Final Project: Top Spotify Tracks of 2018 Analysis

Jacqueline Deprey and Julie Stone

March 10, 2019

```
knitr::opts_chunk$set(echo = TRUE)
library(rvest)
library(tidyr)
library(dplyr)
library(stringr)
library(readr)
library(magrittr)
library(tidyverse)
library(purrr)
```

Introduction and Motivation

Data science covers the multifaceted activities required to create data-centric applications that answer specific questions. This is done through the data science pipeline - the multiple steps needed to answer a question about our data: obtaining, tidying, exploring, modeling, and interpreting our data. By following this pipeline we can hopefully find new patterns and draw meaningful conclusions about our data.

According to Forbes, the global recorded music industry is now worth over 17.2 billion dollars and only continues to grow. In 2018, Drake released his single "God's Plan" which was played over 1.28 billion times since its release and made Drake over $300,000. But what makes a song a hit? With the increasing prevalence of data that is being generated about consumer preferences, one might expect that a formula could be created to come up with the "perfect" song destined to hit the top charts. Because of the profit in formulas such as these, many producers spend lots of time and energy looking into how to do just that.

For this project, we have decided to examine Spotify's data on the Top 100 Hits from 2018. Because more and more people are getting their music content from audio streaming services, we believe looking at Spotify's data would be a good way to retroactively examine this problem. Spotify, a streaming service based on the freemium model, has over 83 million subscribers and has captured roughly 36% of the market share. Since most major artist now have their music on its platform and the top charts on Spotify almost always match those of industry experts looking at all platforms, we believe analyzing Spotify's data would be representative of the industry trends as a whole.

Description of dataset

"Top Spotify Tracks of 2018" is a dataset depicting the audio features of the top songs of the streaming platform. There are 99 entities, or objects that the dataset refers to. Each entity represents a song on the chart. There are 16 different attributes, including:

Attribute	Description
ID	The primary key of the dataset - the Spotify URL of the song
Name	Name of the song
Artist(s)	Artist of the song
Danceability	Danceability describes how suitable a track is for dancing. This is based on a combination of different musical elements such as tempo, rhythm, stability, beat strength, and overall regularity. A value of 0.0 is least danceable and a value of 1.0 is most danceable

Attribute	Description
Energy	A measure from 0.0 to 1.0 representing a perceptual measure of intensity and activity. Energetic tracks feel fast, loud, and noisy. Perceptual features that contribute to this measure are dynamic range, percieved loudness, timbre, onset rate, and general entropy
Key	The key the track is in. Integers map to pitches using the standard pitch class notation. For example, C=0, C#=1, D=2, and so on
Loudness	The overall loudness of a track in decibels. These values are averaged over the course of the song.
Mode	Modality of a song (major vs. minor). Major is represented by 1 and minor is represented by 0.
Speechiness	Detects the presence of spoken words in the track. The more exclusively speech-like the song is, the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks
Acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence.
Instrumentalness	Predicts whether a track contains no vocals. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
Liveness	Detects the presence of an audience in the recording. Higher liveness values represent a greater chance that the track was performed live.
Valence	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive, while tracks with low valence sound more negative.
Tempo	The overall estimated tempo of a track in beats per minute (BPM).
Duration	The duration of the track in milliseconds.
Time Signature	An estimated overall time signature of a track. The time signature is a notational convention to specify how many beats are in each measure.

The dataset was obtained as a CSV file from Kaggle.com. Kaggle is an online community of data scientists, owned by Google LLC. Users find and publish data sets, explore and build models in a data-science environment, work with other data scientists, and enter competitions to solve data science challenges. This specific dataset was uploaded 3 months ago by Nadin Tamer. To load this CSV file into an R data frame, which we can then use for analysis, we must use read_csv method() to read in the data and use the as_data_frame() method to transform it into a data frame, as seen below.

```
csv_file <- "top2018.csv"

spotify_data <- read_csv(csv_file) %>%
  set_colnames(c("id", "name", "artists", "danceability", "energy", "key", "loudness", "mode", "speechi
  as_data_frame()
```

```
## Parsed with column specification:
## cols(
##   id = col_character(),
##   name = col_character(),
##   artists = col_character(),
##   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(),
##   duration_ms = col_double(),
##   time_signature = col_double()
## )
```

```
## Warning: `as_data_frame()` is deprecated, use `as_tibble()` (but mind the new semantics).
## This warning is displayed once per session.
```

```
head(spotify_data)
```

```
## # A tibble: 6 x 16
##   id    name  artists danceability energy   key loudness  mode speechiness
##   <chr> <chr> <chr>          <dbl>  <dbl> <dbl>    <dbl> <dbl>       <dbl>
## 1 6DCZ~ God'~ Drake          0.754  0.449     7    -9.21     1      0.109
## 2 3ee8~ SAD!  XXXTEN~        0.74   0.613     8    -4.88     1      0.145
## 3 0e7i~ rock~ Post M~        0.587  0.535     5    -6.09     0      0.0898
## 4 3swc~ Psyc~ Post M~        0.739  0.559     8    -8.01     1      0.117
## 5 2G7V~ In M~ Drake          0.835  0.626     1    -5.83     1      0.125
## 6 7dt6~ Bett~ Post M~        0.68   0.563    10    -5.84     1      0.0454
## # ... with 7 more variables: acousticness <dbl>, instrumentalness <dbl>,
## #   liveness <dbl>, valence <dbl>, tempo <dbl>, duration_ms <dbl>,
## #   time_signature <dbl>
```

Tidying the Dataset

It is important that before we analyze our data, it is presented in such a manner that it is responsive to analysis, in both modeling and visualization. To do this, we have structured our data according to the Entity-Relationship model. This is a data structure where each attribute forms a column, each entity forms a row, and each type of entity forms a table. Once our data is presented as such, we modified some of the entities to make it easier to analyze. First of all, we added a rank attribute, which numerically shows the song's rank on the chart. Then, we converted the duration of the song from milliseconds to a minutes and seconds format. Lastly, we got dropped all of the "NAs" in the data frame, as they wouldn't be useful to our analysis.

```
spotify_data <- spotify_data %>%
  mutate(rank = seq(1, 100)) %>%
  mutate(duration_min = as.integer(duration_ms / 60000)) %>%
  mutate(duration_sec = as.integer((duration_ms - (duration_ms / 60000)) / 1000)) %>%
  unite("duration_time", duration_min, duration_sec, sep = ":") %>%
  type_convert(col_types = cols(end_datetime = col_datetime(format = "%M:%S"))) %>%
  drop_na()
```

```
## Warning in type_convert_col(char_cols[[i]], specs$cols[[i]],
## which(is_character)[i], : [94, 18]: expected valid date, but got '1:95'
```

```
spotify_data %>% select(rank, everything())
```

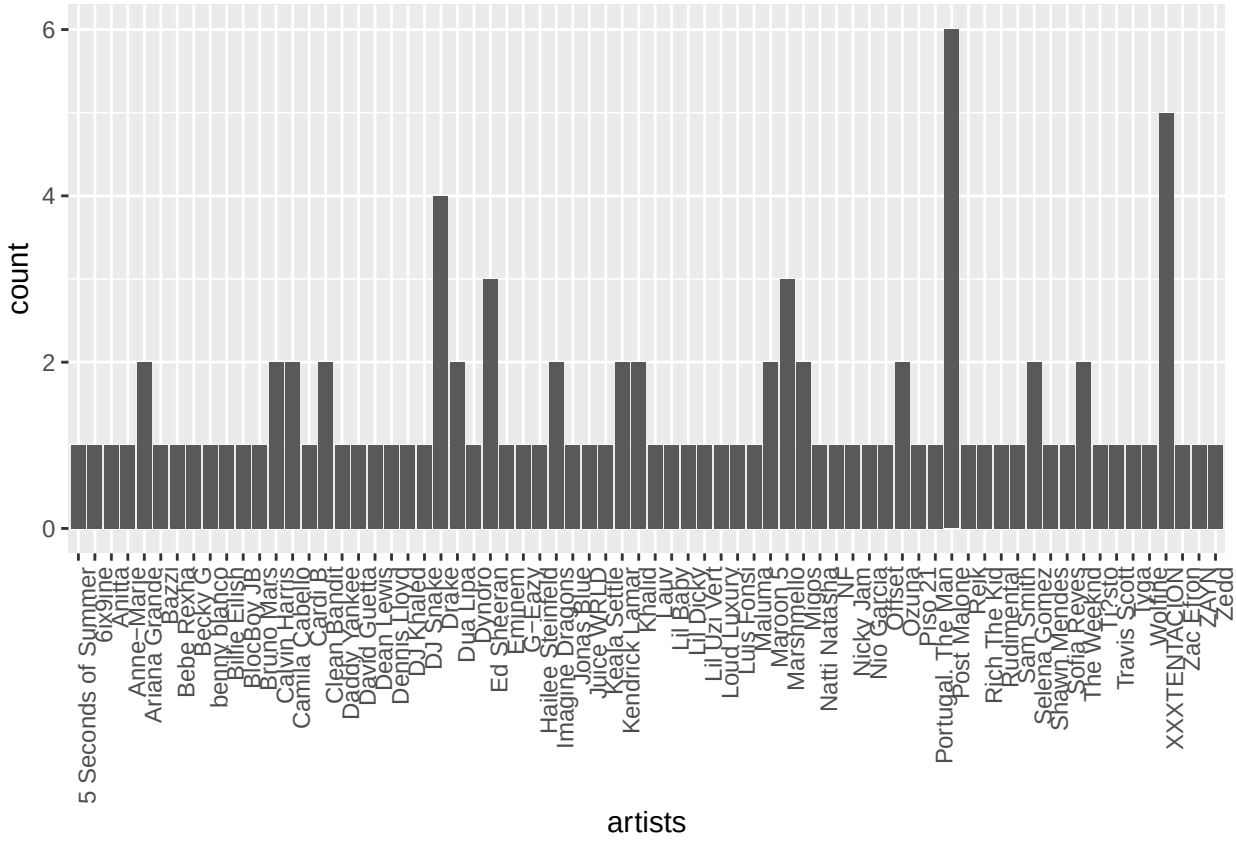
```
## # A tibble: 99 x 18
```

```
##     rank id    name  artists danceability energy   key loudness  mode
##    <int> <chr> <chr> <chr>          <dbl>  <dbl> <dbl>    <dbl> <dbl>
##  1     1 6DCZ~ God'~ Drake          0.754  0.449     7    -9.21     1
##  2     2 3ee8~ SAD!  XXXTEN~        0.74   0.613     8    -4.88     1
##  3     3 0e7i~ rock~ Post M~        0.587  0.535     5    -6.09     0
##  4     4 3swc~ Psyc~ Post M~        0.739  0.559     8    -8.01     1
##  5     5 2G7V~ In M~ Drake          0.835  0.626     1    -5.83     1
##  6     6 7dt6~ Bett~ Post M~        0.68   0.563    10    -5.84     1
##  7     7 58q2~ I Li~ Cardi B        0.816  0.726     5    -4.00     0
##  8     8 7ef4~ One ~ Calvin~        0.791  0.862     9    -3.24     0
##  9     9 76cy~ IDGAF Dua Li~        0.836  0.544     7    -5.98     1
## 10    10 08bN~ FRIE~ Marshm~        0.626  0.88      9    -2.38     0
## # ... with 89 more rows, and 9 more variables: speechiness <dbl>,
## #   acousticness <dbl>, instrumentalness <dbl>, liveness <dbl>,
## #   valence <dbl>, tempo <dbl>, duration_ms <dbl>, time_signature <dbl>,
## #   duration_time <time>
```

General Data Visualization

Data Visualization is a very important aspect of data science. It allows us to gain understanding of dataset characteristics throughout our analysis, and helps us easily communicate discovered insights derived from said analysis. To start off our analysis, we decided to create a simple bar graph. Do certain artists hold a monopoly over the charts?

```
spotify_data %>%
  ggplot(mapping=aes(x=artists)) +
    geom_bar() + theme(axis.text.x=element_text(angle=90, hjust=1))
```



As you can deduce from this graph, most artists only have one or two songs on the Top 100 chart. However, we do seem to have a few outliers. Post Malone seems to be the most popular artist on Spotify in 2018 with a grand total of six songs on the chart, with XXXTENTACION following close behind at five songs, and Drake at 4 songs.

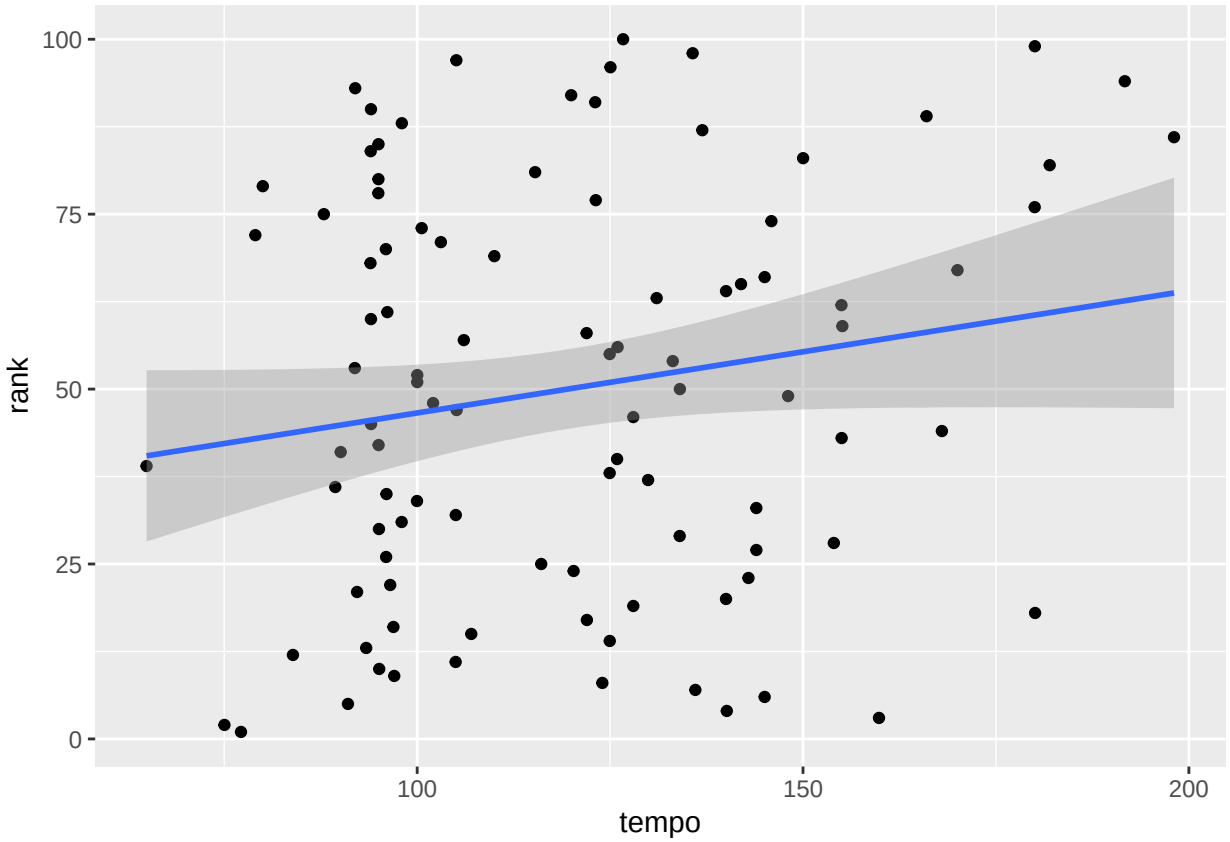
Want to learn how to plot different types of graphs? Check out these notes from Professor Héctor Corrada Bravo from the University of Maryland.

Statistical analysis and methods

Now, we want to see what aspects of a song make it popular. Why are the top songs ranked so high? What about them makes them special? To do this, we can perform a process known as linear regression. It is extensively used in exploratory data analysis and statistic analysis. We are able to analyze the relationship between two variables, and measure if that relationship is significant. The linear regression generates a value called a p-value. If this value is less than .05, then we are able to conclude that the relationship is significant. Here are a few examples using our dataset.

Is there a relationship between a song's tempo and its chart rank?

```
spotify_data %>% ggplot(aes(x = tempo, y = rank)) + geom_point() + geom_smooth(method=lm)
```




```
linear_regression <- lm(rank ~ tempo, data = spotify_data)

linear_regression %>%
  broom::tidy()
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   29.1     12.3         2.37  0.0196
## 2 tempo          0.175    0.0996      1.76  0.0822
```

```
log_regression_tempo <- lm(rank ~ log(tempo), data = spotify_data)

log_regression_tempo %>%
  broom::tidy()
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)    -46.1      58.2    -0.792   0.430
## 2 log(tempo)      20.2      12.2     1.65    0.101
```

```
log_regression_both <- lm(log(rank) ~ log(tempo), data = spotify_data)

log_regression_both %>%
  broom::tidy()
```

```
## # A tibble: 2 x 5
```

```
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)    0.431     1.87      0.230  0.819
## 2 log(tempo)     0.672     0.393     1.71   0.0906
```

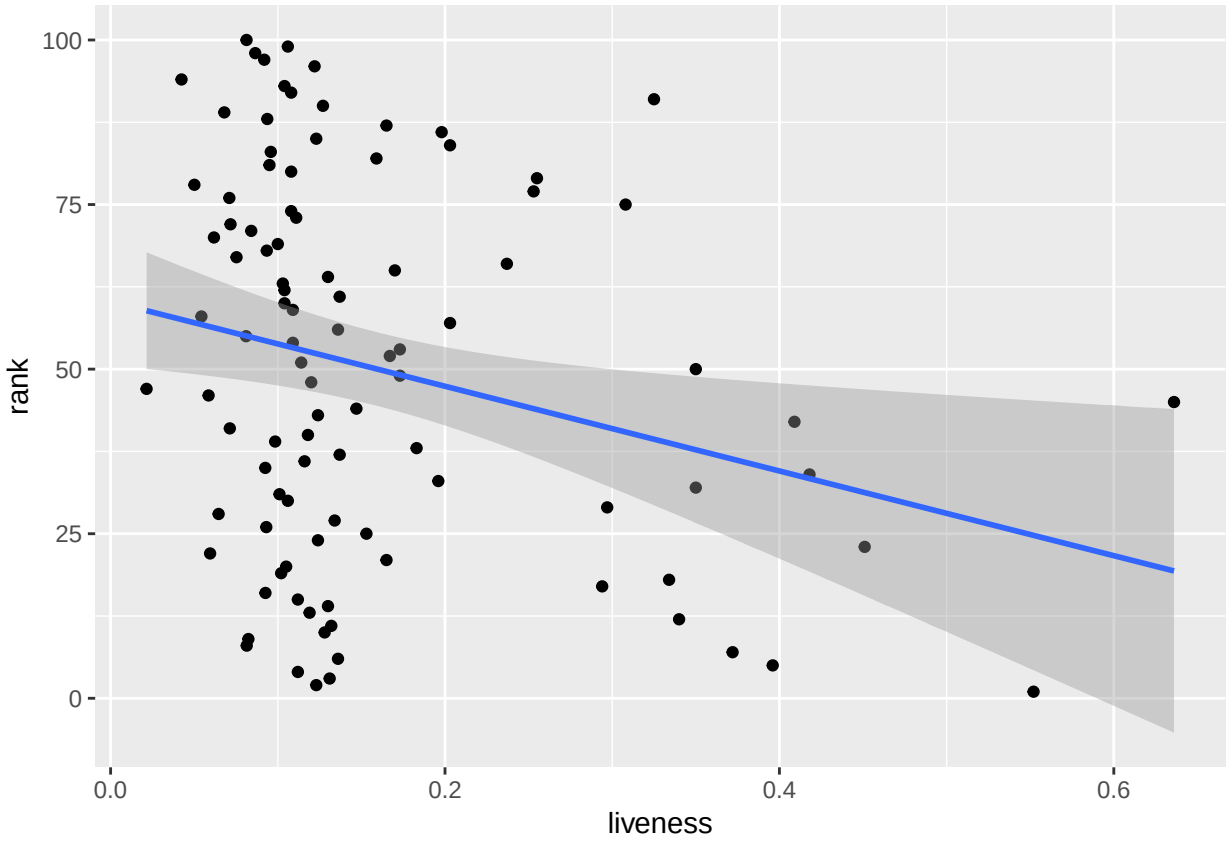
```
log_regression_rank <- lm(log(rank) ~ tempo, data = spotify_data)

log_regression_rank %>%
  broom::tidy()
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  3.00      0.396        7.57 2.14e-11
## 2 tempo        0.00525   0.00321      1.63 1.06e- 1
```

After noticing a slight increase in the trendline, we decided to perform a linear regression to determine if there was a relationship between the tempo of the song and the rank of the song. The linear regression analysis indicated that while the p value for the influence of tempo on rank was small at 0.0822, it was not smaller than our alpha value of 0.05 so we can not conclude that a statistically significant relationship exists between the two variables. We then tried to do a logarithmic regression in case a non-linear relationship existed. Although we tried taking the log of each variable separately and both of them together, again our p-values did not show a statistically significant relationship among these variables.

Is there a relationship between a song's liveness and its chart rank?

```
spotify_data %>% ggplot(aes(x = liveness, y = rank)) + geom_point() + geom_smooth(method=lm)
```



```
linear_regression <- lm(rank ~ liveness, data = spotify_data)

linear_regression %>%
  broom::tidy()
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)     60.3      4.90     12.3  1.65e-21
## 2 liveness       -64.3     25.3      -2.55 1.24e- 2
```

```
log_regression_liveness <- lm(rank ~ log(liveness), data = spotify_data)

log_regression_liveness %>%
  broom::tidy()
```

```
## # A tibble: 2 x 5
##   term          estimate std.error statistic p.value
##   <chr>            <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)       26.0     10.2       2.55  0.0122
## 2 log(liveness)    -11.9      4.84     -2.45  0.0160
```

```
log_regression_both <- lm(log(rank) ~ log(liveness), data = spotify_data)

log_regression_both %>%
  broom::tidy()
```

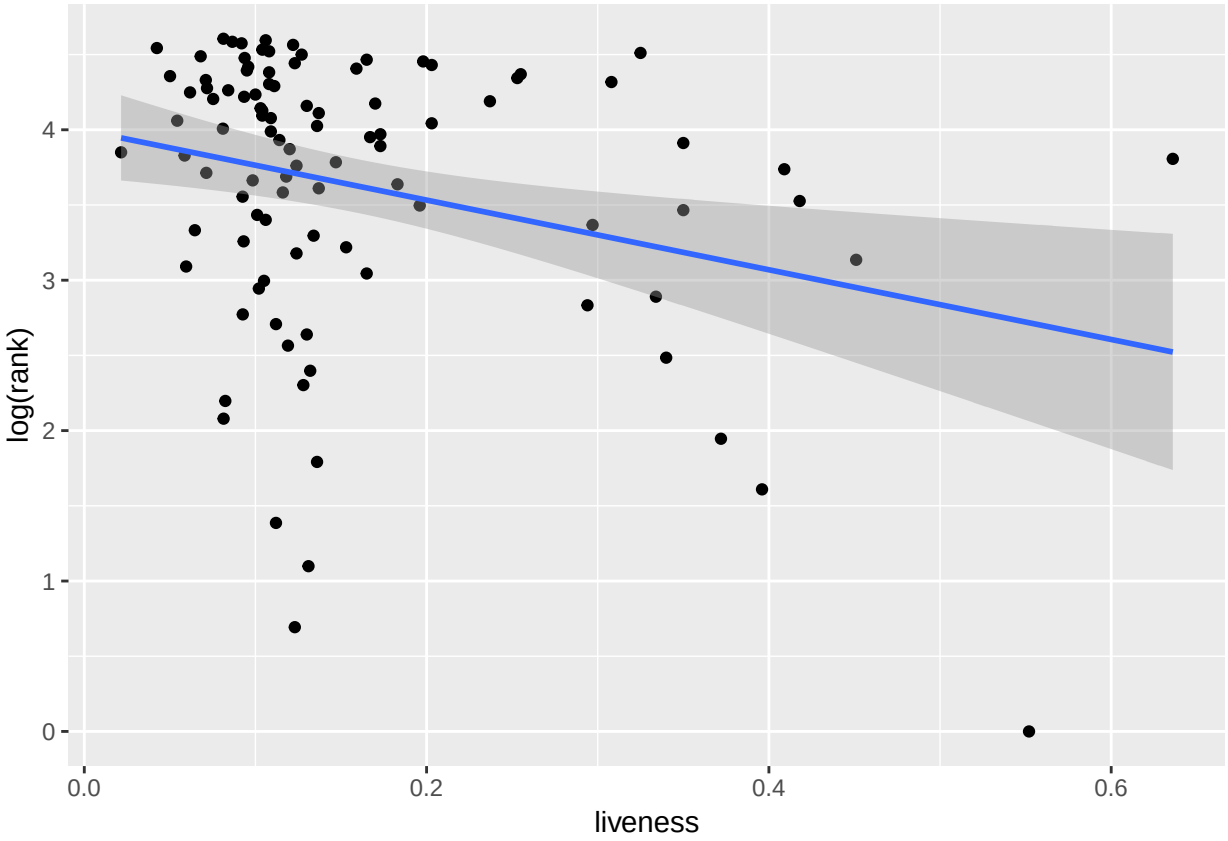
```
## # A tibble: 2 x 5
##   term          estimate std.error statistic  p.value
##   <chr>            <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)      2.81      0.327      8.58 1.55e-13
## 2 log(liveness)   -0.406     0.155     -2.61 1.04e- 2
```

```
log_regression_rank <- lm(log(rank) ~ liveness, data = spotify_data)

log_regression_rank %>%
  broom::tidy()
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)     4.00      0.157     25.5  7.92e-45
## 2 liveness       -2.32      0.807     -2.87 5.02e- 3
```

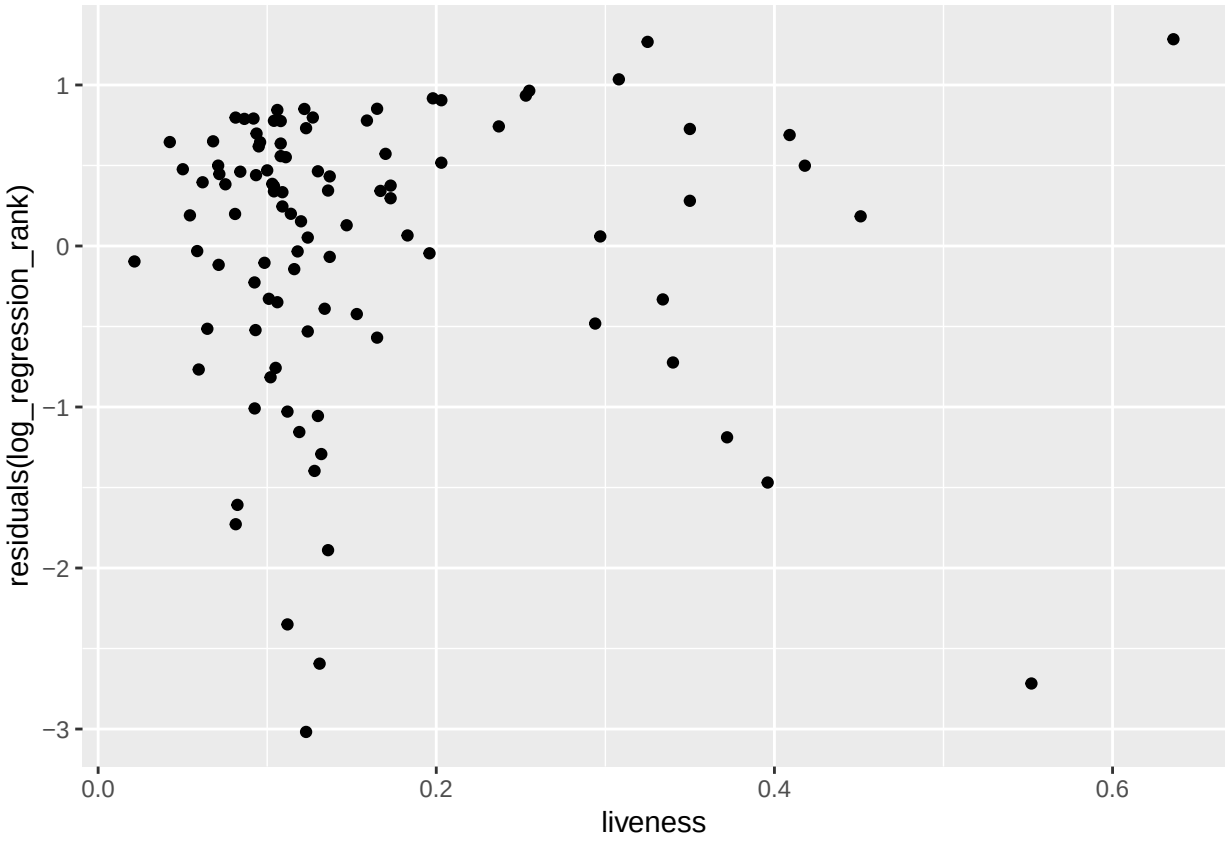
After getting an inconclusive p-value for the relationship between a song's tempo and its chart rank, we began to run regressions on every value in the table to determine what factors did influence a song's rank. While most of the factors did not have an affect, we did find that there was a negative relationship between how "live" a song was and its rank. This can be seen by not only the small p-value of 0.0124 which is smaller than our alpha value of 0.05 proving that this finding is statistically significant, but also by the negative coefficient of the estimate which shows the negative relationship between these two variables. As a result of this we can conclude that songs that sound like they were performed in front of a live audience, such as in concert, do not as well as those produced in a studio. Although we proved a linear relationship existed, we wanted to again see if this was the best fit or if the relationship between liveness and rank could better be described by another function. After taking the logs of both liveness and rank on their own and together, we learned that liveness best describes the log of a song's rank as seen by the lowest p-value at 0.00502.

```
spotify_data %>% ggplot(aes(x = liveness, y = log(rank))) + geom_point() + geom_smooth(method=lm)
```



However, to make sure this is a good model, we also need to look at the residual plot to make sure that the residuals are evenly distributed around 0 and that there are no trends within them.

```
spotify_data %>%
  ggplot(aes(x=liveness, y=residuals(log_regression_rank))) + geom_point()
```



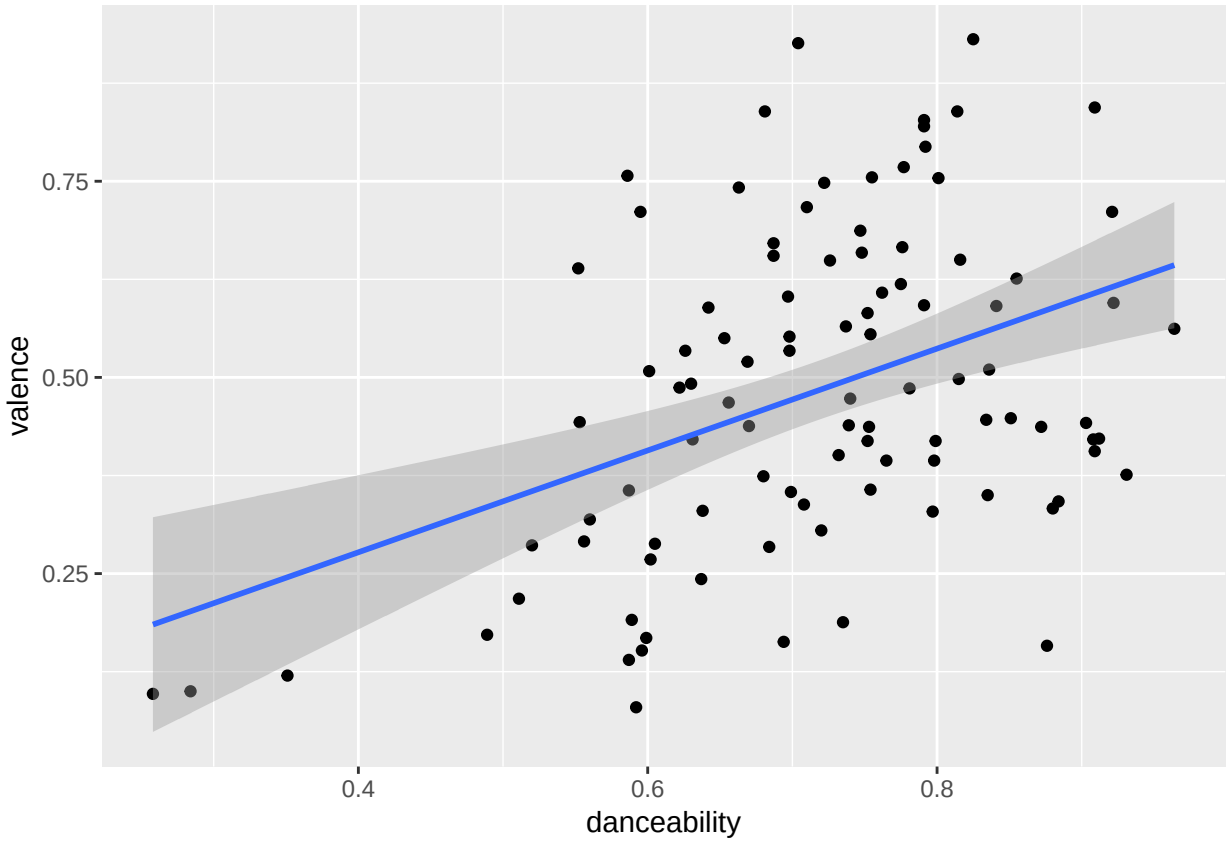

```
  labs(title="Residuals with respect to liveness", x = "liveness", y = "residual")
```

```
## $x
## [1] "liveness"
##
## $y
## [1] "residual"
##
## $title
## [1] "Residuals with respect to liveness"
##
## attr(,"class")
## [1] "labels"
```

Because there are an even number of points above and below 0 as well as no real patter between the residuals as liveness increases, this model is a good fit.

Is there a relationship between danceability and valence?

```
spotify_data %>% ggplot(aes(x = danceability, y = valence)) + geom_point() + geom_smooth(method=lm)
```




```
linear_regression <- lm(valence ~ danceability, data = spotify_data)

linear_regression %>%
  broom::tidy()
```

```
## # A tibble: 2 x 5
##   term         estimate std.error statistic   p.value
##   <chr>           <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)    0.0176     0.105     0.167 0.868
## 2 danceability   0.649      0.145     4.49  0.0000198
```

```
log_regression_danceability <- lm(valence ~ log(danceability), data = spotify_data)

log_regression_danceability %>%
  broom::tidy()
```

```
## # A tibble: 2 x 5
##   term              estimate std.error statistic  p.value
##   <chr>                <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)          0.631    0.0358     17.6  5.95e-32
## 2 log(danceability)    0.419    0.0864      4.85 4.66e- 6
```

```
log_regression_both <- lm(log(valence) ~ log(danceability), data = spotify_data)

log_regression_both %>%
  broom::tidy()
```

```
## # A tibble: 2 x 5
```

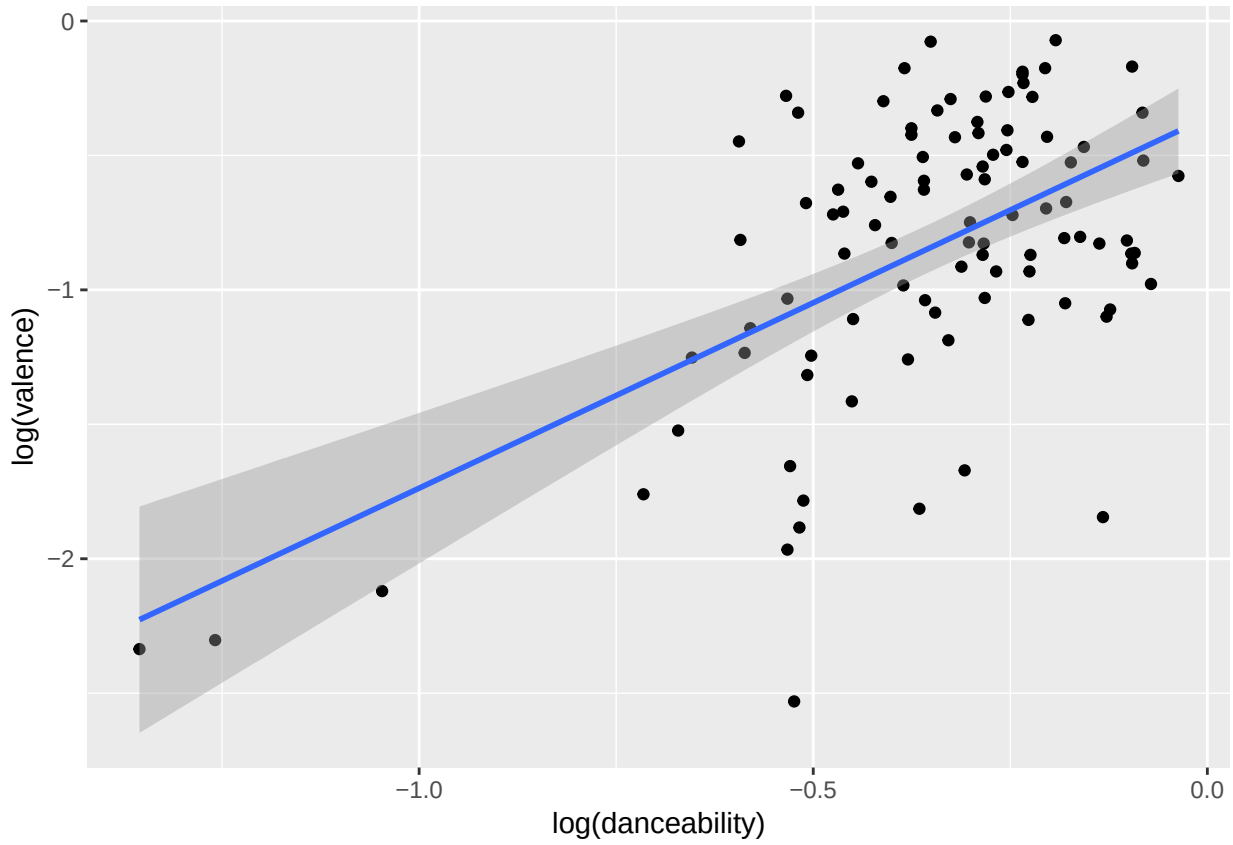
```
##   term              estimate std.error statistic       p.value
##   <chr>                <dbl>     <dbl>     <dbl>         <dbl>
## 1 (Intercept)         -0.358    0.0860     -4.17 0.0000663
## 2 log(danceability)    1.38     0.207       6.66 0.00000000166
```

```
log_regression_valence <- lm(log(valence) ~ danceability, data = spotify_data)

log_regression_valence %>%
  broom::tidy()
```

```
## # A tibble: 2 x 5
##   term         estimate std.error statistic  p.value
##   <chr>           <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)     -2.35     0.258     -9.10 1.16e-14
## 2 danceability     2.09     0.354      5.91 5.05e- 8
```

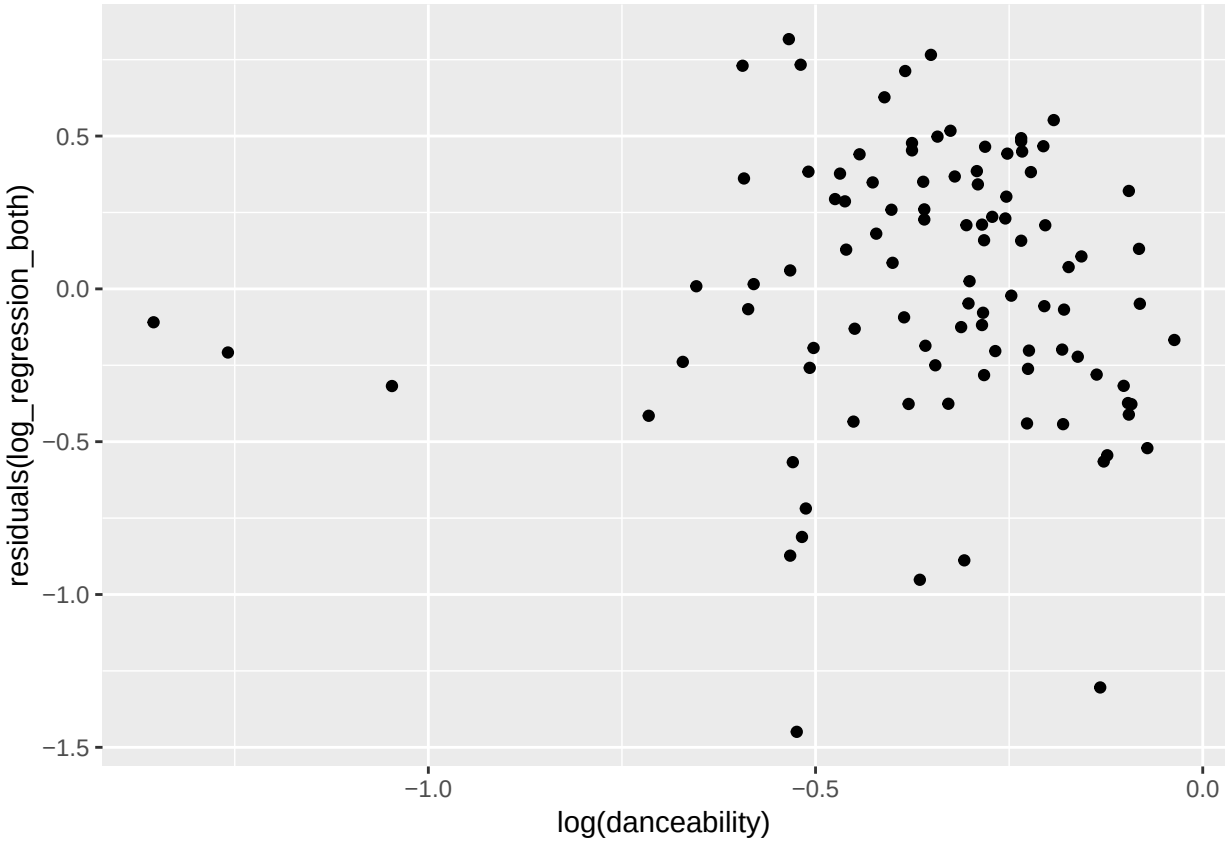
With this project, we wanted to not only look for relationships between different factors and a song's performance, but we were also interested in the relationship between different variables. One hypothesis we created based on personal experiences was that more danceable songs typically came off to us as happier, thereby suggesting they might have more valence. To test this hypothesis, we decided to perform a regression analysis between these two variables. The small p-value of 0.0198 indicated that our hypothesis was true and that there is a relationship between these variables since this p-value is smaller than our alpha value of 0.05. Again, we wanted to see if this was the best relationship between these two variables so we took the log of each individually and the log of them both together. From doing this, we found that the log of danceability and the log of valence are the most similarly related as seen by the p-value of 1.662 e -9.

```
spotify_data %>% ggplot(aes(x = log(danceability), y = log(valence))) + geom_point() + geom_smooth(metho
```



However, to make sure this is a good model, we also need to look at the residual plot to make sure that the residuals are evenly distributed around 0 and that there are no trends within them.

```
spotify_data %>%
  ggplot(aes(x=log(danceability), y=residuals(log_regression_both))) + geom_point()
```




```
  labs(title="Residuals with respect to log(danceability)", x = "log(danceability", y = "residual")
```

```
## $x
## [1] "log(danceability"
##
## $y
## [1] "residual"
##
## $title
## [1] "Residuals with respect to log(danceability)"
##
## attr(,"class")
## [1] "labels"
```

Because there is no pattern amongst the residuals and they are all centered around 0, this relationship is accurate.

For more information about linear regressions and residual plots, check out more information from the statistics department at Yale here.

While we did the regression and residuals plots with R in this tutorial, the same analysis can be done with other programming languages. To learn how to do this same analysis with Python, check out this Towards Data Science tutorial.

Conclusion

In conclusion, while there was not a statistically significant linear relationship between most of the variables and the rank of a song, it can be concluded that the liveness of a song negatively impacts its performance on Spotify. Although we were not able to detect many statistically significant linear relationships, because of how profitable the music industry is, we suggest that artists do more analysis to increase the number of listeners to their music. One way to improve upon our analysis would be to look into other types of relationships between variables and to use a larger dataset. Because our trendlines looked relatively linear to begin with, we to only go forward with linear and log based relationships. Other models though like Poisson's may find that a statistically significant relationship does exist between some of our inconclusive variables. However, if a larger dataset was used than just the top 100 songs, it might be easier to detect if other relationships exist. In addition, because of the larger sample size, it would be easier to prove that relationships found are statistically significant because a larger sample size would decrease the standard error.

For this project, Jacqueline worked on the introduction and industry analysis. Julie researched what each of the industry terms meant to understand what each of the attributes represented. Jacqueline then tidied the data which Julie then analyzed. Both Jacqueline and Julie analyzed the results of their findings and worked to make meaningful conclusions from the results. Jacqueline then uploaded the project to Github to be submitted for the team.