Acquire spotify data

Connor French

Setup

Load packages. There is a tiny bit of python that needs to be done to obtain the billboard records.

```
library(spotifyr)
library(tidyverse)
```

```
## -- Attaching packages ----------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.4.0
```

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

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(reticulate)
library(here)
```

```
## here() starts at /Users/connorfrench/Dropbox/Old_Mac/School_Stuff/CUNY/GCDI/r_data_analysis_2021
```

```
# set up python for billboard api
use_condaenv(condaenv = "py37",
             conda = "/Users/connorfrench/opt/miniconda3/bin/conda",
             required = TRUE)
```

Authenticate account. See here to set up a Spotify Dev account first. From there you will be able to copy-paste your client ID and client secret into these functions.

```
Sys.setenv(SPOTIFY_CLIENT_ID = "alphanumericstring")
Sys.setenv(SPOTIFY_CLIENT_SECRET = "alphanumericstring")

access_token <- get_spotify_access_token()
```

Get used to API

I'm just playing around with the introduction from the spotifyr home page to get used to the API.

Get most recently played tracks.

```
recently_played <- get_my_recently_played(limit = 5) %>%
  mutate(artist.name = map_chr(track.artists, function(x) x$name[1]),
         played_at = as_datetime(played_at)) %>%
  select(track.name, artist.name, track.album.name, played_at)
```

```
## Auto-refreshing stale OAuth token.
```

```
recently_played
```

```
##                                                                                   track.name
## 1                                                                                    Nardis
## 2 The King Of 52nd Street (Chan's Declaration) - Vocal Version Of "Scrapple From The Apple"
## 3                                                                                     Crush
## 4                                                     It Never Entered My Mind - Remastered
## 5                                                                              Besame Mucho
##            artist.name              track.album.name
## 1     Bill Evans Trio                  Explorations
## 2       Melody Gardot The Passion Of Charlie Parker
## 3        Bob Reynolds                       Quartet
## 4 Miles Davis Quartet          Miles Davis (Vol. 2)
## 5        Dave Brubeck                    Love Songs
##              played_at
## 1 2021-02-02 16:36:25
## 2 2021-02-02 16:30:35
## 3 2021-02-02 16:26:54
## 4 2021-02-02 16:20:36
## 5 2021-02-02 16:16:32
```

Here are my top tracks of all time, according to Spotify (seems to be biased towards recent trends. They don't just count the number of plays and sort).

```
top_long <- get_my_top_artists_or_tracks(type = "tracks",
                                         time_range = "long_term", limit = 50) %>%
  mutate(artist.name = map_chr(artists, function(x) x$name[1]))


top_long %>% select(name, artist.name, album.name, popularity) %>% glimpse()
```

```
## Rows: 50
## Columns: 4
## $ name        <chr> "Oxbow", "Belleville", "Fire", "A Colossal Wr...
## $ artist.name <chr> "Waxahatchee", "Knocked Loose", "Waxahatchee"...
## $ album.name  <chr> "Saint Cloud", "A Different Shade of Blue", "...
## $ popularity  <int> 51, 46, 62, 50, 56, 31, 57, 35, 48, 48, 22, 4...
```

Top medium tracks (from the last six months).

```
top_medium <- get_my_top_artists_or_tracks(type = "tracks",
                                           time_range = "medium_term", limit = 50) %>%
  mutate(artist.name = map_chr(artists, function(x) x$name[1]))


top_medium %>% select(name, artist.name, album.name, popularity) %>% glimpse()
```

```
## Rows: 50
## Columns: 4
## $ name        <chr> "A Colossal Wreck", "Oxbow", "two reverse", "...
## $ artist.name <chr> "Every Time I Die", "Waxahatchee", "Adrianne ...
## $ album.name  <chr> "A Colossal Wreck // Desperate Pleasures", "S...
## $ popularity  <int> 50, 51, 31, 30, 31, 22, 48, 49, 46, 70, 37, 2...
```

Top recent tracks (last week).

```
top_recent <- get_my_top_artists_or_tracks(type = "tracks",
                                           time_range = "short_term", limit = 50) %>%
  mutate(artist.name = map_chr(artists, function(x) x$name[1]))


top_recent %>% select(name, artist.name, album.name, popularity) %>% glimpse()
```

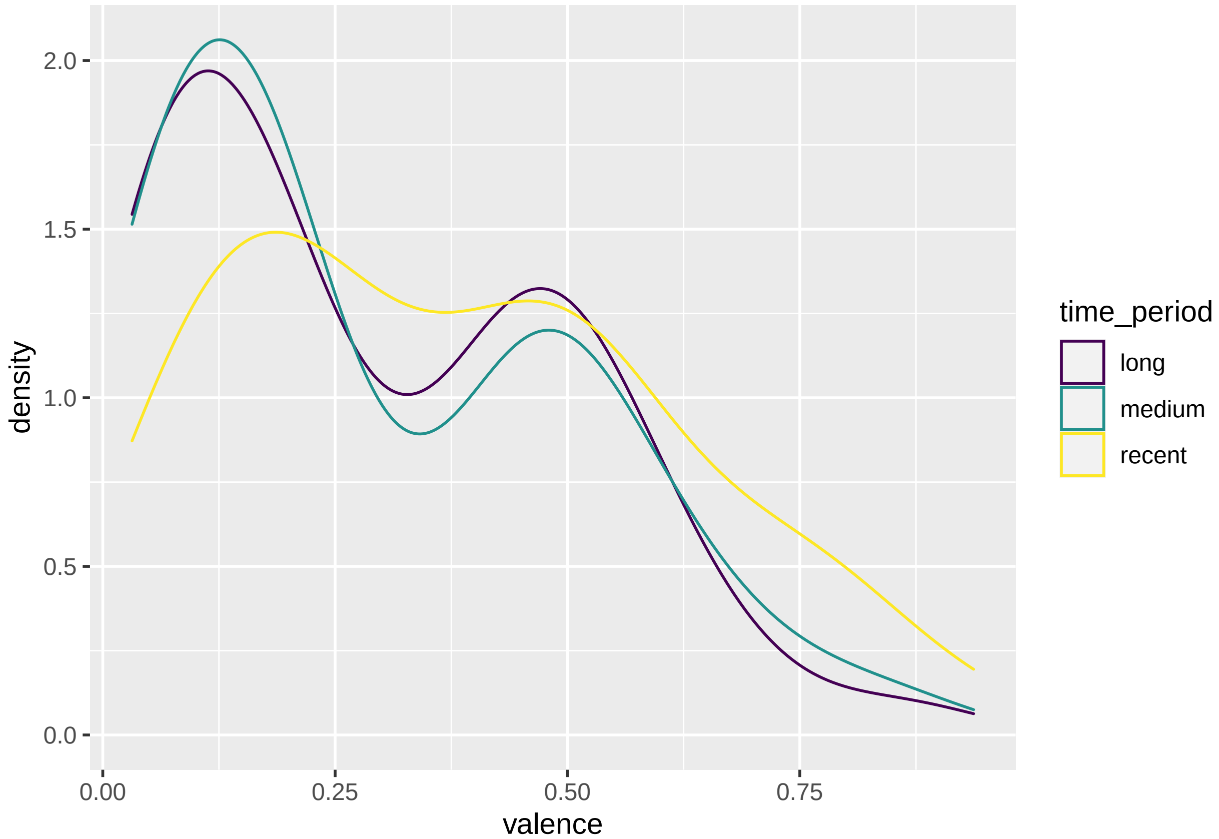
```
## Rows: 50
## Columns: 4
## $ name        <chr> "two reverse", "Freddie Freeloader", "Freddie...
## $ artist.name <chr> "Adrianne Lenker", "Dexter Gordon", "Miles Da...
## $ album.name  <chr> "songs", "Something Different", "Kind Of Blue...
## $ popularity  <int> 52, 1, 52, 41, 50, 37, 60, 51, 38, 61, 48, 25...
```

Get track info for each track from all time periods

```
features_long <- map_df(top_long$id, get_track_audio_features)

features_long_df <- left_join(top_long, features_long, by = "id") %>%
  mutate(time_period = "long")

features_medium <- map_df(top_medium$id, get_track_audio_features)

features_medium_df <- left_join(top_medium, features_medium, by = "id") %>%
  mutate(time_period = "medium")

features_recent <- map_df(top_recent$id, get_track_audio_features)

features_recent_df <- left_join(top_recent, features_recent, by = "id") %>%
  mutate(time_period = "recent")

features_all_df <- bind_rows(
  features_long_df,
  features_medium_df,
  features_recent_df
)
```

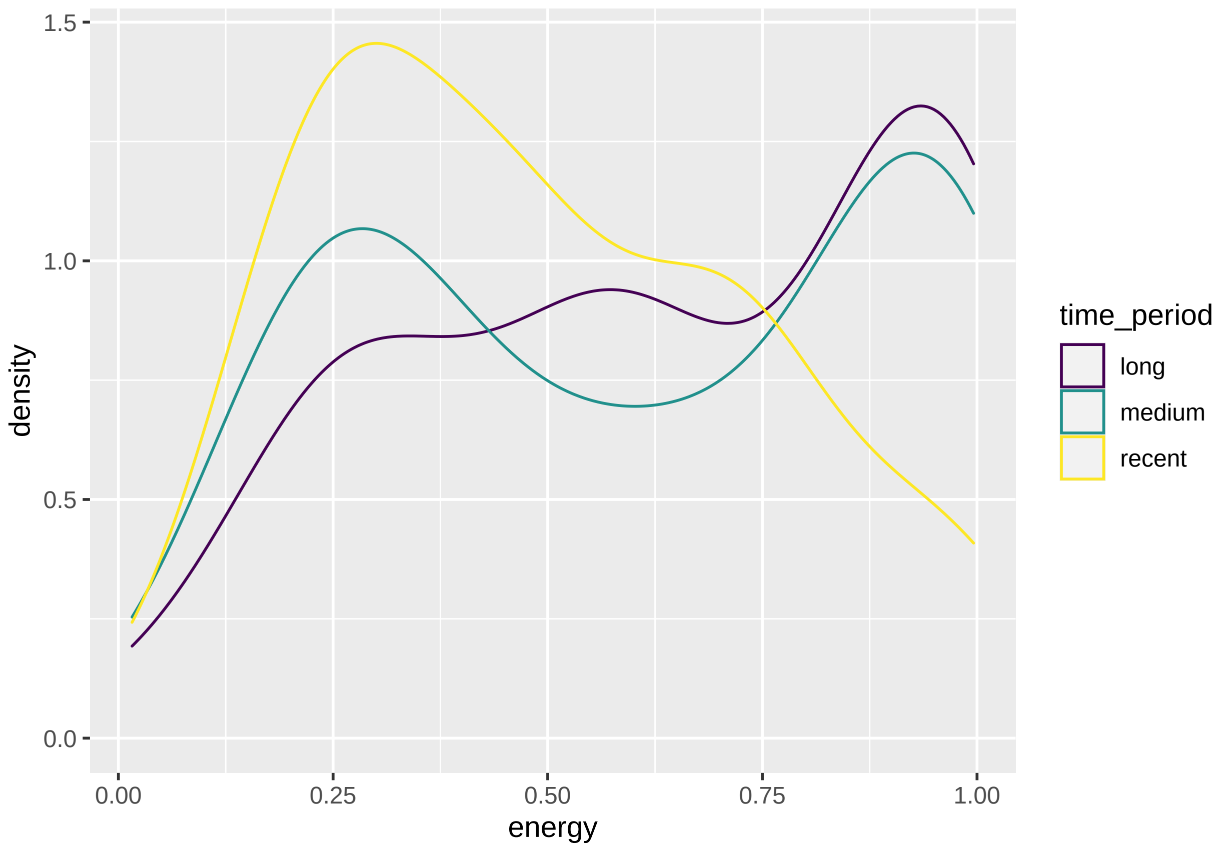
Looks like I've been listening to happier music lately compared to what I usually listen to! Valence is Spotify's measure of the "positivity" of a track.

```
ggplot(data = features_all_df, aes(x = valence, color = time_period)) +
  geom_density() +
  scale_color_viridis_d()
```



The energy level has definitely receded- makes sense. I've listened to a lot of Adrienne Lenker and Waxahatchee, and not much heavier music.

```
ggplot(data = features_all_df, aes(x = energy, color = time_period)) +
  geom_density() +
  scale_color_viridis_d()
```



Get Billboard Top 100

Do a little python to retrieve the billboard hot 100 charts over the last few years

Setup Load billboard package

```
import billboard
```

Get used to the API I'm doing the intro from the billboard.py Python API to get used to it.

```
chart = billboard.ChartData("hot-100")
chart.title
```

```
## 'The Hot 100'
```

```
song = chart[0]
song.title
```

```
## 'Drivers License'
```

```
song.artist
```

```
## 'Olivia Rodrigo'
```

```
song.weeks
```

```
## 3
```

Get the entries and turn into a dataframe with some processing

```
chart_entries <- py$chart$entries

# have to convert to characters before converting to a dataframe
chart_entries_vec <- map_chr(chart_entries, as.character)


chart_entries_df <- enframe(chart_entries_vec,
                            name = "index",
                            value = "entry") %>%
  # every entry splits the track and artist name with " by "
  mutate(track = str_split_fixed(entry, " by ", 2)[,1] %>% str_remove_all("\\'"),
         artist = str_split_fixed(entry, " by ", 2)[,2])

chart_entries_df
```

```
## # A tibble: 100 x 4
##    index entry                       track      artist
##    <int> <chr>                       <chr>      <chr>
##  1     1 'Drivers License' by Olivi~ Drivers L~ Olivia Rodrigo
##  2     2 'Mood' by 24kGoldn Featuri~ Mood       24kGoldn Featuring ia~
##  3     3 'Blinding Lights' by The W~ Blinding ~ The Weeknd
##  4     4 '34+35' by Ariana Grande    34+35      Ariana Grande
##  5     5 'Levitating' by Dua Lipa F~ Levitating Dua Lipa Featuring Da~
##  6     6 'Go Crazy' by Chris Brown ~ Go Crazy   Chris Brown & Young T~
##  7     7 'Positions' by Ariana Gran~ Positions  Ariana Grande
##  8     8 'Holy' by Justin Bieber Fe~ Holy       Justin Bieber Featuri~
##  9     9 'Good Days' by SZA          Good Days  SZA
## 10    10 'Bang!' by AJR              Bang!      AJR
## # ... with 90 more rows
```

Acquire data I’m going to get pre-pandemic (mid-March 2019), early pandemic (mid-March, when covid cases were in all 50 states), first peak (04-25, most deaths), and current (so much worse than the others) Billboard 200 songs

```
pre_chart = billboard.ChartData(name="hot-100", date="2019-03-16")

early_chart = billboard.ChartData(name="billboard-200", date="2020-03-14")

first_chart = billboard.ChartData(name="billboard-200", date="2020-04-25")

current_chart = billboard.ChartData(name="billboard-200", date="2020-01-30")
```

Function to convert entries to dataframes.

```r
# chart_entries must be referring to the python object, e.g. py$chart$entries
# covid_period must be a string, e.g. "early"
# not implementing checks because I'm lazy
chart_to_df <- function(chart_entries, covid_period) {
  # have to convert to characters before converting
  chart_entries_vec <- map_chr(chart_entries, as.character)

  # convert to df
  chart_entries_df <- enframe(chart_entries_vec,
                            name = "index",
                            value = "entry") %>%
  mutate(track = str_split_fixed(entry, " by ", 2)[,1] %>% str_remove_all("\\'"),
         artist = str_split_fixed(entry, " by ", 2)[,2],
         time_period = covid_period)

  return(chart_entries_df)
}
```

Convert charts to a data frame

```r
pre_df <- chart_to_df(py$pre_chart$entries, "pre")
early_df <- chart_to_df(py$early_chart$entries, "early")
first_df <- chart_to_df(py$first_chart$entries, "first")
current_df <- chart_to_df(py$current_chart$entries, "current")

charts_df <- bind_rows(pre_df, early_df, first_df, current_df)
```

Get spotify info

Here, I'm getting the audio features for each song from Spotify. I'm going to assume that the spotify search feature is perfect and I'm going to select the first ID returned for each query. I could do a more robust search and quality control, but this is just for a workshop demo, so not looking for precision.

```r
get_id <- function(entry) {
  spotify_return <- search_spotify(q = entry, type = "track")

  if (length(spotify_return$id) > 0) {
    id <- spotify_return$id[1]
  } else {
    id <- NA
  }

  return(id)
}

# this will return NAs for tracks that don't come up in the search
spotify_ids <- map_chr(charts_df$track, get_id)
charts_df_id <- charts_df %>%
  mutate(id = spotify_ids)
```

It looks like the entries that don't return a result are all albums, which aren't what we're interested in, so it's safe to remove them. They're biased toward the early and current time periods, but I don't think that'll be much of a problem.

```
charts_df_id %>% filter(is.na(id)) %>% knitr::kable()
```

index	entry	track	artist	time_period	id
48	'The Kids Are Coming (EP)' by Tones And I	The Kids Are Coming (EP)	Tones And I	early	NA
157	'Moral Of The Story: Chapter 1 (EP)' by Ashe	Moral Of The Story: Chapter 1 (EP)	Ashe	early	NA
164	'Long Live The Kings (EP)' by Calboy	Long Live The Kings (EP)	Calboy	early	NA
189	'Meet The Woo, V. 1 Mixtape' by Pop Smoke	Meet The Woo, V. 1 Mixtape	Pop Smoke	early	NA
81	'The Kids Are Coming (EP)' by Tones And I	The Kids Are Coming (EP)	Tones And I	first	NA
35	'The Kids Are Coming (EP)' by Tones And I	The Kids Are Coming (EP)	Tones And I	current	NA
90	'Grammy 2020 Nominees' by Various Artists	Grammy 2020 Nominees	Various Artists	current	NA
164	'Marshmello: Fortnite Extended Set' by Marshmello	Marshmello: Fortnite Extended Set	Marshmello	current	NA
192	'Believers Never Die, Volume Two: Greatest Hits' by Fall Out Boy	Believers Never Die, Volume Two: Greatest Hits	Fall Out Boy	current	NA

Removing the NAs

```
charts_df_filt <- charts_df_id %>%
  filter(!is.na(id))
```

Now it's time to get the audio features!

```
features <- map_df(charts_df_filt$id, get_track_audio_features)

features_df <- left_join(charts_df_filt, features, by = "id")

# for some reason the get_track_audio_features returns a bunch of duplicates, so I have to filter those
features_df_filt <- features_df[!duplicated(features_df),]
```

Let's clean up the data a bit for the workshop. Columns to remove:

- entry
- id
- uri
- type
- track_href
- analysis_url

I'm also renaming "index" to "rank"

```
features_df_workshop <- features_df_filt %>%
  select(-c(entry, id, uri, type, track_href, analysis_url)) %>%
  rename(rank = index)

glimpse(features_df_workshop)
```

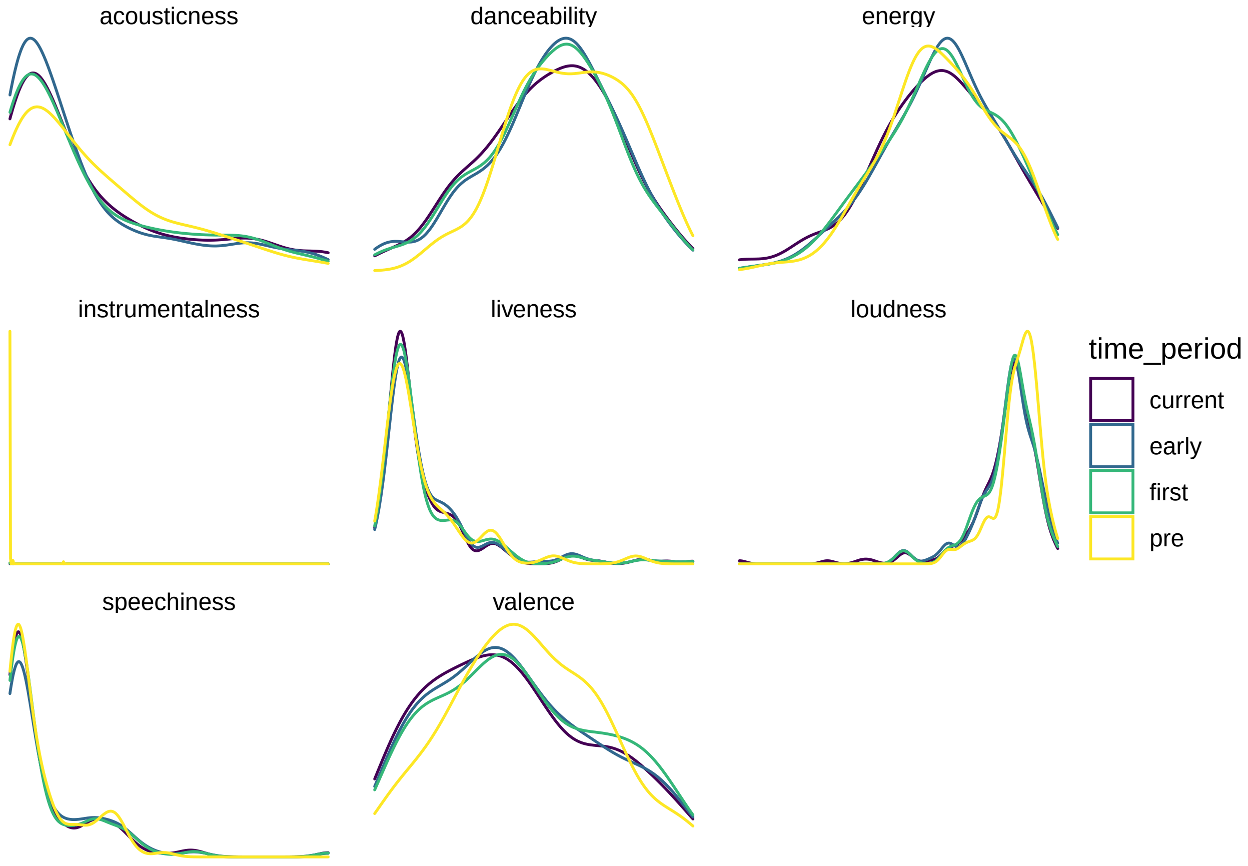
```
##
## -- Column specification ----------------------------------------------
## cols(
##   rank = col_double(),
##   track = col_character(),
##   artist = col_character(),
##   time_period = col_character(),
##   danceability = col_double(),
##   energy = col_double(),
##   key = col_double(),
##   loudness = col_double(),
##   mode = col_double(),
##   speechiness = col_double(),
##   acousticness = col_double(),
##   instrumentalness = col_double(),
##   liveness = col_double(),
##   valence = col_double(),
##   tempo = col_double(),
##   duration_ms = col_double(),
##   time_signature = col_double()
## )
```

```
## Rows: 691
## Columns: 17
## $ rank             <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1...
## $ track            <chr> "Sucker", "7 Rings", "Please Me", "Sunfl...
## $ artist           <chr> "Jonas Brothers", "Ariana Grande", "Card...
## $ time_period      <chr> "pre", "pre", "pre", "pre", "pre", "pre"...
## $ danceability     <dbl> 0.842, 0.778, 0.747, 0.760, 0.752, 0.572...
## $ energy           <dbl> 0.734, 0.317, 0.570, 0.479, 0.488, 0.385...
## $ key              <dbl> 1, 1, 1, 2, 6, 7, 11, 5, 8, 8, 1, 5, 10,...
## $ loudness         <dbl> -5.065, -10.732, -6.711, -5.574, -7.050,...
## $ mode             <dbl> 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0...
## $ speechiness      <dbl> 0.0588, 0.3340, 0.0810, 0.0466, 0.0705, ...
## $ acousticness     <dbl> 0.04270, 0.59200, 0.06420, 0.55600, 0.29...
## $ instrumentalness <dbl> 0.00e+00, 0.00e+00, 0.00e+00, 0.00e+00, ...
## $ liveness         <dbl> 0.1060, 0.0881, 0.0832, 0.0703, 0.0936, ...
## $ valence          <dbl> 0.952, 0.327, 0.650, 0.913, 0.533, 0.323...
## $ tempo            <dbl> 137.958, 140.048, 133.992, 89.911, 136.0...
## $ duration_ms      <dbl> 181027, 178627, 200890, 158040, 201661, ...
## $ time_signature   <dbl> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
```

Quick exploration

Let's see if this is worth making exercises with. Nice! Looks like there are some interesting patterns in the time periods. I wonder what's up with "instrumentalness".

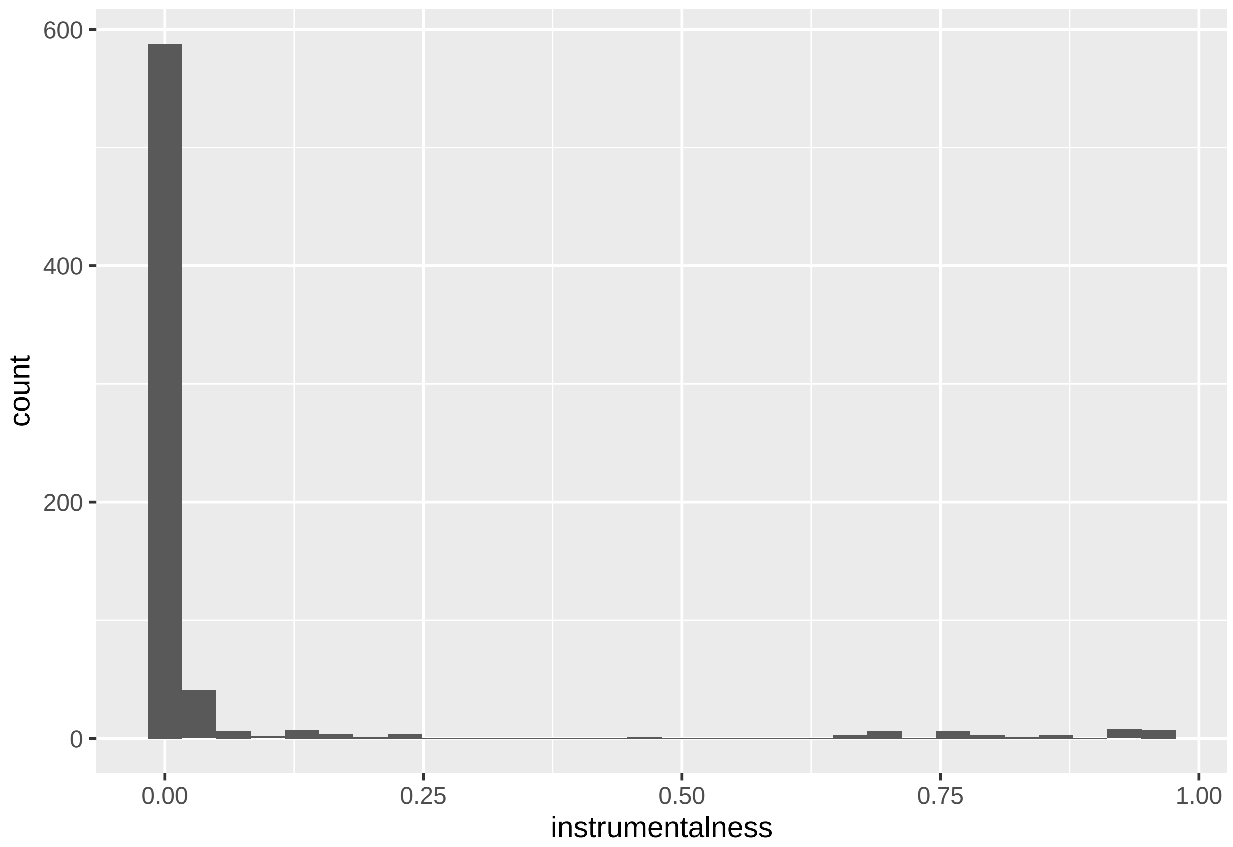
```r
feature_cols <- c("danceability", "energy", "loudness", "speechiness",
                  "acousticness", "instrumentalness", "liveness", "valence")

features_df_workshop %>%
  pivot_longer(cols = all_of(feature_cols), names_to = "feature", values_to = "feature_value") %>%
  ggplot(aes(x = feature_value, color = time_period)) +
  geom_density() +
  scale_color_viridis_d() +
  theme_void() +
  facet_wrap(~feature, scales = "free")
```



Ah, it looks like the vast majority of top 200 songs are predominantly vocal, which makes sense.

```r
ggplot(data = features_df_workshop, aes(x = instrumentalness)) +
  geom_histogram()
```

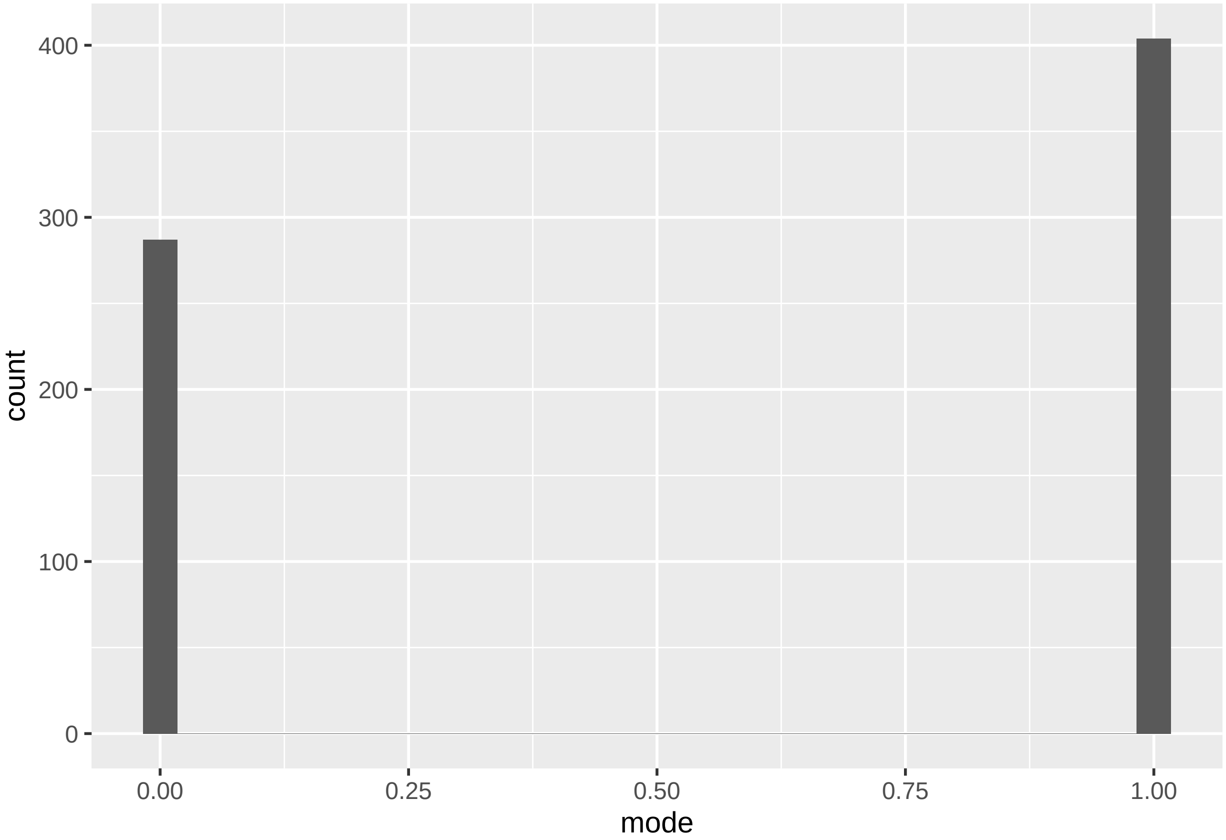
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Just for fun, seeing what mode most top 200 songs are. Looks like they're predominantly major! Makes sense.

```
ggplot(data = features_df_workshop, aes(x = mode)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Okay, let’s write this to file.

```
write_csv(features_df_workshop, here("data", "spotify.csv"))
```