# Final Report

*Luke Novak*

*4/17/2019*

## Spotify and Discogs Music Data Analysis

By Luke Novak, Jessica Cheng, and Matt Lowe For Northeastern University DS4100

### Starting notes:

For reporting purposes, and to keep this report consise, it will focus only on the analysis of the data that we fetched from Spotify and Discogs. If you would like to see the code we used to populate our database with data, as well as how we scraped the Billboard 200 and used the Spotify and Discogs APIs, you are welcome to see the other files included in the report zipfile, or check the repository out on https://github.com/lukenovak/Spotify-Discogs-Data-Analysis. Note that we are using environment variables to obscure database keys.

### A note on Shiny:

This report has an Interactive frontend that has been deployed at fernando.nluken.com/music. If the server is down for some reason, email the webmaster at contact@nluken.com and he will restart the server at that time so that the frontend can be viewed.

### Proposal

#### Background:

Popular music has evolved rapidly since 1960 in a variety of ways, both in style and in consumption. Our project seeks to quantify these changes, and analyze how style (which can be broken down into categories like genre) and release medium (Vinyl into CD into Digital) have changed over time.

The main goals of our project include:

Drawing a timeline of the most popular genres of music from 1960 to the present day.

In addition, we will be analyzing other stylistic properties, like the most popular key within all genres.

Demonstrating the evolution of audio technology (and modern-day vinyl revival) by comparing the number of releases between formats over time.

Seeing if the format of music has impacted the music itself (release length, genre, etc.)

Showing how the digitization of music has impacted the number of releases being put out

#### Methods:

Our data gathering will primarily be done by using the Spotify and Discogs APIs. Spotify has one of the most robust music-related APIs available on the internet and will allow us to gather data about the music itself. The Discogs API is better for tracking release formats, as Discogs tracks each individual physical release of a given piece of music. Thus, the Discogs API will allow us to see how many records are being

released on vinyl vs. CD vs. digitally. We will likely use Python to pull data from these APIs, and we intend to store our data in a PostgreSQL database. That data will then be cleaned up using R to make it consistent. Finally, we will use R to carry out our data analysis.

**Deliverables:**

Our final deliverables will include the R code used for the analysis along with the PostgreSQL database used to store the data. Our project will also include charts and graphs to model our data results and analysis. To bring our project together, we will have a Shiny front-end allowing users to interact with the project and choose what data they want to view.

## Approach

The general approach can be described as following: Get all of the Billboard 200, scrape Spotify for info on all of the albums, then scrape Spotify again for information on all of the songs of all of those albums, and finally, scrape Discogs for data about the physical releases related to these albums. After that we can do data analysis and see if there are any trends in American pop music over the last 60 years.

## General setup:

Libraries, and setting the stage for the database

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  2.0.0     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0
```

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

```
library(dplyr)
library(ggplot2)
library(RPostgreSQL)
```

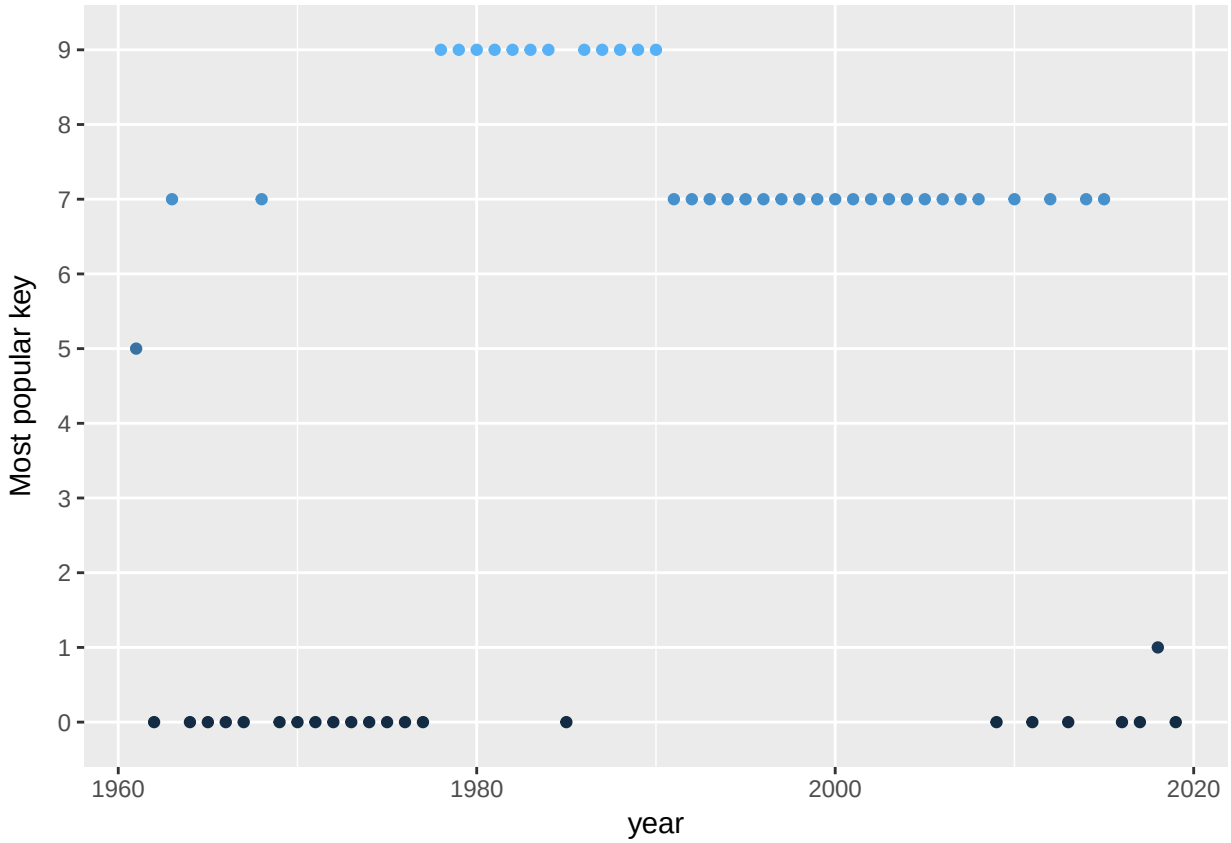
```
## Loading required package: DBI
```

```
library(stringr)
library(spotifyr)
require(DBI)

drv <- dbDriver("PostgreSQL")
db <- dbConnect(drv, user=Sys.getenv("DB_USER"), host=Sys.getenv("DB_HOST"),
                password=Sys.getenv("DB_PW"), dbname='music_analysis')
```

## Key Analysis

To get the key of songs, we carry out an advanced SQL Query and get the count of keys per year, and then plot the most popular key per year

```
## Creates a plot for seeing the most popular key in music since 1960
# Selects year, key, and the count of each key for all songs
keyQuery <- as_tibble(dbGetQuery(db, "SELECT year, key, COUNT(key) AS keyCount FROM songs
                INNER JOIN albums ON songs.album_id = albums.id
                WHERE year > 1960
                GROUP BY year, key
                ORDER BY year DESC, COUNT(key) DESC;
                "))
# Arranges count of keys in decreasing order
keyQuery <- keyQuery %>%
  arrange(desc(keycount))
# Takes unique years to get the max count for every year
keyQueryUnique <- !duplicated(keyQuery["year"])
# Puts the all the unique rows back in a dataframe for plotting
keyQueryDF <- keyQuery[keyQueryUnique,]
# Plots the most popular key in each year
keyPlot <- ggplot(data = keyQueryDF) +
  geom_point(mapping = aes(x=year, y=key, color = key), show.legend = FALSE) +
  scale_y_discrete(name ="Most popular key", limits=c(0:11))
keyPlot
```




The above plot essentially finds the mode within the keys for each year. Since musical keys is ordinal data, the average cannot be taken, hence the mode or most popular key amongst a single year was used as measurement. In addition, this hinders prediction ability, as the numbers cannot be mathematically calculated.

```
## Function allows user to specify start and end year
keyFunc <- function(startYear, endYear) {
```

```
  yearConditionStart <- paste(startYear, " and year < ", sep = '')
  yearConditionEnd <- paste(yearConditionStart, endYear, sep = '')

  queryStart <- paste("SELECT year, key, COUNT(key) AS keyCount FROM songs INNER JOIN albums ON songs.al
                      WHERE year >", yearConditionEnd, sep = '')

  funcTibble <- as_tibble(dbGetQuery(db, paste(queryStart, "GROUP BY year, key ORDER BY year DESC, COUNT

  # Arranges count of keys in decreasing order
  funcTibble <- funcTibble %>%
    arrange(desc(keycount))
  # Takes unique years to get the max count for every year
  tibbleUnique <- !duplicated(funcTibble["year"])
  # Puts the all the unique rows back in a dataframe for plotting
  tibbleFinal <- funcTibble[tibbleUnique,]
  # Plots the most popular key in each year

  ggplot(data = tibbleFinal) +
    geom_point(mapping = aes(x=year, y=key, color = key), show.legend = FALSE) +
    scale_y_discrete(name ="Most popular key", limits=c(0:11))
}
```

Plot produced by function is the same as above general plot, except the start and end years are what the user specifies in function input. This function is one that is going to be used to help run the shiny front-end

### Tempo Analysis

## Song Data

We joined tables songs and albums on album ID in our query to obtain information across both tables such as tempo and duration from songs table and year from albums table since they are the main features we were interested in studying. We queried all song data between 1960 and 2019 as this was the time period we were studying and ordered the data by year.

```
# all tempo data between 1960 and 2019
df_tempo_all <- dbGetQuery(db, "SELECT tempo, duration, year
                           FROM albums INNER JOIN songs ON albums.id = songs.album_id
                           WHERE year BETWEEN 1960 AND 2019
                           ORDER BY year")
```

## Identify outliers in song data and remove them

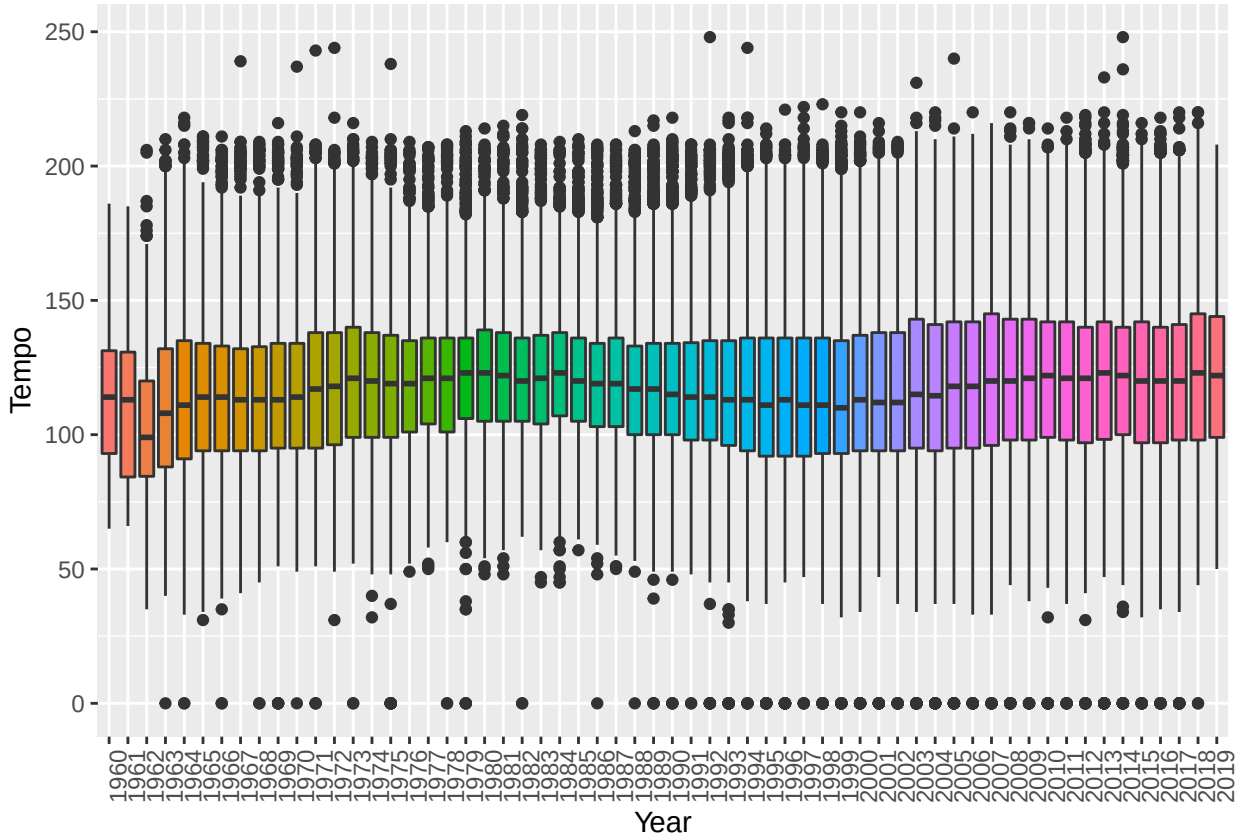
We want to study how tempo has changed over the years. First, we decided to create a boxplot to visualize any tempo outliers from each year.

### Graph tempo by year and visualize outliers in song data

```
# visualize outliers
tempo_all_plot <- ggplot(df_tempo_all,
```

```
                        mapping = aes(x = as.factor(year),
                                      y = tempo,
                                      fill= as.factor(year))) +
  geom_boxplot() +
  labs(x = "Year", y = "Tempo") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1), legend.position="none")
tempo_all_plot
```

```
## Warning: Removed 1 rows containing non-finite values (stat_boxplot).
```




Based on the boxplot produced, we see that there are some tempo outliers. Instead of fully removing these outlier as that would produce uneven column lengths, we decide to reassign each outlier value with NA. A new dataframe was produced which contained NA values of the tempo outliers. Then, the rows containing the NAs or outliers were filtered out so that the dataset would not contain outliers.

```
# function to remove outliers (outside of 1.5*IQR range) and replace them with NAs
remove_outliers <- function(x, na.rm = TRUE) {
  qnt <- quantile(x, probs=c(.25, .75), na.rm = na.rm)
  H <- 1.5 * IQR(x, na.rm = na.rm)
  y <- x
  y[x < (qnt[1] - H)] <- NA
  y[x > (qnt[2] + H)] <- NA
  y
}

#info on df_tempo_all
str(df_tempo_all)
```

```
## 'data.frame':    246778 obs. of  3 variables:
##  $ tempo   : int  186 70 96 81 75 102 174 174 174 142 ...
##  $ duration: num  157560 185106 181226 198866 147506 ...
##  $ year    : int  1960 1960 1960 1960 1960 1960 1960 1960 1960 1960 ...
```

```
# distinct years
years <- df_tempo_all %>% distinct(year)
year_list <- years$year

# function to produce table for given year
table_year <- function(year_num) {
  df_tempo_all %>% filter(year == year_num)
}

# list of tables for each distinct year
list_tables <- lapply(year_list, table_year)

#function that takes in a df and produces a vector of tempo from df
year_tempo <- function(ydf) {
  ydf$tempo
}

# list of list of tempos for each year's tempo table
list_year_tempo <- lapply(list_tables, year_tempo)

# apply replace outliers with NAs to each list of list of tempos from each year
list_no_outliers <- lapply(list_year_tempo, remove_outliers)

# combines lists of tempos into one list
tempo_no_outliers <- unlist(list_no_outliers)
# check length of tempos list
length(tempo_no_outliers) # 246778
```

```
## [1] 246778
```

```
# extracts years from original tempo df
tempo_years <- df_tempo_all$year
# make sure length of years same as length of tempos list
length(tempo_years) #246778
```

```
## [1] 246778
```

```
# create df of tempo data
df <- cbind(tempo_no_outliers, tempo_years)
tempo_table <- as.data.frame(df)
names(tempo_table) <- c("tempo", "year")
head(tempo_table)
```

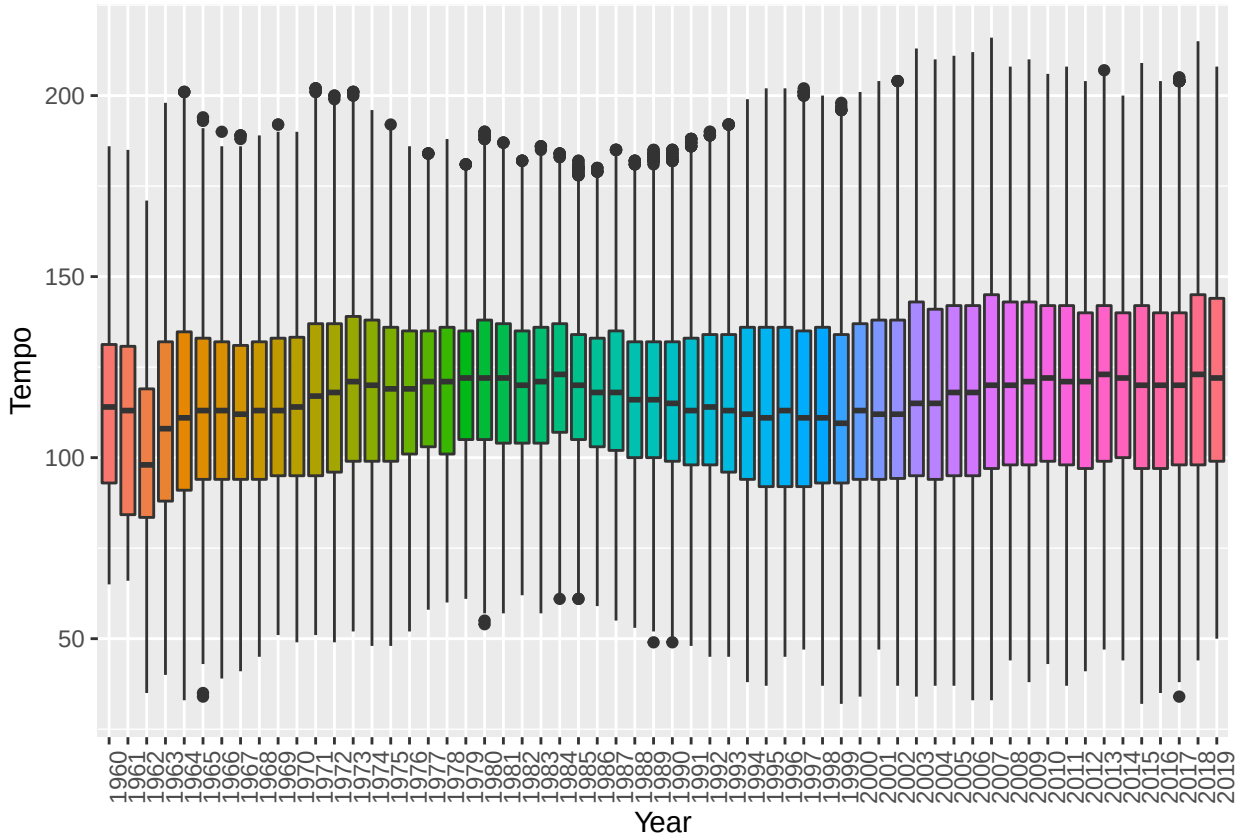
```
##   tempo year
## 1   186 1960
## 2    70 1960
## 3    96 1960
## 4    81 1960
## 5    75 1960
## 6   102 1960
```

```
#count how many outliers as NAs
na_tempo_count <- tempo_table %>%
  summarise(na_count = sum(is.na(tempo)))
# tempo df with outlier/NAs removed
tempo_df <- tempo_table %>% filter(!is.na(tempo))
#check na rows have been removed
na_tempo_new_count <- tempo_df %>%
  summarise(na_count = sum(is.na(tempo)))
```

## Visualize boxplot of new song dataframe with outliers removed

The new dataframe still contains a few outliers but as the size of the data is large enough, it was determined that they should not have a large effect on the mean tempo for each year.

```
# new boxplot with NAs outliers removed
tempo_df_plot <- ggplot(tempo_df,
                        mapping = aes(x = as.factor(year),
                                      y = tempo,
                                      fill= as.factor(year))) +
  geom_boxplot() +
  labs(x = "Year", y = "Tempo") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1), legend.position="none")
tempo_df_plot
```



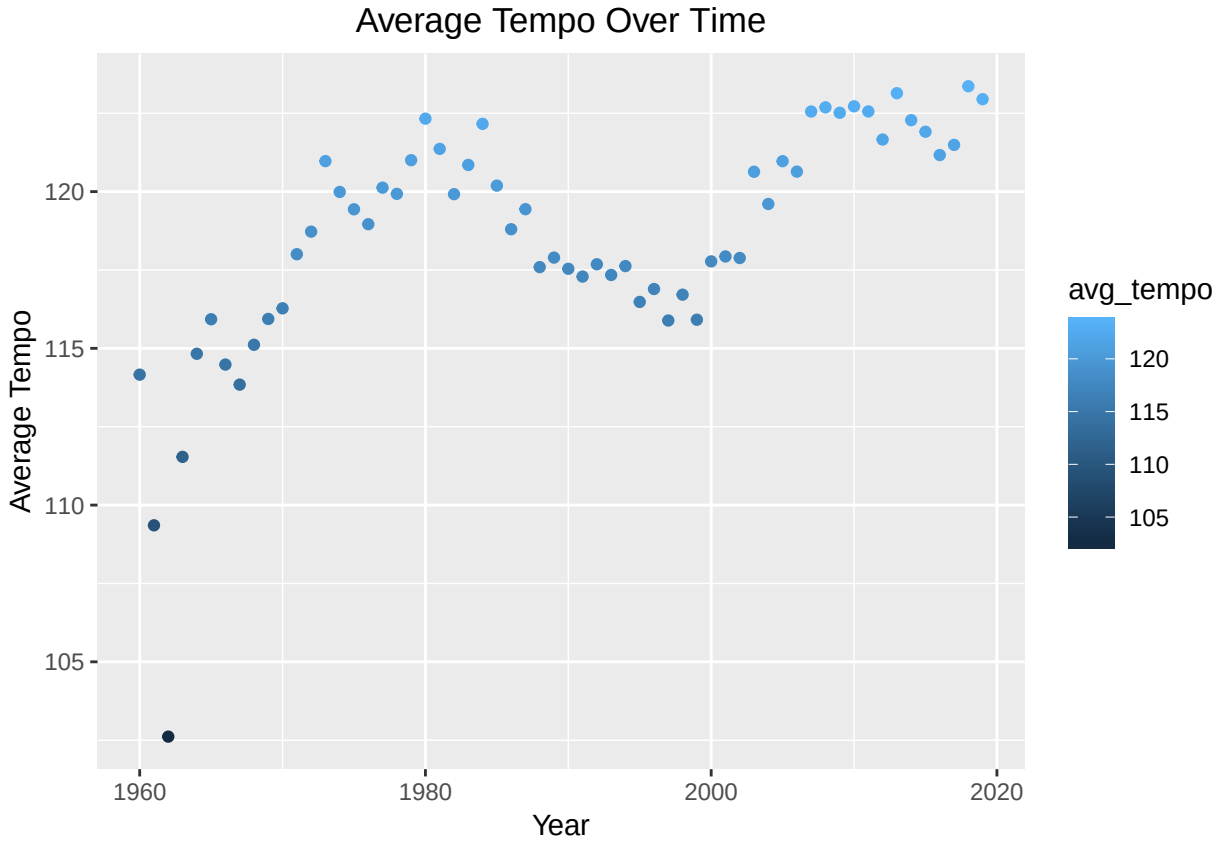

# Analysis of tempo

## Analyze relationship between tempo over year

### Graph of average tempos over year

As our question involves studying how tempo has changed over time, we decided to determine the average tempo for each year. Based on observation alone, the average tempo of songs seems to increase over time from 1960 until 1980. Between 1980 and 1999 the average tempo of the songs decreased. Then, the average tempo seems to generally increase again by 2000. The lowest tempos were around 1960s and around 1980 and recent years, the average tempos were quite high.

```
#Average tempos
tempo_avg <- tempo_df %>% group_by(year) %>%
  summarise(avg_tempo = mean(tempo))
# plot avg tempo over time with outliers/ NAs removed
avg_tempo_plot <- ggplot(tempo_avg,
                         mapping = aes(x = year,
                                       y = avg_tempo,
                                       color = avg_tempo)) +
  geom_point() +
  labs(x = "Year", y = "Average Tempo") +
  ggtitle("Average Tempo Over Time") +
  theme(plot.title = element_text(hjust = 0.5))
avg_tempo_plot + scale_color_gradient()
```




### Analyze correlation and model of tempo and year Correlation is a statistical measure that suggests the level of linear dependence between two variables. The correlation between the average tempo and year was 0.65. There is a moderately positive correlation between average tempo and year. In general, as year increased since 1960, so did tempo. However, between 1980 and 1990, there was a decrease in tempo which

lowered the positive correlation between average tempo and year. One of the other factors we want to study is genre. Certain genres have a specific range of tempos. The popularity of certain genres per year may explain why the average tempo changed per year.

Now that we have seen the linear relationship pictorially in the scatter plot and by computing the correlation, we decide to build the linear model. We have established the relationship between the predictor and response in the form of a mathematical formula for tempo as a function for duration. To ensure that the regression model is statistically significant, we observe the p-value from summary statistics. We can consider a linear model to be statistically significant only when both these p-Values are less that the pre-determined statistical significance level, which is ideally 0.05. This is visually interpreted by the significance stars at the end of the row. The more the stars beside the variable's p-Value, the more significant the variable. The p-value from this model was 2.03e-08 so year is significant.

When there is a p-value, there is a hull and alternative hypothesis associated with it. In Linear Regression, the Null Hypothesis is that the coefficients associated with the variables is equal to zero. The alternate hypothesis is that the coefficients are not equal to zero. There exists a relationship between the independent variable in question (year in this case) and the dependent variable (tempo). As the p-value is below 0.05, this means that the null hypothesis is rejected and there appears to be a relationship between year and average tempo.

What R-Squared tells us is the proportion of variation in the dependent (response) variable that has been explained by this model. The R-squared is 0.42. About 40% of the variation in the average tempo has been explained by this model.

```
# correlation
cor(tempo_avg$avg_tempo, tempo_avg$year) # around 0.65
```

```
## [1] 0.6490909
```

```
#lm
tempo_mod <- lm(avg_tempo ~ year, data = tempo_avg)
summary(tempo_mod)
```

```
##
## Call:
## lm(formula = avg_tempo ~ year, data = tempo_avg)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.2481  -1.3496   0.1389   1.6274   5.0027
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -153.69988   41.90881  -3.667 0.000534 ***
## year           0.13688    0.02106   6.498 2.03e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.826 on 58 degrees of freedom
## Multiple R-squared:  0.4213, Adjusted R-squared:  0.4113
## F-statistic: 42.23 on 1 and 58 DF,  p-value: 2.029e-08
```

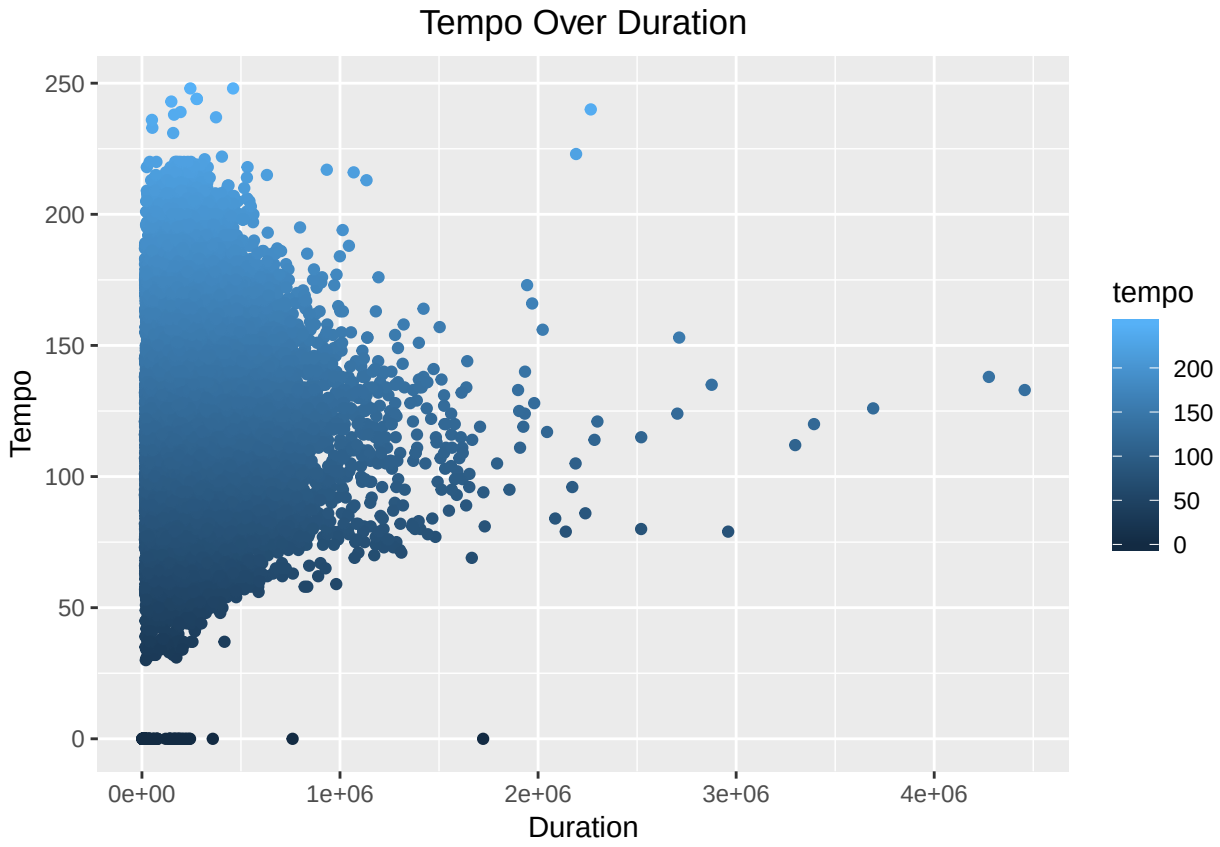
## Analyze relationship between tempo and duration

### Scatterplot of duration vs tempo

It was hypothesized that the duration of songs may also have an effect on tempo. For instance, a longer song may have slower tempo. We decided to create a scatterplot to observe if there is a linear relationship. There did not seem to be a linear relationship. What was interesting was that the range of tempos did seem to decrease over duration based on observation alone. However, further analysis was needed.

```
# duration vs tempo
dur_tempo_plot <- ggplot(df_tempo_all,
                         mapping = aes(x = duration,
                                       y = tempo,
                                       color = tempo)) +
  geom_point() +
  labs(x = "Duration", y = "Tempo") +
  ggtitle("Tempo Over Duration") +
  theme(plot.title = element_text(hjust = 0.5))
dur_tempo_plot + scale_color_gradient()
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```




### Analyze correlation and model for duration and tempo The correlation between duration and tempo was 0.003. A value closer to 0 suggests a weak relationship between the variables. A low correlation (-0.2 < x < 0.2) probably suggests that much of variation of the response variable, tempo, is unexplained by the predictor, duration, in which case, we should probably look for better explanatory variables.

Just in case though, we decide to model duration vs tempo and create a linear regression model. The null hypothesis is that the coefficients associated with the variables is equal to zero. The alternate hypothesis is

that the coefficients are not equal to zero. There exists a relationship between the independent variable in question (duration in this case) and the dependent variable (tempo). As the p-value is above 0.05, this means that the null hypothesis is accepted and there does not appear to be a relationship between duration and average tempo. The linear model is not statistically significant as the p-value of 0.173 is higher than the significance level 0.05.

Finally, the R-squared value is extremely low at 7.6e-06 meaning almost none of the variation in the average tempo has been explained by this model.

```
# bind all columns together from original data with tempo outliers removed into a df
all <- cbind(tempo_no_outliers, tempo_years, df_tempo_all$duration)
df_all <- as.data.frame(all)
# rename columns
names(df_all) <- c("tempo", "year", "duration")
head(df_all)
```

```
##   tempo year duration
## 1   186 1960   157560
## 2    70 1960   185106
## 3    96 1960   181226
## 4    81 1960   198866
## 5    75 1960   147506
## 6   102 1960   188826
```

```
#count how many outliers as NAs
na_df_all_count <- df_all %>%
  summarise(na_count = sum(is.na(tempo)))

# all variables df with outlier/NAs removed
df_all_df <- df_all %>% filter(!is.na(tempo))

#correlation between tempo and duration
cor(df_all_df$tempo, df_all_df$duration) # no correlation
```

```
## [1] 0.002750481
```

```
# model tempo and duration
tempo_dur_model <- lm(tempo ~ duration, data = df_all_df)
summary(tempo_dur_model)
```

```
##
## Call:
## lm(formula = tempo ~ duration, data = df_all_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -87.981 -23.177  -1.408  19.014  95.888
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.199e+02  1.619e-01 740.644   <2e-16 ***
## duration    8.573e-07  6.298e-07   1.361    0.173
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 29.32 on 244908 degrees of freedom
## Multiple R-squared:  7.565e-06,  Adjusted R-squared:  3.482e-06
```

```
## F-statistic: 1.853 on 1 and 244908 DF,  p-value: 0.1735
```

## Discogs Data

To analyze the genre and format data we gathered through discogs (and the python script scraper, we first need to grab our data from the database and clean it up a little (although the data has already been cleaned)

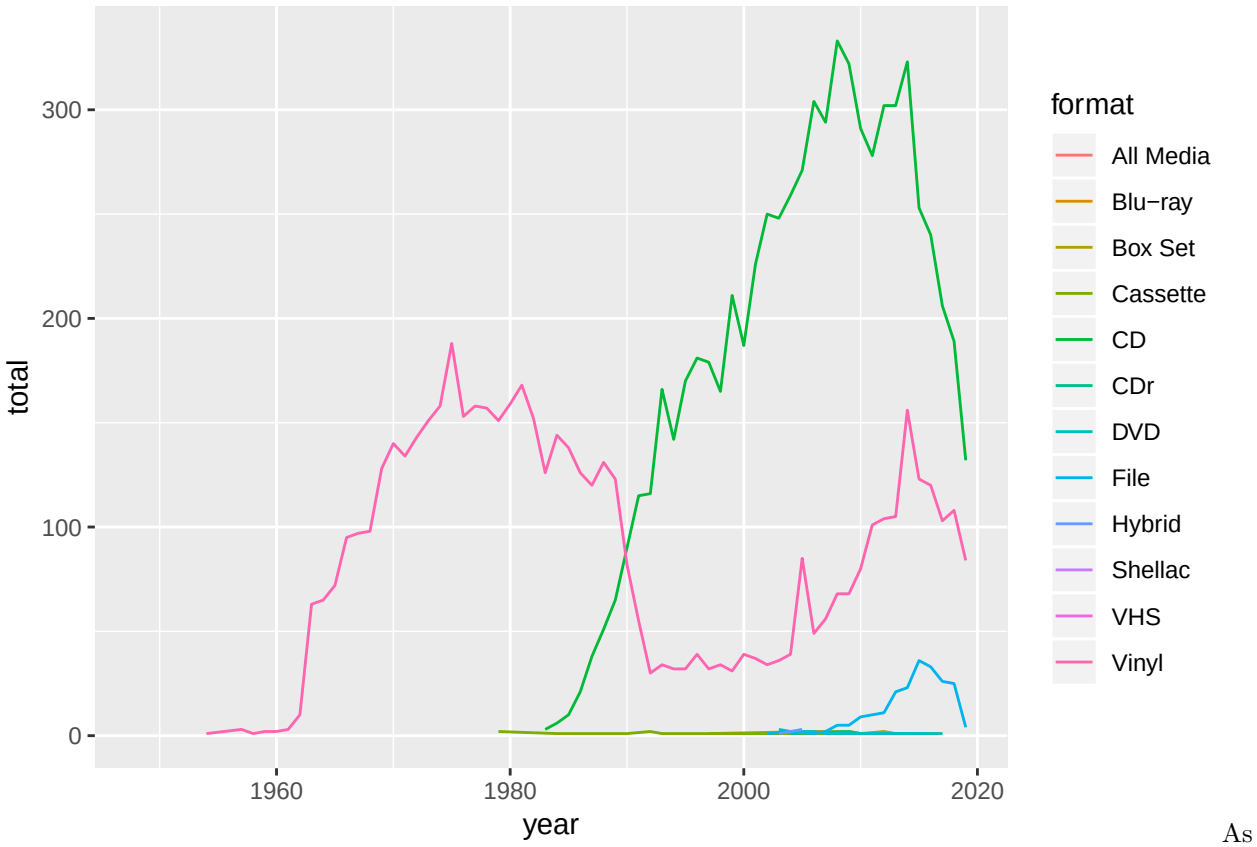
To do this, we join the formats graph with the albums graph so we can see what albums were released in what format

```
formats <- dbGetQuery(db, "SELECT * FROM releases;")
all_albums <- dbGetQuery(db, "SELECT * FROM albums;")
formats_joined <- inner_join(formats, all_albums, by = 'id')
```

## Format analysis

To analyze the format of graphs over time (this was one of the main goals of the project) we take the formats of albums and the year in which they were released and group them together, summarizing them to get the count of each format per year. We then proceed to remove outliers, and do a simple imputation of data for the rest of 2019 to normalize it. Finally, we plot the data.

```
count_format_per_year <- formats_joined %>% select(format, year) %>%
  group_by(format, year) %>% summarize(total = n())

## We need to remove some outliers, like CDs prior to their inventin in 1982
count_format_per_year <- count_format_per_year %>%
  filter(format != 'CD' | year > 1981)
## multiply the current year by 3 to extrapolate out
count_format_per_year <- count_format_per_year %>%
  mutate(total = ifelse(year == 2019, total * 4, total))

ggplot(count_format_per_year, aes(x = year, y=total, color=format)) + geom_line()
```



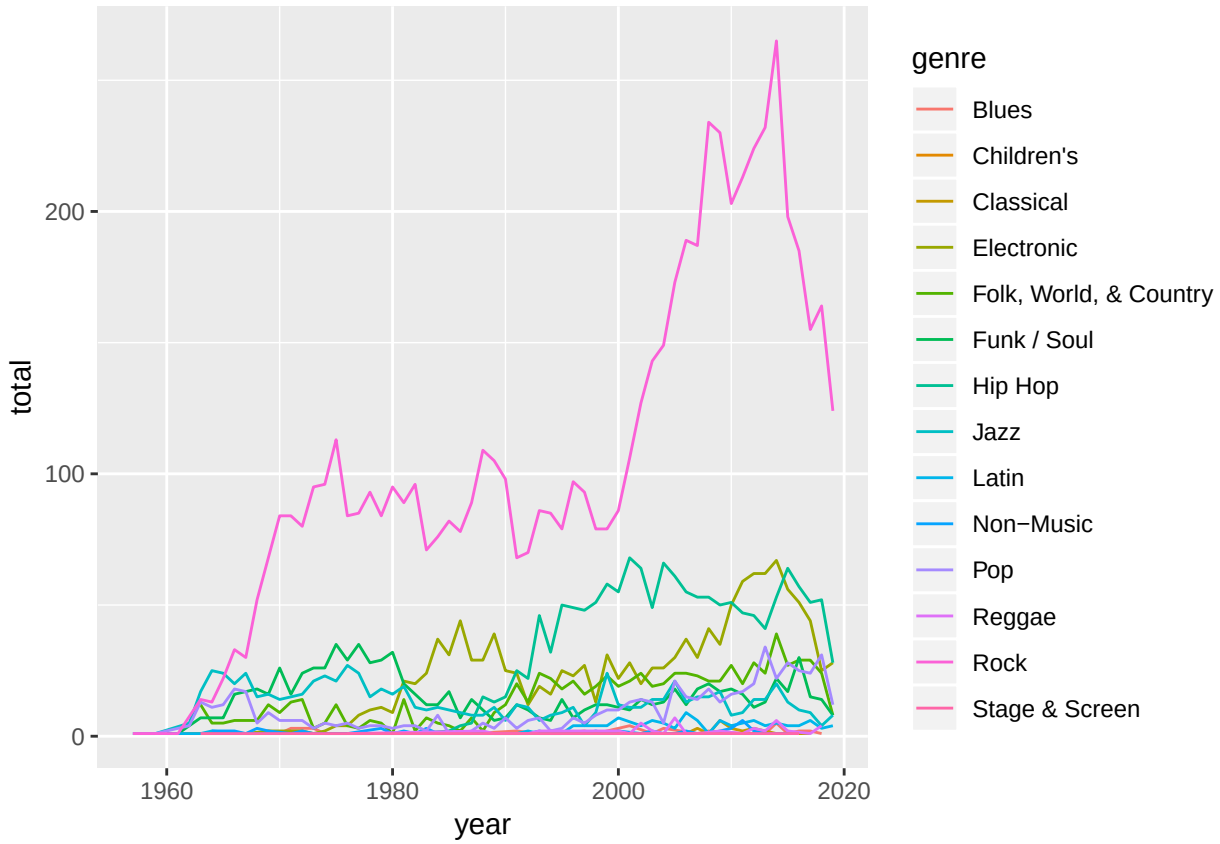

As we can see, the format of albums has changed greatly over time. While Vinyl is the only true contender for the beginning of the graph, we see the rapid rise in CDs as time goes on, and the eventual downfall of CDs as well as the vinyl revival in the mid 2000s. Finally we see the rise of digitial in the 2010s as well.

It is worth noting that Discogs is a biased site, and is thus heavily skewed towards physical releases (vinyl especially), and so we must consider that when tagging the primary release of an album, users will try to refrain from using the file tag unless no physical release is available. Thus, the file data is much lower than expected

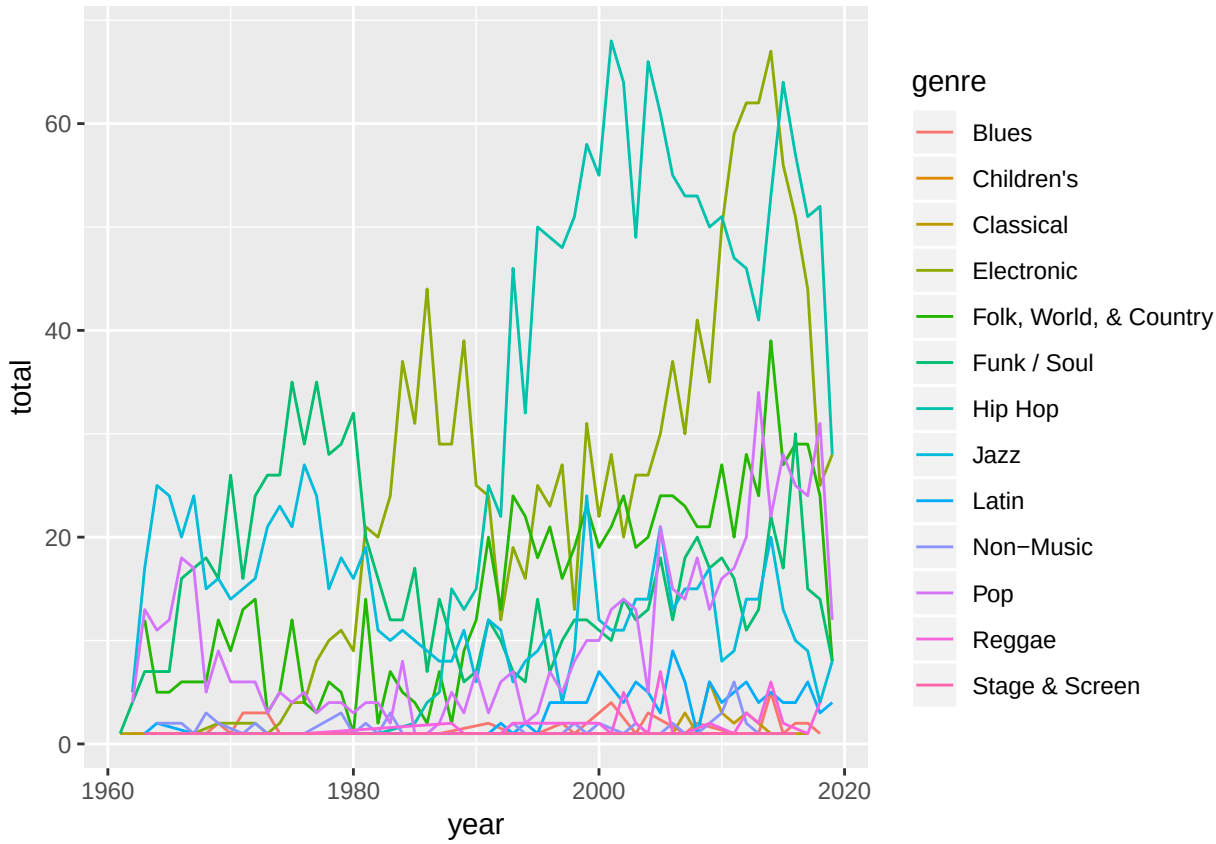
## Genre analyisis

To analyze genres, we tidy the data in a similar manner to the format tidying, dropping all other columns, summarizing after a grouping with year, and imputing for the rest of 2019. Since rock was the dominant genre, I also take a look at non-rock genres, and then remove the "small genres" at the bottom of the graph to make it easier to read. This can be treated like outlier removal

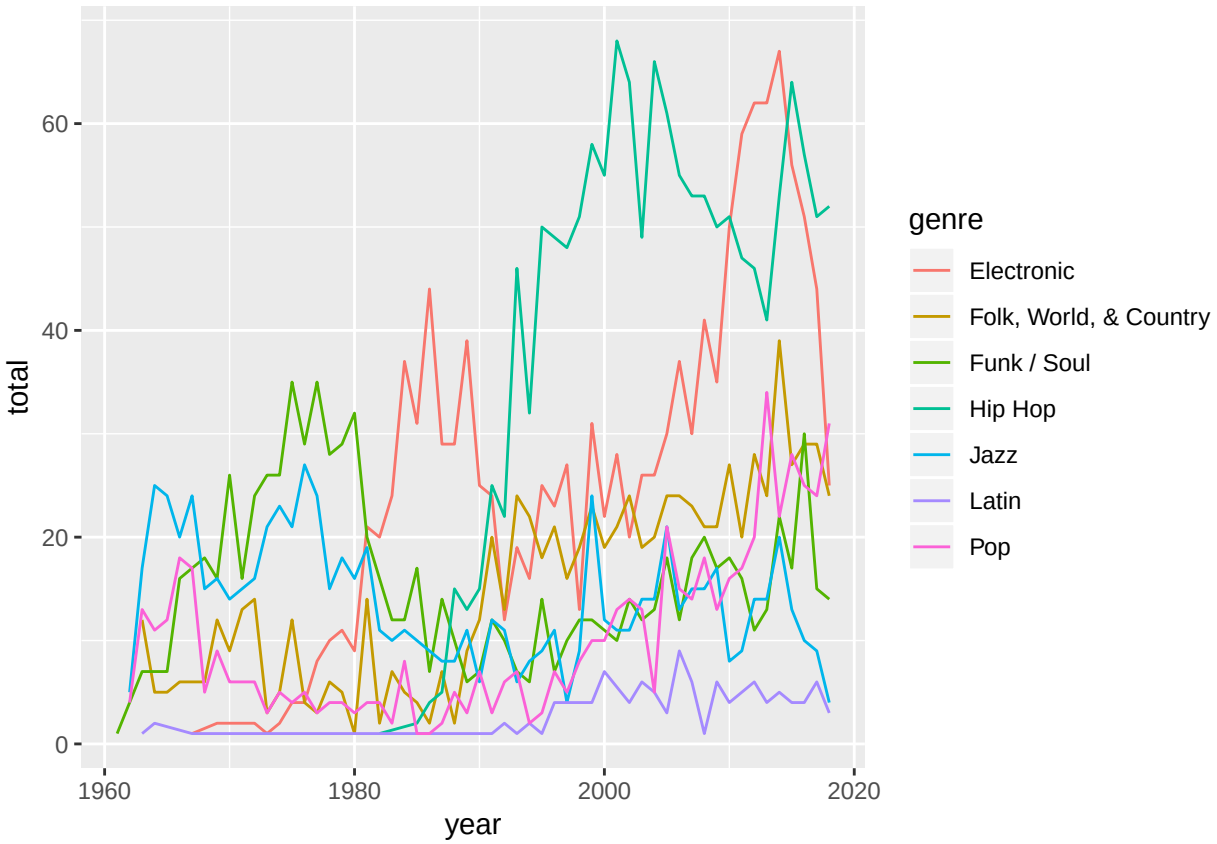
```
genres <- all_albums %>% filter(genre != 'None')
count_genre_per_year <- genres %>% select(genre, year) %>%
  group_by(genre, year) %>% summarize(total = n())

count_genre_per_year <- count_genre_per_year %>%
  mutate(total = ifelse(year == 2019, total * 4, total)) %>% filter(year > 1955)


ggplot(count_genre_per_year, aes(x = year, y=total, color=genre)) + geom_line()
```




```
count_genre_no_rock <- count_genre_per_year %>% filter(genre != 'Rock' & year > 1960)

ggplot(count_genre_no_rock, aes(x = year, y=total, color=genre)) + geom_line()
```




```
count_genre_just_big_ones <- count_genre_no_rock %>%
  filter (genre != 'Non-Music' & genre != 'Children\'s' & genre != 'Classical' &
          genre != 'Stage & Screen' & genre != 'Reggae' & genre != 'Blues' & year != 2019)

ggplot(count_genre_just_big_ones, aes(x = year, y=total, color=genre)) + geom_line()
```




The genre graph was by far one of the more interesting that we had, as it shows a vast number of cultural trends. There are too many to name here, but a few examples to observe is the rise in hip-hop since the 80s (and ensuing fall of rock in the 2010s that industry experts have been parroting for years), the two Electronic booms, one in the 80s (sythwave), and a massive boom/bust in the 2010s (the EDM Boom). Also observe the rise and fall and rise again of Funk/Soul in the 70s, then returning in the 2010s.

## Challenges, limitations, and biases

This project had a number of challenges, limitations and biases that must be considered that affected the data that we were working with.

### Biases

First, it is important to remember that we only looked at american pop music of the last 60 years, and not international music or music from other time periods.

Another bias that was previously mentioned has to do with Discogs and how their tagging system works. Since discogs is a website dedicated to the tracking of physical releases, users are heavily biased to tag an albums "primary release" with a physical release even when a file or digital release might be more important to the work as a whole. This would help explain the low amount of file data that we got

### Difficulties

Gathering the data in the method that we did proved to be extremely difficult as the three APIs that we used were almost completely disparate save for the fact that they tracked music. The first "API" was really

just the Billboard website and data from their webisite was very unclean, often with extra spaces, symbols, or incorrect capitalization. This was fixed in the database stage, with an SQL `TRIM` command.

Additionally, there was a lot of difficuly in actually getting the data due to Rate Limiting. Discogs epecially only allowed around 30 requests per minute, so it was necessary to optomize the script to run on muliple IPs as the API was IP limited

## Shiny Frontend

Please feel free to interact with our frontend at fernando.nluken.com/music

## Final Takeaways

The ways that Americans consume music, and what kind of music they listen to are constantly changing, and our data confirms that. From the rise of CDs to the similarites in tempo across the years, trends in American music will continue to drive its development