Final Project Draft

Introduction

ASK ABOUT PUTTING CODEBOOK IN THE README FILE AND NOT IN THE PROJECT ITSELF email

"The Billboard Hot 100 is the music industry standard record chart in the United States for songs, published weekly by Billboard magazine. Chart rankings are based on sales (physical and digital), radio play, and online streaming in the United States."

The data was found on TidyTuesday and is from Data.World with the original data points found on Billboard.com and Spotify. The cases are songs from the certain week(s) in which they appeared on the Billboard 100 chart. It includes every weekly Hot 100 singles chart from Billboard.com. Relevant variables from Billboard include song, performer, instance (ordinal for which appearance on the chart it is for the song), previous_week_position, and more. Relevant variables from Spotify include spotify_genre, spotify_track_duration, danceability (double 0-1; factoring in tempo, rhythm stability, beat strength), energy (double 0-1; perceptual measure of intensity and activity), key, acousticness (double 0-1), and valence (double 0-1; "musical positiveness"), and more. Both include a song name in the variable "song" that we'll use for joining.

From this dataset, we're interested in examining songs from 2000 to today and songs that made it to the #1 song position on the Billboard 100. Since there are so many song attributes in the dataset, we decided to focus on attributes that are easily identifiable through hearing by the general population: genre, danceability, energy, speechiness, valence, and tempo. Our main research question is: what factors play a role in a song becoming #1 on the Billboard 100 and the longevity of songs on the Billboard 100 after peaking at #1?

Variable description: Genre - is based on what genre Spotify puts the song into. Genre are categories that a song can put into based on musical techniques used, the cultural context behind the song and the spirit of the themes.

Danceability - describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.

Energy - a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.

Speechiness - detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.

Valence - A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

Tempo - The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.

```
library(tidyverse)
```

```
## Warning in system("timedatectl", intern = TRUE): running command 'timedatectl'
## had status 1

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidymodels)
```

```
## Registered S3 method overwritten by 'tune':
##   method                   from
##   required_pkgs.model_spec parsnip

## -- Attaching packages -------------------------------------- tidymodels 0.1.4 --

## v broom        0.7.9      v rsample      0.1.0
## v dials        0.0.10     v tune         0.1.6
## v infer        1.0.0      v workflows    0.2.4
## v modeldata    0.1.1      v workflowsets 0.1.0
## v parsnip      0.1.7      v yardstick    0.0.8
## v recipes      0.1.17

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Search for functions across packages at https://www.tidymodels.org/find/
```

```
library(stringr)
```

Setting Up Our Project

```
billboard <-
  readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2021/2021-
```

```
## Rows: 327895 Columns: 10

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): url, week_id, song, performer, song_id
## dbl (5): week_position, instance, previous_week_position, peak_position, wee...

##
## 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
audio_features <-
  readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2021/2021-0
```

```
## Rows: 29503 Columns: 22
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): song_id, performer, song, spotify_genre, spotify_track_id, spotify...
## dbl (14): spotify_track_duration_ms, danceability, energy, key, loudness, mo...
## lgl  (1): spotify_track_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.
```

To conduct our work, we first need to combine our datasets, which is slightly complicated. For the billboard dataset, each entry is not a song, but rather the position of a song for each week. For example, a song is on the Billboard Top 100 for multiple weeks, it would be entered multiple times, each time with it's week's ranking. In contrast, each of the audio_feature's rows is a song. Songs are not entered in multiple times. To be able to combine the datasets together, we have to condense the Billboard 100 to having each row represent a song. Since we only care about #1 songs and songs from the 2000s, we plan on filtering the billboard data set so that there is only one song that appears for each week, the #1 song. If there are songs that remain #1 for multiple weeks we will filter these out so they only appear once. After creating the new dataset, we will hopefulle be able to merge the 2 data sets creating a final large dataset with x number of observations and 31 variables. (we need to discuss with the TA and professor smith about how to do this)

```r
billboard2 <- billboard %>%
  mutate(date = as.Date(week_id, format = "%m/%d/%Y", origin = "1958-08-02")) %>%
  mutate(month = format(date, "%m")) %>%
  mutate(year= format(date, "%Y"))

all_combined <- audio_features %>%
  right_join(billboard2, by= c("song", "performer"))

combined <- all_combined %>%
  filter(year > 1999)
```

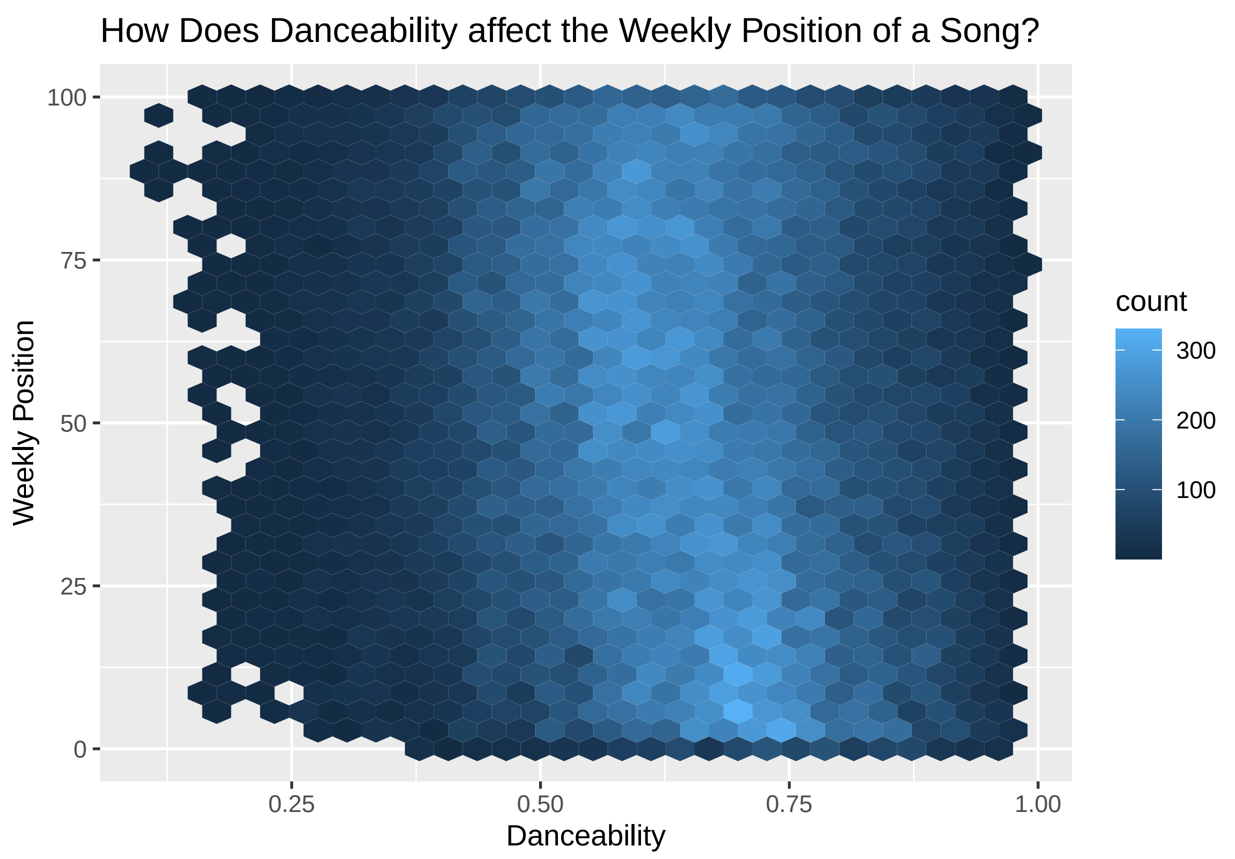
Attributes and Time

##By Rank

We looked at the differences between attributes of song ranks.

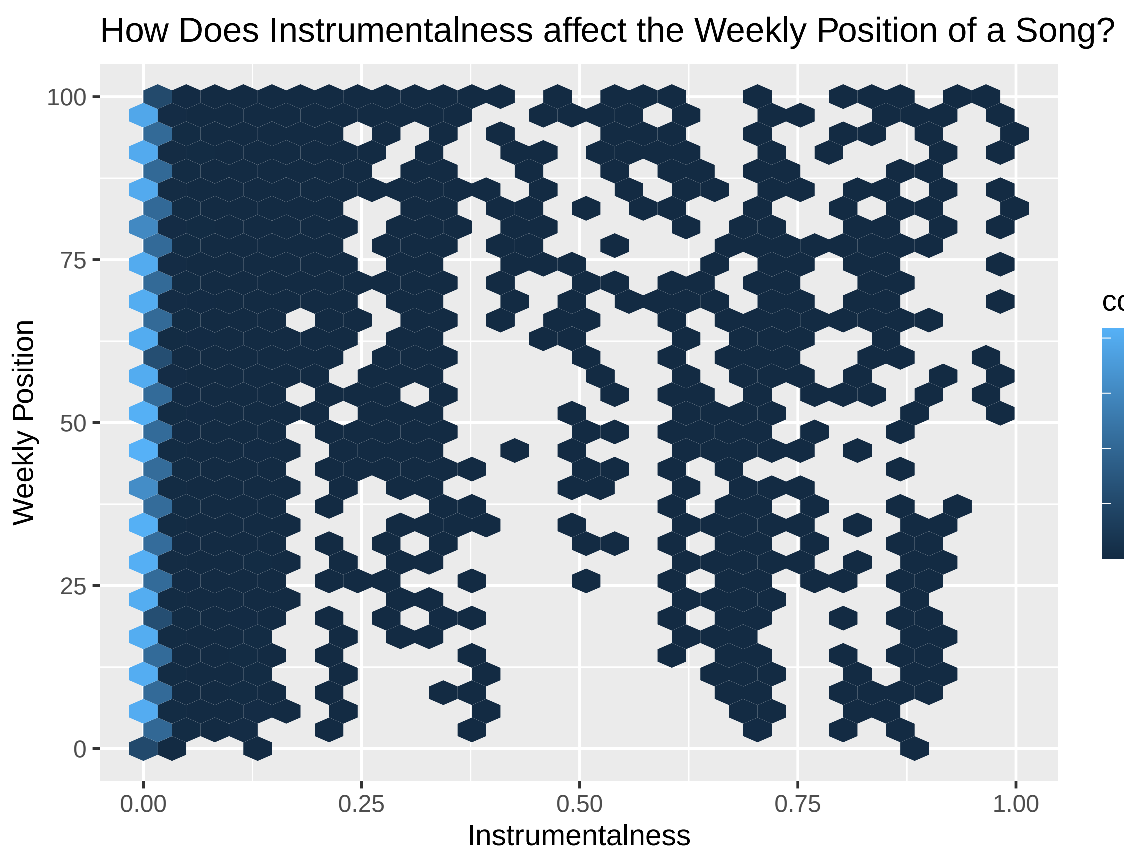
```r
combined %>%
  ggplot(mapping = aes(x = danceability, y = week_position)) +
  geom_hex() +
  labs(title = "How Does Danceability affect the Weekly Position of a Song?",
       x = "Danceability", y = "Weekly Position")
```

```
## Warning: Removed 8584 rows containing non-finite values (stat_binhex).
```



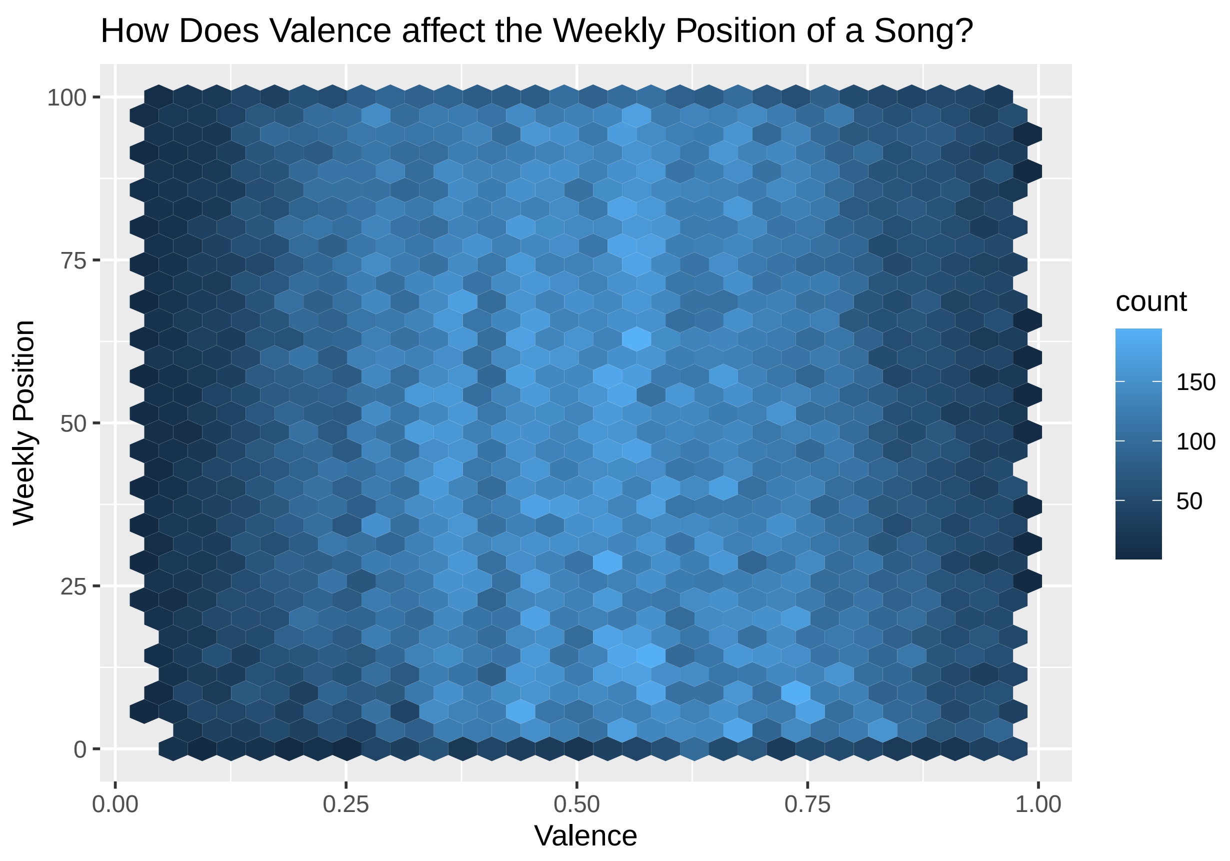
```
combined %>%
  ggplot(mapping = aes(x = instrumentalness, y = week_position)) +
  geom_hex() +
  labs(title = "How Does Instrumentalness affect the Weekly Position of a Song?",
       x = "Instrumentalness", y = "Weekly Position")
```

```
## Warning: Removed 8584 rows containing non-finite values (stat_binhex).
```



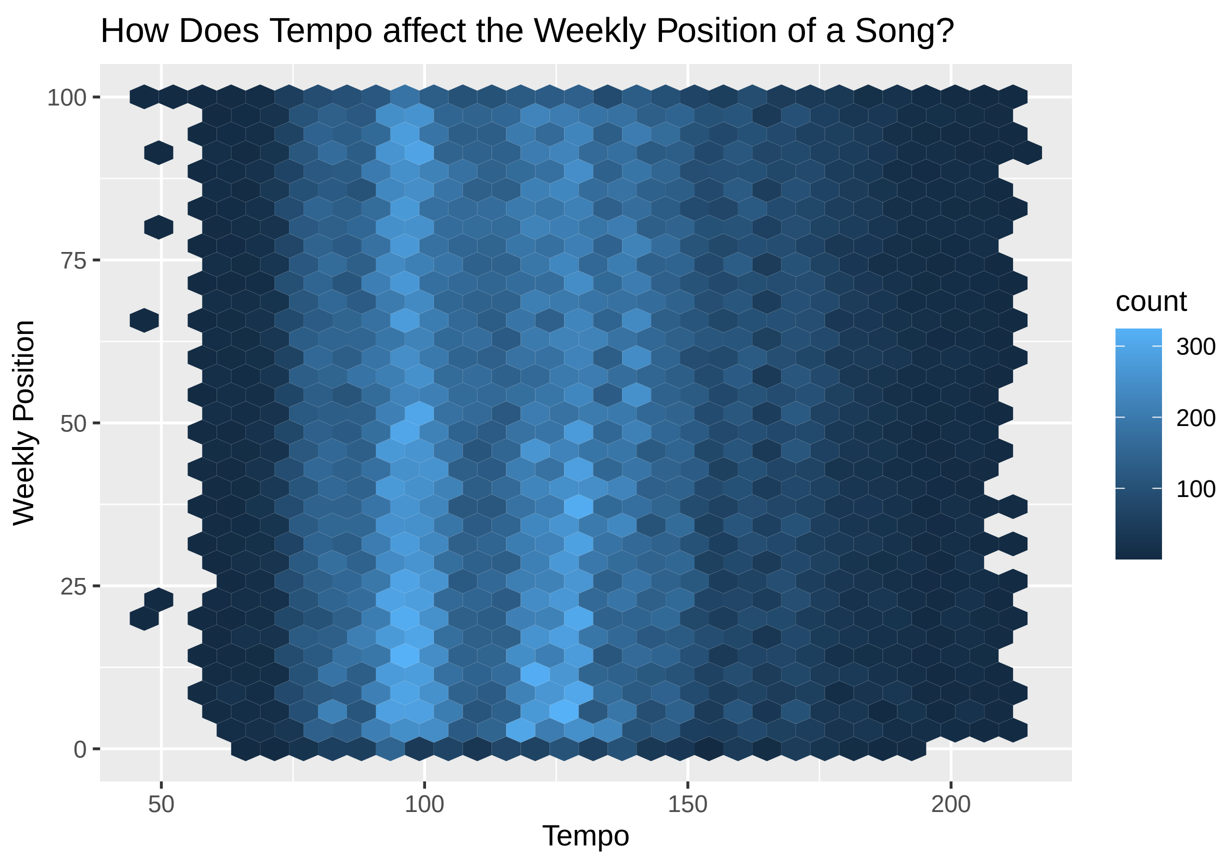
```
combined %>%
  ggplot(mapping = aes(x = valence, y = week_position)) +
  geom_hex() +
  labs(title = "How Does Valence affect the Weekly Position of a Song?",
       x = "Valence", y = "Weekly Position")
```

```
## Warning: Removed 8584 rows containing non-finite values (stat_binhex).
```



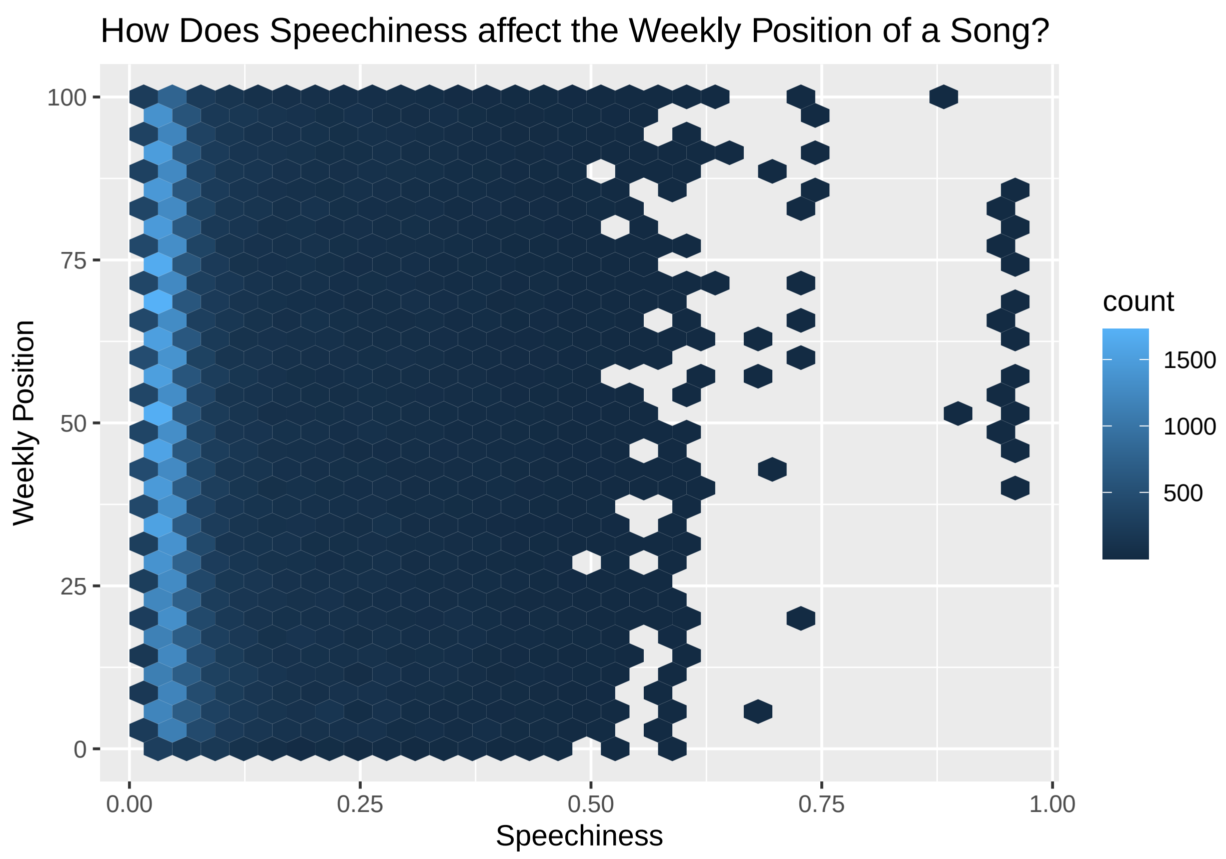
```
combined %>%
  ggplot(aes(x = tempo, y = week_position)) +
  geom_hex() +
  labs(title = "How Does Tempo affect the Weekly Position of a Song?",
       x = "Tempo", y = "Weekly Position")
```

```
## Warning: Removed 8584 rows containing non-finite values (stat_binhex).
```



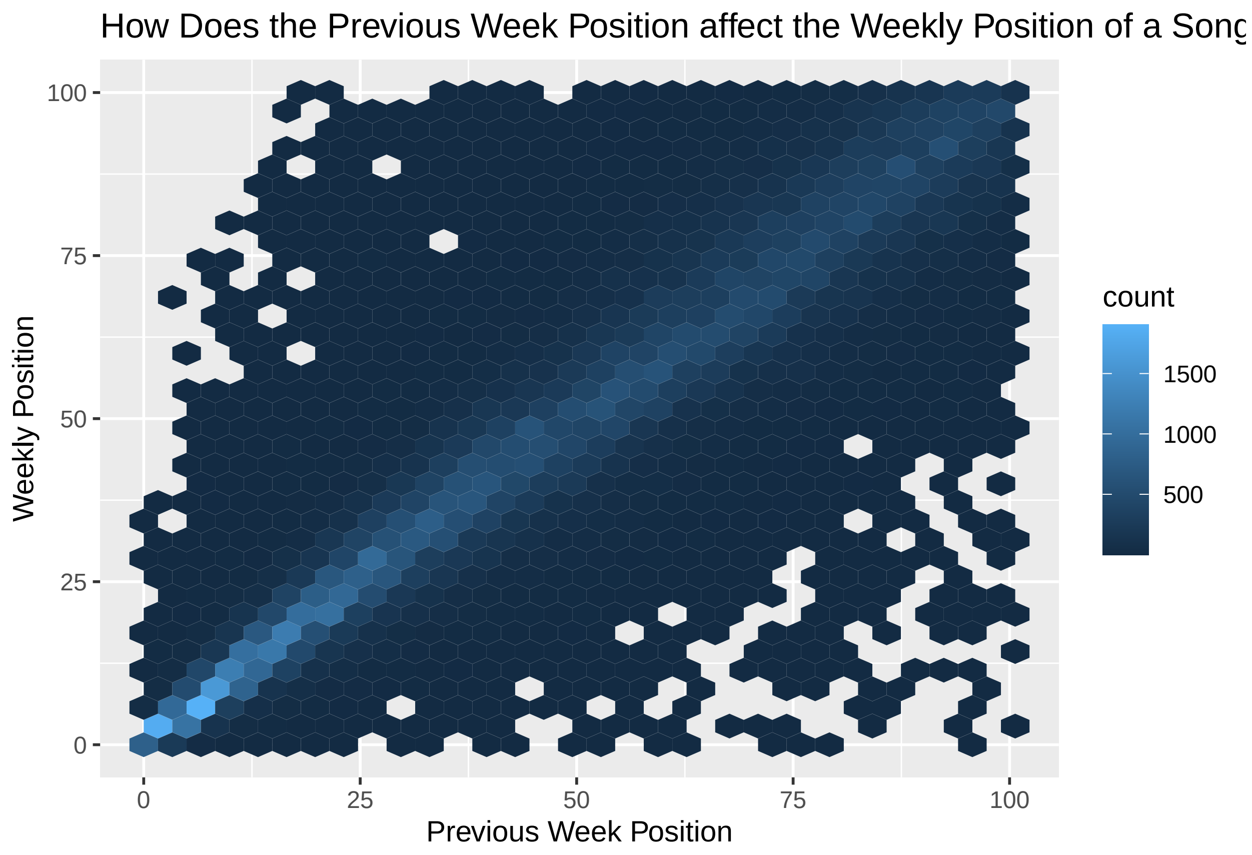
```
combined %>%
  ggplot(mapping = aes(x = speechiness, y = week_position)) +
  geom_hex() +
  labs(title = "How Does Speechiness affect the Weekly Position of a Song?",
       x = "Speechiness", y = "Weekly Position")
```

```
## Warning: Removed 8584 rows containing non-finite values (stat_binhex).
```



```
combined %>%
  ggplot(mapping = aes(x = previous_week_position, y = week_position)) +
  geom_hex() +
  labs(
    title = "How Does the Previous Week Position affect the Weekly Position of a Song?",
       x = "Previous Week Position", y = "Weekly Position")
```

```
## Warning: Removed 10855 rows containing non-finite values (stat_binhex).
```



Insert discussion about the graph

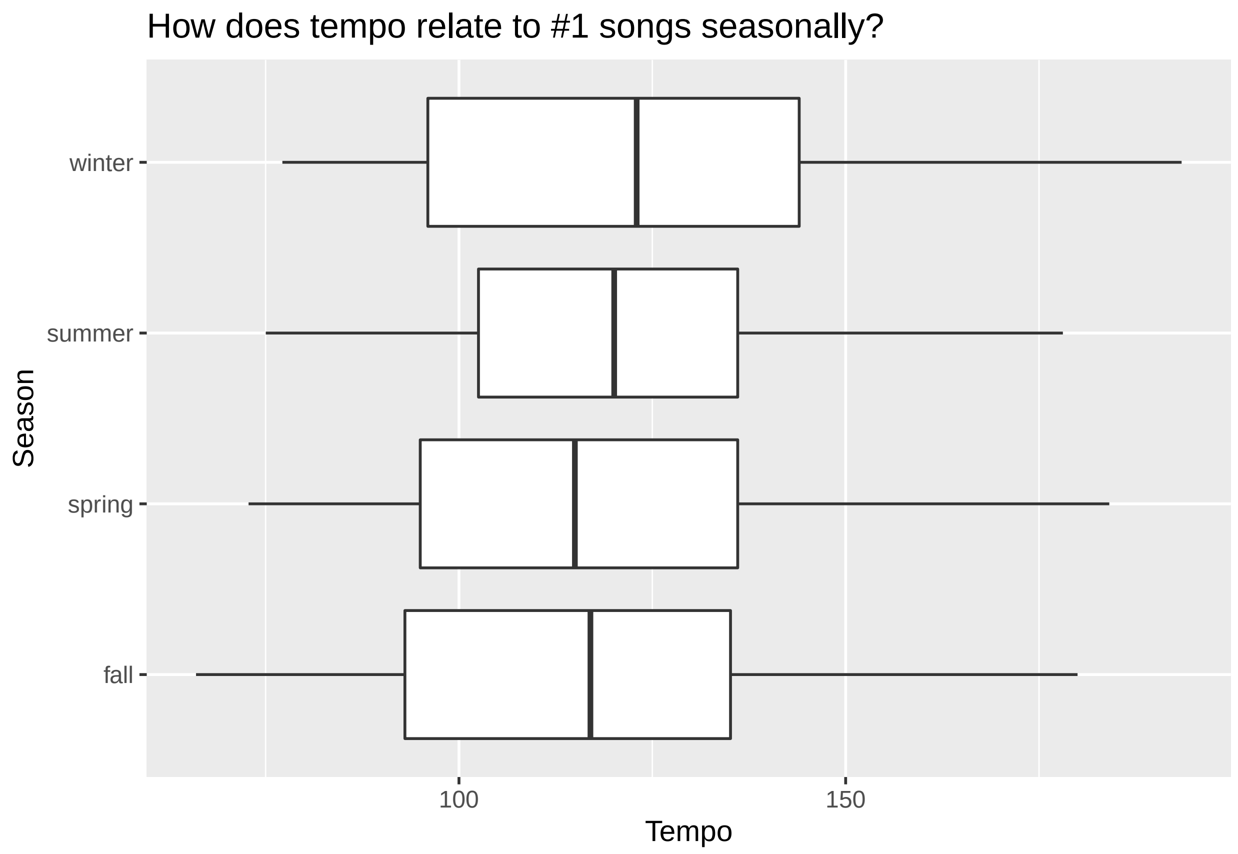
By Season

Seasons can influence what song attributes are preferred. For example, winter has many holidays, so that could mean that happier songs with medium tempos are preferred. To further examine attributes and #1 songs, we separated the dataset by seasons and looked at the distribution of attributes' values. We defined fall as from September to November, winter as from December to February, spring as March to May, and summer as June to August.

```
combined$month <- as.numeric(combined$month)
season_data <- combined %>%
  filter(week_position == 1) %>%
  mutate(season = case_when(month == (9) ~ "fall",
                            month == (10) ~ "fall",
                            month == (11) ~ "fall",
                            month == (12) ~ "winter",
                            month == (1) ~ "winter",
                            month == (2) ~ "winter",
                            month == (3) ~ "spring",
                            month == (4) ~ "spring",
                            month == (5) ~ "spring",
                            TRUE ~ "summer"))
```

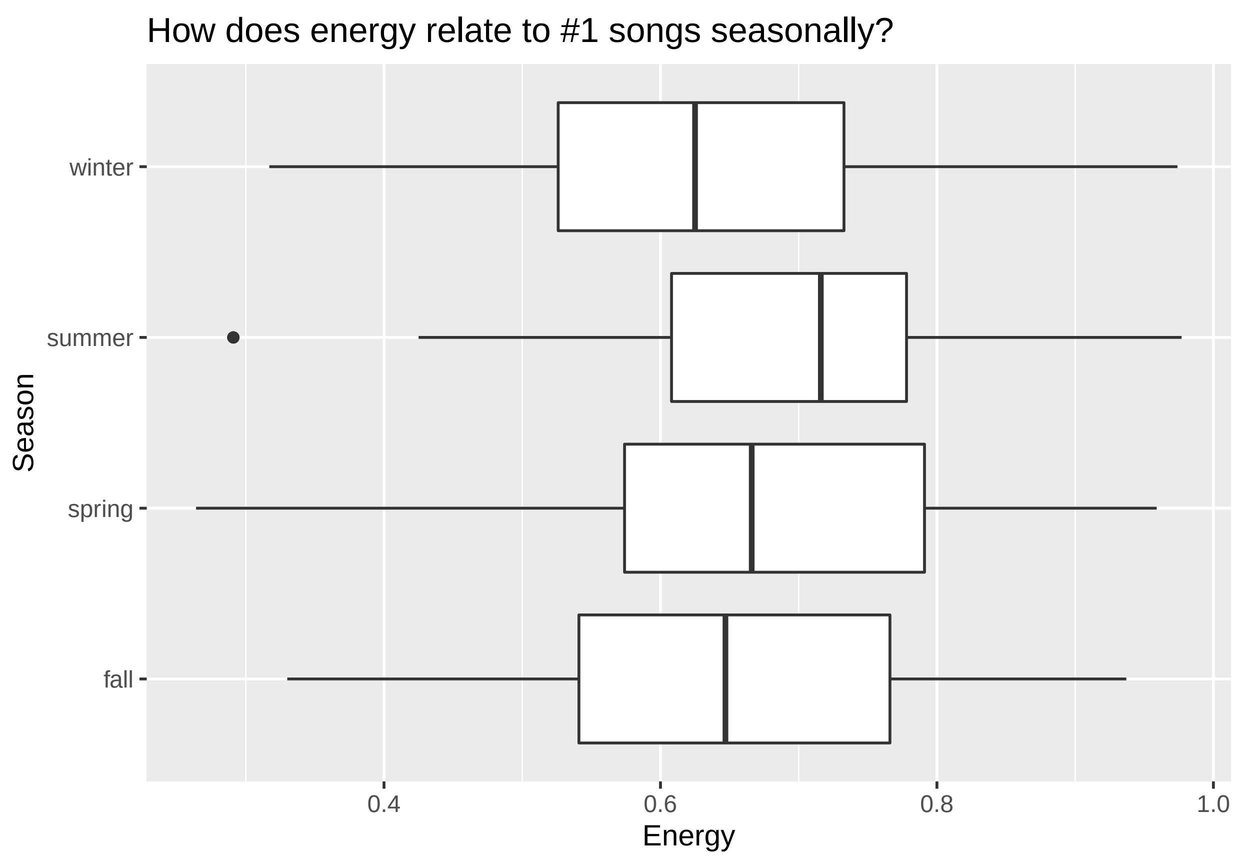
```
ggplot(data = season_data,
       mapping = aes(x = season, y = tempo)) +
  geom_boxplot() + coord_flip() +
  labs(title = "How does tempo relate to #1 songs seasonally?",
       x = "Season", y = "Tempo")
```

```
## Warning: Removed 95 rows containing non-finite values (stat_boxplot).
```



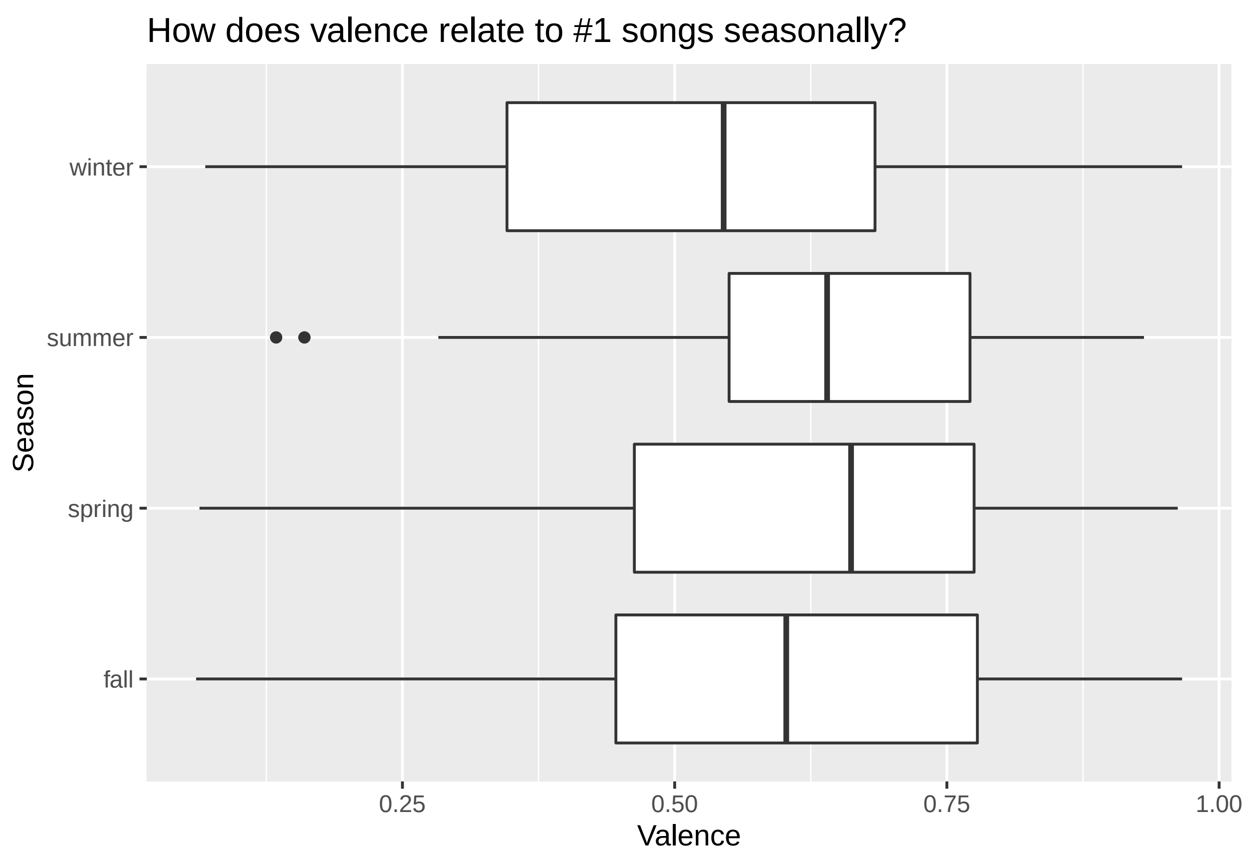
```
ggplot(data = season_data,
       mapping = aes(x = season, y = energy)) +
  geom_boxplot() + coord_flip() +
  labs(title = "How does energy relate to #1 songs seasonally?",
       x = "Season", y = "Energy")
```

```
## Warning: Removed 95 rows containing non-finite values (stat_boxplot).
```



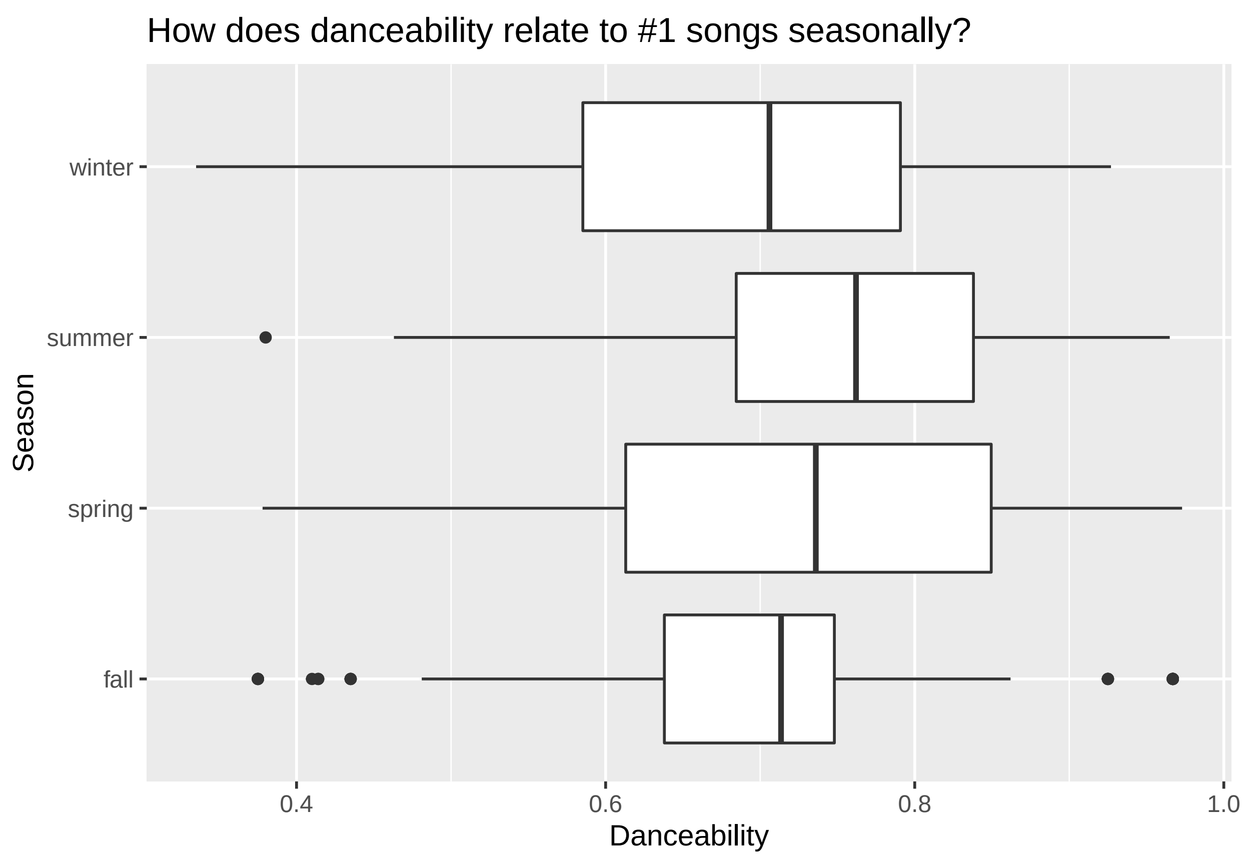
```
ggplot(data = season_data,
       mapping = aes(x = season, y = valence)) +
  geom_boxplot() + coord_flip() +
  labs(title = "How does valence relate to #1 songs seasonally?",
       x = "Season", y = "Valence")
```

```
## Warning: Removed 95 rows containing non-finite values (stat_boxplot).
```



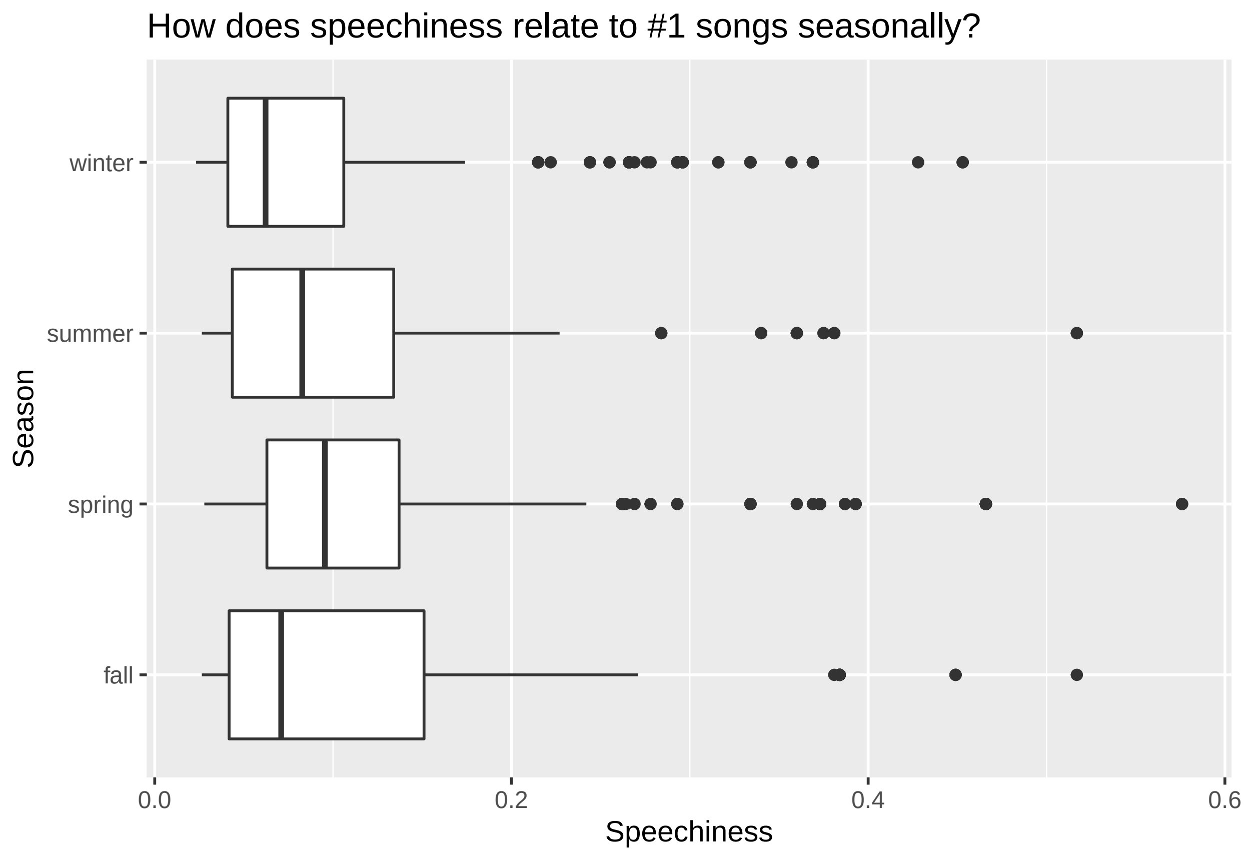
```
ggplot(data = season_data,
       mapping = aes(x = season, y = danceability)) +
  geom_boxplot() + coord_flip() +
  labs(title = "How does danceability relate to #1 songs seasonally?",
       x = "Season", y = "Danceability")
```

```
## Warning: Removed 95 rows containing non-finite values (stat_boxplot).
```



```
ggplot(data = season_data,
       mapping = aes(x = season, y = speechiness)) +
  geom_boxplot() + coord_flip() +
  labs(title = "How does speechiness relate to #1 songs seasonally?",
       x = "Season", y = "Speechiness")
```

```
## Warning: Removed 95 rows containing non-finite values (stat_boxplot).
```



## Genre and Billboard #1 Songs

If this happens: Since it seems like *insert factor or factors* don't really follow a pattern, we decided to remove *insert factor or factors* in the below analyses.

Testing Pop Genre

To examine the correlation between genre and Billboard rankings, we created a null and alternative hypothees to test with simulation-based methods. Since the labels "pop genre" and "popular genre" tend to be interchangeable, we predict that the majority of songs that are #1 on the Billboard are of the pop genre.

Null Hypothesis: 50% of the #1 Billboard songs are of the pop genre. Alternative Hypothesis: Over 50% of the #1 Billboard songs are of the pop genre.

$$H_0 : p_{pop} = 0.5 v.s. H_a : p_{pop} > 0.5$$

To test our hypotheses, we first created a dataset of just the #1 Billboard songs. We then checked to see if the word "pop" appears in the spotify genre column. Though the column entries are formatted like a list, it is actually a string, as evidenced by the result of the typeof(). So, we used str_detect to detect the word "pop". If "pop" does appear, the is_pop variable will have a "Yes". If not, the is_pop variable will have a "No". With our dataset set up, we constructed a null distribution. We set the seed to make this reproducible.

```
combined_1 <- combined %>%
  filter(week_position == 1)

typeof(combined_1$spotify_genre)
```

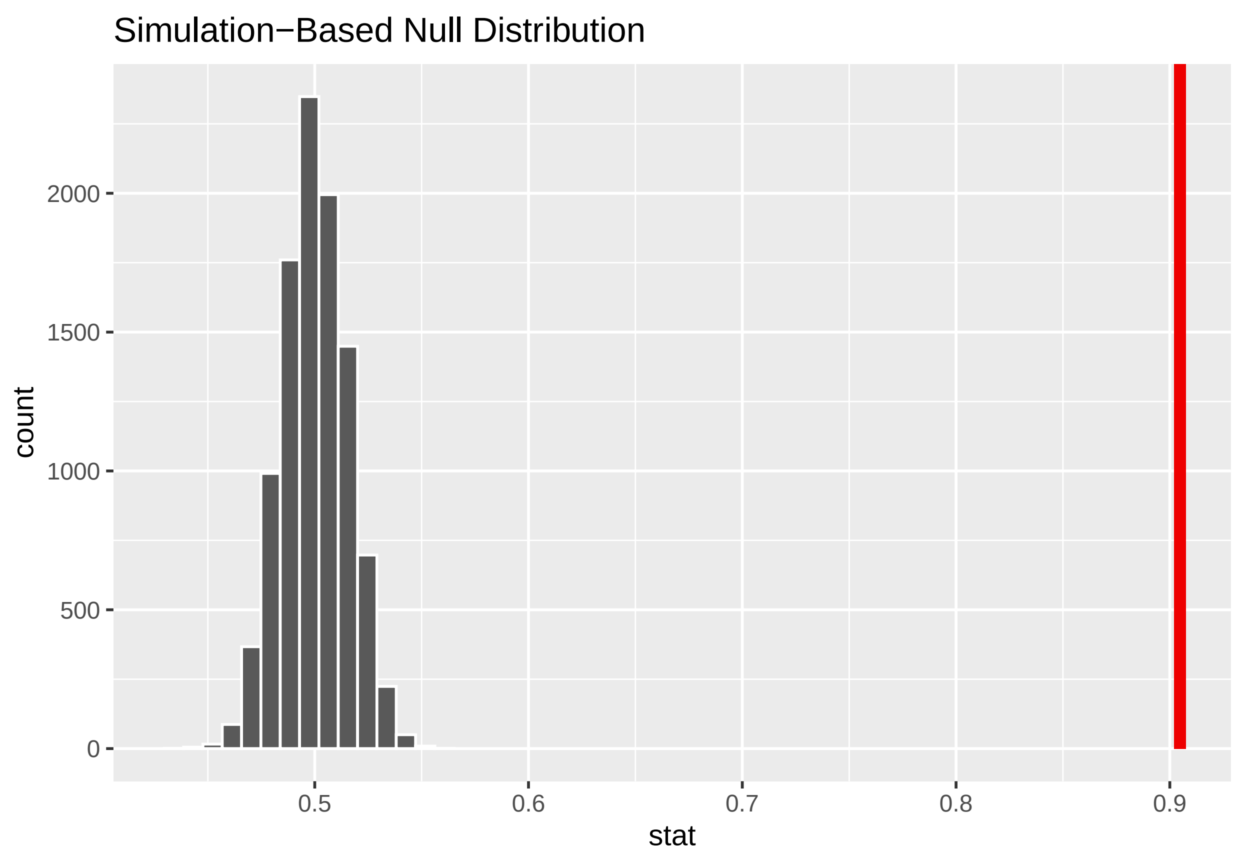
```
## [1] "character"
```

```r
combined_1 <- combined_1 %>%
  filter(spotify_genre != "[]") %>%
  mutate(is_pop= ifelse(str_detect(spotify_genre, "pop")== TRUE, "Yes", "No"))
```

```r
set.seed(1)
null_dist <- combined_1 %>%
  specify(response= is_pop, success = "Yes") %>%
  hypothesize(null= "point", p= 0.5) %>%
  generate(reps= 10000, type= "draw") %>%
  calculate(stat= "prop")

p_hat <- combined_1 %>%
  count(is_pop) %>%
  mutate(prob= n/ sum(n)) %>%
  filter(is_pop== "Yes") %>%
  pull()
```

```r
visualize(null_dist) +
  shade_p_value(obs_stat= p_hat, direction= "greater")
```



```r
p_value <- null_dist %>%
  get_p_value(obs_stat= p_hat, direction= "greater") %>%
  pull()
```

```
## Warning: Please be cautious in reporting a p-value of 0. This result is an
## approximation based on the number of `reps` chosen in the `generate()` step. See
## `?get_p_value()` for more information.
```

```
p_value
```

```
## [1] 0
```

Using the typical significance level of 0.05, we reject the null hypothesis. Given that the p-value is 0, there is strong evidence that the majority of #1 Billboard songs are of the pop genre.

ask Professor Smith This: why 98 confidence interval over other intervals? To verify our conclusion from above, we also decided to perform a confidence interval Test. Since our p-value is 0, we decided to do a 99% confidence interval test since that would give us less precision but a greater likelihood of containing our true proportion.

```
set.seed(2)
boot_dist <- combined_1 %>%
  specify(response= is_pop, success= "Yes") %>%
  generate(reps= 10000, type= "bootstrap") %>%
  calculate(stat= "prop")
```

```
CI <- boot_dist %>%
  summarize(lower= quantile(stat, 0.01),
            upper= quantile(stat, 0.99))

CI
```

```
## # A tibble: 1 x 2
##   lower upper
##   <dbl> <dbl>
## 1 0.884 0.925
```

We are 98% confident that the proportion of #1 Billboard songs that are of the pop genre is between 0.884 and 0.925. Since this interval does not have 0.5 between it, we reject our null hypothesis, further providing evidence that the majority of #1 Billboard songs are of the pop genre.

Testing Predictors

linear models To further examine the relationship between attributes and the weekly position of songs on the Billboard top 100, we decided to use linear regression. We thought that it would be interesting to observe which attribute is the strongest predictor of a song doing well on the Billboard. To do this we decided to test with logistic regression how the attributes; danceability, energy, speechiness, valence, and tempo predicted how well a song would do in becoming #1 on the Billboard charts.

First, we will show how the attribute danceability impacts the weekly position of a song.

```
danceability_model <- linear_reg() %>%
  set_engine("lm") %>%
  fit(danceability ~ week_position, data = combined)
tidy(danceability_model)
```

```
## # A tibble: 2 x 5
##   term           estimate std.error statistic p.value
##   <chr>             <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)    0.673    0.000876      769.        0
## 2 week_position -0.000635 0.0000151     -42.0       0
```

```
energy_model <- linear_reg() %>%
  set_engine("lm") %>%
  fit(energy ~ week_position, data = combined)
tidy(energy_model)
```

```
## # A tibble: 2 x 5
##   term          estimate std.error statistic  p.value
##   <chr>            <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   0.679    0.00101       672.  0
## 2 week_position 0.000231 0.0000174      13.3 3.86e-40
```

```
speechiness_model <- linear_reg() %>%
  set_engine("lm") %>%
  fit(speechiness ~ week_position, data = combined)
tidy(speechiness_model)
```

```
## # A tibble: 2 x 5
##   term             estimate std.error statistic  p.value
##   <chr>               <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    0.104      0.000625     167.   0
## 2 week_position -0.0000973 0.0000108     -9.01 2.05e-19
```

```
valence_model <- linear_reg() %>%
  set_engine("lm") %>%
  fit(valence ~ week_position, data = combined)
tidy(valence_model)
```

```
## # A tibble: 2 x 5
##   term            estimate std.error statistic  p.value
##   <chr>              <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    0.547    0.00136       403.  0
## 2 week_position -0.000385 0.0000234     -16.5 7.89e-61
```

```
tempo_model <- linear_reg() %>%
  set_engine("lm") %>%
  fit(tempo ~ week_position, data = combined)
tidy(tempo_model)
```

```
## # A tibble: 2 x 5
##   term          estimate std.error statistic  p.value
##   <chr>            <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   119.       0.180       663.  0
## 2 week_position   0.0425   0.00310      13.7 1.26e-42
```

Calculation of R^2 Values of the different predictors.

```
glance(danceability_model)$r.squared
```

```
## [1] 0.01639905
```

```
glance(energy_model)$r.squared
```

```
## [1] 0.001661981
```

```
glance(speechiness_model)$r.squared
```

```
## [1] 0.000767592
```

```
glance(valence_model)$r.squared
```

```
## [1] 0.002557178
```

```
glance(tempo_model)$r.squared
```

```
## [1] 0.001769466
```

Insert Discussion Talk about other factors that make a song popular that are influential such as social media or there is not variable called catchiness which could show how catchy a song is.

Things We Want Feedback on from Peer Review

Is our research question clear?

What do you think of the tests and visualizations that we're planning on performing?

How do you feel about our organization? Do you have suggestions on how we should be organizing our things?

Are there other tests/hypotheseses we should look at?

Note: this is just a basic template of our project. Some of the things that we are examining or variables could change just keep this in mind. Let us know what we can do to improve our project!