Analysis of the Top 5000 ALbums of All Time

Marcus Garica and Luke Petet

2022-12-12

Overview

The goal of this project was to use a data set of the Top 5000 Albums of All Time, break it down and analyze it by genre, rating, and age of the album. Our initial goal was to break down this data set three different ways. The first was to take the descriptors of the album and use them to determine which descriptors are most often paired with what genres. Our second goal was to use the rating of the album, the age, and the genres to define the highest rated genres over the years and to view how with time the genres shifted. Finally, an attempt was made to use a linear model to predict what rating an album would receive based on how old the album was. Initially it was our intent to use genres to predict an albums rating, but we decided to shift gears and move in the direction of age instead. Overall, our idea for this project was to break down the components of an album that make it unique and use them to discover more about their genres, what role age plays in the success of an album, and how music has changed over time.

Introduction

The motivation behind doing this project is very straightforward: both partners enjoy music. Specifically, both respect music as well as the time and the effort that goes into making it. So when given a chance to take a deep dive into the Top 5000 Albums of All Time, it was an easy choice. There is a bit of background that needs to be known about where this data set came from. This data was collected from the website, Rate Your Music, where people are able to rate and review albums of their choice based on a five-star rating system. Essentially, the rankings of the data set are arbitrary so certain albums may be ranked higher or lower only based off of personal preference.

Analysis

Genres and Descriptors

The primary goal of this section of the project was to find out which descriptors are most popular with which genres and vice versa. We started first by breaking down the genres and descriptors at the commas because each album takes on several genres and descriptors. After that was finished, we could then match genres to descriptors and count how many times each descriptor occurred in that specific genre. The results are sorted by highest amount below.

```
head(sortedDf, 20)
```

```
## # A tibble: 20 x 3
## # Groups:   primary_genres [10]
```

```
##    primary_genres     descriptors   count
##    <chr>              <chr>         <int>
##  1 Alternative Rock   malevocals      630
##  2 Pop Rock           malevocals      570
##  3 Pop Rock           melodic         558
##  4 Singer-Songwriter  malevocals      520
##  5 Indie Rock         malevocals      471
##  6 Alternative Rock   melodic         450
##  7 Neo-Psychedelia    psychedelic     400
##  8 Singer-Songwriter  melancholic     380
##  9 Art Rock           malevocals      378
## 10 Alternative Rock   energetic       370
## 11 Art Pop            femalevocals    363
## 12 East Coast Hip Hop malevocals      342
## 13 Singer-Songwriter  melodic         342
## 14 Indie Rock         melodic         341
## 15 Experimental Rock  malevocals      332
## 16 Singer-Songwriter  poetic          324
## 17 Art Pop            melodic         315
## 18 Singer-Songwriter  bittersweet     306
## 19 Singer-Songwriter  introspective   306
## 20 Indie Pop          melodic         296
```

It is clear from the table that male vocals seems to dominate the alternative rock genre, but not only that unfortunately it appears that male vocals is a common theme among all of the specific genres of rock. Though looking at the table, I think it is safe to assume that alternative rock can be defined as energetic and melodic with typically male vocals. But this can be taken a step further as we can then subset out a specific genre or descriptor that we wish to look at. For example, if we wanted to look at one of my favorite genres, ska punk:

```
subset(pggroupDf, primary_genres == "Ska Punk")
```

```
## # A tibble: 5 x 3
## # Groups:   primary_genres [1]
##   primary_genres descriptors count
##   <chr>          <chr>       <int>
## 1 Ska Punk       malevocals      9
## 2 Ska Punk       playful         9
## 3 Ska Punk       summer          8
## 4 Ska Punk       energetic       7
## 5 Ska Punk       drugs           6
```

We see that the few albums that did make the top 500 list were characterized as male vocals, playful, summer, energetic, and drugs (how is this a descriptor). The reverse is also true. We can subset out a specific descriptor as well. Say you are feeling melancholic and you want to know which genre would best fit your mood:

```
subset(dsgroupDf, descriptors == "melancholic")
```

```
## # A tibble: 5 x 3
## # Groups:   primary_genres [5]
##   primary_genres    descriptors count
```

```
##   <chr>             <chr>       <int>
## 1 Singer-Songwriter melancholic   380
## 2 Indie Rock        melancholic   282
## 3 Alternative Rock  melancholic   281
## 4 Art Pop           melancholic   204
## 5 Art Rock          melancholic   173
```

The best fit for your melancholic mood would be to try the genres singer-songwriter, indie rock, alt rock, art pop, and art rock. Overall, I think that the results of fitting genres to descriptors turned out exactly as we had hoped and even better. Our original goal was purely to find out what genres fit with what descriptors, but in the end we made something better that allows us to subset by genre and descriptor.

Genres Over Time

In the same pattern as the earlier code, our code cleans the RYM data, giving us a data frame of only the primary genres from each of the top 5000 records, where each of the (usually) multiple genres per album is given an individual row. Using the already ordered position of albums, it then collects the top 50 albums in a given year. (This isn't entirely true though - more on this later). We limit it to 50, as it caps the given data to about 3000 albums, so that our data won't be influenced by somewhat critically successful, yet ultimately un-influential records. From this, we count the top 2 genres, including ties. Because the categories for genres are so specific, this results in anywhere from 2 to 20 top genres of the year. 2 was settled on, rather than 1, because specific instances seemed inaccurate, although statistically valid.

```
countedDf <- dateDf %>%
  count(release_date, primary_genres) %>%
  group_by(release_date) %>%
  top_n(1, n)

subset(countedDf, release_date == "2020")
```

```
## # A tibble: 1 x 3
## # Groups:   release_date [1]
##   release_date primary_genres        n
##   <chr>        <chr>             <int>
## 1 2020         Singer-Songwriter     5
```

For instance, in the top 50 albums of 2020, the most prevalent genre was "Singer-songwriter", without tie. Opening up the data to top 2 genres helps broaden the data, giving us more genres related to electronic for 2020, which is what one would expect.

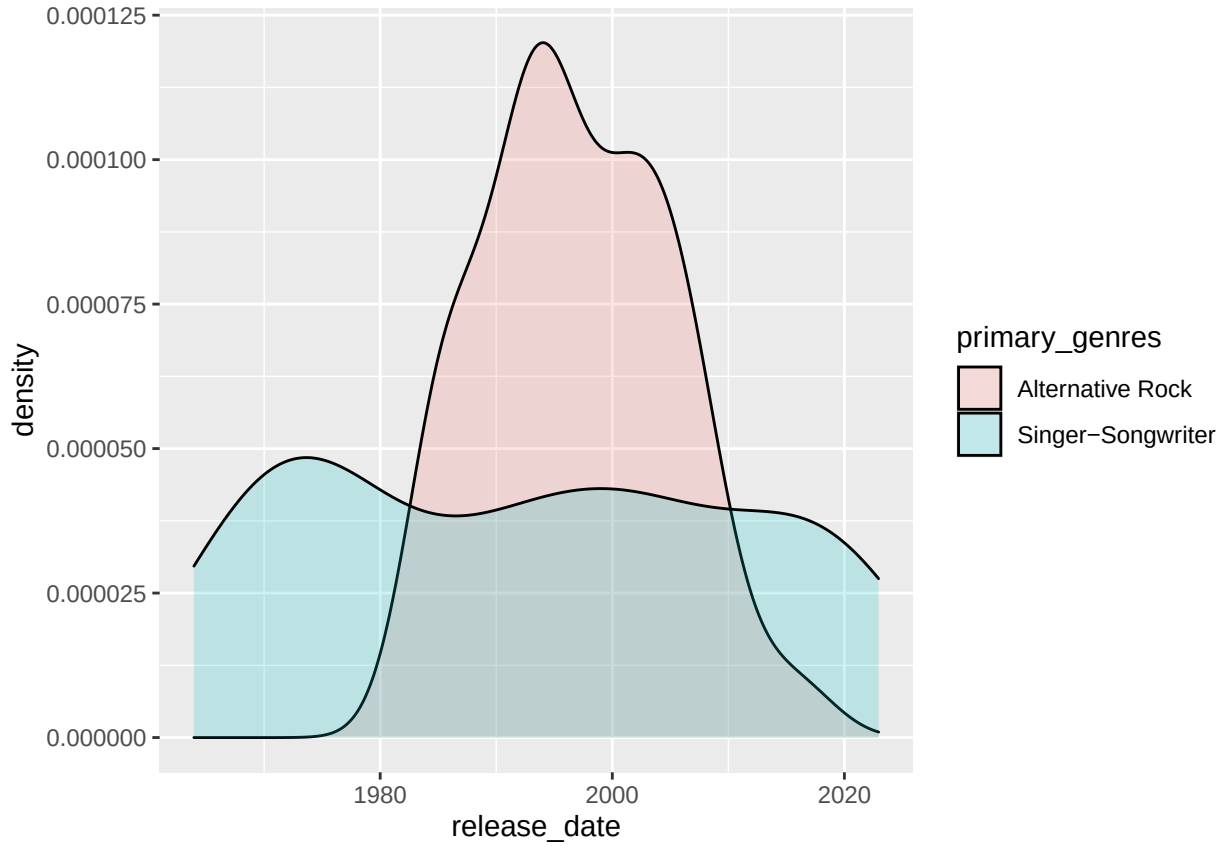
```
countedDf <- dateDf %>%
  count(release_date, primary_genres) %>%
  group_by(release_date) %>%
  top_n(2, n)

subset(countedDf, release_date == "2020")
```

```
## # A tibble: 4 x 3
## # Groups:   release_date [1]
##   release_date primary_genres        n
##   <chr>        <chr>             <int>
```

```
## 1 2020          Art Pop               3
## 2 2020          Dance-Pop             3
## 3 2020          Singer-Songwriter     5
## 4 2020          Synthpop              3
```

For obtaining data from a particular year on genre, this data table is extremely useful. It was expected that this data table would be useful for tracking trends across years, as well. However, the result was that too many genres were included for the data table itself to provide anything of value. With no real association between genres, filtering this further to track trends seemed like the wrong approach.

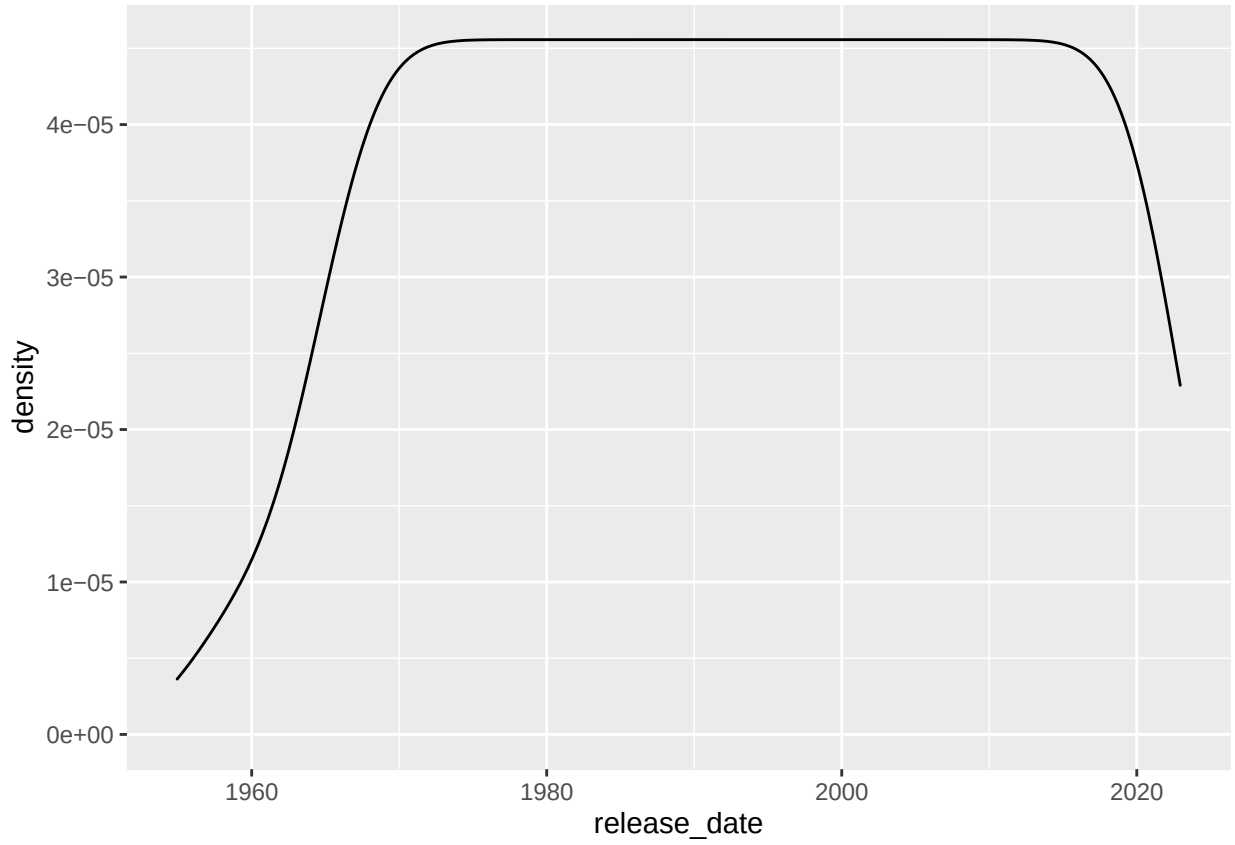
```
genre = list("Singer-Songwriter", "Alternative Rock")

genreTrack <- dateDf[(dateDf$primary_genres==genre),]
genreTrack$release_date <- as.Date(genreTrack$release_date, "%Y")
ggplot(genreTrack, aes(release_date, fill = primary_genres)) +
  geom_density(alpha = 0.2) +
  scale_x_date()
```



However, this code provides a meaningful, easy to change method of tracking genres across years. One may input a single genre, or many, and be returned an easy-to-use graph. The following graph, a comparison between alternative-rock and singer-songwriter, shows the lasting influence of one genre compared to the very timely and dated nature of another.

Traversing the data according to year, one might notice that somewhat of a recency bias affects the data. For the 1950's, only between 2-20 releases are available, but this greatly opens up as we enter the 1960's. We can see the distribution of albums included in the top 5000 by the following graph.

```
genreTrack <- dateDf
genreTrack$release_date <- as.Date(genreTrack$release_date, "%Y")
ggplot(genreTrack, aes(release_date, fill = release_date)) +
  geom_density(alpha = 0.2) +
  scale_x_date()
```



As shown, it is obvious that some sort of recency bias affects the number of ratings, and quality of those ratings, for between 1980-2010. It makes sense that, given the open source nature of RYM voting, users would want to vote for music they are familiar with. It's only logical that most using RYM grew up with the music in this time span, and therefore, often rank records in that period. RYM is an internet only application, which would deter those unfamiliar with internet from voting on older albums.

Linear Model

In this section, we attempted to use a linear model to predict a rating for an album based on how old it is. First, we needed to calculate the length of time between the present and when the album came out using the function difftime(). This gives the span of time in days between now and the album so this needed to be divided by 365. Once this had been set, our model could be created which gave:

```
summary(lmMusic)
```

```
##
## Call:
## lm(formula = avg_rating ~ timeDiffYears, data = RatingMusic)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.72638 -0.10245  0.06115  0.20155  0.91347
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.317482   0.008777  377.99   <2e-16 ***
## timeDiffYears 0.008910   0.000292   30.51   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3438 on 4998 degrees of freedom
## Multiple R-squared:  0.157,  Adjusted R-squared:  0.1569
## F-statistic:   931 on 1 and 4998 DF,  p-value: < 2.2e-16
```

This basic model only accounts for the amount of time lapsed between release date and now which cannot amount for too good of a prediction. Since rating is directly connected to age, the older an album gets the higher the rating should be. Obviously, this will not always be true as age is not the only factor in how good an album will be. In any case, the model states that a 20 year old album should receive a rating of 3.50.

If we change our model and factor in the rating count, which measures how many people rated the album, we can achieve a more accurate model.

```
summary(lmtransMusic)
```

```
##
## Call:
## lm(formula = rating ~ timeDiffYears + sqrt_rating_count, data = transRatingMusic)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.66875 -0.08409  0.07248  0.19039  0.69216
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.0365492  0.0140221  216.56   <2e-16 ***
## timeDiffYears     0.0083543  0.0002764   30.22   <2e-16 ***
## sqrt_rating_count 0.0040541  0.0001633   24.83   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3244 on 4997 degrees of freedom
## Multiple R-squared:  0.2496, Adjusted R-squared:  0.2493
## F-statistic:   831 on 2 and 4997 DF,  p-value: < 2.2e-16
```

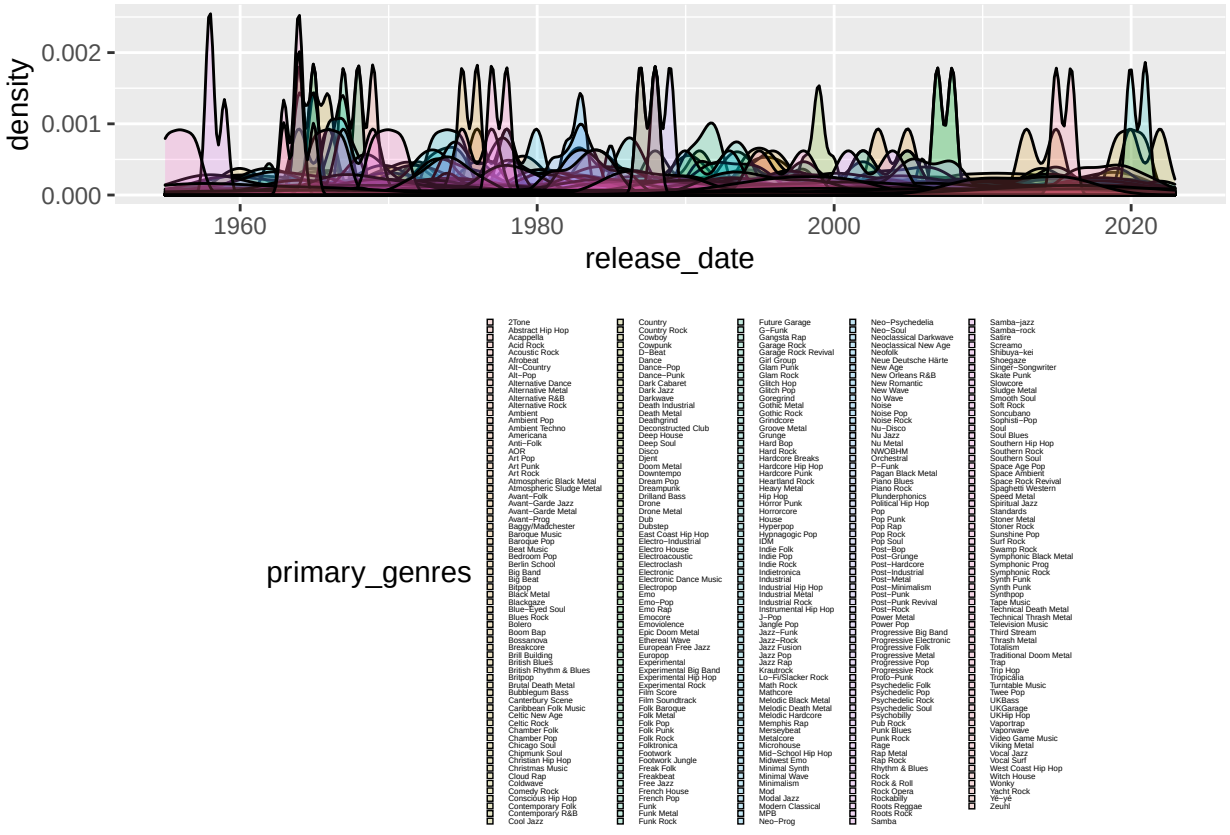
Now with this model, the rating of an album does not scale evenly with age but instead now also factors in how many people rated it. To show that age does not play as much of a big factor now, we can analyze the third ranked album, Pink Floyd's Wish You Were Here. This album is 47 years old with a rating count of 56,776 which would rate is at 4.39. This rating is way closer to the rank from the data set, 4.31, than the first liner model which would have given a score of 3.74.

Overall, we were happy with the outcomes of the linear model. We didn't have many expectations going in as we were unsure of how the model would work but factoring in the rating count helped to exceed expectations that weren't even there.

Reflections

We both believe the project went well, although like most, it was not without its issues. For starters, the smartest move with this project would have been to track relations of genres from another data-set after we extracted the top records from RYM, or to somehow filter down the amount of genres. The sheer amount of genres and descriptors, while no doubt provide accuracy and specificity, made all of the tasks we performed extremely difficult. For instance, while tracking genre within the linear model would have been ideal, as per our original goal, the variability of genres made this near impossible. Additionally, the descriptors had many of the same issues. We did not have the experience with models to get this working properly, but we settled on a close approximation using the age of an album. A separate data set that links genres would have greatly simplified tracking genres across time, as well. Even choosing only limiting genre to only the top single genre per year, you are still met with this unreadable graph:

```
genreTrack <- dateDf
genreTrack$release_date <- as.Date(genreTrack$release_date, "%Y")
ggplot(genreTrack, aes(release_date, fill = primary_genres)) +
  geom_density(alpha = 0.2) +
  scale_x_date() +
  theme(legend.key.size = unit(0.1, 'cm'),
        legend.text = element_text(size=3),
        legend.position = "bottom")
```



This is why we settled on letting a user choose genres to track, instead of trying to keep track of all data at the same time.

Doing this project again, we would have settled on a more concise data set, with only a few possible genres that an album can fall into. This would have simplified the project massively, and we would have been able

to achieve what our original goals with the project entailed. Moreover, we would look for a data set that contained not only the highest rated albums, but a random sampling of albums across all scores. With using only the highest ranked records, you can only create a model that will generally give a high score, as that is all it is trained on. When creating a linear model with the top records in genres, we would get high scores in genres that critics generally dislike, such as nu-metal and AOR.

The above considered, we believe that with our available knowledge, we did as well as we could have operating within this data-set. We did not achieve a single model, or method of tracking, that incorporates all available data into a single algorithm, but the approximations show proficiency in R.

Attribution

Both partners worked on the project a fair amount and delegation of work was split evenly. We met up to work on the first section of the project which was genres and descriptors. After meeting, we decided to each take one of the two remaining tasks. Luke worked on the highest rated genres over time and Marcus worked on building a linear model for rating. We did this separately, and would bounce our code between each other to check for bugs. We would also talk about features of the project together, although only one person would end up implementing the idea. We wanted to make sure we would be on the same track for all the features, and that we would understand how they worked. Both partners collaborated on the final project write up.