Data Analysis Project

Max Herman

12/14/2022

1 Data Analysis Project

1.1 Introduction

Since its genesis in 2006, Spotify has grown to be the largest music streaming platform in the world. Spotify services over 433 million users across 184 countries. As such, many trust Spotify to curate their playlists, suggest weekly tunes, and rank the world's top songs. In order to provide these services, Spotify leverages machine learning on its vast amount of data. In addition to basic information and metadata, Spotify calculates several data points for each song including simple features like tempo and others as abstract as danceability. In this paper, we will be exploring what features make up songs in Spotify's weekly top 200 list.

1.2 Research Question

How well can the features of a song such as energy, streams, and acousticness predict the weekly popularity ranking of songs on Spotify?

1.3 Significance of this study

In this study, we will be exploring popularity ranking as our outcome variable. The results of this study may be valuable because it will potentially provide insight into what makes a song popular. The ability to identify the features behind the most popular songs on Spotify has immense fiscal value. Top ranked songs on Spotify average around 3 million streams netting those artists north of $12,000 (Spotify pays $0.004 per stream on average). Hopefully, the results of this study can inform artists working to monetize their art on ways to make more popular music and hence make more money.

1.4 Data

This data set consists of the top 200 songs on Spotify every week from 02/04/2021 ~ 07/14/2022. The data set has these rankings not only for the United States but for every country in which Spotify is offered as well as the global aggregate. There are several features reported in this data set but the ones of importance are explained on below:

- album_num_tracks - number of tracks in the album that the track is from
- weeks_on_chart - number of weeks the song was on Spotify Charts (in a given country)
- streams - number of streams
- 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
- key - the key of the song
- loudness - The overall loudness of a track in decibels (dB)
- 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.
- acousticness - A confidence measure from 0.0 to 1.0 of whether the track is acoustic
- instramentalness - Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context
- liveness - Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live
- valence - A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound

more negative (e.g. sad, depressed, angry).
- tempo - tempo of the song in BPM (beats per minute)
- duration - length of the song in milliseconds
- release_date - day the song was released on Spotify

For more in depth descriptions of these features, refer to the Spotify documentation. https://developer.spotify.com/documentation/web-api/reference/#/operations/get-several-audio-features (https://developer.spotify.com/documentation/web-api/reference/#/operations/get-several-audio-features)

To download this data set, visit the following Kaggle page: https://www.kaggle.com/datasets/yelexa/spotify200 (https://www.kaggle.com/datasets/yelexa/spotify200)

Please note that the data set I submitted is not the same as the data set above. This is because the original data set is 800MB large. Instead, I have submitted the data set after I reduced its scope.

1.4.1 Data Wrangling

In order to complete my analysis, the following steps were taken to organize the data

1. Called in data
2. Narrowed scope of data
3. Removed duplicate songs
4. Calculated days since release
5. Removed null observations

1.4.2 Calling in the data

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✔ ggplot2 3.3.5     ✔ purrr   0.3.4
## ✔ tibble  3.1.6     ✔ dplyr   1.0.7
## ✔ tidyr   1.1.4     ✔ stringr 1.4.0
## ✔ readr   2.0.2     ✔ forcats 0.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
spotify <- read_csv("final.csv")
```

```
## New names:
## * `` -> ...1
```

```
## Rows: 1787999 Columns: 36
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (13): uri, artist_names, artist_individual, artist_id, artist_genre, ar...
## dbl  (22): ...1, rank, artists_num, collab, album_num_tracks, peak_rank, pre...
## date  (1): week
##
## 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.
```

```
dim(spotify)
```

```
## [1] 1787999      36
```

```
str(spotify)
```

```
## spec_tbl_df [1,787,999 × 36] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ...1              : num [1:1787999] 0 1 2 3 4 5 6 7 8 9 ...
##  $ uri               : chr [1:1787999] "spotify:track:2gpQi3hbcUAcEG8m2dlgfB" "spotif
y:track:2x8oBuYaObjqHqgGuIUZ0b" "spotify:track:2SJZdZ5DLtlRosJ2xHJJJa" "spotify:track:1O
2pcBJGej0pmH2Y9XZMs6" ...
##  $ rank              : num [1:1787999] 1 2 3 5 6 11 17 20 23 24 ...
##  $ artist_names      : chr [1:1787999] "Paulo Londra" "WOS" "Paulo Londra" "Cris Mj"
...
##  $ artists_num       : num [1:1787999] 1 1 1 1 1 1 1 1 1 1 ...
##  $ artist_individual: chr [1:1787999] "Paulo Londra" "WOS" "Paulo Londra" "Cris Mj"
...
##  $ artist_id         : chr [1:1787999] "spotify:artist:3vQ0GE3mI0dAaxIMYe5g7z" "spotif
y:artist:5YCc6xS5Gpj3EkaYGdjyNK" "spotify:artist:3vQ0GE3mI0dAaxIMYe5g7z" "spotify:artis
t:1Yj5Xey7kTwvZla8sqdsdE" ...
##  $ artist_genre      : chr [1:1787999] "argentine hip hop" "argentine indie" "argentin
e hip hop" "urbano chileno" ...
##  $ artist_img        : chr [1:1787999] "https://i.scdn.co/image/ab6761610000e5ebf796a9
76c5597baf6f7b786c" "https://i.scdn.co/image/ab6761610000e5eb75e1511f68e988110962dd9c"
"https://i.scdn.co/image/ab6761610000e5ebf796a976c5597baf6f7b786c" "https://i.scdn.co/im
age/ab6761610000e5eb8f4ebcf4a5d23a2515374f89" ...
##  $ collab            : num [1:1787999] 0 0 0 0 0 0 0 0 0 0 ...
##  $ track_name        : chr [1:1787999] "Plan A" "ARRANCARMELO" "Chance" "Una Noche en
Medellín" ...
##  $ release_date      : chr [1:1787999] "2022-03-23" "2022-04-06" "2022-04-06" "2022-01
-21" ...
##  $ album_num_tracks : num [1:1787999] 1 1 2 1 1 1 14 1 15 1 ...
##  $ album_cover       : chr [1:1787999] "https://i.scdn.co/image/ab67616d0000b2737e1179
e64539bedc938933ef" "https://i.scdn.co/image/ab67616d0000b273d8c9945c63f1806031dae6f0"
"https://i.scdn.co/image/ab67616d0000b273274a28ec692ca28a73da1288" "https://i.scdn.co/im
age/ab67616d0000b273697ed12671078b5dee48f0ad" ...
##  $ source            : chr [1:1787999] "WEA Latina" "DOGUITO Records / DALE PLAY Recor
ds" "WEA Latina" "Nabru Records LLC" ...
##  $ peak_rank         : num [1:1787999] 1 2 3 5 6 6 14 11 13 2 ...
##  $ previous_rank     : num [1:1787999] 1 129 59 5 9 6 16 15 17 21 ...
##  $ weeks_on_chart    : num [1:1787999] 4 2 2 8 3 2 47 8 6 9 ...
##  $ streams           : num [1:1787999] 3003411 2512175 2408983 2080139 1923270 ...
##  $ week              : Date[1:1787999], format: "2022-04-14" "2022-04-14" ...
##  $ danceability      : num [1:1787999] 0.583 0.654 0.721 0.87 0.761 0.52 0.651 0.771
0.812 0.596 ...
##  $ energy            : num [1:1787999] 0.834 0.354 0.463 0.548 0.696 0.731 0.731 0.467
0.736 0.71 ...
##  $ key               : num [1:1787999] 0 5 1 10 7 6 7 5 4 6 ...
##  $ mode              : num [1:1787999] 1 1 0 0 0 0 1 0 0 1 ...
##  $ loudness          : num [1:1787999] -4.88 -7.36 -9.48 -5.25 -3.82 ...
##  $ speechiness       : num [1:1787999] 0.0444 0.0738 0.0646 0.077 0.0505 0.0557 0.0549
0.123 0.0833 0.136 ...
##  $ acousticness      : num [1:1787999] 0.0495 0.724 0.241 0.0924 0.0811 0.342 0.116 0.
375 0.152 0.243 ...
##  $ instrumentalness : num [1:1787999] 0.00 0.00 0.00 4.60e-05 6.25e-05 1.01e-03 0.00
9.74e-03 2.54e-03 0.00 ...
##  $ liveness          : num [1:1787999] 0.0658 0.134 0.0929 0.0534 0.101 0.311 0.0708
0.112 0.0914 0.204 ...
```

```
## $ valence          : num [1:1787999] 0.557 0.262 0.216 0.832 0.501 0.662 0.653 0.256
0.396 0.632 ...
## $ tempo            : num [1:1787999] 173.9 82 137.9 96 95.1 ...
## $ duration         : num [1:1787999] 178203 183547 204003 153750 133895 ...
## $ country          : chr [1:1787999] "Argentina" "Argentina" "Argentina" "Argentina"
...
## $ region           : chr [1:1787999] "South America" "South America" "South America"
"South America" ...
## $ language         : chr [1:1787999] "Spanish" "Spanish" "Spanish" "Spanish" ...
## $ pivot            : num [1:1787999] 0 0 0 0 0 0 0 0 0 0 ...
## - attr(*, "spec")=
##   .. cols(
##   ..   ...1 = col_double(),
##   ..   uri = col_character(),
##   ..   rank = col_double(),
##   ..   artist_names = col_character(),
##   ..   artists_num = col_double(),
##   ..   artist_individual = col_character(),
##   ..   artist_id = col_character(),
##   ..   artist_genre = col_character(),
##   ..   artist_img = col_character(),
##   ..   collab = col_double(),
##   ..   track_name = col_character(),
##   ..   release_date = col_character(),
##   ..   album_num_tracks = col_double(),
##   ..   album_cover = col_character(),
##   ..   source = col_character(),
##   ..   peak_rank = col_double(),
##   ..   previous_rank = col_double(),
##   ..   weeks_on_chart = col_double(),
##   ..   streams = col_double(),
##   ..   week = col_date(format = ""),
##   ..   danceability = col_double(),
##   ..   energy = col_double(),
##   ..   key = col_double(),
##   ..   mode = col_double(),
##   ..   loudness = col_double(),
##   ..   speechiness = col_double(),
##   ..   acousticness = col_double(),
##   ..   instrumentalness = col_double(),
##   ..   liveness = col_double(),
##   ..   valence = col_double(),
##   ..   tempo = col_double(),
##   ..   duration = col_double(),
##   ..   country = col_character(),
##   ..   region = col_character(),
##   ..   language = col_character(),
##   ..   pivot = col_double()
##   .. )
## - attr(*, "problems")=<externalptr>
```

Clearly, this data set is massive so we are going to reduce its scope. To do so, we will limit our search to only the charts in the United States and within a specific date range. We will select the weeks of 5/6/21 and 5/5/22 for this study. Note that this selection is arbitrary other than the selection of the same week in consecutive years. This choice was to ensure that there was no variability caused by the time of the year.

1.4.3 Narrowing scope of data

```
week_choice_1 <- as.Date("2021-05-06")
week_choice_2 <- as.Date("2022-05-05")

spotify_small <- spotify %>%
  filter(country == "United States") %>%
  filter(week == week_choice_1 | week == week_choice_2)

dim(spotify_small)
```

```
## [1] 592  36
```

We have now reduced our dataset from 1,787,999 observations down to 592! Now, let’s clean up this tibble.

1.4.4 Cleaning the tibble

```
# some songs have multiple observations to represent each involved artist
# we include week to prevent songs that are on the top charts in both years from being e
xcluded
spotify_distinct_songs <- spotify_small %>%
  distinct(track_name, week, .keep_all = T)

# define a variable that tracks days since release
spotify_distinct_songs$release_date <- as.Date(spotify_distinct_songs$release_date)

spotify_mutated <- spotify_distinct_songs %>%
  mutate(days_since_release = difftime(week, release_date, units = "days"))

spotify_mutated$days_since_release <- as.numeric(spotify_mutated$days_since_release)

# Remove rows where days since release is NA (this happens when release_day is NA)
spotify_mutated <- spotify_mutated %>%
  filter(!is.na(days_since_release))
```

Finally, we will select the columns we will use in our analysis.

```
spotify_final_outcome <- spotify_mutated %>%
  select(album_num_tracks, weeks_on_chart, streams, danceability, energy, key, loudness,
         speechiness, acousticness, instrumentalness, liveness, valence, tempo, duratio
n, days_since_release, rank)

# Write out final df to file for submission (original df is too large)
write.csv(spotify_final_outcome, "spotify_final.csv")
```

1.5 Visualize predictors by the rank

Now that we have wrangled and cleaned our data into one tibble, let's visualize how our predictors relate to our outcome variable, rank, using a matrix of scatter plots.

```
library(PerformanceAnalytics)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

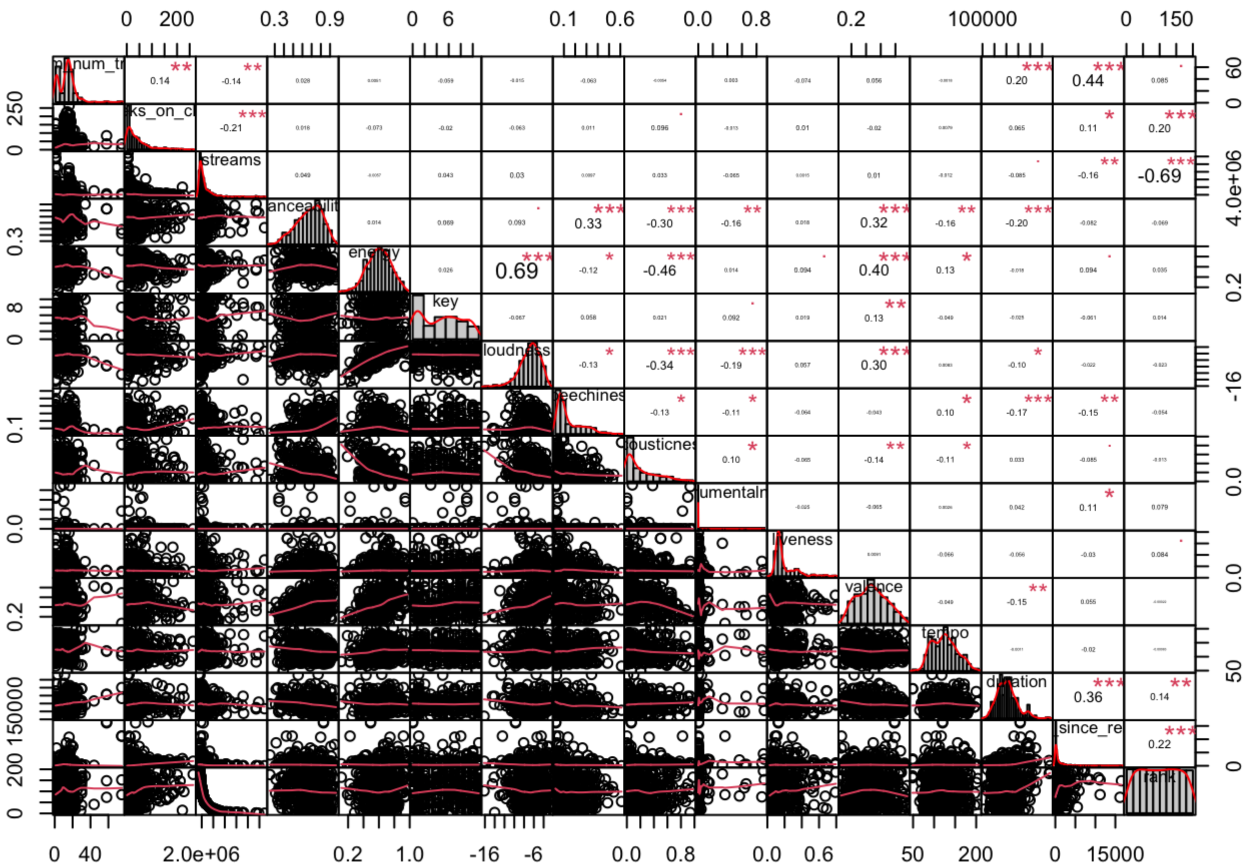
```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
##
## Attaching package: 'PerformanceAnalytics'
```

```
## The following object is masked from 'package:graphics':
##
##     legend
```

```
chart.Correlation(spotify_final_outcome, histogram = TRUE, method = "pearson")
```



1.6 Principle Component Analysis

In the following section, we will conduct a principal component analysis (PCA) on our data. The purpose of a PCA is to reduce the complexity of high dimensional data while maintaining the underlying patterns that exist within it. With 15 different features, a PCA of our Spotify data will help us identify the overall trends that exist in our data set.

For our PCA analysis, we will not need our outcome variable so let's remove it.

```
spotify_final <- spotify_final_outcome %>%
  select(-rank)
```

1.6.1 Check for multicollinearity

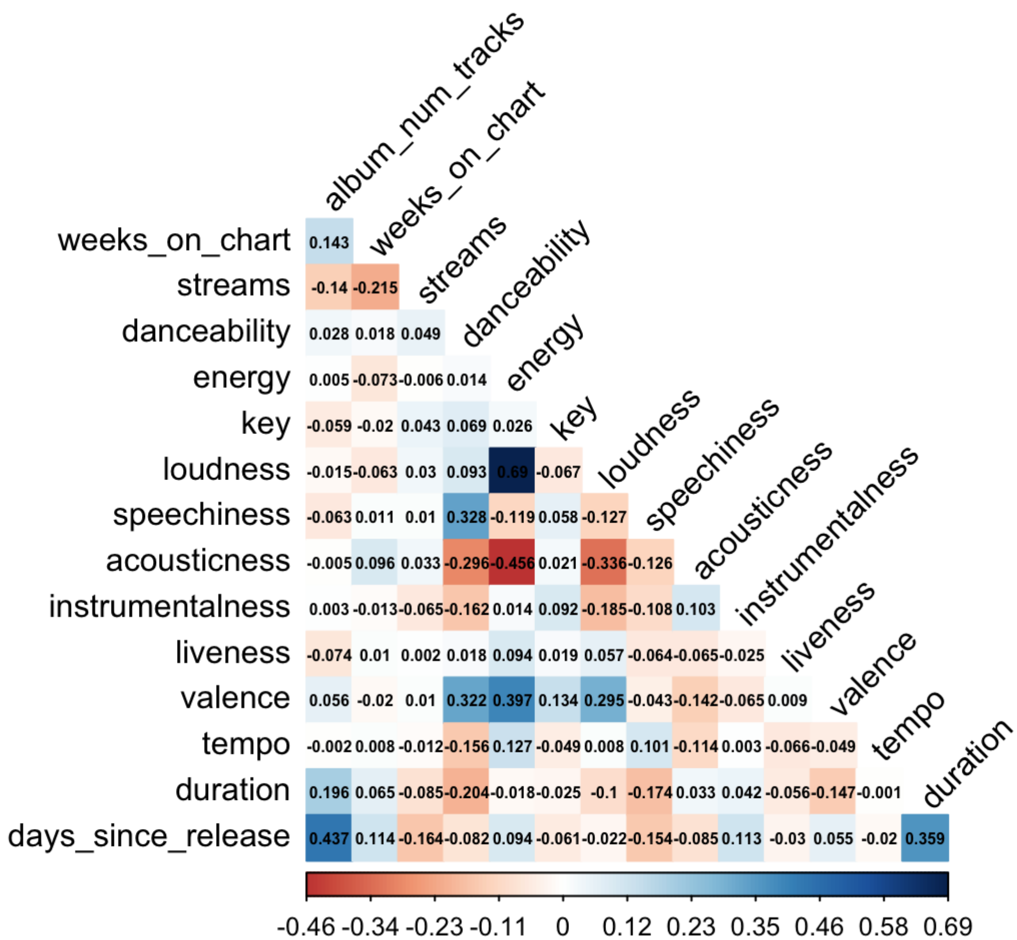
```
spotify_corr <- cor(spotify_final)

# Write correlations out to a file for easier inspection
write.csv(spotify_corr, "spotify_corr.csv")
```

Let's visualize these correlations as well.

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
corrplot(spotify_corr,
         type="lower", #put color strength on bottom
         tl.pos = "ld", #Character or logical, position of text labels, 'ld'(default if type=='lower') means left and diagonal,
         tl.cex = 1, #Numeric, for the size of text label (variable names).
         method="color",
         addCoef.col="black",
         diag=FALSE,
         tl.col="black",
         tl.srt=45,
         is.corr = FALSE,
         number.cex = 0.5,
         number.digits = 3)
```



None of the variables have r > 0.899 so we can assume there are no problems with multicollinearity.

1.6.2 Scale the variables

It is important that we scale our variables before performing any principle component analysis. This is because as it stands, some of our variables are on completely different scales. Some have units of days where others have arbitrary units defined by Spotify. Thus, we must scale our numeric predictors such that each has a mean of 0 and a standard deviation of 1.

```r
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
spotify_sc <- spotify_final %>%
  mutate_all(~(scale(.) %>% as.vector))
```

Let’s check that was successful by ensuring the mean and standard deviation of each feature is 0 and 1 respectively.

```r
psych::describe(spotify_sc)
```

```
##                    vars   n mean sd median trimmed  mad   min  max range  skew
## album_num_tracks      1 395    0  1   0.06   -0.08 0.62 -1.20 6.57  7.78  1.97
## weeks_on_chart        2 395    0  1  -0.33   -0.20 0.66 -0.84 4.75  5.59  1.90
## streams               3 395    0  1  -0.33   -0.21 0.36 -0.71 6.76  7.47  3.60
## danceability          4 395    0  1   0.10    0.05 1.07 -2.86 1.88  4.74 -0.41
## energy                5 395    0  1   0.01    0.01 1.01 -3.25 2.27  5.52 -0.16
## key                   6 395    0  1  -0.06   -0.02 1.21 -1.42 1.57  2.99  0.06
## loudness              7 395    0  1   0.13    0.07 0.94 -4.34 1.76  6.10 -0.83
## speechiness           8 395    0  1  -0.48   -0.17 0.48 -0.86 3.96  4.82  1.32
## acousticness          9 395    0  1  -0.40   -0.16 0.71 -0.92 3.27  4.19  1.17
## instrumentalness     10 395    0  1  -0.17   -0.17 0.00 -0.17 9.07  9.24  7.36
## liveness             11 395    0  1  -0.41   -0.19 0.35 -1.13 4.76  5.89  2.12
## valence              12 395    0  1  -0.02   -0.02 1.12 -1.84 2.25  4.09  0.16
## tempo                13 395    0  1  -0.02   -0.04 1.14 -2.38 2.69  5.07  0.27
## duration             14 395    0  1  -0.09   -0.07 0.88 -2.19 4.19  6.38  0.79
## days_since_release   15 395    0  1  -0.36   -0.23 0.20 -0.51 7.92  8.43  4.26
##                    kurtosis   se
## album_num_tracks      10.44 0.05
## weeks_on_chart         3.78 0.05
## streams               16.21 0.05
## danceability          -0.54 0.05
## energy                -0.25 0.05
## key                   -1.34 0.05
## loudness               1.10 0.05
## speechiness            0.90 0.05
## acousticness           0.47 0.05
## instrumentalness      56.52 0.05
## liveness               4.91 0.05
## valence               -0.80 0.05
## tempo                 -0.60 0.05
## duration               1.12 0.05
## days_since_release    23.72 0.05
```

1.6.3 Conudct initial PCA examination

Let's evaluate the proportion of the variance described by each of the possible components.

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WB
a
```

```
viz_pca <- prcomp(spotify_sc, center = TRUE,scale. = TRUE)

summary(viz_pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5    PC6     PC7
## Standard deviation     1.5425 1.3968 1.21010 1.10996 1.05590 1.0406 0.98813
## Proportion of Variance 0.1586 0.1301 0.09762 0.08213 0.07433 0.0722 0.06509
## Cumulative Proportion  0.1586 0.2887 0.38632 0.46846 0.54279 0.6150 0.68008
##                            PC8     PC9    PC10    PC11    PC12    PC13   PC14
## Standard deviation     0.93990 0.90736 0.85799 0.80840 0.76076 0.71236 0.6317
## Proportion of Variance 0.05889 0.05489 0.04908 0.04357 0.03858 0.03383 0.0266
## Cumulative Proportion  0.73897 0.79386 0.84293 0.88650 0.92508 0.95891 0.9855
##                           PC15
## Standard deviation     0.46610
## Proportion of Variance 0.01448
## Cumulative Proportion  1.00000
```

```
viz_pca$rotation #show the loadings for each component by variable
```

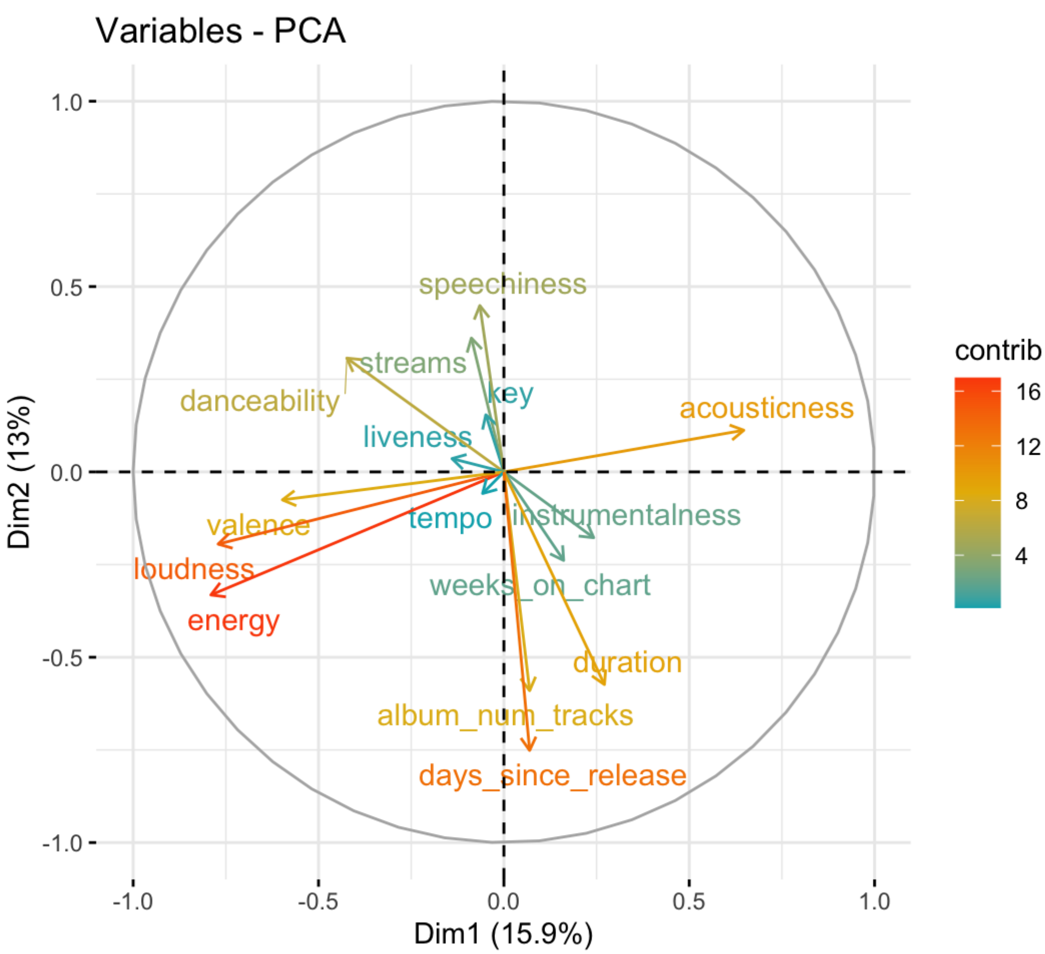
```
##                             PC1         PC2         PC3         PC4         PC5
## album_num_tracks     0.04501961 -0.42246470  0.34833009 -0.02139360  0.01058424
## weeks_on_chart       0.10454333 -0.17092406  0.32219078  0.02780063  0.19188053
## streams             -0.05684246  0.25849813 -0.23531796 -0.09617758 -0.01545140
## danceability        -0.27467750  0.21977232  0.53618607 -0.13833993  0.02970178
## energy              -0.51238244 -0.23759233 -0.22642938  0.02729692 -0.11979215
## key                 -0.03159014  0.11027096  0.05954593 -0.45455585 -0.51368393
## loudness            -0.49973717 -0.13941220 -0.20509891  0.04859279  0.16757571
## speechiness         -0.04207025  0.32128256  0.44881298  0.30420040 -0.22945313
## acousticness         0.41959686  0.08047085 -0.11737397 -0.25994470  0.12318522
## instrumentalness     0.15706248 -0.12705797 -0.17318219 -0.24650701 -0.53567251
## liveness            -0.09073948  0.02568305 -0.12300331 -0.19198329  0.34652449
## valence             -0.38734649 -0.05400581  0.14705650 -0.36132886 -0.10200620
## tempo               -0.03722832 -0.04164354 -0.12210988  0.60573739 -0.40490852
## duration             0.17581592 -0.41062434 -0.02413561  0.04008512 -0.02277678
## days_since_release   0.04501101 -0.53793798  0.19405428 -0.06195489 -0.07696346
##                              PC6         PC7          PC8         PC9
## album_num_tracks    -0.212694594 -0.04835823 -0.026261708  0.39429000
## weeks_on_chart       0.523207566 -0.26948040 -0.298939247  0.05654300
## streams             -0.556395862  0.05752480 -0.285107662  0.42259566
## danceability        -0.121774414  0.10245756  0.134022844  0.02706686
## energy               0.104890306 -0.02843138 -0.009980267 -0.04155289
## key                  0.120543181  0.09389026 -0.543171682 -0.24422269
## loudness             0.023458941 -0.17540897 -0.022429989 -0.12266482
## speechiness          0.007771324  0.18775377  0.007045594 -0.02783845
## acousticness         0.038884956 -0.38951014 -0.136237762  0.20859259
## instrumentalness     0.231827880  0.11198018  0.533134812  0.18281637
## liveness             0.385398856  0.68628582 -0.162174147  0.36034214
## valence             -0.003039252 -0.32075012 -0.045461716  0.22653166
## tempo                0.161580830 -0.05935462 -0.307348424  0.38064964
## duration            -0.256717070  0.25122421 -0.296651141 -0.40534757
## days_since_release -0.186162291  0.17558044  0.037197989  0.13239723
##                             PC10        PC11         PC12         PC13
## album_num_tracks    -0.221945975  0.46364122  0.007291038  0.432874714
## weeks_on_chart       0.597123824  0.03642510 -0.098747698 -0.057366015
## streams              0.526116899  0.07417264 -0.017307037 -0.068044281
## danceability         0.128148580 -0.22833632 -0.217887905  0.298328647
## energy               0.111217968  0.08753806  0.218597512  0.002784584
## key                 -0.148671926  0.29445093 -0.122191528 -0.032411108
## loudness             0.088857626  0.27881898  0.248673179  0.126033500
## speechiness          0.068795692  0.11470682  0.671894660 -0.135868047
## acousticness        -0.148945938 -0.07848491  0.498892469  0.080564684
## instrumentalness     0.349405313 -0.01050157  0.086051441  0.207975830
## liveness            -0.124575086 -0.07584513  0.124989715  0.063502733
## valence             -0.177722292 -0.53908725  0.127366029 -0.075092296
## tempo               -0.160459837 -0.24808249 -0.154599041  0.108582319
## duration             0.191672386 -0.41878681  0.231728058  0.381877248
## days_since_release   0.001405999 -0.03527057  0.026130400 -0.680631616
##                             PC14        PC15
## album_num_tracks    -0.205366912  0.00538612
## weeks_on_chart      -0.091543614 -0.01497536
## streams             -0.069244849 -0.01732987
```

```
## danceability        0.522646180  0.21568909
## energy             -0.087002718  0.72499486
## key                 0.093583418 -0.03513441
## loudness            0.411376160 -0.52807322
## speechiness        -0.138819239 -0.07202477
## acousticness        0.407640500  0.23609051
## instrumentalness    0.052711626 -0.14803676
## liveness            0.008736609 -0.04868170
## valence            -0.357557676 -0.23722285
## tempo               0.238673008 -0.04651598
## duration           -0.037605903 -0.05813300
## days_since_release  0.333780980 -0.01350404
```

1.6.4 Visualize the PCA

Now that we have built our initial PCA, let's visualize it through a graph of the variables.

```
fviz_pca_var(viz_pca,
             col.var = "contrib", # Color by contributions to the PC
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE #Avoid overlapping text if possible
             )
```



From this graph, we can see the beginnings of our PCA based on the direction that the vectors for each variable are pointing. Let's now do some validation and pruning of our PCA.

1.6.5 Bartlett's Test

```
cortest.bartlett(spotify_sc, 395)
```

```
## R was not square, finding R from data
```

```
## $chisq
## [1] 943.5711
##
## $p.value
## [1] 6.404062e-135
##
## $df
## [1] 105
```

After conducting Bartlett's test, we calculated a p-value of 6.4e-135 which is much smaller than the standard alpha of 0.05. This allows us to reject the null hypothesis that our matrix is not an identity matrix. This means that there exist some relationships between the variables in our data set. Thus, we can continue with the principal component analysis.

1.6.6 KMO

We will now run a KMO on our data. We will be looking for variables with an index value lower than 0.5 and removing those variables.

```
KMO(spotify_sc)
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = spotify_sc)
## Overall MSA =  0.56
## MSA for each item = 
##   album_num_tracks     weeks_on_chart            streams       danceability 
##               0.62               0.60               0.64               0.48 
##             energy                key           loudness        speechiness 
##               0.54               0.51               0.59               0.57 
##       acousticness   instrumentalness           liveness            valence 
##               0.60               0.43               0.50               0.55 
##              tempo           duration days_since_release 
##               0.39               0.67               0.60
```

Tempo has the lowest KMO index value so we will remove that first. Then, we will rerun the KMO and repeat this process until all variables have an index value above 0.5.

```
spotify_filt <- spotify_sc %>%
  select(-tempo)
```

```
KMO(spotify_filt)
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = spotify_filt)
## Overall MSA =  0.57
## MSA for each item =
##   album_num_tracks     weeks_on_chart            streams       danceability
##               0.62               0.61               0.64               0.48
##             energy                key           loudness        speechiness
##               0.53               0.51               0.60               0.58
##       acousticness   instrumentalness           liveness            valence
##               0.59               0.43               0.51               0.55
##           duration days_since_release
##               0.67               0.60
```

Remove instrumentalness.

```
spotify_filt2 <- spotify_filt %>%
  select(-instrumentalness)
```

```
KMO(spotify_filt2)
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = spotify_filt2)
## Overall MSA =  0.58
## MSA for each item =
##   album_num_tracks     weeks_on_chart            streams       danceability
##               0.62               0.61               0.64               0.46
##             energy                key           loudness        speechiness
##               0.55               0.47               0.63               0.59
##       acousticness           liveness            valence           duration
##               0.60               0.54               0.55               0.69
## days_since_release
##               0.60
```

Remove danceability.

```
spotify_filt3 <- spotify_filt2 %>%
  select(-danceability)
```

```
KMO(spotify_filt3)
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = spotify_filt3)
## Overall MSA =  0.62
## MSA for each item = 
##   album_num_tracks     weeks_on_chart            streams             energy 
##               0.63               0.63               0.66               0.60 
##                key           loudness        speechiness       acousticness 
##               0.43               0.63               0.52               0.68 
##           liveness            valence           duration days_since_release 
##               0.58               0.69               0.62               0.59
```

Remove key.

```
spotify_filt4 <- spotify_filt3 %>%
  select(-key)
```

```
KMO(spotify_filt4)
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = spotify_filt4)
## Overall MSA =  0.63
## MSA for each item = 
##   album_num_tracks     weeks_on_chart            streams             energy 
##               0.62               0.63               0.66               0.61 
##           loudness        speechiness       acousticness           liveness 
##               0.64               0.52               0.68               0.58 
##            valence           duration days_since_release 
##               0.70               0.62               0.59
```

Now all of our variables have a KMO index value above 0.5 so we can proceed. Let’s now run our baseline PCA.

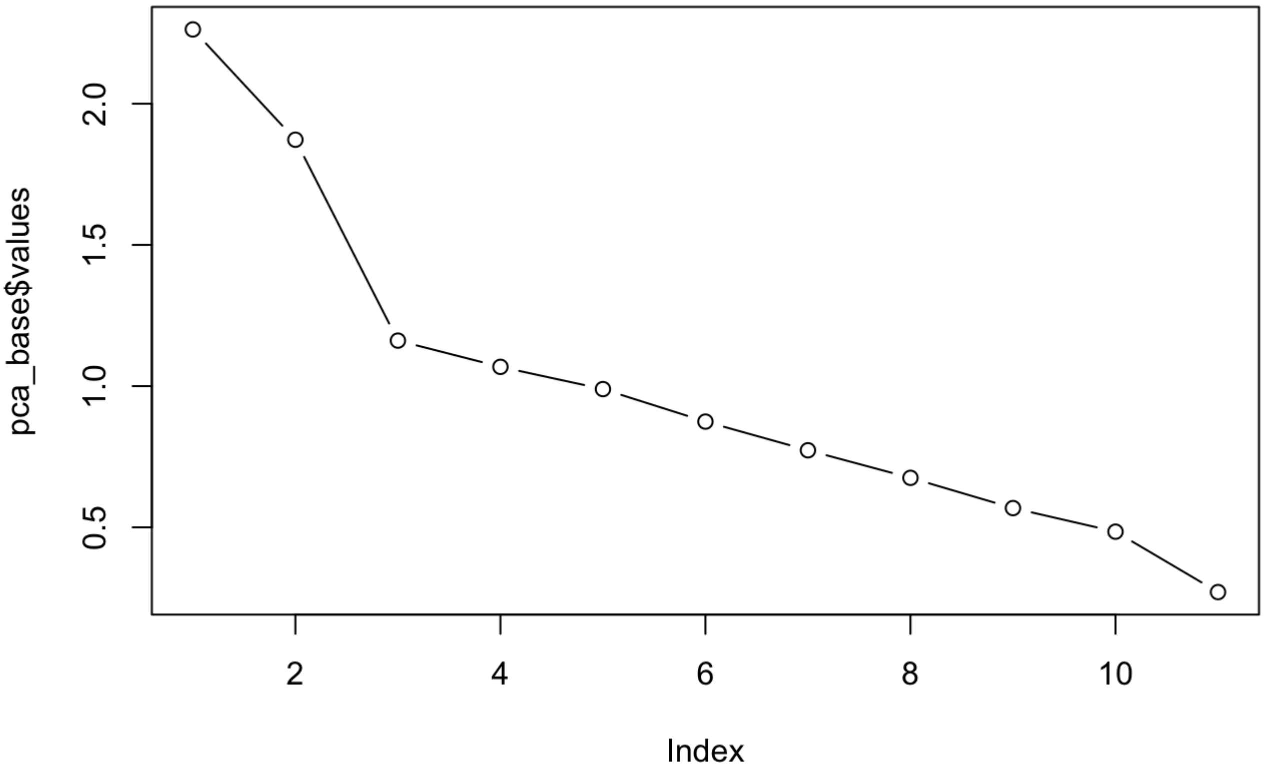
1.6.7 Baseline PCA

```
pca_base <- principal(spotify_filt4, nfactors = 11, rotate = "none")
pca_base
```

```
## Principal Components Analysis
## Call: principal(r = spotify_filt4, nfactors = 11, rotate = "none")
## Standardized loadings (pattern matrix) based upon correlation matrix
##                      PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10
## album_num_tracks    0.00  0.69  0.10 -0.20  0.12  0.41 -0.04 -0.40  0.21 -0.28
## weeks_on_chart     -0.15  0.35  0.51  0.43  0.21 -0.09  0.59  0.02 -0.14 -0.02
## streams             0.03 -0.41 -0.52 -0.29  0.07  0.39  0.56 -0.03 -0.09  0.01
## energy              0.89  0.08 -0.03  0.02 -0.02 -0.12  0.09  0.01  0.12  0.08
## loudness            0.82 -0.03 -0.05  0.07  0.08 -0.20  0.15 -0.21  0.30  0.14
## speechiness        -0.12 -0.31  0.65 -0.43 -0.26  0.21  0.12  0.20  0.31  0.14
## acousticness       -0.62 -0.02 -0.23  0.27  0.48  0.07 -0.04  0.05  0.42  0.25
## liveness            0.15 -0.09 -0.08  0.68 -0.51  0.48 -0.04  0.06  0.10 -0.01
## valence             0.57  0.02  0.11  0.01  0.51  0.33 -0.17  0.49 -0.07 -0.14
## duration           -0.14  0.61 -0.35 -0.12 -0.29 -0.25  0.19  0.43  0.25 -0.20
## days_since_release  0.07  0.79 -0.07 -0.16 -0.10  0.22 -0.05  0.06 -0.22  0.48
##                     PC11 h2       u2 com
## album_num_tracks    0.03  1  6.7e-16 3.4
## weeks_on_chart      0.01  1 -1.6e-15 4.3
## streams             0.01  1  4.4e-16 4.4
## energy              0.40  1 -4.4e-16 1.5
## loudness           -0.31  1  2.2e-16 2.1
## speechiness         0.00  1  1.0e-15 4.1
## acousticness        0.09  1  8.9e-16 4.1
## liveness           -0.02  1 -8.9e-16 3.0
## valence            -0.08  1  2.4e-15 4.1
## duration           -0.04  1  3.3e-16 4.9
## days_since_release -0.04  1  0.0e+00 2.2
##
##                        PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9 PC10 PC11
## SS loadings           2.26 1.87 1.16 1.07 0.99 0.87 0.77 0.68 0.57 0.48 0.27
## Proportion Var        0.21 0.17 0.11 0.10 0.09 0.08 0.07 0.06 0.05 0.04 0.02
## Cumulative Var        0.21 0.38 0.48 0.58 0.67 0.75 0.82 0.88 0.93 0.98 1.00
## Proportion Explained  0.21 0.17 0.11 0.10 0.09 0.08 0.07 0.06 0.05 0.04 0.02
## Cumulative Proportion 0.21 0.38 0.48 0.58 0.67 0.75 0.82 0.88 0.93 0.98 1.00
##
## Mean item complexity =  3.5
## Test of the hypothesis that 11 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0
##  with the empirical chi square  0  with prob <  NA
##
## Fit based upon off diagonal values = 1
```

The first four components are the only ones with SS loadings greater than 1. As such, we will only consider the first four components for our final analysis. Let’s graph a scree plot to check this intuition.

```
plot(pca_base$values, type = "b")
```



In the scree plot, the inflection point appears to be around 3-4. Since the first four components all have SS loadings greater than one, we will use four components in our final analysis.

1.6.8 Check that residuals are normally distributed

```
# Perform PCA to get residuals
pca_resid <- principal(spotify_filt4, nfactors = 4, rotate = "none")
pca_resid
```

```
## Principal Components Analysis
## Call: principal(r = spotify_filt4, nfactors = 4, rotate = "none")
## Standardized loadings (pattern matrix) based upon correlation matrix
##                     PC1   PC2   PC3   PC4   h2   u2 com
## album_num_tracks    0.00  0.69  0.10 -0.20 0.53 0.47 1.2
## weeks_on_chart     -0.15  0.35  0.51  0.43 0.59 0.41 3.0
## streams             0.03 -0.41 -0.52 -0.29 0.52 0.48 2.5
## energy              0.89  0.08 -0.03  0.02 0.80 0.20 1.0
## loudness            0.82 -0.03 -0.05  0.07 0.68 0.32 1.0
## speechiness        -0.12 -0.31  0.65 -0.43 0.72 0.28 2.3
## acousticness       -0.62 -0.02 -0.23  0.27 0.51 0.49 1.7
## liveness            0.15 -0.09 -0.08  0.68 0.50 0.50 1.2
## valence             0.57  0.02  0.11  0.01 0.34 0.66 1.1
## duration           -0.14  0.61 -0.35 -0.12 0.53 0.47 1.8
## days_since_release  0.07  0.79 -0.07 -0.16 0.66 0.34 1.1
##
##                        PC1  PC2  PC3  PC4
## SS loadings           2.26 1.87 1.16 1.07
## Proportion Var        0.21 0.17 0.11 0.10
## Cumulative Var        0.21 0.38 0.48 0.58
## Proportion Explained  0.36 0.29 0.18 0.17
## Cumulative Proportion 0.36 0.65 0.83 1.00
##
## Mean item complexity =  1.6
## Test of the hypothesis that 4 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.11
##  with the empirical chi square  517.02  with prob <  4.9e-99
##
## Fit based upon off diagonal values = 0.63
```

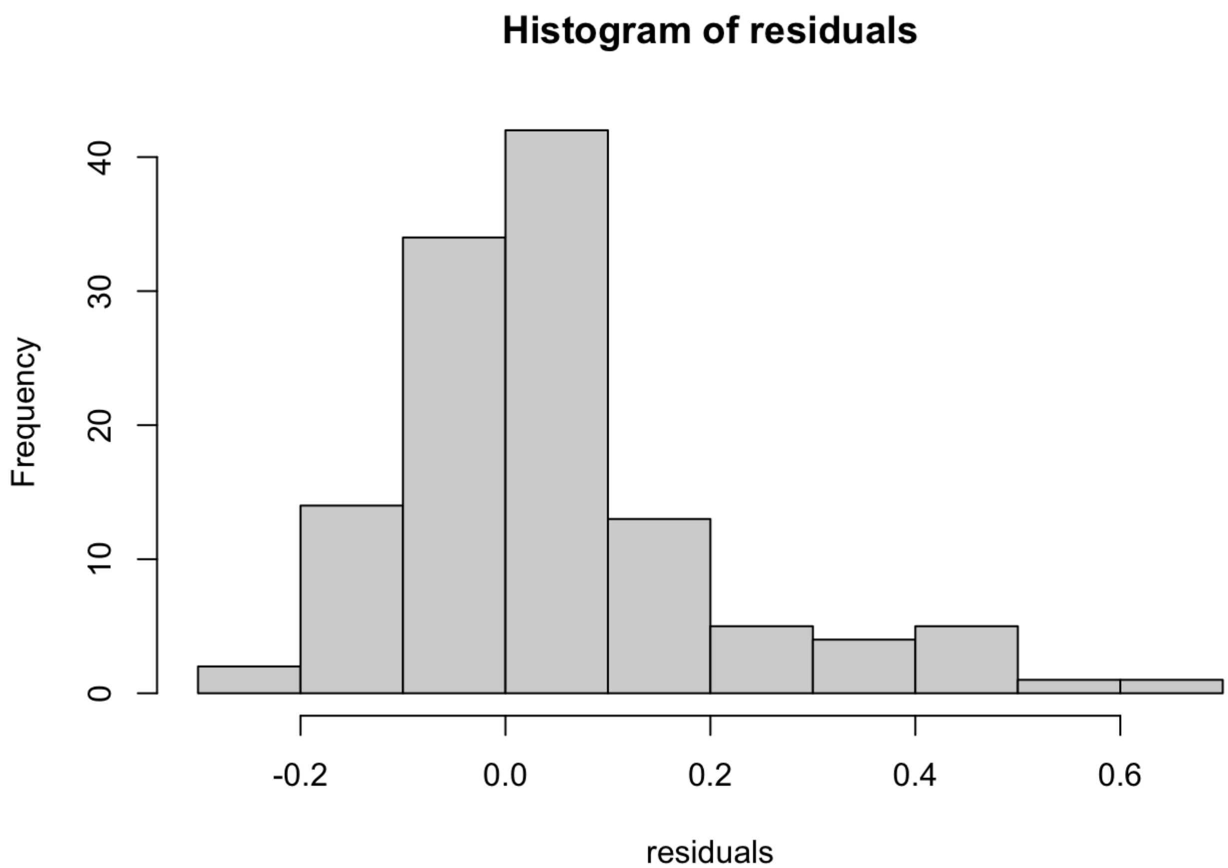
```
# Create a correlation matrix that will be used to calculate residuals below
corMatrix<-cor(spotify_filt4)
corMatrix
```

```
##                    album_num_tracks weeks_on_chart      streams       energy
## album_num_tracks        1.000000000     0.14340875 -0.140458046  0.005125119
## weeks_on_chart          0.143408751     1.00000000 -0.214810843 -0.072905319
## streams                -0.140458046    -0.21481084  1.000000000 -0.005719318
## energy                  0.005125119    -0.07290532 -0.005719318  1.000000000
## loudness               -0.014584432    -0.06289865  0.029992726  0.690468828
## speechiness            -0.063198328     0.01114231  0.009666619 -0.118545724
## acousticness           -0.005414122     0.09596121  0.033120929 -0.456251438
## liveness               -0.073842826     0.01006404  0.001510068  0.093985043
## valence                 0.055555333    -0.02014435  0.010256996  0.396853682
## duration                0.195913736     0.06519036 -0.084765271 -0.017501476
## days_since_release      0.436971596     0.11417871 -0.163981367  0.094207946
##                       loudness  speechiness acousticness     liveness
## album_num_tracks   -0.01458443 -0.063198328 -0.005414122 -0.073842826
## weeks_on_chart     -0.06289865  0.011142305  0.095961215  0.010064038
## streams             0.02999273  0.009666619  0.033120929  0.001510068
## energy              0.69046883 -0.118545724 -0.456251438  0.093985043
## loudness            1.00000000 -0.127223964 -0.336030155  0.057272308
## speechiness        -0.12722396  1.000000000 -0.125593004 -0.063959299
## acousticness       -0.33603015 -0.125593004  1.000000000 -0.064559248
## liveness            0.05727231 -0.063959299 -0.064559248  1.000000000
## valence             0.29548383 -0.043245417 -0.142014729  0.009080177
## duration           -0.10019687 -0.173596792  0.032858407 -0.056293235
## days_since_release -0.02223250 -0.153667885 -0.084987603 -0.029918725
##                        valence    duration days_since_release
## album_num_tracks    0.055555333  0.19591374         0.43697160
## weeks_on_chart     -0.020144348  0.06519036         0.11417871
## streams             0.010256996 -0.08476527        -0.16398137
## energy              0.396853682 -0.01750148         0.09420795
## loudness            0.295483833 -0.10019687        -0.02223250
## speechiness        -0.043245417 -0.17359679        -0.15366789
## acousticness       -0.142014729  0.03285841        -0.08498760
## liveness            0.009080177 -0.05629324        -0.02991872
## valence             1.000000000 -0.14738514         0.05532113
## duration           -0.147385138  1.00000000         0.35937473
## days_since_release  0.055321127  0.35937473         1.00000000
```

```
# Create an object from the correlation matrix and the PCA loading that contains the fac
tor residuals
residuals<-factor.residuals(corMatrix, pca_resid$loadings)
```

Let’s visualize these residuals to confirm that they are normally distributed.

```
hist(residuals)
```



The residuals look mostly normally distributed so we may continue.

1.6.9 Informed PCA

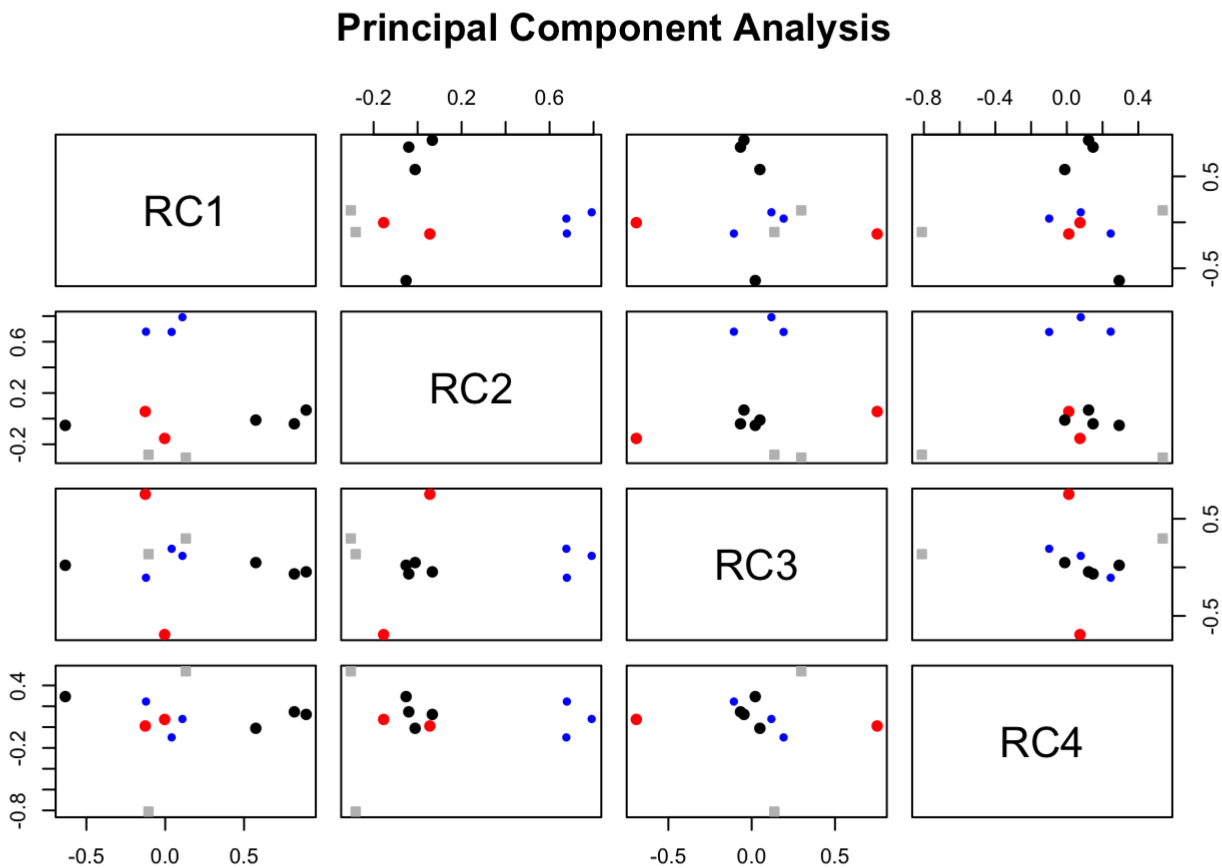
We will now perform our final PCA.

```
pca_final <- principal(spotify_filt4, nfactors = 4, rotate = "promax")

# Print these with formatting to increase legibility
print.psych(pca_final, cut = 0.3, sort = TRUE)
```

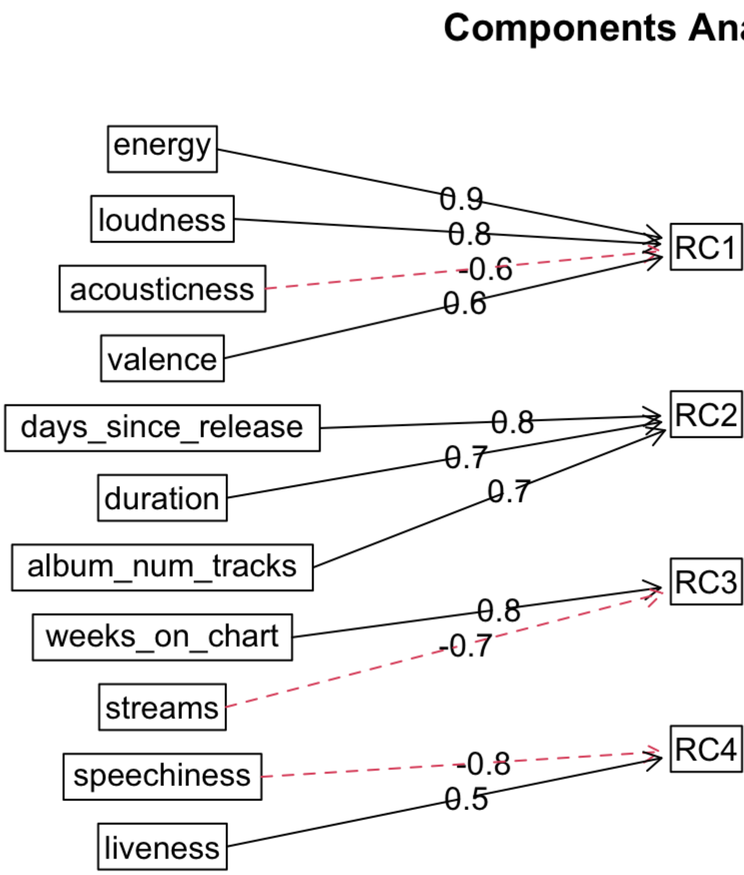
```
## Principal Components Analysis
## Call: principal(r = spotify_filt4, nfactors = 4, rotate = "promax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##                    item   RC1   RC2   RC3   RC4   h2   u2 com
## energy                4  0.89                    0.80 0.20 1.1
## loudness              5  0.82                    0.68 0.32 1.1
## acousticness          7 -0.63                    0.51 0.49 1.4
## valence               9  0.57                    0.34 0.66 1.0
## days_since_release   11        0.79              0.66 0.34 1.1
## duration             10        0.68              0.53 0.47 1.4
## album_num_tracks      1        0.68              0.53 0.47 1.2
## weeks_on_chart        2              0.75        0.59 0.41 1.1
## streams               3             -0.69        0.52 0.48 1.1
## speechiness           6                   -0.81 0.72 0.28 1.3
## liveness              8       -0.30        0.54 0.50 0.50 2.4
##
##                        RC1  RC2  RC3  RC4
## SS loadings           2.26 1.75 1.23 1.13
## Proportion Var        0.21 0.16 0.11 0.10
## Cumulative Var        0.21 0.36 0.48 0.58
## Proportion Explained  0.35 0.27 0.19 0.18
## Cumulative Proportion 0.35 0.63 0.82 1.00
##
##  With component correlations of
##        RC1   RC2  RC3   RC4
## RC1  1.00 -0.03 0.04 -0.05
## RC2 -0.03  1.00 0.06 -0.06
## RC3  0.04  0.06 1.00  0.02
## RC4 -0.05 -0.06 0.02  1.00
##
## Mean item complexity =  1.3
## Test of the hypothesis that 4 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.11
##  with the empirical chi square  517.02  with prob <  4.9e-99
##
## Fit based upon off diagonal values = 0.63
```

Let's visualize this analysis through a PCA plot and a factor loading graph.

```
plot(pca_final)
```




```
fa.diagram(pca_final)
```



1.6.10 Collect factor scores

Finally, lets gather these factor loadings into a data frame that we can use in further analysis.

```
# Create tibble from PCA scores and rename columns
pca_final_scores <- as_tibble(pca_final$scores) %>%
  rename(Vibe = RC1, ListenTime = RC2, RecentPopularity = RC3, AudienceInteraction = RC
4)

# Pull out the outcome variable from our original data
outcome <- spotify_final_outcome %>%
  select(rank)

# bind outcome variable to PCA
pca_scores_outcome <- bind_cols(pca_final_scores, outcome)
```

```
pca_scores_outcome
```

```
## # A tibble: 395 × 5
##       Vibe ListenTime RecentPopularity AudienceInteraction  rank
##      <dbl>      <dbl>            <dbl>               <dbl> <dbl>
##  1 -0.690      -1.25           -3.77               -0.216     2
##  2 -0.0267     -1.82           -2.23                0.884     3
##  3 -3.50       -0.559          -3.16                1.69      4
##  4 -1.05       -1.03           -1.95                1.65      7
##  5  0.227      -1.13           -1.71                0.147     8
##  6 -1.09       -0.668          -1.14                1.20     10
##  7  1.19       -0.896          -1.38               -1.47     12
##  8 -0.645       0.478          -1.42                0.246    13
##  9 -0.351      -0.508          -0.919               0.196    16
## 10 -2.03       -0.218          -1.57                1.10     17
## # … with 385 more rows
```

```
# Write these scores out to a file for easier inspection
write.csv(pca_scores_outcome,"pca_scores.csv", row.names=FALSE)
```

1.6.11 PCA Discussion

This component analysis is very interesting in the way that components were grouped. Component one makes logical sense in the grouping of variables. All but acousticness are positively correlated which allows us to group these features into whether the song is upbeat/happy. We will classify this as the song's vibe.

The second component is less clear in the way the variables are grouped. One would think that days since release would have no relation to duration or the number of tracks on the album. Still, these variables can be generally grouped into the category of music length because it combines how long the songs are with how many songs are on the album as well as the time since release.

The third and fourth components are much more clear in their relation to each other. Weeks on chart and streams make sense to be related as a song on the top charts for more weeks would likely have more streams. We can group these features into Recent Popularity. The last component combines speechiness and liveness. It is likely that live recordings of songs would have more moments where the artist speaks to the audience. As such, these features can be combined into Audience Interaction.

These components do a decent job explaining the variance in our data. The components explain 21, 17, 11, and 10 percent of the variance respectively. Collectively, all four components explain 58% of the variance present in our data.

Using this component analysis, we can now perform a regression analysis on this data using the selected components.

1.7 Linear Regression

We will now conduct a linear regression on the data using the composite features we created in the principal component analysis above. Before moving forward, let's recap the underlying features baked into these new variables.

- Vibe - energy, loudness, acousticness, valence
- ListenTime - days_since_release, duration, album_num_tracks
- RecentPopularity - weeks_on_chart, streams

- AudienceInteraction - speechiness, liveness

1.7.1 Get descriptives (mean/SD) for numeric predictor variables

Let's ensure that the principal components created above are scaled correctly.

```
psych::describe(pca_scores_outcome)
```

```
##                     vars   n   mean    sd median trimmed   mad   min    max
## Vibe                   1 395   0.00  1.00   0.09    0.06  0.97 -3.81   2.20
## ListenTime             2 395   0.00  1.00  -0.15   -0.10  0.73 -1.98   6.16
## RecentPopularity       3 395   0.00  1.00  -0.09   -0.01  0.68 -4.20   3.25
## AudienceInteraction    4 395   0.00  1.00   0.11    0.03  0.87 -3.21   3.29
## rank                   5 395 100.22 57.89 100.00  100.15 74.13  1.00 200.00
##                      range  skew kurtosis   se
## Vibe                  6.01 -0.70     0.90 0.05
## ListenTime            8.15  2.41    11.34 0.05
## RecentPopularity      7.45 -0.24     2.76 0.05
## AudienceInteraction   6.50 -0.28     0.77 0.05
## rank                199.00  0.00    -1.21 2.91
```

Great! All of the variables have a mean of 0 and standard deviation of 1.

1.7.2 Correlation among variables for multi-collinearity

Let's find the correlation between all numeric features in our data.

```
cor(pca_scores_outcome)
```

```
##                            Vibe  ListenTime RecentPopularity
## Vibe                 1.00000000 -0.03048685       0.03706681
## ListenTime          -0.03048685  1.00000000       0.05855518
## RecentPopularity     0.03706681  0.05855518       1.00000000
## AudienceInteraction -0.05264242 -0.05606038       0.02443141
## rank                 0.02825924  0.19316460       0.53253230
##                     AudienceInteraction       rank
## Vibe                        -0.05264242 0.02825924
## ListenTime                  -0.05606038 0.19316460
## RecentPopularity             0.02443141 0.53253230
## AudienceInteraction          1.00000000 0.04977253
## rank                         0.04977253 1.00000000
```

None of the variables clearly exhibit multi-collinearity between them so we will not remove any of them for now.

1.7.3 Split into train and test sets

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
set.seed(1234)

data_final <- pca_scores_outcome %>%
  mutate(id = row_number()) # create an id field so that it is easy to anti join for the
test set

train <- data_final %>%
  sample_frac(0.8)

# Create testing set by removing all rows from train set from the original data and remo
ve ID column
test <- anti_join(data_final, train, by="id") %>%
  select(-id)

# Remove the ID column
train <- train %>%
  select(-id)
```

1.7.4 Build linear model

We will now calculate a linear regression on our training data.

```
spotify_lm <- lm(rank~., train)
summary(spotify_lm)
```

```
##
## Call:
## lm(formula = rank ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.167 -37.235  -5.842  37.572 120.086
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           99.292      2.667  37.225  < 2e-16 ***
## Vibe                   2.089      2.775   0.753 0.451975
## ListenTime             9.271      2.595   3.573 0.000408 ***
## RecentPopularity      31.633      2.864  11.045  < 2e-16 ***
## AudienceInteraction    0.609      2.697   0.226 0.821496
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 47.23 on 311 degrees of freedom
## Multiple R-squared:  0.3074, Adjusted R-squared:  0.2985
## F-statistic: 34.51 on 4 and 311 DF,  p-value: < 2.2e-16
```

The most significant variables appear to be ListenTime and RecentPopularity. Keeping that in mind, let’s remove variables with high VIF scores.

1.7.5 Checking for VIF

We will now check that the VIF scores for each variable are below 5.

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:psych':
##
##     logit
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
car::vif(spotify_lm)
```

```
##                Vibe          ListenTime    RecentPopularity AudienceInteraction 
##            1.003310            1.006310            1.001980            1.005252
```

Since the VIF score for each variable is below 5, we will leave them all in our regression. If we had any VIF scores greater than 5, we would have removed them one by one from our model starting with the highest VIF score (provided that wasn't our most significant variable).

Now that we have accounted for VIF, let's check for supression effects.

1.7.6 Check for supression effects

```
cor(pca_scores_outcome)
```

```
##                            Vibe  ListenTime RecentPopularity
## Vibe                 1.00000000 -0.03048685       0.03706681
## ListenTime          -0.03048685  1.00000000       0.05855518
## RecentPopularity     0.03706681  0.05855518       1.00000000
## AudienceInteraction -0.05264242 -0.05606038       0.02443141
## rank                 0.02825924  0.19316460       0.53253230
##                     AudienceInteraction       rank
## Vibe                        -0.05264242 0.02825924
## ListenTime                  -0.05606038 0.19316460
## RecentPopularity             0.02443141 0.53253230
## AudienceInteraction          1.00000000 0.04977253
## rank                         0.04977253 1.00000000
```

```
summary(spotify_lm)
```

```
##
## Call:
## lm(formula = rank ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.167 -37.235  -5.842  37.572 120.086
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           99.292      2.667  37.225  < 2e-16 ***
## Vibe                   2.089      2.775   0.753 0.451975
## ListenTime             9.271      2.595   3.573 0.000408 ***
## RecentPopularity      31.633      2.864  11.045  < 2e-16 ***
## AudienceInteraction    0.609      2.697   0.226 0.821496
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 47.23 on 311 degrees of freedom
## Multiple R-squared:  0.3074, Adjusted R-squared:  0.2985
## F-statistic: 34.51 on 4 and 311 DF,  p-value: < 2.2e-16
```

The direction of the correlation between each of the features and the outcome variable is the same in our regression so supression effects are not present. Finally, let's remove any insignificant variables from our model.

1.7.7 Removing insignificant variables

```
summary(spotify_lm)
```


```
##
## Call:
## lm(formula = rank ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.167 -37.235  -5.842  37.572 120.086
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           99.292      2.667  37.225  < 2e-16 ***
## Vibe                   2.089      2.775   0.753 0.451975
## ListenTime             9.271      2.595   3.573 0.000408 ***
## RecentPopularity      31.633      2.864  11.045  < 2e-16 ***
## AudienceInteraction    0.609      2.697   0.226 0.821496
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 47.23 on 311 degrees of freedom
## Multiple R-squared:  0.3074, Adjusted R-squared:  0.2985
## F-statistic: 34.51 on 4 and 311 DF,  p-value: < 2.2e-16
```

```
# Remove AudienceInteraction because it the most insignificant
spotify_lm2 <- lm(rank~., train[, c(1:3,5)])
summary(spotify_lm2)
```

```
##
## Call:
## lm(formula = rank ~ ., data = train[, c(1:3, 5)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.388 -37.494  -5.915  37.855 120.321
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        99.281      2.663  37.284  < 2e-16 ***
## Vibe                2.067      2.768   0.747 0.455888
## ListenTime          9.238      2.586   3.572 0.000411 ***
## RecentPopularity   31.653      2.858  11.074  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 47.15 on 312 degrees of freedom
## Multiple R-squared:  0.3073, Adjusted R-squared:  0.3007
## F-statistic: 46.14 on 3 and 312 DF,  p-value: < 2.2e-16
```

```
# Remove Vibe because it is still insignificant
spotify_lm3 <- lm(rank~., train[, c(2:3,5)])
summary(spotify_lm3)
```

```
##
## Call:
## lm(formula = rank ~ ., data = train[, c(2:3, 5)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.937 -35.641  -5.943  36.235 118.340
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        99.403      2.656  37.426  < 2e-16 ***
## ListenTime          9.152      2.582   3.544 0.000454 ***
## RecentPopularity   31.661      2.856  11.085  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 47.12 on 313 degrees of freedom
## Multiple R-squared:  0.3061, Adjusted R-squared:  0.3016
## F-statistic: 69.03 on 2 and 313 DF,  p-value: < 2.2e-16
```

AudienceInteraction was the most insignificant variable so it was removed first. After rerunning the regression, Vibe was still insignificant so it was removed as well. This leaves us ith a regression using two variables.

1.7.8 Recheck VIF and supression effects for new model

```
car::vif(spotify_lm3)
```

```
##       ListenTime RecentPopularity
##         1.001007         1.001007
```

```
summary(spotify_lm3)
```

```
##
## Call:
## lm(formula = rank ~ ., data = train[, c(2:3, 5)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.937 -35.641  -5.943  36.235 118.340
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        99.403      2.656  37.426  < 2e-16 ***
## ListenTime          9.152      2.582   3.544 0.000454 ***
## RecentPopularity   31.661      2.856  11.085  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 47.12 on 313 degrees of freedom
## Multiple R-squared:  0.3061, Adjusted R-squared:  0.3016
## F-statistic: 69.03 on 2 and 313 DF,  p-value: < 2.2e-16
```

All variables still have VIF values below 5 and there are no problems with supression effect so we may proceed with our analysis.

1.7.9 Interpretting the model

Recall that in our case, increasing rank is a negative outcome if an artist wants their song to be at the top of the charts. Since our variable are scaled, we cannot derive the exact meaning of these coefficients. However, we can ascertain that RecentPopularity has a far stronger effect on rank than ListenTime.

```
# Update the training and test sets for future use
train2 <- train %>%
  select(-Vibe, -AudienceInteraction)

test2 <- test %>%
  select(-Vibe, -AudienceInteraction)
```

1.7.10 Homoscedasticity check

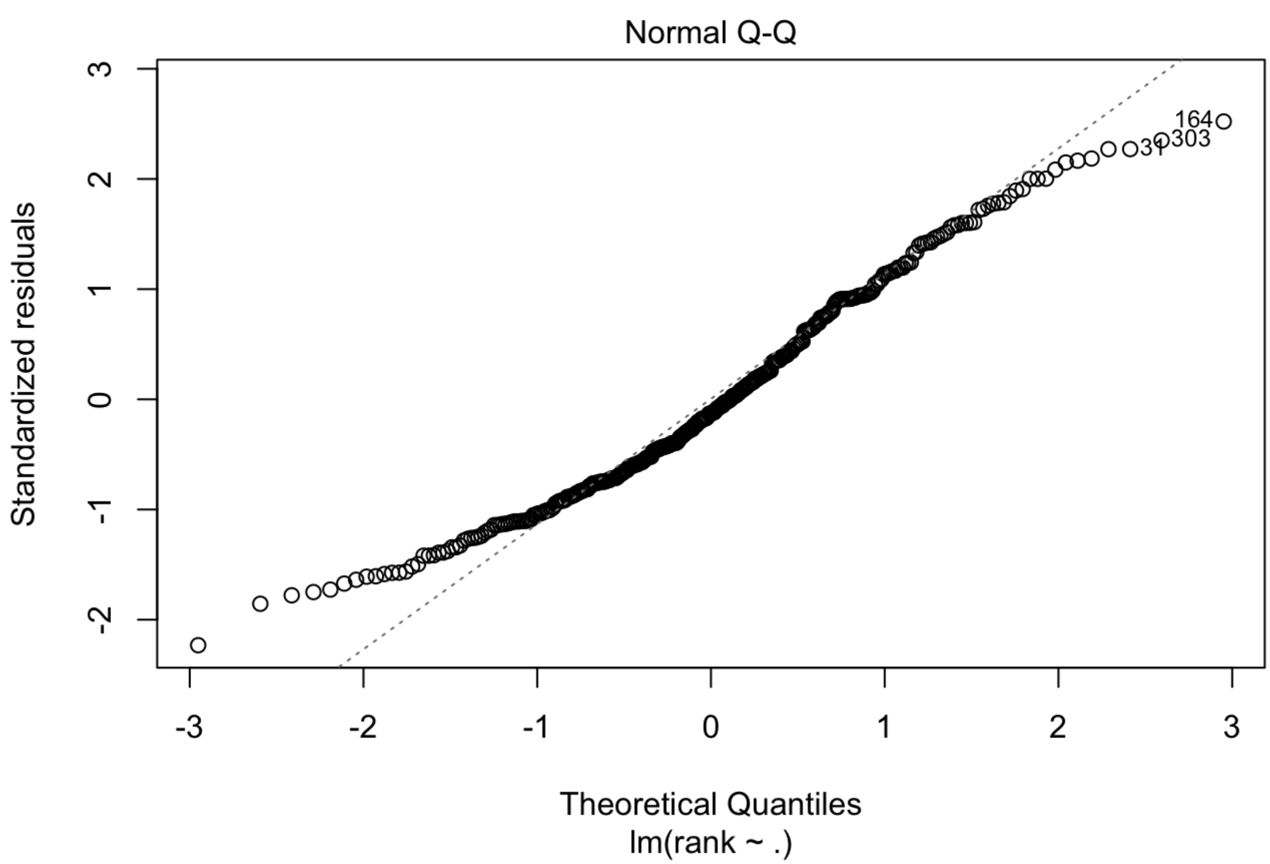
Now all of our variables are significant. We will now check for homoscedasticity through a series of tests.

```
# Shapiro test
shapiro.test(residuals(spotify_lm3))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(spotify_lm3)
## W = 0.97473, p-value = 2.35e-05
```

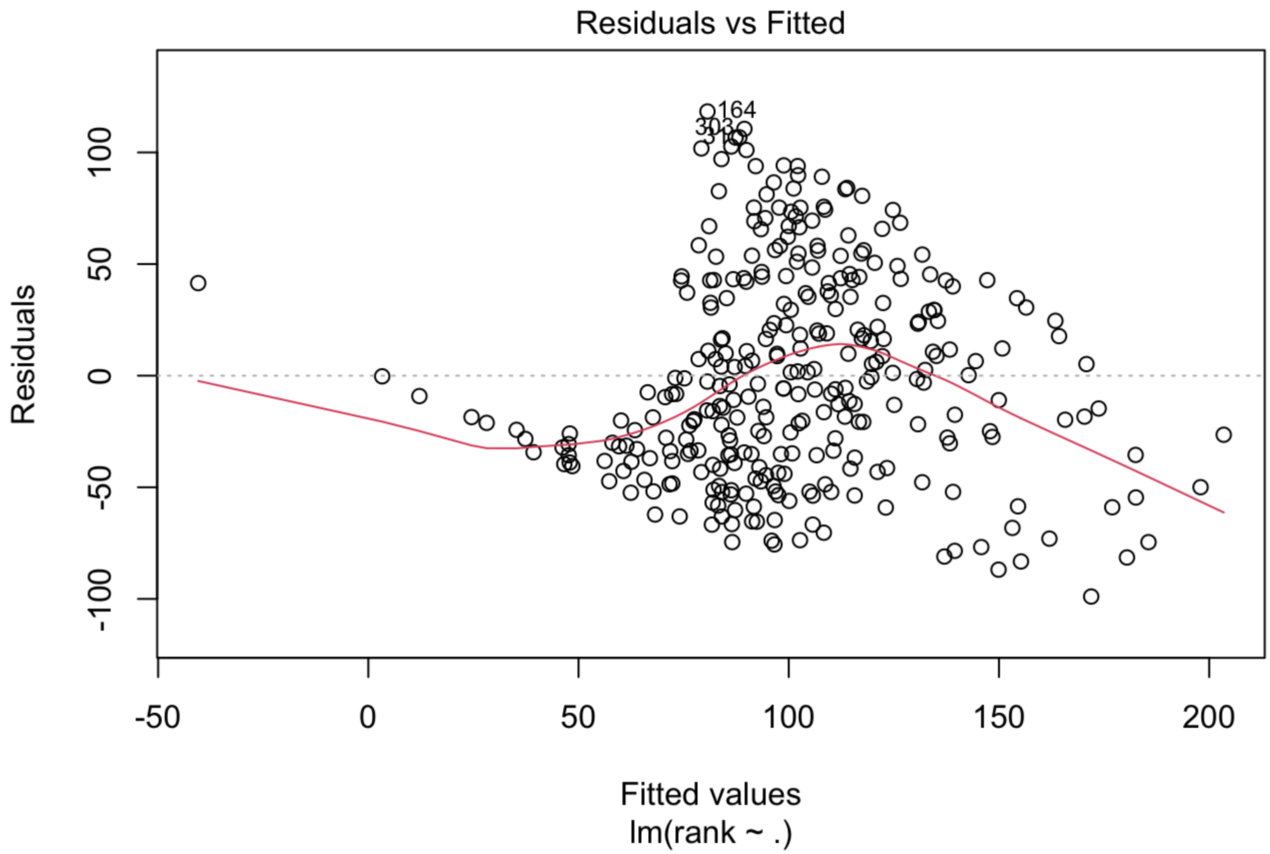
Since we have a p-value of 2.35e-05 which is far smaller than alpha, we reject the null hypothesis meaning that the residuals likely do not follow a normal distribution.

```
# Q-Q plot
plot(spotify_lm3, which=2)
```



Since the points almost all fall along a straight line, we can assume the data is normally distributed.

```
# Residuals vs fitted plot
plot(spotify_lm3, which=1)
```

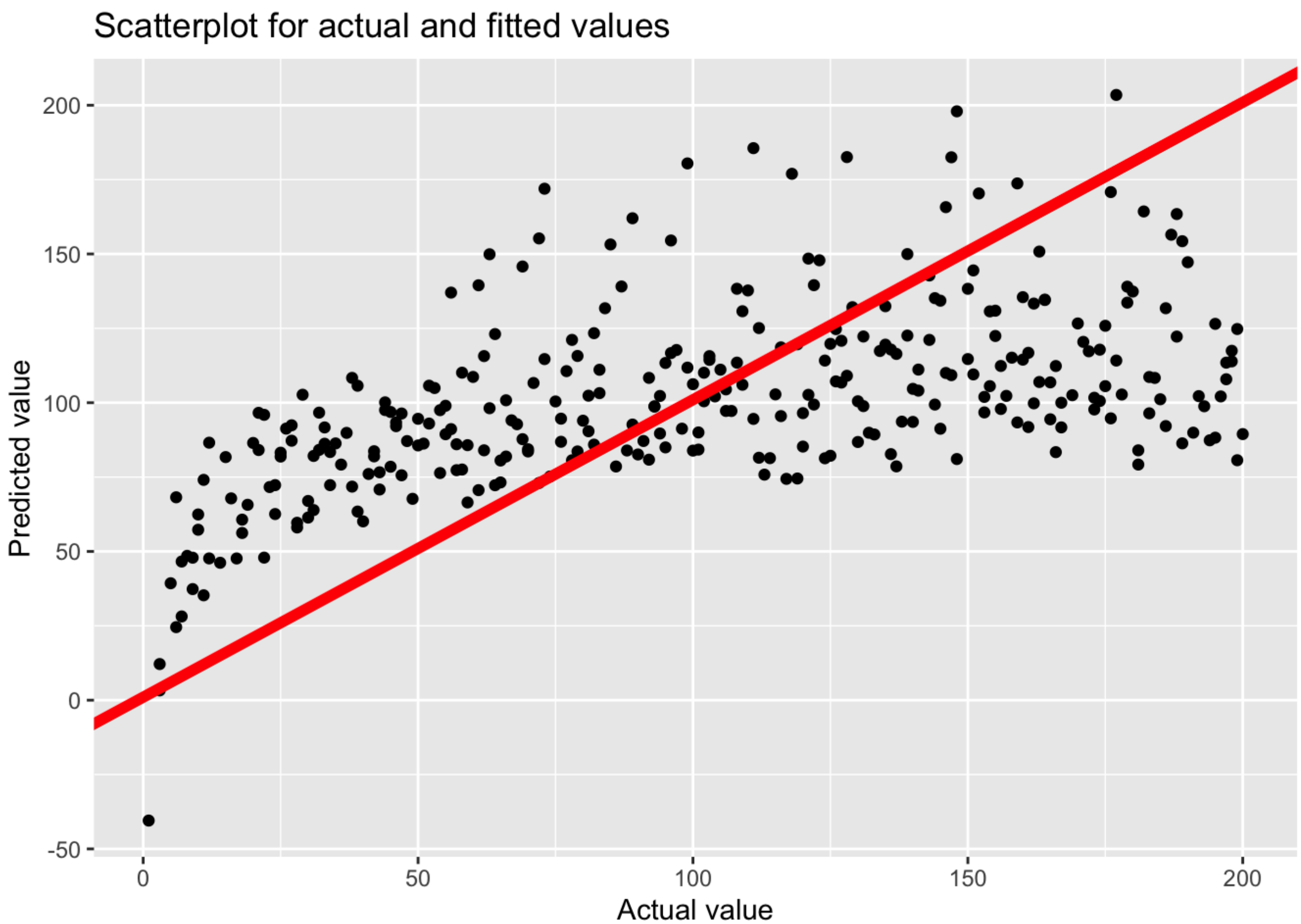


In this plot, the points appear to be located randomly about the 0 line in no distinct pattern. Thus, we can reasonably assume that the relationship between the data is linear.

1.7.11 Visualizing model fit

Let’s visualize how well our model did by comparing our fitted values to the actual values on a scatter plot.

```
actual <- train2$rank
fitted <- unname(spotify_lm3$fitted.values)
act_fit <- cbind.data.frame(actual, fitted)

ggplot(act_fit, aes(x = actual, y = fitted)) +
  geom_point() +
  xlab("Actual value") +
  ylab("Predicted value") +
  ggtitle("Scatterplot for actual and fitted values") +
  geom_abline(intercept = 1,
              slope = 1,
              color = "red",
              size = 2)
```



Let's quantify that error using the root mean squared error (RMSE). Since this metric is somewhat meaningless on its own, let's also find the RMSE of a sample where the rankings are completely randomly assigned and compare it to the RMSE of our fitted values.

```
# Calculate RMSE
sqrt(mean((actual - fitted) ^ 2))
```

```
## [1] 46.89676
```

```
# Compare to completely random distribution
random_guess <- sample(1:200, length(actual), replace=TRUE)
sqrt(mean((actual - random_guess) ^ 2))
```

```
## [1] 82.07846
```

The RMSE of the random distribution is almost twice as large as the fitted from our model. That suggests that our model has a tangible ability to predict the rank of a song on Spotify. Let's now fit this model to our test data set to check our findings.

```
test_actual <- test2$rank
test_fit <- unname(predict(spotify_lm3, newdata = test2))
sqrt(mean((test_actual - test_fit) ^ 2))
```

```
## [1] 52.42269
```

As expected, the RMSE for our fitted data from our test set is slightly larger than that of the training set. This is because our model is built to fit the training data. However, the RMSE for this test data is still considerably lower than the random guesser which is promising.

Let's also compare the R2 values of our training and testing data.

```
# train R2
cor(actual, fitted) ^ 2
```

```
## [1] 0.3060806
```

```
# test R2
cor(test_actual, test_fit) ^ 2
```

```
## [1] 0.3235118
```

Interestingly, the model fits our test data slightly better than our training data. Our model explains 32.35% of the variance in our testing data.

1.8 Discussion

I found the outcome of this regression to be surprising. Before building the model, I would have expected the "vibe" of a song to have a significant impact on a songs ranking due to the trends in modern popular music. However, the effect of this was likely dampened because all of the songs in our data set were in the top 200 list.

Furthermore, I was surprised how well our regression predicted the rank of songs. Music is an incredibly subjective and creative art that one would expect could not be quantified and predicted purely by numbers. However, our model does a relatively impressive job fitting our data. The model boasts a somewhat low RMSE and explains 32.4% of the variance in the data. Moreover, the tests for homoscedasticity lead us to confirm that a linear model is valid for modeling the relationships between the features of a song and its rank.

1.9 Limitations

Though the findings of this study were interesting, it is important to acknowledge the limitations that faced this analysis. First, we had a rather small data set to work with. In order to control for differences in time of year and song preference, the same week of the year was used for all of the data. However, since the data set only contained two years worth of data, we only had two observations of each rank type. In future studies, it would be likely informative if one could acquire more data to work with. Second, this study compares popular songs against other popular songs. In other words, only songs that were in the top 200 were a part of this data set. This is potentially an issue because it is likely that these songs had a lot in common because we know they were all popular. This is likely why many of the variables we used in our regression were found to be insignificant.

1.10 Future Studies

In the future, it would be interesting to take this study further and down other avenues. For example, if we acquired a larger data set with non top charts songs, one could evaluate what elements make up more popular songs versus songs that aren't popular at all. It would also be interesting to pair this analysis with a sentiment analysis of tweets related to the artist to see if how the public perceives an artist affects a song's ranking.