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)

recently_played
```

```
##                  track.name        artist.name    track.album.name           played_at
## 1               Monotonous Arthur Hnatek Trio              Static 2021-02-15 17:00:13
## 2                    Trust       Roy Hargrove     Nothing Serious 2021-02-15 16:52:19
## 3     Drop Your Anchor Down       Chris Potter     There is a Tide 2021-02-15 16:46:23
## 4            Down in a Hole       Jason Isbell Sirens Of The Ditch 2021-02-15 16:40:27
## 5 Brand New Kind of Actress       Jason Isbell Sirens Of The Ditch 2021-02-15 16:37:21
```

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", "A Colossal Wreck", "Belleville", "Fire", "Can't Do M...
## $ artist.name <chr> "Waxahatchee", "Every Time I Die", "Knocked Loose", "Waxahatch...
## $ album.name  <chr> "Saint Cloud", "A Colossal Wreck // Desperate Pleasures", "A D...
## $ popularity  <int> 51, 50, 46, 62, 55, 31, 57, 36, 47, 22, 29, 47, 30, 45, 46, 45...
```

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", "Burning Myrrh", "Oxbow", "Aria of Jeweled...
## $ artist.name <chr> "Every Time I Die", "Full Of Hell", "Waxahatchee", "Full Of He...
## $ album.name  <chr> "A Colossal Wreck // Desperate Pleasures", "Weeping Choir", "S...
## $ popularity  <int> 50, 29, 51, 22, 49, 27, 31, 49, 70, 46, 38, 20, 68, 25, 30, 17...
```

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()
```

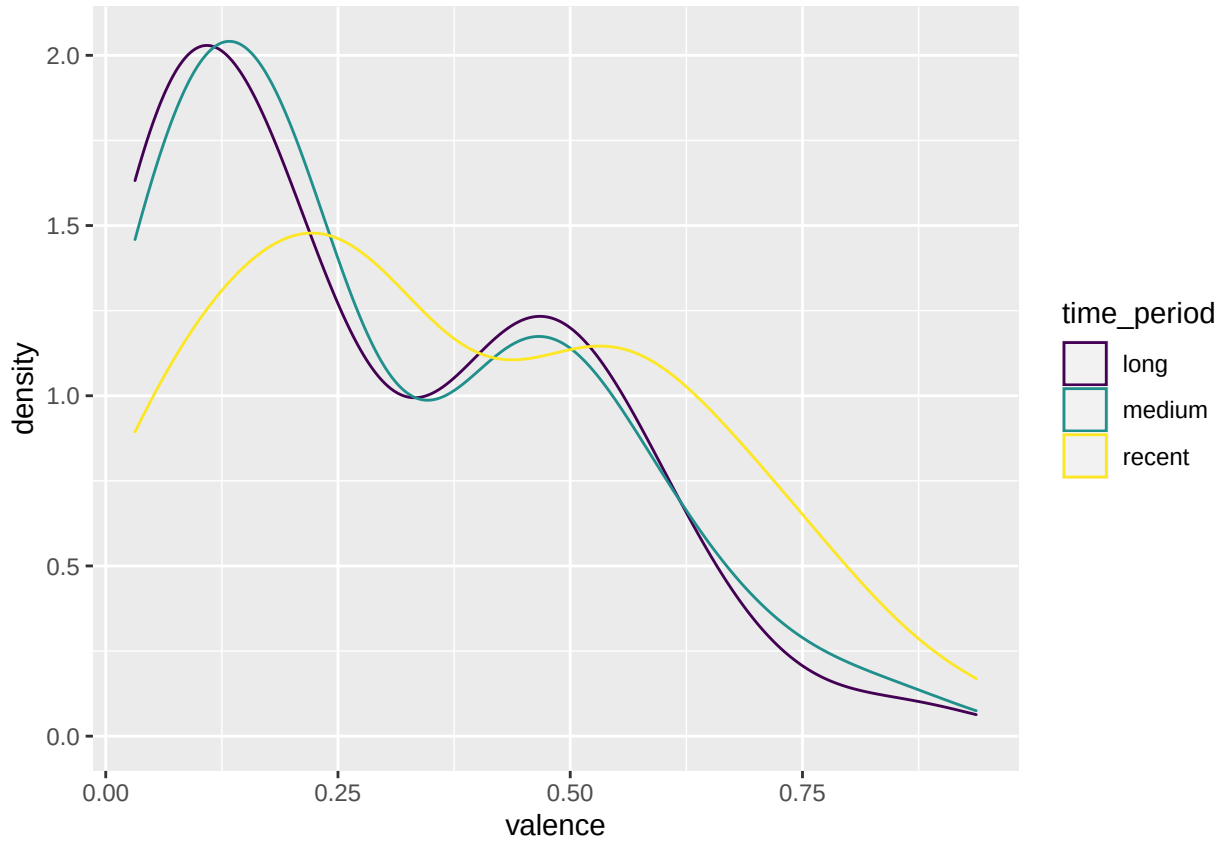
```
## Rows: 50
## Columns: 4
## $ name        <chr> "A Colossal Wreck", "Freddie Freeloader", "Freddie Freeloader ...
## $ artist.name <chr> "Every Time I Die", "Dexter Gordon", "Miles Davis", "Adrianne ...
## $ album.name  <chr> "A Colossal Wreck // Desperate Pleasures", "Something Differen...
## $ popularity  <int> 50, 1, 52, 51, 54, 41, 68, 30, 36, 37, 33, 61, 49, 25, 48, 59,...
```

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
)
```

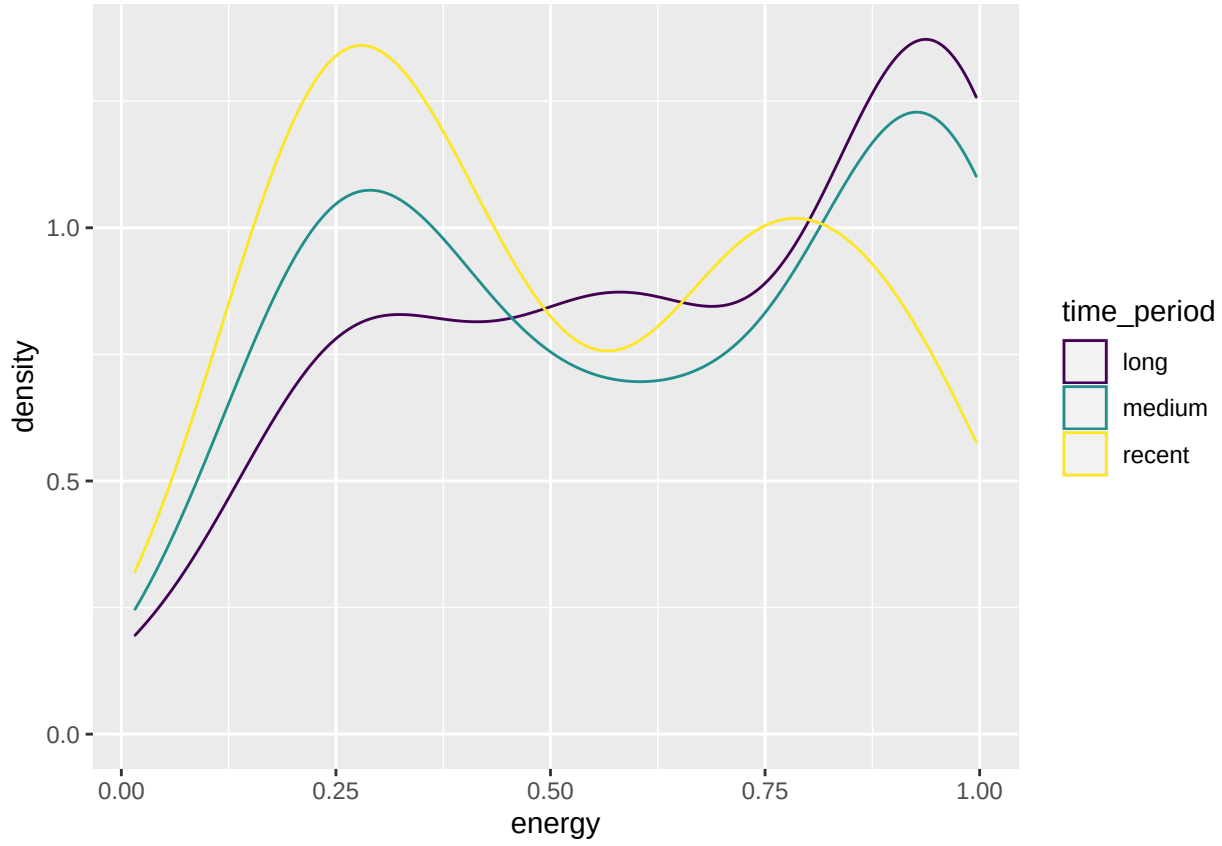
Looks like I've been listening to happier music lately compared to what I usually listen to! Valence is Spotify's measure of the "positivity" of a track.

```
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
```

```
## 4
```

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 Olivia Rodrigo      Drivers Li~ Olivia Rodrigo
##  2     2 'Mood' by 24kGoldn Featuring iann d~     Mood        24kGoldn Featuring iann dior
##  3     3 'Blinding Lights' by The Weeknd          Blinding L~ The Weeknd
##  4     4 '34+35' by Ariana Grande                 34+35       Ariana Grande
##  5     5 'Levitating' by Dua Lipa Featuring ~     Levitating  Dua Lipa Featuring DaBaby
##  6     6 'Go Crazy' by Chris Brown & Young T~     Go Crazy    Chris Brown & Young Thug
##  7     7 'Positions' by Ariana Grande             Positions   Ariana Grande
##  8     8 'Save Your Tears' by The Weeknd          Save Your ~ The Weeknd
##  9     9 'Holy' by Justin Bieber Featuring C~     Holy        Justin Bieber Featuring Chan~
## 10    10 'Whoopty' by CJ                          Whoopty     CJ
## # ... with 90 more rows
```

Acquire data I'm going to get Hot 100 songs for every week since 2019.

First, I need to get a list of weeks since the present day.

```
year_weeks <- seq(ymd('2019-02-12'),ymd('2021-02-12'), by = '1 week') %>% as.character()

head(year_weeks)
```

```
## [1] "2019-02-12" "2019-02-19" "2019-02-26" "2019-03-05" "2019-03-12" "2019-03-19"
```

Now, I'm looping through the weeks and extracting the billboard Hot 100 charts for each week. This takes forever, so grab a coffee if you're running this yourself.

```
chart_data = list()

for i in range(len(r.year_weeks)):
  dat = billboard.ChartData(name="hot-100", date=r.year_weeks[i])
  chart_data.append(dat)
```

I need a function to convert entries to dataframes to make the next part easier.

```
# 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_num) {
  chart_entries <- py$chart_data[[chart_num]]$entries
  # 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],
         week = py$chart_data[[chart_num]]$date)

  return(chart_entries_df)
}
```

Now, I'm looping through the charts, converting them to data frames, then binding them together.

```
charts_df <- map_df(1:length(py$chart_data), chart_to_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.

I'm also only searching for unique song names (1306) to reduce the search time. This still takes a while, so I hope you're ready for another cup of coffee.

```
# this will return NAs for tracks that don't come up in the search
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)
}

# I'm only searching for unique track names to minimize search time. 1306 vs 10,500 searches
spotify_ids <- map_chr(unique(charts_df$track), get_id) %>%
  enframe(value = "id", name = NULL) %>%
  mutate(track = unique(charts_df$track))

charts_df_id <- left_join(charts_df, spotify_ids, by = "track")
```

"Mr. Solo Dolo III" didn't get pulled from spotify based on the full track name. I'm using a more flexible search criterion to grab it here.

```
mr_solo <- search_spotify("Mr. Solo Dolo", type = "track")

charts_df_id[str_detect(charts_df_id$track, "Solo Dolo"),]$id <- mr_solo[1,]$id
```

Now it's time to get the audio features! This takes quite a while too. Might be a good time to catch up on email.

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

features_df <- left_join(charts_df_id, 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)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   track = col_character(),
##   artist = col_character(),
##   week = col_date(format = ""),
##   month = col_character(),
##   season = col_character(),
##   covid_period = col_character()
## )
## i Use `spec()` for the full column specifications.
```

```r
# season function I got off of this stackoverflow post:
# https://stackoverflow.com/questions/9500114/find-which-season-a-particular-date-belongs-to

getSeason <- function(DATES) {
    WS <- as.Date("2012-12-15", format = "%Y-%m-%d") # Winter Solstice
    SE <- as.Date("2012-3-15",  format = "%Y-%m-%d") # Spring Equinox
    SS <- as.Date("2012-6-15",  format = "%Y-%m-%d") # Summer Solstice
    FE <- as.Date("2012-9-15",  format = "%Y-%m-%d") # Fall Equinox

    # Convert dates from any year to 2012 dates
    d <- as.Date(strftime(DATES, format="2012-%m-%d"))

    ifelse (d >= WS | d < SE, "Winter",
      ifelse (d >= SE & d < SS, "Spring",
        ifelse (d >= SS & d < FE, "Summer", "Fall")))
}

# function to determine the number of days pre-covid or post-covid, where pre-covid is number of days b
# negative numbers indicate pre-covid time
covid_test <- function(x) {
  first_case <- as.Date("2020-01-20")
  week_date <- as.Date(x)
  date_dif <- week_date - first_case
  date_dif = as.numeric(date_dif)
  return(date_dif)
}

# adding some temporal features. covid_period is whether the chart is pre-covid or post-covid
features_df_dates <- features_df_workshop %>%
  mutate(month = lubridate::month(week, label = TRUE),
         day = lubridate::day(week),
         year = lubridate::year(week),
         season = getSeason(week),
         time_since_covid = covid_test(week),
         covid_period = if_else(time_since_covid >= 0, "post_covid", "pre_covid")) %>%
  select(-value)

glimpse(features_df_dates)
```

```
## Rows: 10,500
## Columns: 23
## $ rank             <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17...
## $ track            <chr> "7 Rings", "Happier", "Without Me", "Sunflower (Spider-Ma...
## $ artist           <chr> "Ariana Grande", "Marshmello & Bastille", "Halsey", "Post...
## $ week             <date> 2019-02-16, 2019-02-16, 2019-02-16, 2019-02-16, 2019-02-...
## $ danceability     <dbl> 0.778, 0.687, 0.752, 0.760, 0.834, 0.579, 0.717, 0.837, 0...
## $ energy           <dbl> 0.317, 0.792, 0.488, 0.479, 0.730, 0.904, 0.653, 0.364, 0...
## $ key              <dbl> 1, 5, 6, 2, 8, 5, 1, 8, 11, 0, 6, 1, 1, 8, 4, 7, 8, 0, 6,...
## $ loudness         <dbl> -10.732, -2.749, -7.050, -5.574, -3.714, -2.729, -5.634, ...
## $ mode             <dbl> 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, ...
## $ speechiness      <dbl> 0.3340, 0.0452, 0.0705, 0.0466, 0.2220, 0.0618, 0.0658, 0...
## $ acousticness     <dbl> 0.59200, 0.19100, 0.29700, 0.55600, 0.00513, 0.19300, 0.2...
## $ instrumentalness <dbl> 0.00e+00, 0.00e+00, 9.11e-06, 0.00e+00, 0.00e+00, 0.00e+0...
```

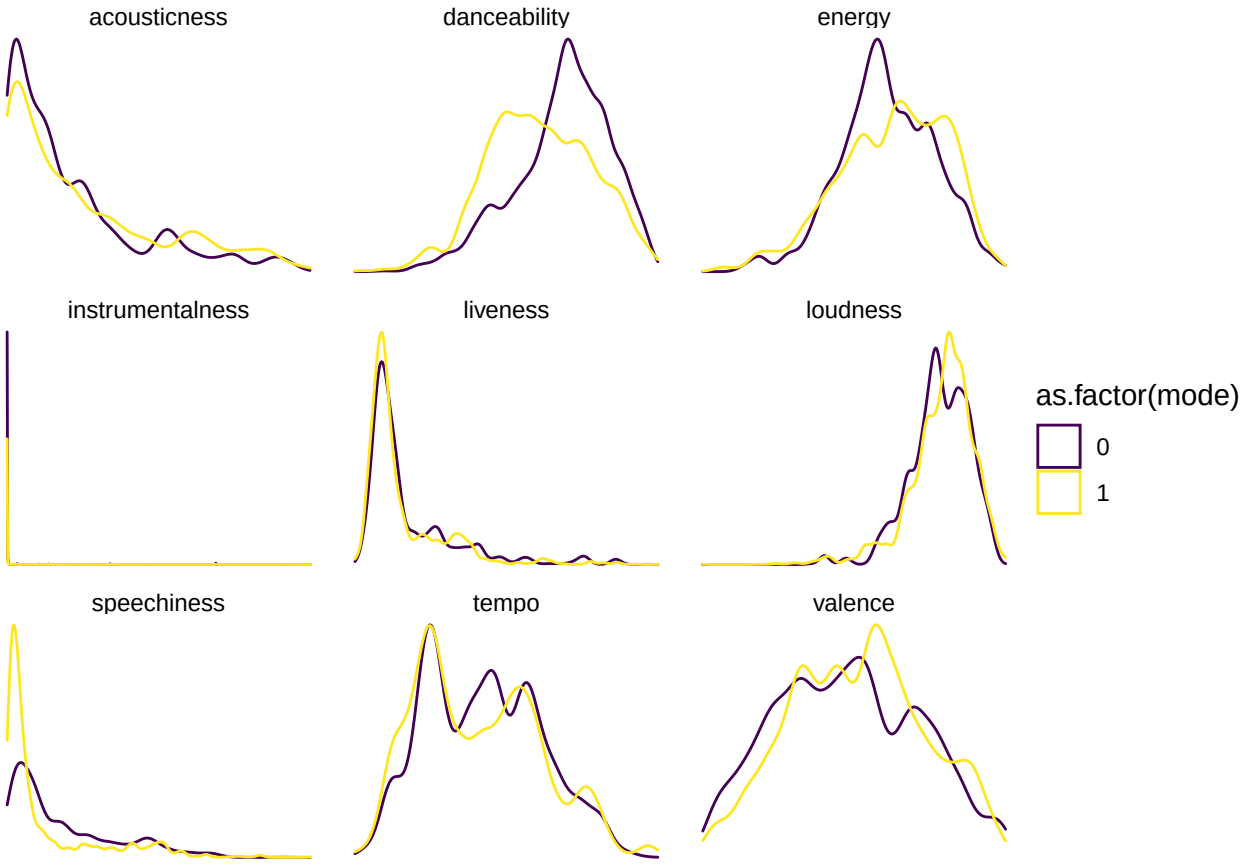
```
## $ liveness         <dbl> 0.0881, 0.1670, 0.0936, 0.0703, 0.1240, 0.0640, 0.1010, 0...
## $ valence          <dbl> 0.327, 0.671, 0.533, 0.913, 0.446, 0.681, 0.412, 0.463, 0...
## $ tempo            <dbl> 140.048, 100.015, 136.041, 89.911, 155.008, 82.014, 106.9...
## $ duration_ms      <dbl> 178627, 214290, 201661, 158040, 312820, 190947, 207320, 2...
## $ time_signature   <dbl> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, ...
## $ month            <chr> "Feb", "Feb", "Feb", "Feb", "Feb", "Feb", "Feb", "Feb", "...
## $ day              <dbl> 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 1...
## $ year             <dbl> 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 201...
## $ season           <chr> "Winter", "Winter", "Winter", "Winter", "Winter", "Winter...
## $ time_since_covid <dbl> -338, -338, -338, -338, -338, -338, -338, -338, -338, -33...
## $ covid_period     <chr> "pre_covid", "pre_covid", "pre_covid", "pre_covid", "pre_...
```

Quick exploration

Let's see if this is worth making exercises with. There don't appear to be strong patterns in the data across the factors, except in the danceability and energy between major and minor keys.

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

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

```
## Warning: Removed 9 rows containing non-finite values (stat_density).
```



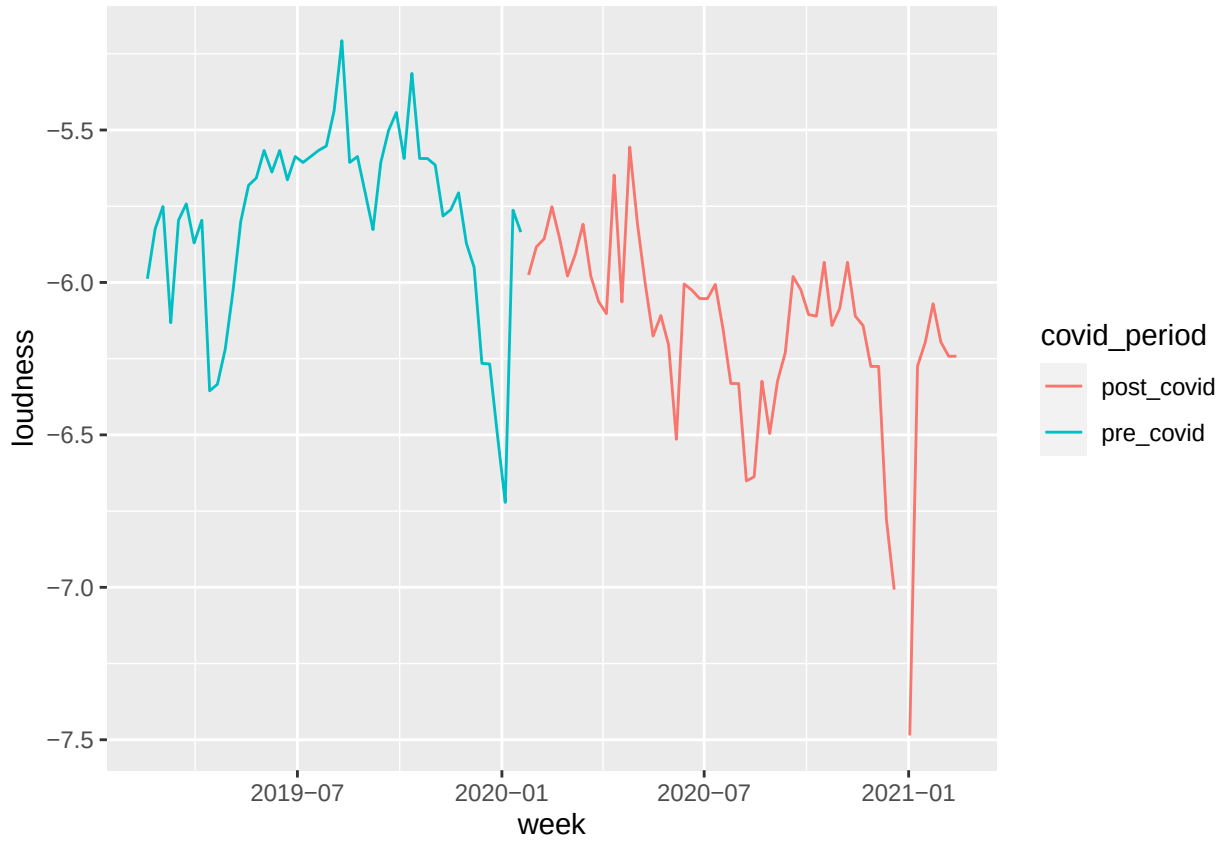
Maybe some of the within-week noise obscures general patterns. I'll group by weeks and summarize the variables to see if other patterns get teased out.

```
# R doesn't have a built-in function for grabbing the mode
get_mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

features_df_sum <- features_df_dates %>%
  group_by(week) %>%
  summarize(
    valence = median(valence),
    tempo = median(tempo),
    speechiness = median(speechiness),
    instrumentalness = median(instrumentalness),
    liveness = median(liveness),
    loudness = median(loudness),
    acousticness = median(acousticness),
    danceability = median(danceability),
    energy = median(energy),
    mode_mode = get_mode(mode),
    major = sum(mode == 1),
    minor = sum(mode == 0),
    duration_ms = median(duration_ms),
    time_since_covid = median(time_since_covid),
```

```
    covid_period = get_mode(covid_period)
  ) %>%
  mutate(week = as.Date(week),
    prop_minor = minor / (major + minor))
```

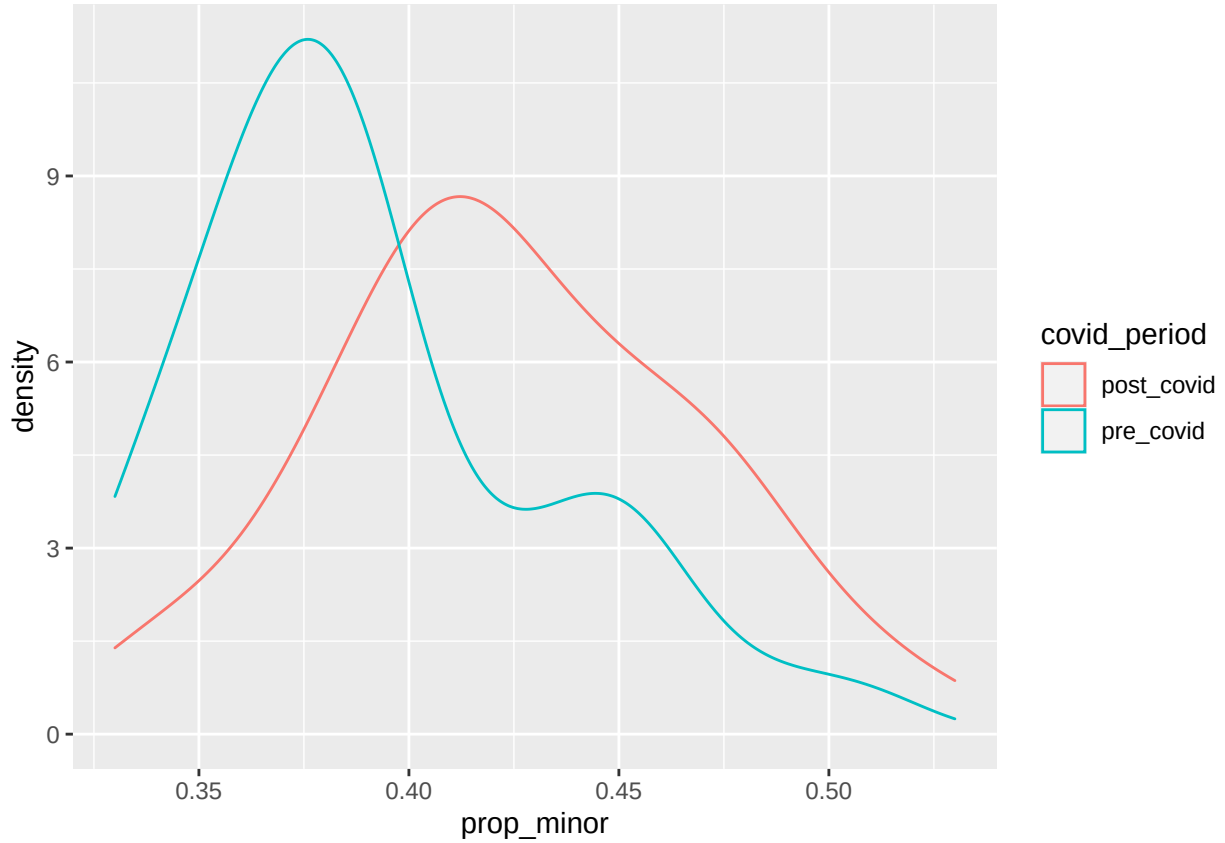
Looks like there are some interesting patterns across time too!

```
features_df_sum %>%
  ggplot(aes(x = week, y = loudness, color = covid_period)) +
  geom_line()
```




```
features_df_sum %>%
  ggplot(aes(x = prop_minor, color = covid_period)) +
  geom_density()
```

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

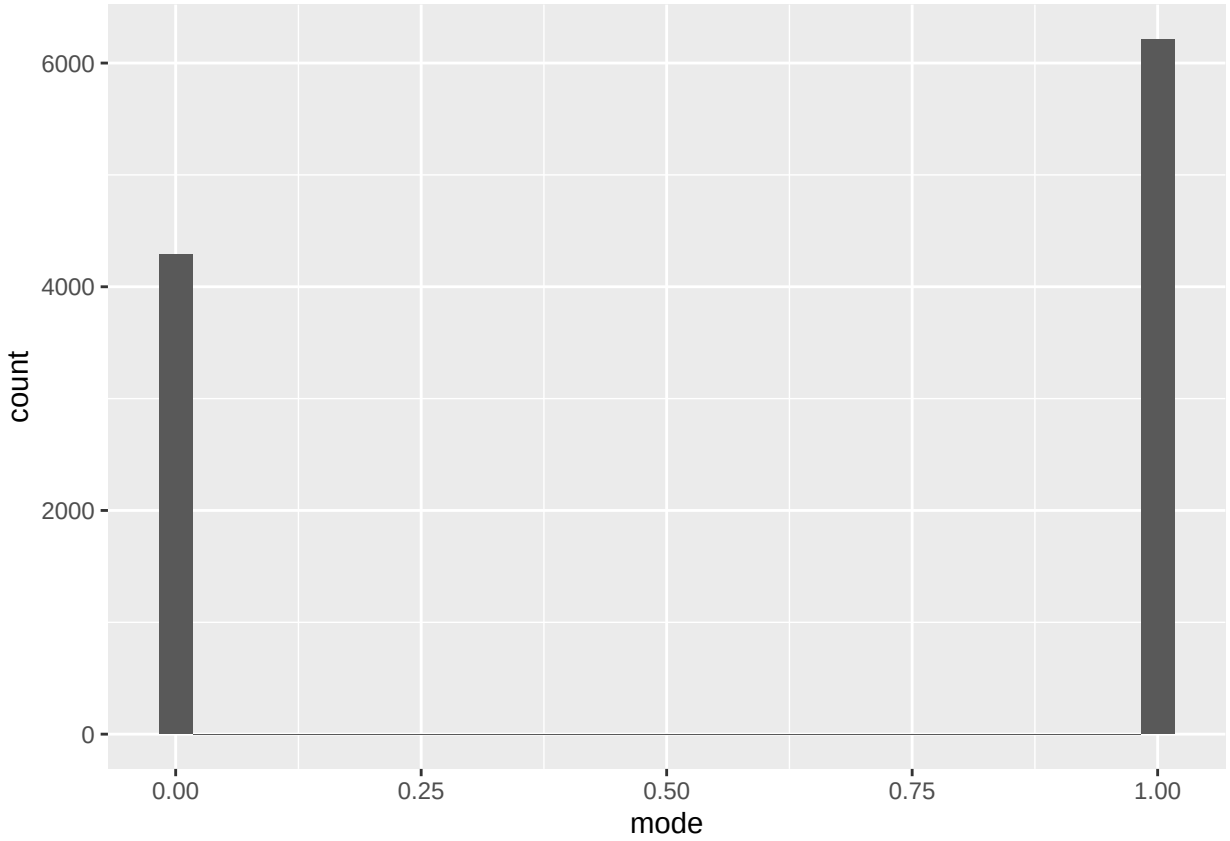


Just for fun, seeing what mode most top 100 songs are. Looks like they're predominantly major! Makes sense.

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

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

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



Okay, let's write this to file.

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