Case Study - Song Performance on the Billboard Hot 100 from 1991-2021

Caitlin Connolly, Conrad Sorenson, William Liang, and Isabella Gonzalez

Introduction

The Billboard Hot 100 is the the "music industry's standard record chart in the United States for songs, published weekly by Billboard magazine". Billboard Magazine is known to be the most influential music media brand not only across the United States, but across the rest of the globe as well. This is due to the brand's exceptional reports on latest music news across all different music genres.

In 1958, Billboard published The Hot 100 for the first time, it combined all music genres song's info into a single chart and is ranked by radio airplay audience impressions measured by Nielsen BDS. Now, chart rankings are based on sales (physical or digital), radio play, and online streaming in the United States. The data collection process for these individual aspects differs. An artist or band's ability to have hits in the Hot 100 is the first sign of success, while being able to maintain music on the chart throughout time is recognized as long term success and the ability to adapt to new audiences.[1]

Our group wanted to take the opportunity to explore the datasets containing information about songs in The Hot 100. The data has been collected from tidytuesday on Git Hub, and provides information about songs on the Hot 100. The two datasets from tidytuesday comes from "Data.World by way of Sean Miller, Billboard.com and Spotify."[2]

More information about these datasets can be found in the Data section.

[1]Reference: https://en.wikipedia.org/wiki/Billboard_Hot_100#

[2] tidytuesday September 14, 2021 https://github.com/rfordatascience/tidytuesday/blob/master/data/2021/2021-09-14/readme.md

Questions

The questions we will be exploring with our datasets are:

1. How have song genre popularity's changed over time? What were the most popular genre's in each decade?
2. How has the length of a song changed over time in terms of popularity? Are songs today shorter than in previous decades?
3. What are the songs that lasted longest in the top 10 on the chart?
4. Is there any relationship between a song's popularity and its audio features?

Load packages

```
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
library(tidyverse)
library(tidyr)
library(pdftools)
library(readxl)
library(ggrepel)
library(naniar)
library(plm)
library(broom)
library(stats)
library(ggcorrplot)
library(ggplot2)
library(rsample)
library(GGally)
library(car)
library(mlbench)
library(glmnet)
library(lubridate)
library(caTools)
library(caret)
```

The Data

The first data set provides information such as the position the song got that week, the song name, the artist name, the instances the song has appeared on the chart before, the song's previous week position, the song's peak position, and how many weeks it has been on the chart. The second data set provides information about the song's audio features such as genre, duration, duration, and other features such as danceability, loudness, energy, and various others.[3]

[3] From the Data Dictionary on the tidytuesday GitHub pages https://github.com/rfordatascience/tidytuesday/blob/master/data/2021/2021-09-14/readme.md#data-dictionary

Data Import

Since the datasets we chose were so large, we saved the raw data to a data folder after reading in the .csv files using readr and loaded it from there. The commented code is that process and only needs to be run once. The uncommented code is how we subsequently loaded the data from the folder.

```
# read in once for data folder, then just load

# billboard <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/da
# audio_features <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/mast

# save(billboard, file="data/raw/billboard.rda")
# save(audio_features, file="data/raw/audio_features.rda")


load("data/raw/billboard.rda")
load("data/raw/audio_features.rda")
```

Let's take a look at the `billboard` columns.

```
glimpse(billboard)
```

```
## Rows: 327,895
## Columns: 10
## $ url                    <chr> "http://www.billboard.com/charts/hot-100/1965-0~
## $ week_id                <chr> "7/17/1965", "7/24/1965", "7/31/1965", "8/7/196~
## $ week_position          <dbl> 34, 22, 14, 10, 8, 8, 14, 36, 97, 90, 97, 97, 9~
## $ song                   <chr> "Don't Just Stand There", "Don't Just Stand The~
## $ performer              <chr> "Patty Duke", "Patty Duke", "Patty Duke", "Patt~
## $ song_id                <chr> "Don't Just Stand TherePatty Duke", "Don't Just~
## $ instance               <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ previous_week_position <dbl> 45, 34, 22, 14, 10, 8, 8, 14, NA, 97, 90, 97, 9~
## $ peak_position          <dbl> 34, 22, 14, 10, 8, 8, 8, 8, 97, 90, 90, 90, 90,~
## $ weeks_on_chart         <dbl> 4, 5, 6, 7, 8, 9, 10, 11, 1, 2, 3, 4, 5, 6, 1, ~
```

Let's take a look at the audio_features columns.

```
glimpse(audio_features)
```

```
## Rows: 29,503
## Columns: 22
## $ song_id                   <chr> "-twistin'-White Silver SandsBill Black's Co~
## $ performer                 <chr> "Bill Black's Combo", "Augie Rios", "Andy Wi~
## $ song                      <chr> "-twistin'-White Silver Sands", "¿Dònde Està~
## $ spotify_genre             <chr> "[]", "['novelty']", "['adult standards', 'b~
## $ spotify_track_id          <chr> NA, NA, "3tvqPPpXyIgKrm4PR9HCf0", "1fHHq3qHU~
## $ spotify_track_preview_url <chr> NA, NA, "https://p.scdn.co/mp3-preview/cef48~
## $ spotify_track_duration_ms <dbl> NA, NA, 166106, 172066, 211066, 208186, 2055~
## $ spotify_track_explicit    <lgl> NA, NA, FALSE, FALSE, FALSE, FALSE, TRUE, FA~
## $ spotify_track_album       <chr> NA, NA, "The Essential Andy Williams", "Comp~
## $ danceability              <dbl> NA, NA, 0.154, 0.588, 0.759, 0.613, NA, 0.64~
## $ energy                    <dbl> NA, NA, 0.185, 0.672, 0.699, 0.764, NA, 0.68~
## $ key                       <dbl> NA, NA, 5, 11, 0, 2, NA, 2, NA, NA, 7, NA, 1~
## $ loudness                  <dbl> NA, NA, -14.063, -17.278, -5.745, -6.509, NA~
## $ mode                      <dbl> NA, NA, 1, 0, 0, 1, NA, 0, NA, NA, 1, NA, 0,~
## $ speechiness               <dbl> NA, NA, 0.0315, 0.0361, 0.0307, 0.1360, NA, ~
## $ acousticness              <dbl> NA, NA, 0.91100, 0.00256, 0.20200, 0.05270, ~
## $ instrumentalness          <dbl> NA, NA, 2.67e-04, 7.45e-01, 1.31e-04, 0.00e+~
## $ liveness                  <dbl> NA, NA, 0.1120, 0.1450, 0.4430, 0.1970, NA, ~
## $ valence                   <dbl> NA, NA, 0.150, 0.801, 0.907, 0.417, NA, 0.95~
## $ tempo                     <dbl> NA, NA, 83.969, 121.962, 92.960, 160.015, NA~
## $ time_signature            <dbl> NA, NA, 4, 4, 4, 4, NA, 4, NA, NA, 4, NA, 4,~
## $ spotify_track_popularity  <dbl> NA, NA, 38, 11, 77, 73, 61, 40, NA, NA, 31, ~
```

Data Wrangling

First merge the two datasets on the song_id column and store it in the variable "full_billboard".

```
full_billboard <- merge(billboard, audio_features, by = 'song_id')
```

Remove repeat columns and rename song.y and performer.y to song and performer respectively.

```
full_billboard <- full_billboard |>
   select(-song.x, -performer.x) |>
   rename('song'='song.y', 'performer' = 'performer.y')
```

Create a "spotify_track_duration_seconds" column using mutate and dividing the values in the spotify_track_duration_ms column by 1000 to get the duration in seconds. Also create a spotify_track_duration_min column by dividing the values in the "spotify_track_duration_seconds" column by 60.

These will be helpful if we want to see the length of songs in units other than milliseconds (since we rarely use milliseconds when quantifying song length).

```
full_billboard <- full_billboard |>
   mutate(spotify_track_duration_seconds = spotify_track_duration_ms /1000) # get seconds
```

```
full_billboard <- full_billboard |>
  mutate(spotify_track_duration_min = spotify_track_duration_seconds / 60) # get minutes
```

Using `mutate`, create a week column and extract the week from the week_id column using `mdy` from lubridate to transform the weeks into dates.

This transformation will allow us to more easily work with dates for necessary analyses.

```
full_billboard <- full_billboard |>
  mutate(week = mdy(week_id)) # turn into date type
```

Pull the week_id from the `full_billboard` dataframe and split the dates on the slash to get a list of dates split into month, day, and year and store it into a variable. Then unlist those lists into three columns in a matrix, keeping the rows intact so the first column is all months, second is all days, and third is all years and transom the matrix into a dataframe. Select the third column and rename it "year". Apply this column to a new column called "year" in full_billboard.

This "year" column will allow us to easily look at the trajectory of songs over the years and how the positions and popularity changes across different variables like genre.

Remove unnecessary columns from full_billboard so we only have the variables we need to perform EDA and other data analyses.

```
day_listed <- full_billboard |>
  pull("week_id") |>
  strsplit("/") # get list of dates without /

year_col <-
  data.frame(matrix(unlist(day_listed), ncol = 3, byrow=TRUE)) |> # got from stack overflow
  select("X3") |>
  rename("year" = "X3")

 # https://stackoverflow.com/questions/4227223/convert-a-list-to-a-data-frame
 # from the Update July 2020: portion of the first answer

full_billboard['year'] <- year_col

#removing unnecessary or incomplete variables
full_billboard <- subset(full_billboard, select = -c(previous_week_position,spotify_track_id, spotify_t
```

Finally filter the songs so that they were charting after the year 1990. Remove any NA values in the dataset.

We checked that all NA values were removed because any NA values are not able to be used in analysis and it made our dataset more manageable for our purposes. We chose to look at songs on the chart from 1991 on due to our dataset being so large it would crash when running and knitting the rmd file.

```
#Filtering for songs in the Hot 100 after 1990
full_billboard <- filter(full_billboard, year > 1990) |>
  drop_na()
#making double sure there are no NA values
if (any(is.na(full_billboard))) {
  print("there are NA values in dataset")
} else {print("No NA values")}
```

```
## [1] "No NA values"
```

Let's take a look at our wrangled dataset to double check everything is wrangled correctly.

```
glimpse(full_billboard)
```

```
## Rows: 143,962
## Columns: 27
## $ song_id                      <chr> "...Baby One More TimeBritney Spears", ~
## $ week_position                <dbl> 3, 11, 4, 29, 1, 8, 13, 38, 9, 5, 31, 4~
## $ instance                     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ peak_position                <dbl> 3, 1, 4, 1, 1, 1, 1, 1, 9, 1, 1, 4, 8, ~
## $ weeks_on_chart               <dbl> 10, 18, 9, 27, 11, 17, 19, 29, 4, 16, 2~
## $ performer                    <chr> "Britney Spears", "Britney Spears", "Br~
## $ song                         <chr> "...Baby One More Time", "...Baby One M~
## $ spotify_genre                <chr> "['dance pop', 'pop', 'post-teen pop']"~
## $ spotify_track_duration_ms    <dbl> 211066, 211066, 211066, 211066, 211066,~
## $ spotify_track_explicit       <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALS~
## $ spotify_track_album          <chr> "...Baby One More Time (Digital Deluxe ~
## $ danceability                 <dbl> 0.759, 0.759, 0.759, 0.759, 0.759, 0.75~
## $ energy                       <dbl> 0.699, 0.699, 0.699, 0.699, 0.699, 0.69~
## $ loudness                     <dbl> -5.745, -5.745, -5.745, -5.745, -5.745,~
## $ mode                         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ speechiness                  <dbl> 0.0307, 0.0307, 0.0307, 0.0307, 0.0307,~
## $ acousticness                 <dbl> 0.202, 0.202, 0.202, 0.202, 0.202, 0.20~
## $ instrumentalness             <dbl> 0.000131, 0.000131, 0.000131, 0.000131,~
## $ liveness                     <dbl> 0.443, 0.443, 0.443, 0.443, 0.443, 0.44~
## $ valence                      <dbl> 0.907, 0.907, 0.907, 0.907, 0.907, 0.90~
## $ tempo                        <dbl> 92.96, 92.96, 92.96, 92.96, 92.96, 92.9~
## $ time_signature               <dbl> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, ~
## $ spotify_track_popularity     <dbl> 77, 77, 77, 77, 77, 77, 77, 77, 77, 77,~
## $ spotify_track_duration_seconds <dbl> 211.066, 211.066, 211.066, 211.066, 211~
## $ spotify_track_duration_min   <dbl> 3.517767, 3.517767, 3.517767, 3.517767,~
## $ week                         <date> 1999-01-23, 1999-03-20, 1999-01-16, 19~
## $ year                         <chr> "1999", "1999", "1999", "1999", "1999",~
```

Save the wrangled data in data folder. Can load the data afterwards so you don't have to clean the data each time when needing to get original wrangled data when performing data analysis.

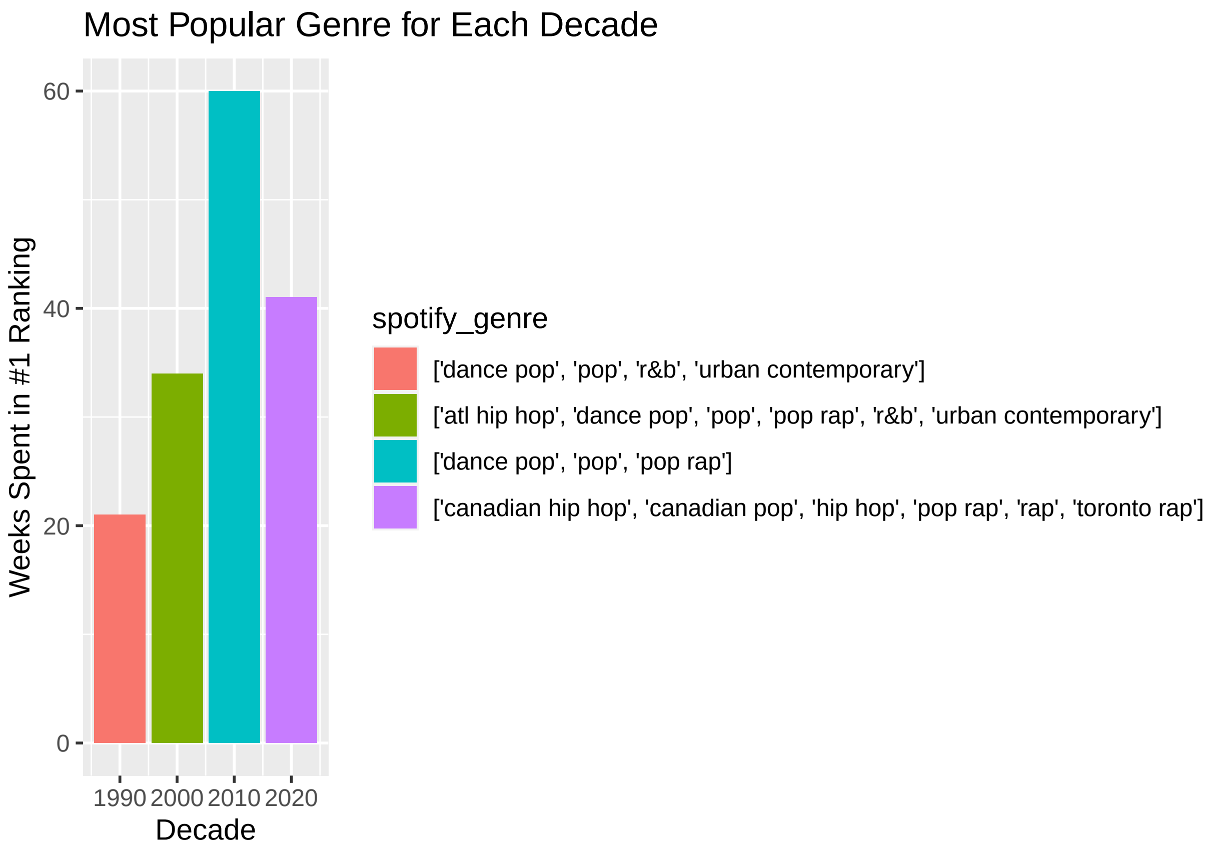
```
save(full_billboard, file="data/wrangled/full_billboard.rda")
load("data/wrangled/full_billboard.rda")
```

Analysis

Exploratory Data Analysis

```
by_genre_decade <- full_billboard |>
   filter(!is.na(spotify_genre)) |> # get genres not NA
   group_by(spotify_genre,
            decade = round(year(week)/10) *10) |> # get decade
    summarize(total_weeks_in_top1 = sum(week_position == 1)) |> # get totals weeks at number 1
   arrange(desc(total_weeks_in_top1))

by_genre_decade |>
   group_by(decade) |>
   slice_max(total_weeks_in_top1, n = 1) |>
   mutate(spotify_genre = fct_reorder(spotify_genre, total_weeks_in_top1)) |>
   ggplot(aes(decade, total_weeks_in_top1, fill=spotify_genre)) +
   geom_col() +
   labs(x = 'Decade',
        y = 'Weeks Spent in #1 Ranking',
        title = 'Most Popular Genre for Each Decade')
```

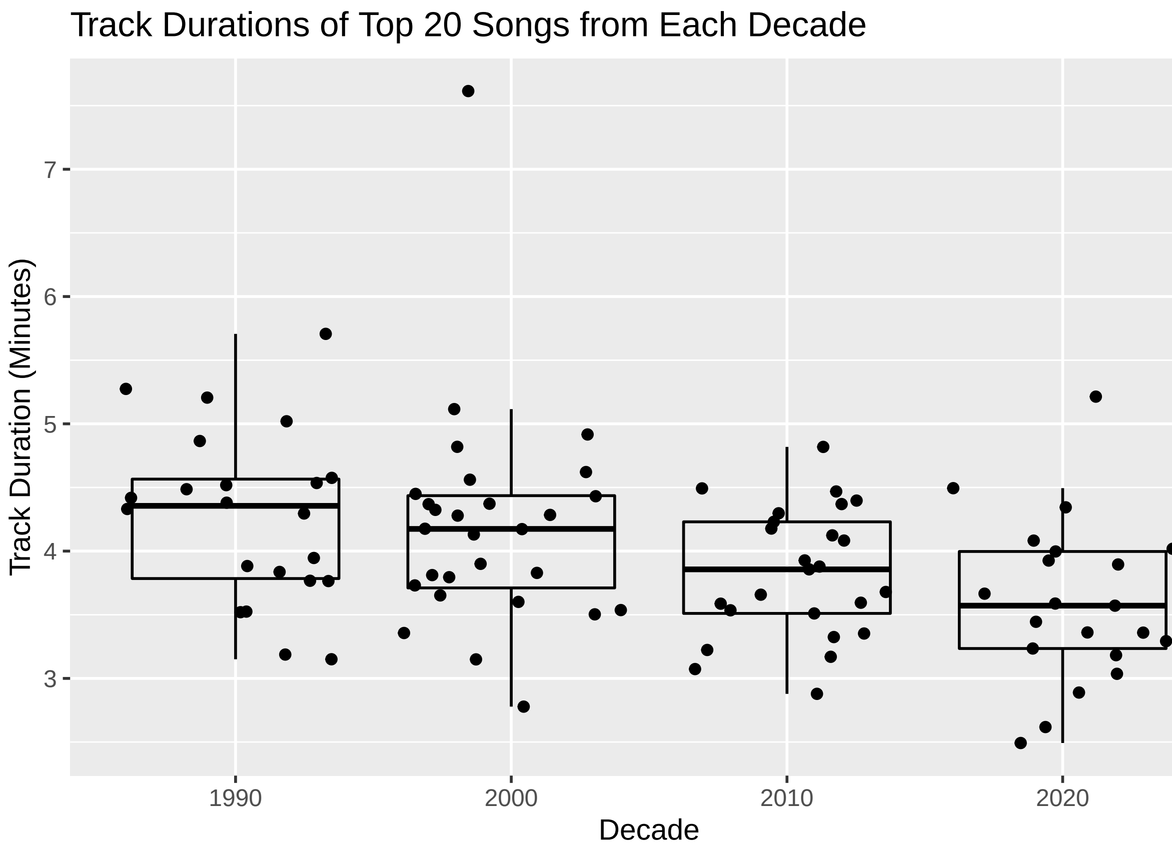
Q1: How has song genre popularity changed over time? What was the most popular genre in each decade?



This figure shows the most popular genre for the 1990s, 2000s, 2010s, and 2020s. These genres were determined to be most popular by computing the genres that had the most number of weeks recorded in the number one spot for the decade.

```
decade_song_duration <- full_billboard |>
   group_by(song_id,
        song,
        performer,
        spotify_track_duration_min,
            decade = round(year(week)/10) *10) |>
    summarize(total_weeks_in_top1 = sum(week_position <= 10)) |>
   arrange(desc(total_weeks_in_top1))

decade_song_duration |>
  group_by(decade) |>
  slice_max(total_weeks_in_top1, n = 20) |>
mutate(song = fct_reorder(song, spotify_track_duration_min)) |>
ggplot(aes(factor(decade), spotify_track_duration_min)) +
    geom_point(position='jitter') +
  geom_boxplot(alpha=0, colour="black") +
   labs(x = 'Decade',
        y = 'Track Duration (Minutes)',
        title = 'Track Durations of Top 20 Songs from Each Decade')
```

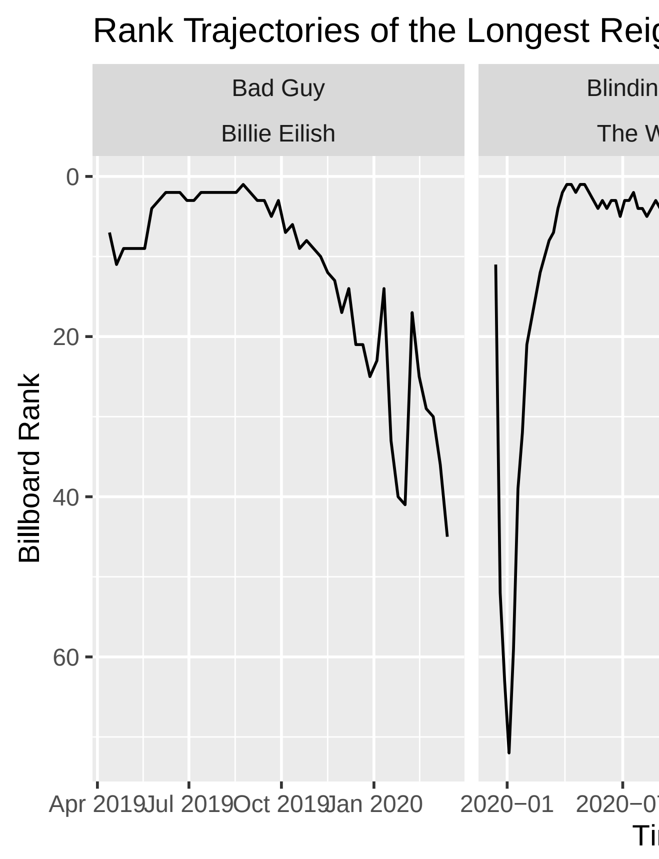
Q2: How has the length of popular songs changed over time? Are songs today shorter than in



previous decades?

This figure shows the track duration of the top 20 songs from each decade.

```
longest_top_10 <- full_billboard |>
    filter(week_position <= 10) |>
    count(song_id, song, performer, sort = TRUE)

full_billboard |>
   semi_join(head(longest_top_10, 3), by = 'song_id') |>
    ggplot(aes(week, week_position, group = instance)) +
    geom_line() +
   facet_wrap(~ song + performer, scales = 'free_x') +
    scale_y_reverse() +
   labs(x = 'Time',
        y = 'Billboard Rank',
        title = 'Rank Trajectories of the Longest Reigning Top 10 Hits')
```

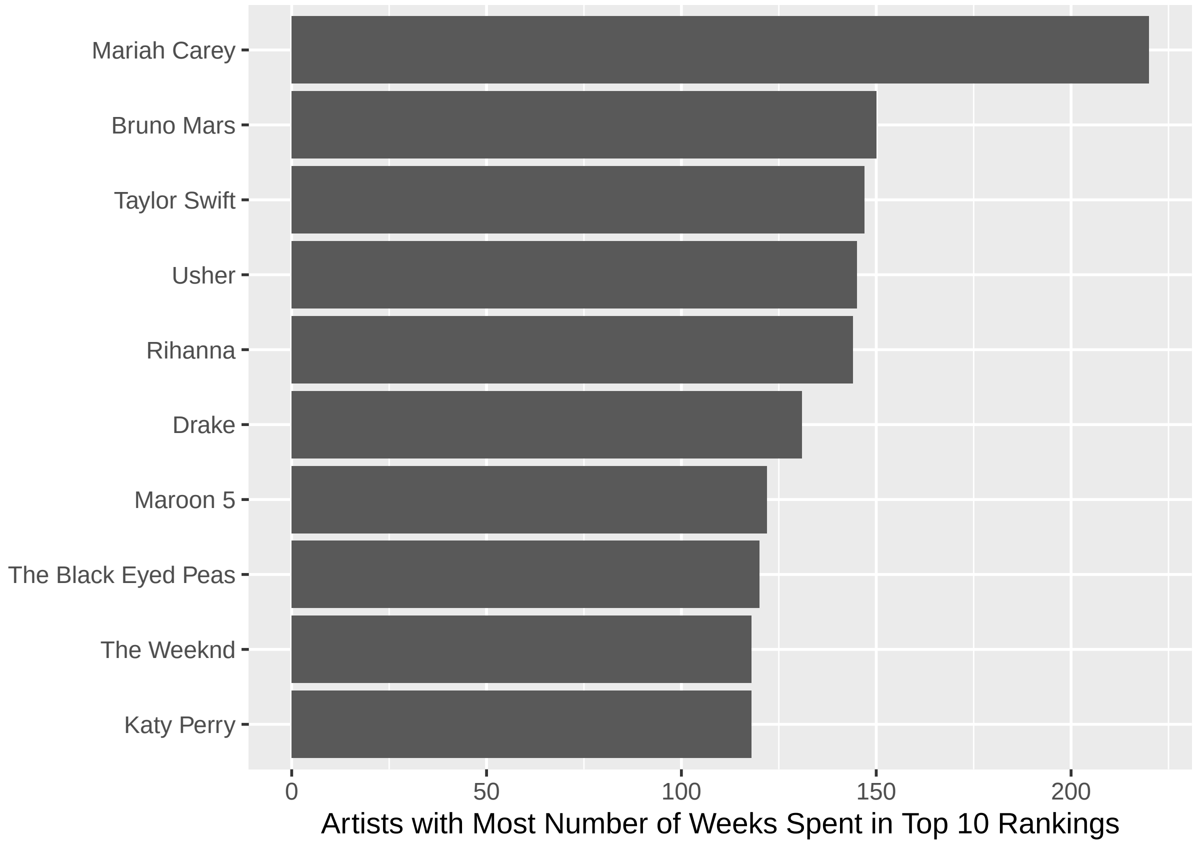


Q3a: What are the songs that lasted longest in the top 10?

This figure shows the top 3 songs of all time that had the most weeks in the top 10 rankings.

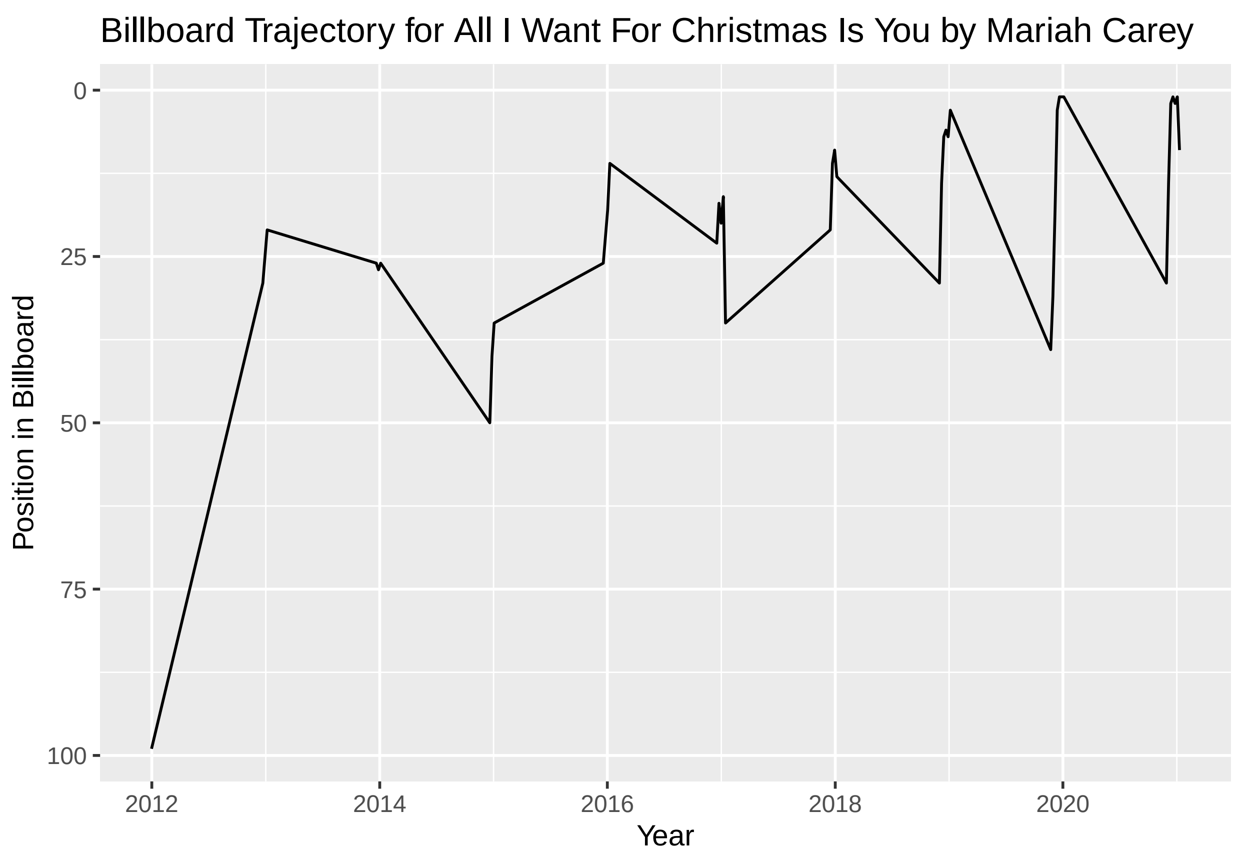
```r
top_10_performers <- full_billboard |>
   group_by(performer) |>
   summarize(total_weeks_in_top10 = sum(week_position <= 10)) |>
   arrange(desc(total_weeks_in_top10))

top_10_performers |>
   arrange(desc(total_weeks_in_top10)) |>
   head(10) |>
   mutate(performer = fct_reorder(performer, total_weeks_in_top10)) |>
   ggplot(aes(total_weeks_in_top10, performer)) +
   geom_col() +
   labs(x = 'Artists with Most Number of Weeks Spent in Top 10 Rankings',
        y = '')
```

Q3b: Who are the artists with the most number of weeks spent in the Top 10?



This figure shows the list of artists with the most number of weeks spent in the Top 10 rankings in all of time. All the artists are household names that are to be expected on this list, and most of them share similarly high numbers. One outlier is Mariah Carey, who is significantly ahead of everyone else in the number of weeks in top 10 rankings. This is because Mariah Carey's Christmas song, All I Want For Christmas Is You, reaches the top rankings during the holiday season every year, while the other artists' songs are generally ranked in the charts centered around the time of their release.

```
full_billboard |>
    filter(song == 'All I Want For Christmas Is You', year > 2010) |>
    ggplot(aes(week, week_position)) +
    geom_line() +
    scale_y_reverse() +
  labs(x = 'Year',
       y = 'Position in Billboard',
      title = 'Billboard Trajectory for All I Want For Christmas Is You by Mariah Carey')
```



As observed in the figure above, All I Want for Christmas began to gain popularity in the early 2010s, and is consistently in the Top 100 every single year, often breaking into the Top 10. As such, this is consistent with the observation that Mariah Carey holds the top spot of the artist with the most number of weeks in the Top 10.

Data Analysis

Q4: Is there any relationship between a song's popularity and its audio features? Experimenting with music feature variables: Taking a look at how audio features affect weeks on chart. Because the data has different variables that determine popularity differently like week position, or peak position as of the week before. Due to time constraints, we chose to determine popularity as the number of weeks a single song has been on the chart. So we determined popularity by the max number of weeks of every song on the hot 100. Created a normalization function for the data since variables had a range of different values.

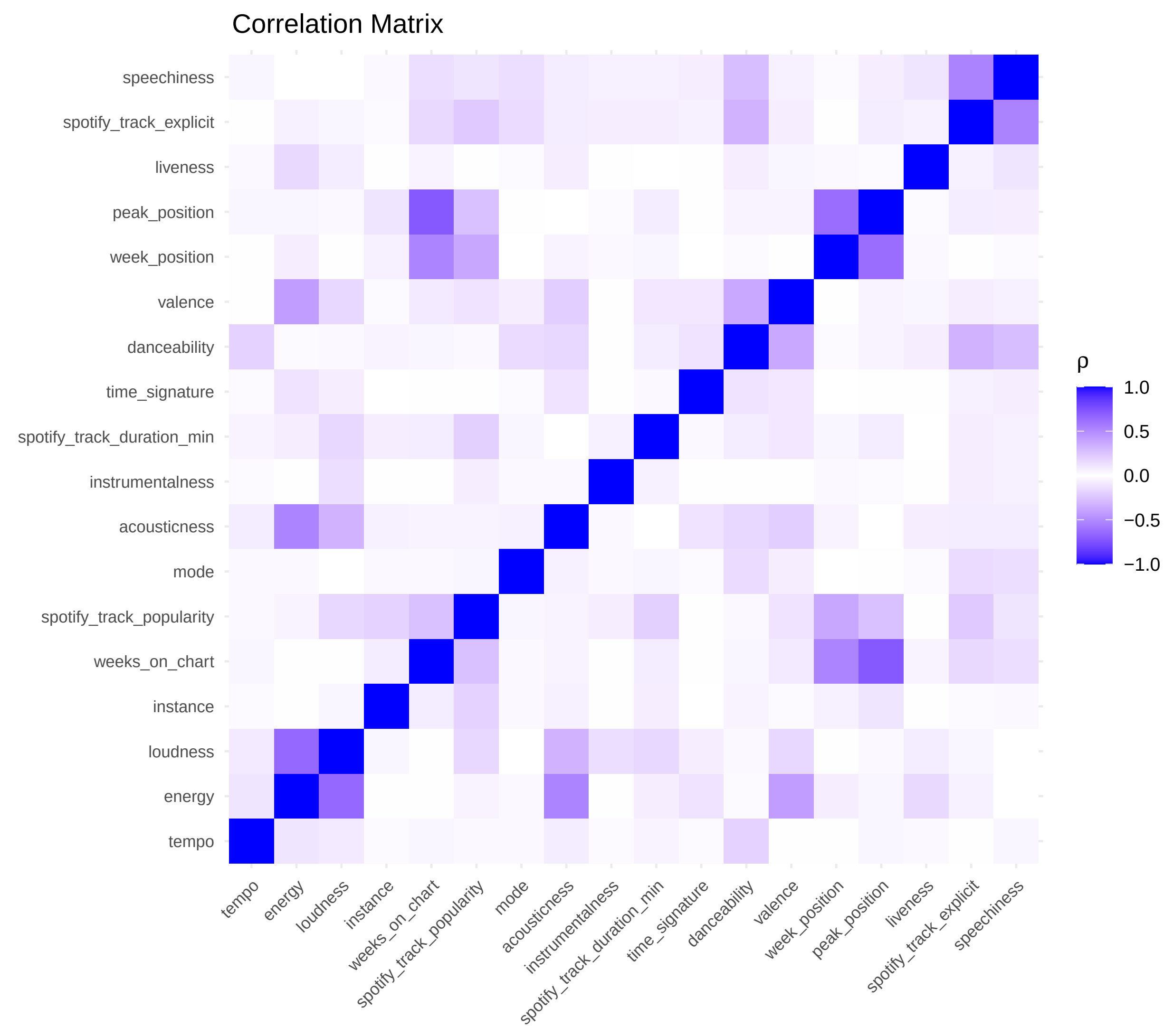
Then we excluded variables with reason, the data set had multiple song duration variables by different metrics of time, so if we kept them it would not assume the law of independence, so we chose to include only one duration variable. Then we removed variables that were strings that could not be converted to numeric such as the name of an album or the name of the artist. Lastly, we removed spotify genre variable because creating dummy columns for each genre created too large of a data set due to the high number of genres spotify has.

```r
#normalizing data

min_max_norm <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

#taking out out variables i think cannot be included and dont assume the law of independence, or that c
```

```
r <- full_billboard |>
   select(-c("performer", 'spotify_track_album', 'spotify_track_duration_seconds', 'spotify_track_durat

#popularity determined by how many weeks a song has been on chart. trying to filter each song for its h
#grouping by song and finding the maximum value for each weeks on chart for every song.
r <- r |>
  group_by(song) |>
  top_n(1, weeks_on_chart)
r <- ungroup(r)
r2 <- r |>
  select(-song)

bill_norm <- as.data.frame(lapply(r2[2:19], min_max_norm))

#making formula
BILL_variables <- bill_norm |>
  select(-c("weeks_on_chart")) |>
  colnames()
fmla <-
  as.formula(paste("weeks_on_chart ~", paste(BILL_variables, collapse = " + ")))
```

Then we wanted to check for any correlation between variable pairs:

```
cor_BILL <- cor(bill_norm)
cor_BILL <- as.data.frame(cor_BILL)
ggcorrplot(cor_BILL,
  tl.cex = 8,
  hc.order = TRUE,
  colors = c(
    "blue",
    "white",
    "blue"
  ),
  outline.color = "transparent",
  title = " Correlation Matrix",
  legend.title = expression(rho)
)
```



There is different degrees of correlation between variable pairs, some have higher correlation then others. The next step was to create a regression model and check for Variance Inflation Factors to make sure there's no multicollinearity.

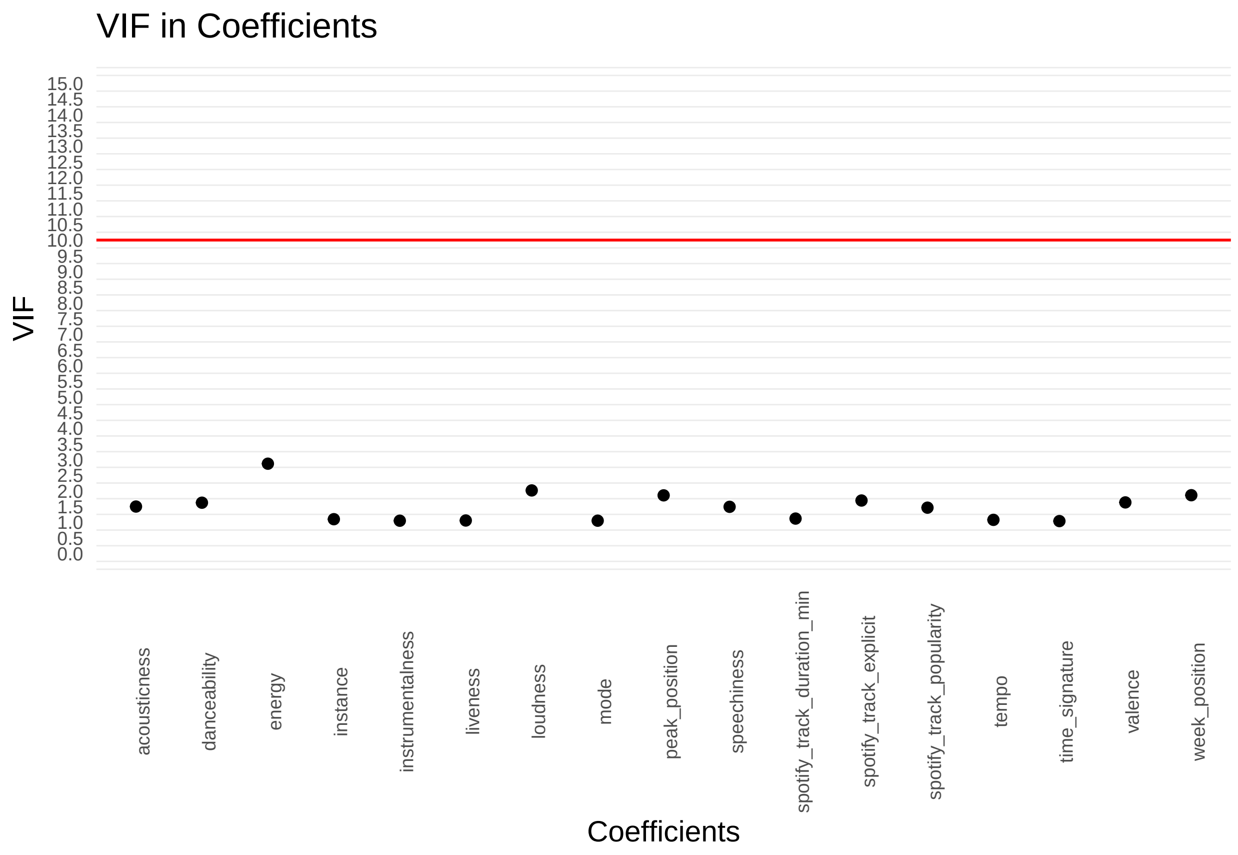
First created a regression model:

```
#creating regression model
model_output <- lm(fmla, data = r)
model_tidy <- tidy(model_output, conf.int = 0.95)
model_tidy
```

```
## # A tibble: 18 x 7
##    term                   estimate std.error statistic  p.value conf.low conf.high
##    <chr>                     <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
##  1 (Intercept)            20.7       1.50       13.8   3.46e-43 17.8      23.7
##  2 week_position          -0.0473    0.00467  -10.1    5.00e-24 -0.0565   -0.0382
##  3 instance                2.52      0.135     18.6    6.73e-76  2.25      2.78
##  4 peak_position          -0.224     0.00324  -69.1    0        -0.230    -0.217
##  5 spotify_track_expli~ -2.28      0.211     -10.8    6.33e-27 -2.69     -1.86
##  6 danceability            1.78      0.627      2.83   4.60e- 3  0.548     3.01
##  7 energy               -0.243     0.708     -0.343  7.32e- 1 -1.63      1.15
##  8 loudness              0.0241    0.0394     0.613  5.40e- 1 -0.0530    0.101
##  9 mode                  0.320     0.158      2.03   4.22e- 2  0.0113    0.629
## 10 speechiness          -5.69      0.829     -6.87   6.78e-12 -7.32     -4.07
## 11 acousticness         -3.08      0.422     -7.30   3.17e-13 -3.91     -2.25
## 12 instrumentalness     -0.0511    0.876     -0.0583 9.53e- 1 -1.77      1.67
## 13 liveness             -2.31      0.528     -4.38   1.19e- 5 -3.35     -1.28
## 14 valence               2.38      0.411      5.80   6.98e- 9  1.58      3.19
## 15 tempo                -0.00388   0.00258   -1.51   1.32e- 1 -0.00893   0.00117
## 16 time_signature        0.0726    0.277      0.261  7.94e- 1 -0.471     0.616
## 17 spotify_track_popul~  0.0507    0.00506   10.0    1.63e-23  0.0408    0.0606
## 18 spotify_track_durat~  0.595     0.0785     7.59   3.64e-14  0.441     0.749
```

Checking for VIF:

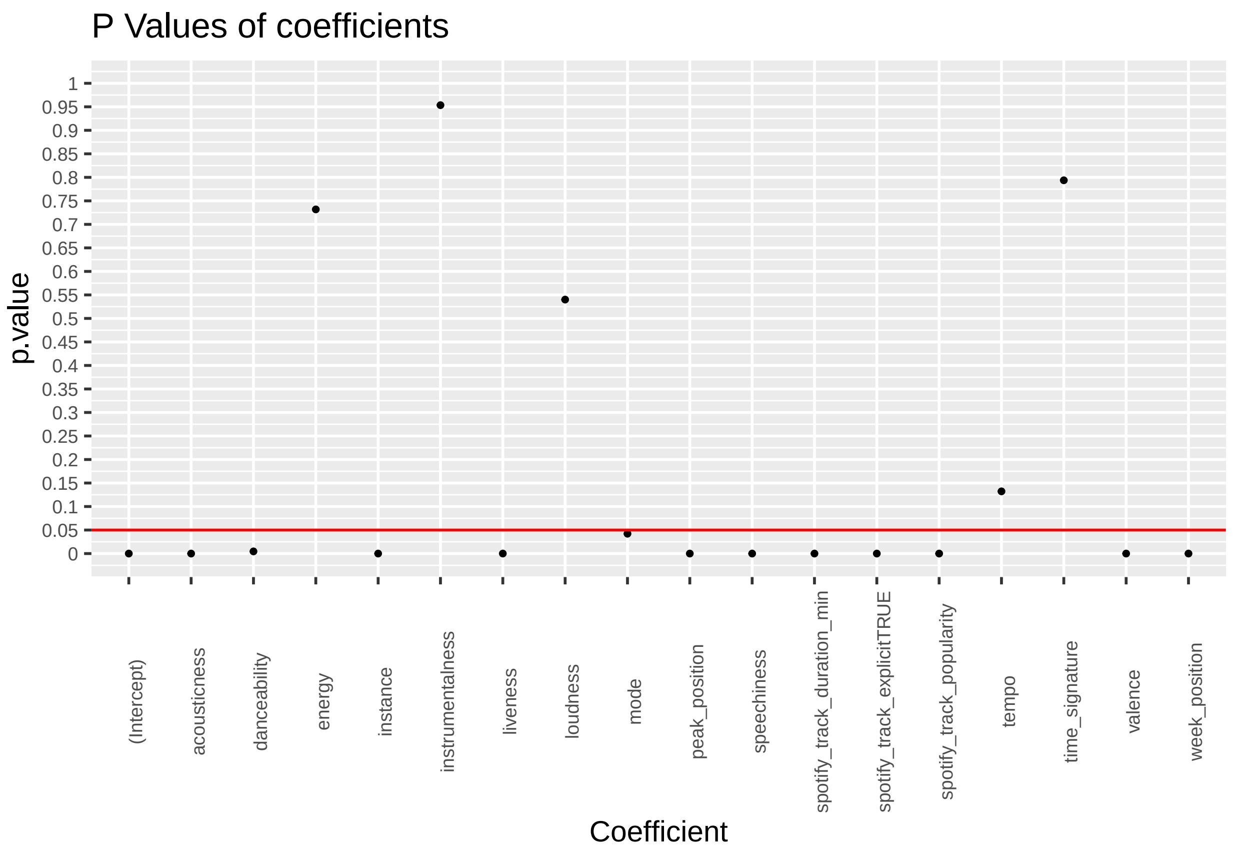
```
#Checking for variance inflation factors
vif <- vif(model_output)
vif <- as.data.frame(vif)
#adding a name column to vif dataframe in order to plot

names <- c("week_position", "instance","peak_position" ,"spotify_track_explicit", "danceability", "energ

vif$names <- names
#plotting VIF
ggplot(data = vif, aes(x = names, y = vif)) +
   geom_point() +
  geom_hline(yintercept = 10, color = "red")  +
  scale_y_continuous(
    breaks = seq(0, 15, by = 0.5),
    limits = c(0, 15)) +
   labs(title = "VIF in Coefficients", x = "Coefficients", y = "VIF") +
  theme(legend.position = "none") +
   theme_minimal() +
   theme(axis.text.x = element_text(angle = 90, size = 7)) +
   theme(axis.text.y = element_text(size = 7)) +
   theme(panel.grid.major = element_blank())
```



```
#they are all near one which indicates there is no multicollinearity.
```

No variables have worrying VIF values which indicates low mulitcollinearity among variables. A value pass 10 is indicated as a high VIF.

```
#plotting p values
ggplot(model_tidy) +
  geom_point(aes(x = term, y = p.value), size = .7) +
   scale_y_continuous(
      breaks = seq(0.0, 1.0, by = 0.05),
      labels = seq(.0, 1.0, by = 0.05),
      limits = c(0.0, 1.0)) +
  geom_hline(yintercept = 0.05, color = "red") +
   theme(axis.text.x = element_text(angle = 90, size = 7)) +
   theme(axis.text.y = element_text(size = 7)) +
   labs(title = "P Values of coefficients", x = "Coefficient")
```



Here we can see that after running our regression most variables are significant because they are less than our alpha value 0.05, there are a couple of variables that are not statistically significant this means for we retain the null hypothesis which states there is no relationship between the outcome variable and those independent variables with non significant p values. The ideal solution would be to remove the variables from the model but we fear it could underfit the model.

Now we have to check the r-squared value to see how much variance our model accounts for:

```
summary(model_output)
```

```
##
## Call:
## lm(formula = fmla, data = r)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -25.895  -3.738  -0.045   3.578  59.689
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 20.727205   1.496675  13.849  < 2e-16 ***
## week_position               -0.047329   0.004669 -10.138  < 2e-16 ***
## instance                     2.517604   0.135310  18.606  < 2e-16 ***
## peak_position               -0.223721   0.003240 -69.060  < 2e-16 ***
## spotify_track_explicitTRUE -2.278077   0.211371 -10.778  < 2e-16 ***
## danceability                 1.778062   0.627287   2.835   0.0046 **
## energy                      -0.242754   0.708370  -0.343   0.7318
## loudness                     0.024114   0.039354   0.613   0.5401
## mode                         0.320073   0.157535   2.032   0.0422 *
## speechiness                 -5.693004   0.828554  -6.871 6.78e-12 ***
## acousticness                -3.081838   0.422302  -7.298 3.17e-13 ***
## instrumentalness            -0.051103   0.875940  -0.058   0.9535
## liveness                    -2.312660   0.527870  -4.381 1.19e-05 ***
## valence                      2.380600   0.410682   5.797 6.98e-09 ***
## tempo                       -0.003880   0.002578  -1.505   0.1323
## time_signature               0.072552   0.277487   0.261   0.7937
## spotify_track_popularity     0.050726   0.005062  10.020  < 2e-16 ***
## spotify_track_duration_min   0.595063   0.078452   7.585 3.64e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.033 on 9316 degrees of freedom
## Multiple R-squared:  0.5772, Adjusted R-squared:  0.5765
## F-statistic: 748.2 on 17 and 9316 DF,  p-value: < 2.2e-16
```

R squared is 57% which means that our independent variables are only accounting for 57% of the variation in our model. The residual error is quite low which is good.

Our regression equation is:

weeks_on_chart = 0.2522541 + week_position(-0.0544834) + instance (0.2634702) + peak_position (-0.2575398) + spotify_track_explicit(-0.0264893) + danceability(0.0180494) + energy(-0.0027513) + loudness(0.0065045) + mode (0.0037218) + speechiness (-0.0573272) + acousticness(-0.0353694) + instrumentalness(-0.0005835) + liveness(0.0261894) + valence(0.0261894) + tempo(-0.0079877) + time_signature(0.0042181) + spotify_track_popularity(0.0589842) + spotify_track_duration_min(0.3516720)

Although our model has a low R Squared, we wanted to test and train the data to take a look at our model results. We did this by splitting our data into test data and train data with a 20:80 ratio. Then we ran a regression using the training set, and used our test set to make predictions. Lastly, we checked for correlation accuracy, R Squared, and Root Standard Mean Error.

Testing and Training Model:

```r
# Using function to create 80 - 20 split into test and train
sample_size = round(nrow(bill_norm)*.80) # setting what is 80%

train <- sample_n(bill_norm, sample_size)
sample_id <- as.numeric(rownames(train)) # rownames() returns character so as.numeric
test <- bill_norm[-sample_id,]
train_y <- train['weeks_on_chart']
train_x <- train |>
   select(-c('weeks_on_chart'))
test_y <- test['weeks_on_chart']
test_x <- test |>
   select(-c('weeks_on_chart'))
#building model to train
lmMod <- lm(fmla, data=train)
#predicting outcome variables
Predictions_test <- predict(lmMod, test)
summary(lmMod)
```

```
##
## Call:
## lm(formula = fmla, data = train)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.30205 -0.04316 -0.00067  0.04131  0.69418
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 0.247428   0.017955  13.780  < 2e-16 ***
## week_position              -0.058455   0.005980  -9.776  < 2e-16 ***
## instance                    0.254180   0.015821  16.066  < 2e-16 ***
## peak_position              -0.256159   0.004133 -61.976  < 2e-16 ***
## spotify_track_explicit     -0.025588   0.002735  -9.355  < 2e-16 ***
## danceability                0.019918   0.007081   2.813 0.004923 **
## energy                     -0.006995   0.008867  -0.789 0.430201
## loudness                    0.013657   0.011826   1.155 0.248204
## mode                        0.001134   0.002038   0.557 0.577865
## speechiness                -0.064064   0.009243  -6.931 4.53e-12 ***
## acousticness               -0.038777   0.005376  -7.213 6.00e-13 ***
## instrumentalness            0.002454   0.010919   0.225 0.822212
## liveness                   -0.023039   0.006654  -3.462 0.000538 ***
## valence                     0.027025   0.005008   5.397 6.99e-08 ***
## tempo                      -0.013129   0.005925  -2.216 0.026719 *
## time_signature              0.020525   0.017767   1.155 0.248030
## spotify_track_popularity    0.054774   0.006546   8.368  < 2e-16 ***
## spotify_track_duration_min  0.315588   0.049638   6.358 2.17e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08124 on 7449 degrees of freedom
## Multiple R-squared:  0.5818, Adjusted R-squared:  0.5808
## F-statistic: 609.5 on 17 and 7449 DF,  p-value: < 2.2e-16
```

```
#binding actual values and predicted values from model
actuals_pred <- data.frame(cbind(actuals=test$weeks_on_chart, predicteds=Predictions_test))  # make act
head(actuals_pred)
```

```
##        actuals predicteds
## 7468 0.1046512  0.0802130
## 7469 0.1279070  0.0589771
## 7470 0.1511628  0.2176568
## 7471 0.0000000  0.1105991
## 7472 0.0000000  0.1477083
## 7473 0.2209302  0.2469189
```

```
#looking at correlation accuracy
correlation_accuracy <- cor(actuals_pred)
correlation_accuracy
```

```
##              actuals predicteds
## actuals    1.0000000  0.7672637
## predicteds 0.7672637  1.0000000
```

```
#Printing RMSE and R2
v <- RMSE(Predictions_test, test$weeks_on_chart)
print(c('The RMSE is:', v))
```

```
## [1] "The RMSE is:"       "0.0816371242762786"
```

```r
v <- R2(Predictions_test, test$weeks_on_chart)
print(c('R Squared is:', v))
```

```
## [1] "R Squared is:"     "0.588693651730561"
```

Our tests continue to have similar R squared and low RMSE.

Additionally, we wanted to try Leave One Out Cross Validation to test our model. LOOCV takes out one observation and runs the model multiple times to see if the model functions differently.

Leave One Out Cross Validation:

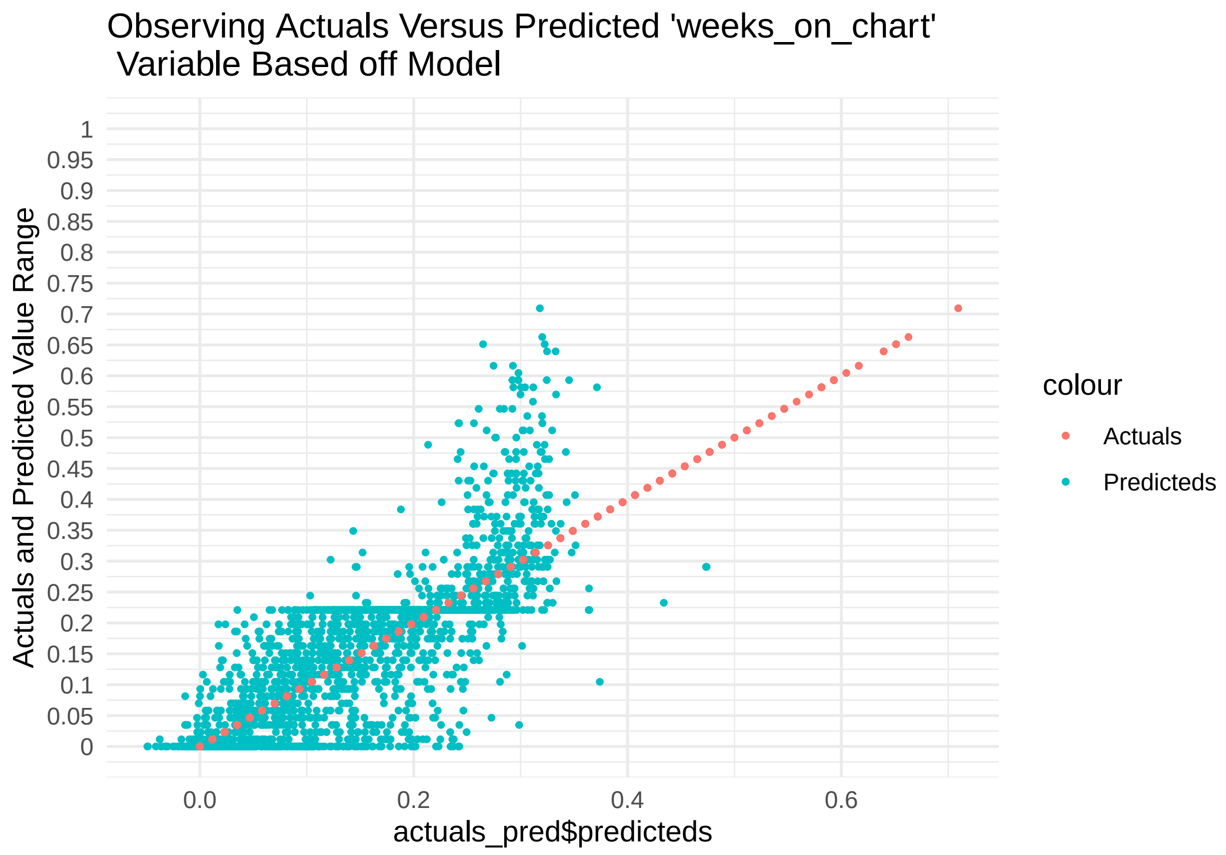
```r
#Leave one out cross vaidation:

#specify the cross-validation method
ctrl <- trainControl(method = "LOOCV")
#fit a regression model and use LOOCV to evaluate performance
model <- train(fmla, data = bill_norm, method = "lm", trControl = ctrl)
print(model)
```

```
## Linear Regression
##
## 9334 samples
##   17 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 9333, 9333, 9333, 9333, 9333, 9333, ...
## Resampling results:
##
##   RMSE        Rsquared   MAE
##   0.08190253  0.5751036  0.05873213
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

The R Squared and RMSE Values are similar to the original output.

Now Graphing Actuals and Predicteds to see variation:

```r
#plotting actuals and residuals from test
ggplot(data = actuals_pred, aes(y = actuals)) +
   geom_point(aes(x = actuals_pred$predicteds, color = 'Predicteds'), size = 0.7) +
   geom_point(aes(x = actuals_pred$actuals, color = 'Actuals'), size = 0.7) +
   scale_y_continuous(
      breaks = seq(0.0, 1.0, by = 0.05),
      labels = seq(.0, 1.0, by = 0.05),
      limits = c(0.0, 1.0)) +
   labs(title =                                  "Observing Actuals Versus Predicted 'weeks_on_chart'
 Variable Based off Model", y = "Actuals and Predicted Value Range") +
   theme_minimal()
```



Although the model has a low RMSE, because the R squared is around 60%, there is 40% of the variance of the dependent variable that is not accounted for, therefore the predictions are scattered. The next step to create a better model would be to include more variables in the data frame such as gender and turn them numerical to see if these excluded variables are important in determining the outcome. We do see however, that some of these audio features do have a significant effect in the outcome of the maximum number of weeks a song has been in the top 100.

Results/Conclusion

Q1 For the 1990s, dance-pop/pop/r&b/urban contemporary was the most popular genre. For the 2000s, atl hip hop/dance pop/pop/pop rap/r&b/urban contemporary was the most popular genre. For the 2010s, dance pop/pop/pop rap was the most popular genre. For the 2020s, canadian hip hop/canadian pop/hip hop/pop rap/rap/toronto rap.

These results are consistent with our expectations, as the genres match a lot of the top artists from each corresponding decade. While there are subtle differences in the classification of the genres, it is apparent that 'pop' is a recurring genre between the decades. This is consistent with expectations, as most mainstream music on the radio is pop music, and most of the biggest stars in the music industry are pop singers.

There is also an increasing trend when observing the Weeks Spent in #1 Ranking for each successive decade, followed by a drop off for the 2020s. First, we theorize that the increase is due to the widespread availability of music with the technological advancements of each decade, especially with streaming services in the 2010s. Secondly, we theorize that the observed sharp drop off for the 2020s is not actually a drop off, but rather an artifact due to us still being relatively early into the 2020s. If the current trend holds, the 2020s should see a further increase from the numbers shown for the 2010s. More data as the decade progresses will reveal.

Q2 For each decade, the spread of duration is approximately the same, and it can be observed that there is an apparent decrease in the median track duration with each decade. From the data, songs today in the 2020s are shorter than those from previous decades. This is consistent with our observations, as songs today are typically around the median value shown of about 3.5 minutes long.

While we had expected there to be a difference in the length of songs between decades, we did not expect there to be such an apparent trend from each passing decade. One plausible explanation for this is due to the change in how musicians make money. While in the past, musicians sold CDs and received payments for CD purchases, many musicians today make money with each stream of their songs on streaming services such as Spotify. This could explain motivation for musicians to have more tracks that are of shorter duration, rather than to release longer tracks. Further analysis of the economics of the music industry may reveal insights. In addition, comparing artists with many shorts songs and artists with a few longer songs on Spotify and analyzing the revenue generated.

Q3 The songs are Bad Guy by Billie Eilish, Blinding Lights by The Weeknd, and Without Me by Halsey. Their respective trajectories on the billboard can also be observed. A gradual decline can be observed for each song, but it is interesting to note that there is also a common trend of dropping off a little and going back up the rankings. For each of the top 3 songs however, after they drop off, they do not return to the top rankings again.

A worthwhile note is that every year dating back to around 2012 Mariah Carey has made it to the billboard top 100 with "All I want for Christmas is you." It has made the top 10 every year from 2018 to present day.

Q4 There is an indication that audio features may have a relationship with how many weeks an individual song has been on the top 100 chart. Features such as energy, instrumentalness, time signature, tempo, and loudness do not have significant effects. Also, due to our regression's low r squared value signifies our model is not accounting for much of the change in the dependent variable, this can be due to variables that are not included and outside factors. Additionally, one thing to note is the later a song came out, the less time it had to be on the chart, this would be something to look at if research is furthered.

Extended Analysis We attempted to find a correlation between the subsequent data that was included from the audio feature but after running a VIF we found that there was no discernible multicollinearity, From this point we checked to see if there the multiple R-squared value would support this and we did find that though there were some statistically significant variables, they only accounted for just over half of the variation which indicates no correlation. As stated above by our predicted vs actual values, we could try to make predictions but they would be a guess at best. We believe that there are other factors that could and should be looked into for future research. The next steps in this research process would be to include genre as a variable and create dummy columns for every genre there is (because we believe song genre affects its popularity) then looking at the regression results and removing the insignificant variables (if any) from the model. After this, testing for different models to see which one fits this data the best to make accurate predictions about song audio features and its popularity. Lastly, there were multiple variables that determined popularity in a different way, for instance there was week position, which gave a song a ranking of 1 to 100 for a said week. Using one of these other variables as the dependent variable could change the results, so further research on this could yield significant results.