Use reshape2 to melt the matrix
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
melted_cor <- melt(cor_mat)

# Plot heatmap
ggplot(data = melted_cor, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  scale_fill_viridis(option = "magma")  +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(
    title="Correlation Heatmap",
    x = "Variable 1",
    y = "Variable 2"
  )
```

This plot shows the correlation between each numeric variable in our dataset (as a matrix heatmap). The darker colors show negative correlation while the warmer colors (more yellow) show more positive correlation. We can see that many variable in the heatmap are notably correlated (either positively or negatively) .

For example, the variable 'Energy' is positively correlated with loudness and very negatively correlated what acoustiness. Futhermore, loudess and acoustiness, unsurprisingly, are also very negative correlated. So, to avoid multicollinearity, we likely don't want to put more than one of these variables in our model without careful consideration.

Another example is, 'Speechiness', 'Explicit_numeric', and 'Hip Hop' are all positively correlated. This makes intuitive sense as we expect hip hop to be more 'wordy' and likely contain more explicit language.

Again, we have to be careful with including more than one of these variables in a model, considering their correlation.

To get a closer understanding of the correlations and relationships of our variables, we constructed a few pairs plots and considered some transformations of our variables.

```
##  [1] "artist"           "song"             "duration_ms"      "explicit"
##  [5] "year"             "popularity"       "danceability"     "energy"
##  [9] "key"              "loudness"         "mode"             "speechiness"
## [13] "acousticness"     "instrumentalness" "liveness"         "valence"
## [17] "tempo"            "duration_scale"   "loudness_scale"   "explicit_numeric"
## [21] "key_numeric"      "pop"              "rock"             "Other"
## [25] "hip hop"          "R&B"              "Dance/Electronic" "pop_high"
## [29] "pop_moderate"     "pop_low"
```

Pairs Plots

```r
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(ggplot2)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
#DATA TRANSFORMATIONS (Explanataion will come later in the report)

# encoded_data$log_popularity <- log(song_train$popularity + 1)
# encoded_data$exp_popularity <- 2^(song_train$popularity)
# encoded_data$sqrt_popularity <- sqrt(song_train$popularity)

encoded_data$log_speechiness<- log(encoded_data$speechiness)

encoded_data$valence_square<- (encoded_data$valence)^2

encoded_data$log_liveness<- log(encoded_data$liveness)
encoded_data$sqrt_liveness<- sqrt(encoded_data$liveness)

encoded_data$log_acousticness<- log(encoded_data$acousticness)

encoded_data$sqrt_instrumentalness<- (encoded_data$instrumentalness)^(1/4)
encoded_data$log_instrumentalness<- log(encoded_data$instrumentalness + 1, base = 10)
encoded_data$sqrtlog_instru <- (encoded_data$log_instrumentalness)^(1/2)
encoded_data$loglog_instru <- log(encoded_data$log_instrumentalness + 1, base = 10)
```

```r
encoded_data$danceability_square <- (encoded_data$danceability)^2
encoded_data$energy_square = (encoded_data$energy)^2
encoded_data$tempo_sqrt = sqrt(encoded_data$tempo)


# Split data for cross-validation, model builging and future predictions
song_split <- initial_split(encoded_data, prop = .90)
song_train <- training(song_split)
song_test <- testing(song_split)

ggpairs(song_train, columns = c("danceability", "acousticness", "liveness", "speechiness", "energy"),
        title = "Pairwise Plot of Song Characteristics",
        upper = list(continuous = wrap("cor", size = 4)),
        lower = list(continuous = wrap("smooth", method = "lm", color = "blue", se = TRUE)))
```

```r
ggpairs(song_train, columns = c("danceability", "valence", "loudness_scale", "energy_square", "log_acous
        title = "Pairwise Plot of Song Characteristics 2 (with transformations)",
        upper = list(continuous = wrap("cor", size = 4)),
        lower = list(continuous = wrap("smooth", method = "lm", color = "blue", se = TRUE)))
```

```
ggpairs(song_train, columns = c("danceability", "duration_scale", "tempo", "sqrt_instrumentalness", "pop
        title = "Pairwise Plot of Song Characteristics 3",
        upper = list(continuous = wrap("cor", size = 4)),
        lower = list(continuous = wrap("smooth", method = "lm", color = "blue", se = TRUE)))
```

From the pairs plots, it's evident that danceability exhibits varying degrees of correlation with several audio features. Notably, it shows moderate positive correlations with valence (0.4) and log_speechiness (0.208), suggesting that higher danceability tends to be associated with higher valence and speechiness levels, respectively. Conversely, danceability demonstrates weak negative correlations with variables such as acousticness (-0.078), energy (-0.1), and tempo (-0.184), implying that higher danceability may be associated with lower acousticness, energy, and tempo values. However, the correlations with loudness_scale, duration_scale, sqrt_instrumental, and popularity appear to be negligible, indicating minimal linear relationships between these variables and danceability. Even though certain variables exhibit negligible correlations with danceability, it's important to still consider them for model building due to their potential non-linear relationships, contextual importance, and role in mitigating multicollinearity to help create a more comprehensive understanding of the factors influencing danceability. Furthermore, we see evidence of non-constant variance, non-linearity and possible violations of the assumption of normal errors for certain predictors (in particular, look at first pair-wise plot of characteristics).

Transformations:

In our analysis, we applied logarithmic and square root transformations to certain variables to address the conditions of a normal model. These transformations helped to stabilize the variance of residuals and achieve a more symmetric distribution, thereby enhancing the adherence of our models to the assumptions of linear regression. Through these transformations, we aimed to improve the reliability and interpretability of our regression analyses.

No Transformations

```r
library(broom)
library(purrr)
# Confirm transformations / which predictors aren't following the normal errors model?
#(consider lm(danceability ~ some_variable)) residual plots to find out. Look for non-constant variace


# Model 1: Danceability vs Valence
m1 <- lm(danceability ~ valence, data = song_train)

# Model 9: Danceability vs Tempo
m10 <- lm(danceability ~ tempo, data = song_train)


# Set up a 2x2 layout for plots
par(mfrow = c(2, 2))

# Plot diagnostics for each model
plot(m1, main = "Danceability vs Valence")
```

```r
plot(m10, main = "Danceability vs Tempo")
```

In cases where variables meet the model conditions based on diagnostic plots, transformation are not necessary as the assumptions of linearity, normality, and constant variance are adequately satisfied. Therefore, applying transformations to these variables may introduce unnecessary complexity (while also losing interpretability) without yielding significant improvement in model performance.

Log Transformations

Danceability vs Liveness
Residuals vs Fitted
Residuals
Fitted values
Danceability vs Liveness
Q−Q Residuals
Standardized residuals
Theoretical Quantiles
Danceability vs Liveness
Scale−Location
√|Standardized residuals|
Fitted values
Danceability vs Liveness
Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage

Danceability vs Log_Liveness
Residuals vs Fitted
Residuals
Fitted values
Danceability vs Log_Liveness
Q−Q Residuals
Standardized residuals
Theoretical Quantiles
Danceability vs Log_Liveness
Scale−Location
√|Standardized residuals|
Fitted values
Danceability vs Log_Liveness
Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage

Danceability vs Acousticness
Residuals vs Fitted
Residuals
Fitted values
901
698
1334
Danceability vs Acousticness
Q−Q Residuals
Standardized residuals
Theoretical Quantiles
Danceability vs Acousticness
Scale−Location
√|Standardized residuals|
Fitted values
Danceability vs Acousticness
Residuals vs Leverage
Standardized residuals
Cook's distance
626
877
901
Leverage

Danceability vs Log_Acousticness
Residuals vs Fitted
Residuals
Fitted values
Danceability vs Log_Acousticness
Q−Q Residuals
Standardized residuals
Theoretical Quantiles
Danceability vs Log_Acousticness
Scale−Location
√|Standardized residuals|
Fitted values
Danceability vs Log_Acousticness
Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage

Danceability vs Speechiness
Residuals vs Fitted
Residuals
Fitted values
Danceability vs Speechiness
Q−Q Residuals
Standardized residuals
Theoretical Quantiles
Danceability vs Speechiness
Scale−Location
√|Standardized residuals|
Fitted values
Danceability vs Speechiness
Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage

However, we encountered skewed distributions and heteroscedasticity in certain predictor variables, as evidenced by the residual vs. fitted plots and scale-location plots. To address these issues, we opted to apply a logarithmic transformation to variables such as liveness, acousticness, and speechiness, which exhibited pronounced skewness. The rationale behind this transformation was to mitigate the influence of extreme values and achieve a more symmetric distribution, aligning with the assumptions of linear regression. After completing the transformations, we observed a marked improvement in the diagnostic plots. The previously skewed residual vs. fitted plots now displayed a more uniform distribution, indicative of a linear relationship between the predictor and response variables. Furthermore, the scale-location plots indicated a reduction in heteroscedasticity, suggesting enhanced adherence to the assumption of constant variance. In summary, the logarithmic transformation of skewed predictor variables yielded significant improvements in the linearity and homoscedasticity of our regression models, thereby enhancing their reliability and predictive capability.

Square Root and Quadratic Transformations

Danceability vs Instrumentalness
Residuals vs Fitted
Residuals
Fitted values
Danceability vs Instrumentalness
Q−Q Residuals
Standardized residuals
Theoretical Quantiles
Danceability vs Instrumentalness
Scale−Location
√|Standardized residuals|
Fitted values
Danceability vs Instrumentalness
Residuals vs Leverage
Standardized residuals
425
1608
59
Cook's distance
0.5
Leverage

Instrumentalness is heaviliy skewed, so we applied a sqrt() transformation to it and found that this transformation helped improve its constant variance and linearity. We chose this over log() as there were more extreme values, and sqrt() can moderate these values more effectively.

Further Improving Assumptions of Normal Errors Model

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

Studentized Residuals vs Fitted Values for energy
Studentized Residuals
2
0
−2
−4
0.64
0.66
0.68
0.70
0.72
Fitted Values

The previous residuals vs fitted values for the 'energy' predictor looked slightly quadtradic, and we confirmed this suspicion by looking at the studentized residuals vs fitted values plots. As seen above, when we don't square 'energy', the studentized residuals heavily favor the fitted values far below the mean (rather than above the mean), especially for large fitted values; this indicates non-linearity. When we look at the same plot, but for a linear model that squares 'energy', we see that there is still a bias towards fitted values below the mean, but it is reduced and there is an improvement in constant variance.

In fact, even after many of previous transformations on the predictors, we still noticed a bias toward studentized residuals below the mean (sometimes more than 2 standard deviations). Therefore, we decided to transform the response variable, 'danceability', into a quadtradic term. Below, you will see more studentized residuals vs fitted values plots that show further improvements to various predictors when we transform 'danceability' in this way (in terms of constant variance, linearity, and particularly normal errors).

Studentized Residuals vs Fitted Values for energy_square (Danceability Squared I
Studentized Residuals
Fitted Values
2
1
0
−1
−2
−3
0.40
0.45
0.50

Studentized Residuals vs Fitted Values for log_acousticness
Studentized Residuals
Fitted Values
2
0
−2
−4
0.625
0.650
0.675

Studentized Residuals vs Fitted Values for log_acousticness (Danceability Squared Response)
Studentized Residuals
Fitted Values
2
1
0
−1
−2
0.400
0.425
0.450
0.475
0.500

Studentized Residuals vs Fitted Values for log_liveness
Studentized Residuals
2
0
−2
−4
0.625
0.650
0.675
0.700
0.725
Fitted Values

Studentized Residuals vs Fitted Values for log_liveness (Danceability Squared Response)
Studentized Residuals
3
2
1
0
−1
−2
0.40
0.45
0.50
Fitted Values

Studentized Residuals vs Fitted Values for log_speechiness
Studentized Residuals
2
0
−2
0.65
0.70
0.75
Fitted Values

Studentized Residuals vs Fitted Values for log_speechiness (Danceability Squared Response)
Studentized Residuals
3
2
1
0
−1
−2
0.40
0.45
0.50
0.55
Fitted Values

```
## [[1]]
## TableGrob (2 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
##
## [[2]]
## TableGrob (2 x 2) "arrange": 2 grobs
```

```
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
##
## [[3]]
## TableGrob (2 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
##
## [[4]]
## TableGrob (2 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
##
## [[5]]
## TableGrob (2 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

Although transforming the response variable in this way makes the interprebility of our model more nuanced, the studentized residual plots show that this tradeoff is worth it for the signficant improvement in our normal errors model assumptions.

Popularity

```
## [1] "Danceability vs Popularity Plots"
```

Residuals vs Fitted
Residuals
Fitted values
901
698
1334
Q−Q Residuals
Standardized residuals
Theoretical Quantiles
Scale−Location
√|Standardized residuals|
Fitted values
Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage

Popularity, as seen above, is binomially split between songs that have 0 or close to 0 popularity and those who have moderate to high popularity. Due to this fact, and because we found the values of popularity weren't very correlated with our response, we decided perform a one hot encoding of popularity (aka. popularity is now a categorical variable, represented by three binary features: low popularity [0,25), moderate popularity [25, 75), and high popularity [75,100]).

Comparing two models / Cross Validation

With the previous analysis of the relationships in our model, and with various domain knowledge, we constructed two models we thought would explain dancebility well without introducing too much multicollinearity or overfitting. Consider the two models below shown below. For both models, we determined that the variables 'valence','energy_square' and 'log_liveness' were all correlated enough with our response, 'danceability', to always be included. Also, for both models, we determined that the categorical variable 'Dance/Eletronic' should also be included (given the name and previous histograms).
For the first model, we decided that a model that focuses more on tempo and instrumentalness is an intuitive approach to modeling danceability. In particular, we thought that for 'Dance/Eletronic' and Hip Hop songs, that tempo would become very important in terms of predicting danceability (rather than for a genre like rock or jazz). Therefore, we included two interations terms to reflect this hypothesis. The second model focuses more on the speechiness and explicit language of a song. We hypothesized that very wordy songs would be less danceable (kind of like the opposite of instrumentalness). Furthermore, we hypothesized that explicit language would decrease the effects of valance the perceived danceability, as much of the perceived valance of song comes from the language itself; we included to interaction terms to reflect this hypothesis. Also, we should note that, given we included speechiness (and explicit language) in the second model, we removed the categorical variable 'hip hop' to avoid multicollinearity, but added 'rock'. Furthermore, we thought that if a Dance/Electronic song is also considered to be in the genre of pop, these types of songs

may have unique danceability features; we added an interaction term to reflect this hypothesis.

```
model_dance1 <- lm(danceability_square ~ valence + energy_square + log_liveness + sqrt_instrumentalness

model_dance2 <- lm(danceability_square ~ valence + energy_square + log_liveness + log_speechiness +  `Da
```

We wanted to check for multicollinearity and any leverage or influence points using these 2 models as an overall insight for any future models as well.

```
model_dance1 %>% car::vif()
```

```
## there are higher-order terms (interactions) in this model
## consider setting type = 'predictor'; see ?vif

##                  valence            energy_square             log_liveness
##                 1.147600                 1.252271                 1.034539
##    sqrt_instrumentalness       `Dance/Electronic`                    tempo
##                 1.086224                35.817232                 2.026211
##                `hip hop` `Dance/Electronic`:tempo          tempo:`hip hop`
##                20.492221                36.282038                20.875150
```

Multicollinearity varies across variables in the model after calculating the Variance Inflation Factor (VIF). While Valence, Energy Square, Log Liveness, and Sqrt Instrumentalness show low to very low levels, Dance/Electronic and Hip Hop, along with their interaction terms with Tempo, exhibit potential high multicollinearity. This suggests redundancy or strong correlation, potentially affecting model stability and interpretation.

```
model_dance2 %>% car::vif()
```

```
## there are higher-order terms (interactions) in this model
## consider setting type = 'predictor'; see ?vif

##                        valence                  energy_square
##                       1.567049                       1.617424
##                    log_liveness                log_speechiness
##                       1.028419                       1.263461
##             `Dance/Electronic`               explicit_numeric
##                       1.066121                      11.032822
## energy_square:explicit_numeric       valence:explicit_numeric
##                       8.068258                       8.370899
```

In this case, Valence, Energy Square, Log Liveness, Dance/Electronic and Log Speechiness show low to very low levels, Energy Square and Valence along with their interaction terms with Explicit Numeric show medium levels, and with Explicit_Numeric to cause a little concern for multicollinearity.

Afterward, we plotted an influence plot for each model to pinpoint Cook's distance and leverage points to uncover leverage or influential points that could influence the regression model's coefficients. This method helps us understand the model's reliability by highlighting potential outliers and assessing their impact.

```
threshold <- 2 * (ncol(song_train) + 1) / nrow(song_train)
threshold
```

```
## [1] 0.04979734
```

```r
# For model_dance1
car::influencePlot(model_dance1, main = "Influence Plot for Model Dance 1")
```

```
##         StudRes         Hat       CookD
## 112  -0.9112987 0.054026139 0.004743393
## 232  -2.9471239 0.021016891 0.018563097
## 543   3.1923929 0.003420817 0.003479622
## 1082 -1.9684937 0.048508572 0.019722187
## 1469 -3.0831316 0.008674280 0.008276659
```

```r
plot(model_dance1, which = 4)  # Cook's distance plot
abline(h = 4 / nrow(song_train), lty = 2)
```

The influence plot revealed a small handful of observations with leverage values greater than twice the average leverage $(2 \times (k+1)/n) = 0.047$ (chirp chirp), where k is the number of predictors and n is the number of observations. These observations can be considered high leverage points. There are unfortunate a small number that passed this threshold within our first model, indicating leverage points. However, when examining the Cook's distance plot, none of the observations had a Cook's distance value greater than 1, which is a common threshold for identifying influential observations.

```
# For model_dance2
car::influencePlot(model_dance2, main = "Influence Plot for Model Dance 2")
```

```
##         StudRes         Hat       CookD
## 242   2.994400 0.001318043 0.001308792
## 1115 -1.137836 0.020575278 0.003021459
## 1277  3.211967 0.002499371 0.002856725
## 1497  2.592756 0.012730182 0.009599207
## 1549  2.425138 0.028394946 0.019043621
```

```r
plot(model_dance2, which = 4)
abline(h = 4 / nrow(song_train), lty = 2)
```

The influence plot did not reveal any observations with high leverage values exceeding twice the average leverage. Similarly, the Cook's distance plot did not show any observations with a Cook's distance value greater than 1.

In summary, while model_dance1 had a few high leverage points identified through the influence plot, neither model exhibited observations with substantial influence based on the Cook's distance criterion. This suggests that, although there are some high leverage points in model_dance1, they may not be significantly influencing the regression coefficients or model fit. However, it's still recommended to investigate these high leverage points further and consider their potential impact on the model. These are most likely not the exact models we will be using, so it is a bright sign for building a more concise model later.

Then, we performed a 10-fold cross validation of our data to evaluate how well each model performs on unseen data.

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
## Linear Regression
##
## 1727 samples
```

```
##    7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1555, 1555, 1554, 1554, 1554, 1554, ...
## Resampling results:
##
##   RMSE     Rsquared   MAE
##   0.14822  0.3349019  0.1175102
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
## Linear Regression
##
## 1727 samples
##    8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1554, 1554, 1554, 1556, 1555, 1554, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.1481067  0.3375759  0.1176105
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

As seen above, the models performed comparably in terms of RMSE (Root Mean Squared Error), R-Squared, and MAE. The second model appears to perform marginally better in each of the outputted metrics, thus indicating that the second model performs slighly better for predicting unseen data. This may indicate that language and genre feature are marginally more important for modeling danceability than instrumentation and tempo features.

Model Building / Stepwise BIC Selection

We then considered a two different models built with Backwards Stepwise BIC Selection and Forwards Stepwise BIC Selection. Since we are considering so many different variables and interaction terms in our model, we determined BIC was appropriate criterion for the purpose of penalizing overly complex models. Furthermore, many the predictors in our data set are correlated, therefore less complex models also helps us avoid issues with multicollinearity.
Consider the output models from each of these selection processes.

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'

## The following object is masked from 'package:purrr':
##
##     some

## The following object is masked from 'package:dplyr':
##
##     recode

##
## Call:
## lm(formula = danceability_square ~ valence + energy_square +
##     log_liveness + sqrt_instrumentalness + `Dance/Electronic` +
##     `hip hop` + rock + tempo + explicit_numeric + `Dance/Electronic`:tempo +
##     energy_square:explicit_numeric, data = song_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.47300 -0.08940 -0.00490  0.09309  0.44286
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     0.3000172  0.0231291  12.971  < 2e-16 ***
## valence                         0.3746341  0.0168068  22.291  < 2e-16 ***
## energy_square                  -0.1322006  0.0213232  -6.200 7.06e-10 ***
## log_liveness                   -0.0273230  0.0051326  -5.323 1.15e-07 ***
## sqrt_instrumentalness           0.1340318  0.0222922   6.012 2.23e-09 ***
## `Dance/Electronic`              0.1840585  0.0517211   3.559 0.000383 ***
## `hip hop`                       0.0532483  0.0083806   6.354 2.69e-10 ***
## rock                           -0.0835576  0.0114985  -7.267 5.56e-13 ***
## tempo                          -0.0005638  0.0001385  -4.070 4.91e-05 ***
## explicit_numeric                0.1737929  0.0215990   8.046 1.58e-15 ***
## `Dance/Electronic`:tempo       -0.0013344  0.0004121  -3.238 0.001227 **
## energy_square:explicit_numeric -0.2153386  0.0388168  -5.548 3.35e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1427 on 1715 degrees of freedom
## Multiple R-squared:  0.387,  Adjusted R-squared:  0.383
## F-statistic: 98.42 on 11 and 1715 DF,  p-value: < 2.2e-16

##
## Call:
## lm(formula = danceability_square ~ valence + energy_square +
##     `hip hop` + rock + sqrt_instrumentalness + explicit_numeric +
##     log_liveness + tempo + duration_scale + energy_square:explicit_numeric,
##     data = song_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.46493 -0.09051 -0.00420  0.09505  0.44043
```

```
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     0.3817213  0.0318070  12.001  < 2e-16 ***
## valence                         0.3674278  0.0168624  21.790  < 2e-16 ***
## energy_square                  -0.1300924  0.0211741  -6.144 9.98e-10 ***
## `hip hop`                       0.0533085  0.0083954   6.350 2.75e-10 ***
## rock                           -0.0869803  0.0113109  -7.690 2.46e-14 ***
## sqrt_instrumentalness           0.1451345  0.0218375   6.646 4.03e-11 ***
## explicit_numeric                0.1707593  0.0216467   7.888 5.40e-15 ***
## log_liveness                   -0.0275449  0.0051382  -5.361 9.41e-08 ***
## tempo                          -0.0007098  0.0001311  -5.412 7.11e-08 ***
## duration_scale                 -0.0257905  0.0090763  -2.842  0.00454 **
## energy_square:explicit_numeric -0.2083912  0.0389625  -5.349 1.01e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.143 on 1716 degrees of freedom
## Multiple R-squared:  0.3846, Adjusted R-squared:  0.381
## F-statistic: 107.2 on 10 and 1716 DF,  p-value: < 2.2e-16
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic   p.value    df logLik    AIC    BIC
##       <dbl>         <dbl> <dbl>     <dbl>     <dbl> <dbl>  <dbl>  <dbl>  <dbl>
## 1     0.391         0.386 0.142      78.4 1.63e-172    14   923. -1813. -1726.
## # i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic   p.value    df logLik    AIC    BIC
##       <dbl>         <dbl> <dbl>     <dbl>     <dbl> <dbl>  <dbl>  <dbl>  <dbl>
## 1     0.384         0.380 0.143      107. 1.36e-172    10   913. -1803. -1737.
## # i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
## Analysis of Variance Table
##
## Response: danceability_square
##                                 Df Sum Sq Mean Sq  F value    Pr(>F)
## valence                          1  8.902  8.9024 435.0908 < 2.2e-16 ***
## explicit_numeric                 1  4.051  4.0511 197.9903 < 2.2e-16 ***
## energy_square                    1  3.689  3.6893 180.3084 < 2.2e-16 ***
## rock                             1  1.657  1.6567  80.9662 < 2.2e-16 ***
## `hip hop`                        1  0.737  0.7372  36.0287 2.367e-09 ***
## sqrt_instrumentalness            1  1.006  1.0055  49.1424 3.409e-12 ***
## tempo                            1  0.533  0.5332  26.0616 3.675e-07 ***
## log_liveness                     1  0.550  0.5500  26.8806 2.420e-07 ***
## explicit_numeric:energy_square   1  0.632  0.6322  30.8972 3.149e-08 ***
## valence:explicit_numeric         1  0.134  0.1337   6.5364   0.01065 *
## Residuals                     1716 35.111  0.0205
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As seen above, the second model (found with Forward Stepwise BIC) has better criteria metric for adjusted *R^2*, AIC and BIC. Based on these metrics, we determined that the second model would likely better

predict danceability for unseen data and would have better explainability and interpretability. Later in this project, we will evaluate this model in more depth. First, let's consider how we can interpret the coefficient estimates and their signficance.
Each of the predictors' p-values are below our predetermined signficance level of $\alpha = 0.01$; therefore, given all other predictors and interaction terms are held constant, each predictor in our model significantly contributes to the predictive power of the model. Aka. changes in these predictor values results in signficant changes to danceability of a song. The ANOVA table shown above confirms the signficance of the individual predictors in our final modal.

Now, let's consider the intuition behind some of the values of our coefficients. Continuous variables such as 'valance', 'instrumentalness' (4th root), and indicator variables such as 'Dance/Electronic', 'hip hop' and explict langage improve the danceability of songs according to this model. We believe this result to be consistent with our domain knowledge; we initially expected that more positive sounding, instrumental songs would more danceable and we also expected that Dance/Electronic and hip hop music would also be perceived as more danceable. Furthermore, songs with explicit language are more common in social setting where people dance (such as bars and clubs), so more danceable music may be produced with explict language more often.
Increases in energy (squared), liveness (log), tempo and all of our interaction terms, on the other hand, decrease the expected danceability (squared) of a song. Although liveness decreasing danceability isn't particularly surprising (songs we dance to, especially in hip hop and eletronic music, usually aren't recorded live), energy decreasing danceability was a little suprising to us. We suspect that songs with too much energy are too intense to dance to properly. Our tempo coefficient, and interaction term with tempo, indicate that lower tempos are more conducive to dancing, and that, when a song is Dance/Electronic, higher tempos hurt the dancebility of the song more than for other genres. The two interaction terms involving *explicit_numeric* indicate that the positive effect of valence on danceability is diminished in the presence of explicit language. Additionally, explicit language amplifies the negative impact of high energy (squared) on danceability.

Before we evaluate this model on test data we set aside at the start, we will finally consider why certain variables were dropped out of the model during the BIC selection process. For example, if we look back to the correlation heatmap, we can see that 'hip hop' is very positively correlated with speechiness, and that energy is noteably correlated with both acousticness and loudness. You'll notice that none of 'speechiness', 'acousticness' or 'loudness' are included in the final model; these variables were likely dropped as they weren't signficant in our model (in terms of BIC) in the presense their highly correlated predictors 'hip hop' and 'energy'. One interesting observation is that 'energy' and 'valence' are positively correlated, however they are both included in the model. We suspect that both were signficant in the model because, although they are positively correlated, they are *oppositely* correlated to 'danceability.'

Testing the Final Model: *R^2*

Now that we've chosen a final model, we will evaluate how well it performs on the test data (which was set aside before the model building),

```
## Summary of Model Fit:

## Sum of Squared Errors (SSE): 3.7005

## Total Sum of Squares (SST): 6.1428

## Coefficient of Determination (R²): 0.3976
```

```
##
## The model explains 39.76% of the variability in the response variable, based on the test dataset.
```

While does model does not explain the majority of the variability in dancebility, we believe this model is still useful, especially considering all of the various musical, artistic, and cultural aspects contributing to people dancing to particular songs.
A moderate or high *R^2* value doesn't necessarily mean that a linear model is really explaining the variability of the response well. It could be inflated do to a variety of factors, such as multicollinearity, or overfitting. This why we took special consideration in choosing variables and procedures that reduce these effects.

CIs (Mean Predicted Value and Future Predicted Value)

In our study on predicting danceability in songs, we've developed a linear model and now introduce confidence intervals (CIs) to enhance the interpretation of our results. These intervals provide ranges of values within which we're confident the true mean danceability lies for both current data and future predictions. They offer insights into the precision and variability of our model's estimates, aiding stakeholders in understanding the reliability of our predictions.

```r
subset_song_test <- song_test %>%  slice(1:10)
mean_pred_int <- predict(finalmodel1, subset_song_test, interval="confidence", level=0.95)
future_pred_int <- predict(finalmodel1, subset_song_test, interval="prediction", level=0.95  )

mean_pred_int
```

```
##          fit       lwr       upr
## 1  0.2953303 0.2692174 0.3214433
## 2  0.2800702 0.2573105 0.3028300
## 3  0.6012345 0.5642857 0.6381833
## 4  0.5846867 0.5464008 0.6229727
## 5  0.4005126 0.3640895 0.4369357
## 6  0.6166170 0.5920365 0.6411974
## 7  0.6422624 0.6187855 0.6657392
## 8  0.5387052 0.5236869 0.5537236
## 9  0.5716453 0.5383240 0.6049667
## 10 0.4394495 0.4286712 0.4502278
```

```r
future_pred_int
```

```
##          fit         lwr       upr
## 1  0.2953303  0.01356218 0.5770985
## 2  0.2800702 -0.00140697 0.5615475
## 3  0.6012345  0.31825634 0.8842126
## 4  0.5846867  0.30153088 0.8678426
## 5  0.4005126  0.11760264 0.6834226
## 6  0.6166170  0.33498668 0.8982472
## 7  0.6422624  0.36072626 0.9237985
## 8  0.5387052  0.25774799 0.8196625
## 9  0.5716453  0.28911793 0.8541727
## 10 0.4394495  0.15868697 0.7202120
```

Summary of Findings

Within this project, our study aimed to find the the factors that influence the "danceability" of a song from a combination of audio features within the song, its popularity, genre, and other characteristics. We hope our findings will help musical artists and composers have a better understanding on "danceability" and its connection to the music world.

We aimed to create a valuable statistical model danceability as our main variable. Our first step in the process was preprocessing, which included feature engineering, transforming certain variables to fit the assumptions of the normal errors models, and checking for multicollinearity and high influence/leverage point. When it came to building our model, we employed stepwise selection models with f-tests to find the optimal variables to use for our model.

We found that the Forward Selected Stepwise BIC exhibited a better adjusted R^2, AIC, and BIC metrics, as well significant p-values to variable coefficients, indicating better suitability to predict/interpret danceability. Our final model indicates variables like valence, instrumentalness, and genre selection proved valuable positive effects on danceability while energy and tempo showed negative effects.

Further exploration could include the consideration of other model selection techniques such as Lasso or Ridge regression in order to select important predictors for our model better. We could also include a more deeper analysis on the effect of multicollinearity and potential outliers. We may also explore different shrinkage and smoothing methods. Overall, our current model can be seem as valuable for now based on the accuracy of our model and the precaution taken to prevent overfitting and overcomplexity.