Obtaining Lyrics Data

Connor French

3/21/2021

Load packages

```
library(geniusr)
library(tidyverse)
library(tidytext)
library(textdata)
library(here)
```

Get the lyrics for Buck Meek's album Two Saviors (the "positive" music). NOTE: Follow the geniusr instructions first to authenticate yourself. Otherwise, you won't be able to access the API.

```
buck_tracklist <- get_album_tracklist_search("Buck Meek", "Two Saviors")


buck_song_df <- map_df(buck_tracklist$song_lyrics_url, get_lyrics_url) %>%
  group_by(song_name) %>%
  mutate(line_number = row_number()) %>%
  ungroup()
```

Get the lyrics for Full of Hell's album Wheeping Choir (the "negative" music)

```
foh_tracklist <- get_album_tracklist_search("Full of Hell", "Weeping Choir")

foh_song_df <- map_df(foh_tracklist$song_lyrics_url, get_lyrics_url) %>%
  group_by(song_name) %>%
  mutate(line_number = row_number()) %>%
  ungroup()
```

Put both into a single df

```
total_df <- bind_rows(buck_song_df, foh_song_df)
```

This is the dataset for the workshop.

```
write_csv(total_df, here("data", "lyrics.csv"))
```

Example analysis

Get one line per token

```
tidy_df <- total_df %>%
  unnest_tokens(word, line)
```

Remove stop words

```
no_stop_df <- tidy_df %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

Check out the most common words for:

Overall

```
no_stop_df %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 711 x 2
##    word      n
##    <chr> <int>
##  1 blue     13
##  2 eyes     13
##  3 love     13
##  4 jewel    10
##  5 time     10
##  6 mind      9
##  7 doors     8
##  8 heart     8
##  9 eye       7
## 10 hold      7
## # ... with 701 more rows
```

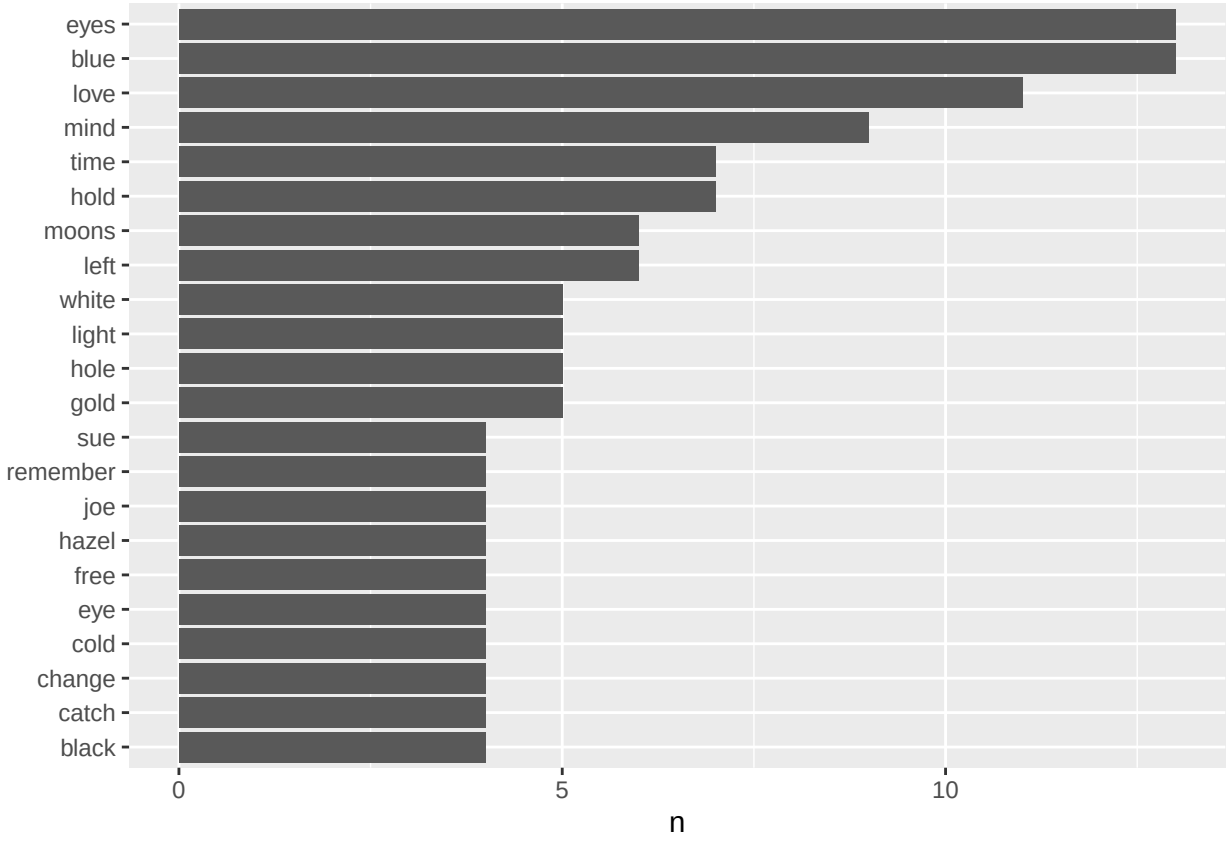
Buck Meek's most common words

```
no_stop_df %>%
  filter(artist_name == "Buck Meek") %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 370 x 2
##    word      n
##    <chr> <int>
##  1 blue     13
##  2 eyes     13
##  3 love     11
##  4 mind      9
##  5 hold      7
##  6 time      7
##  7 left      6
##  8 moons     6
##  9 gold      5
## 10 hole      5
## # ... with 360 more rows
```

Visualize as a bar chart

```
no_stop_df %>%
  filter(artist_name == "Buck Meek") %>%
  count(word, sort = TRUE) %>%
  filter(n > 3) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL)
```



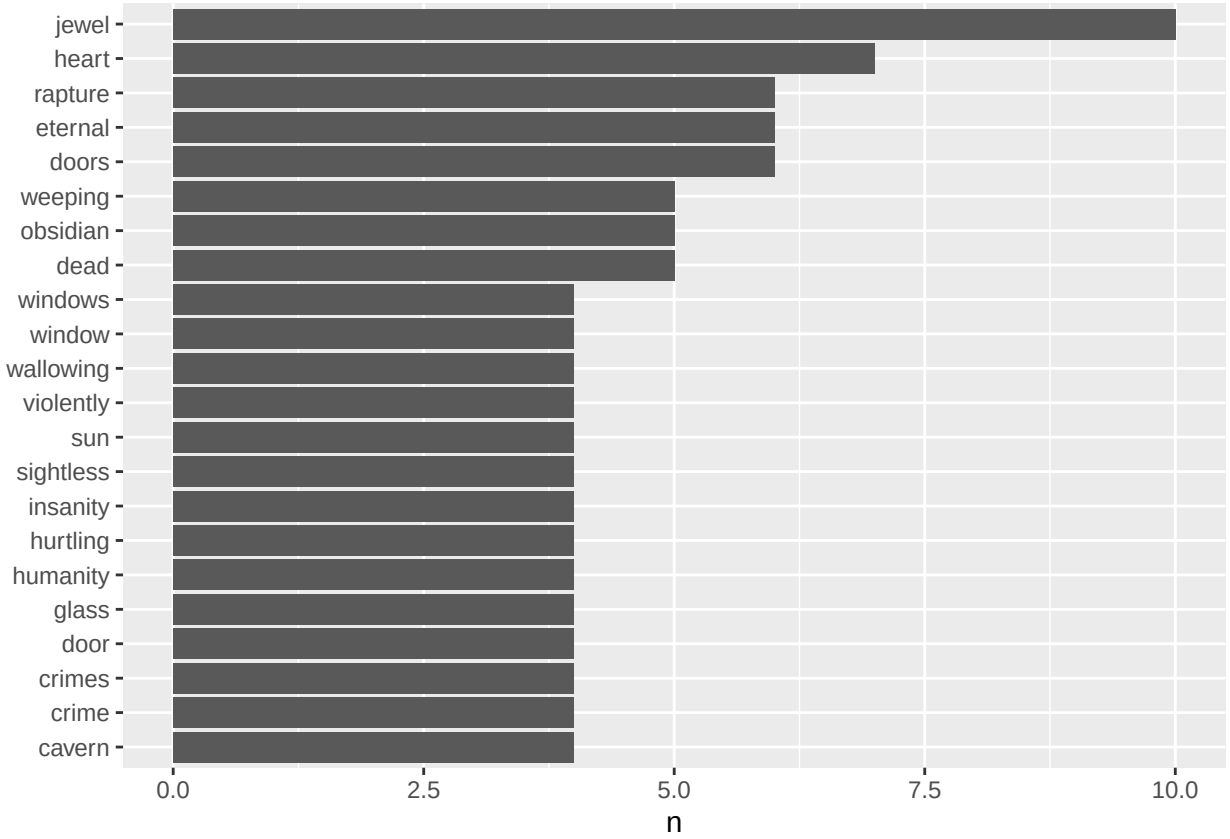
Most common words for Full of Hell

```
no_stop_df %>%
  filter(artist_name == "Full of Hell") %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 389 x 2
##    word         n
##    <chr>    <int>
##  1 jewel       10
##  2 heart        7
##  3 doors        6
##  4 eternal      6
##  5 rapture      6
```

```
##  6 dead         5
##  7 obsidian     5
##  8 weeping      5
##  9 cavern       4
## 10 crime        4
## # ... with 379 more rows
```

```
no_stop_df %>%
  filter(artist_name == "Full of Hell") %>%
  count(word, sort = TRUE) %>%
  filter(n > 3) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL)
```



Sentiment analysis

Let's use the AFINN and Bing lexicons to get both a continuous and categorical idea of how these lyrics are structured.

```
afinn_sent <- get_sentiments("afinn")
bing_sent <- get_sentiments("bing")
```

Now let's join this with the original data frame using an `inner_join`

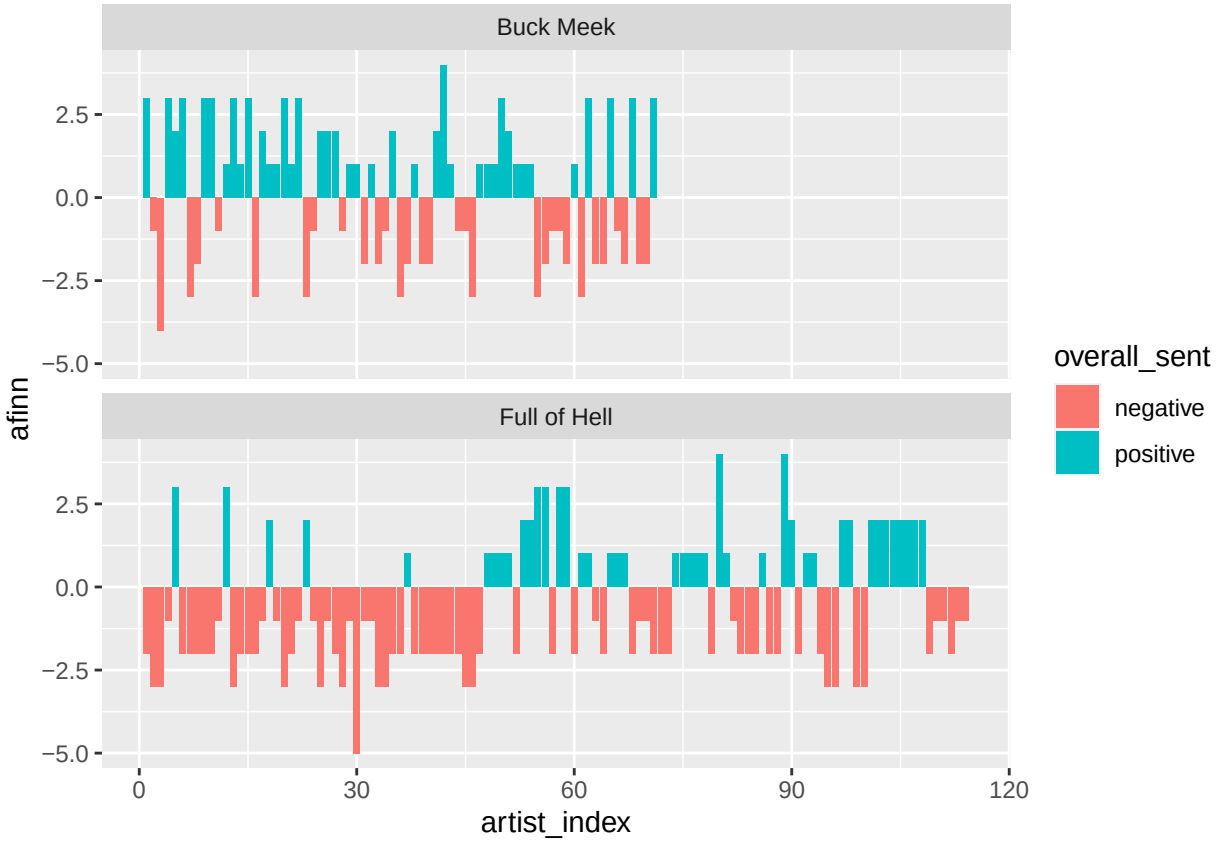
```
afinn_df <- no_stop_df %>%
  inner_join(afinn_sent) %>%
  mutate(index = row_number()) %>%
  rename(afinn = value)
```

```
## Joining, by = "word"
```

Visualize positivity of each artist.

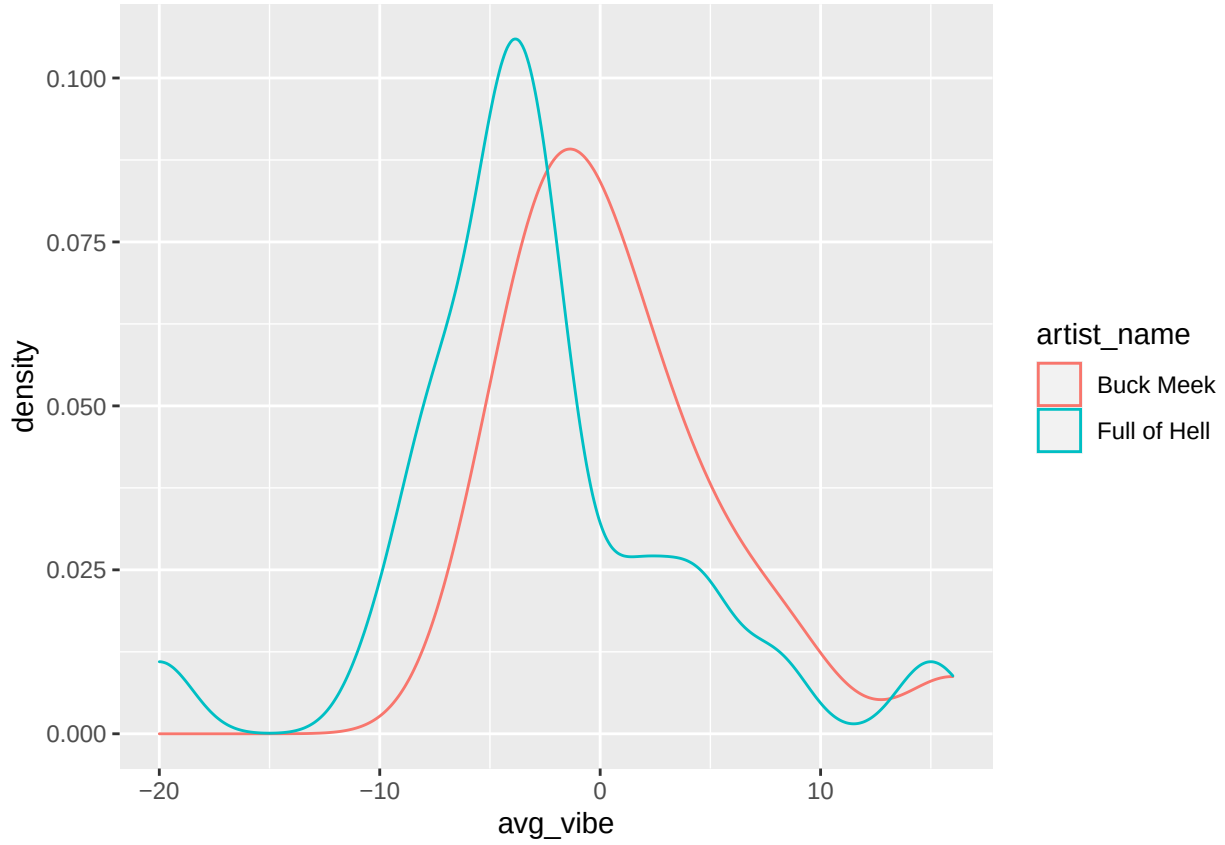
Looks like Full of Hell has relatively more negativity than Buck Meek.

```
afinn_df %>%
  group_by(artist_name) %>%
  mutate(artist_index = row_number(),
         overall_sent = if_else(
           afinn >= 0, "positive", "negative"
         )) %>%
  ggplot(aes(y = afinn, x = artist_index, fill = overall_sent)) +
  geom_col() +
  facet_wrap(~artist_name, nrow = 2)
```



Aggregating single words may not be good enough to get a sense of the sentiment of a body of text. Each song is divided into sections, like the verse, chorus, etc. (although it is not a perfect divide). Let's group the words by their song and section, then summarize the sentiment of each section by taking the sum.

```
afinn_df %>%
  group_by(artist_name, song_name, section_name) %>%
  summarize(avg_vibe = sum(afinn)) %>%
  ggplot(aes(x = avg_vibe, color = artist_name)) +
  geom_density()
```



What about a binary look? We can do this with the Bing lexicon

```
bing_df <- no_stop_df %>%
  inner_join(bing_sent) %>%
  mutate(index = row_number())
```

```
## Joining, by = "word"
```

```
glimpse(bing_df)
```

```
## Rows: 230
## Columns: 9
## $ section_name    <chr> "Pareidolia", "Pareidolia", "Pareidolia", "Pareidolia"~
## $ section_artist  <chr> "Buck Meek", "Buck Meek", "Buck Meek", "Buck Meek", "B~
## $ song_name       <chr> "Pareidolia", "Pareidolia", "Pareidolia", "Pareidolia"~
## $ artist_name     <chr> "Buck Meek", "Buck Meek", "Buck Meek", "Buck Meek", "B~
## $ song_lyrics_url <chr> "https://genius.com/Buck-meek-pareidolia-lyrics", "htt~
## $ line_number     <int> 3, 8, 13, 20, 24, 25, 28, 3, 4, 6, 7, 12, 15, 15, 20, ~
## $ word            <chr> "fast", "paradise", "burning", "froze", "hell", "lucky~
```

```
## $ sentiment       <chr> "positive", "positive", "negative", "negative", "negat~
## $ index           <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,~
```

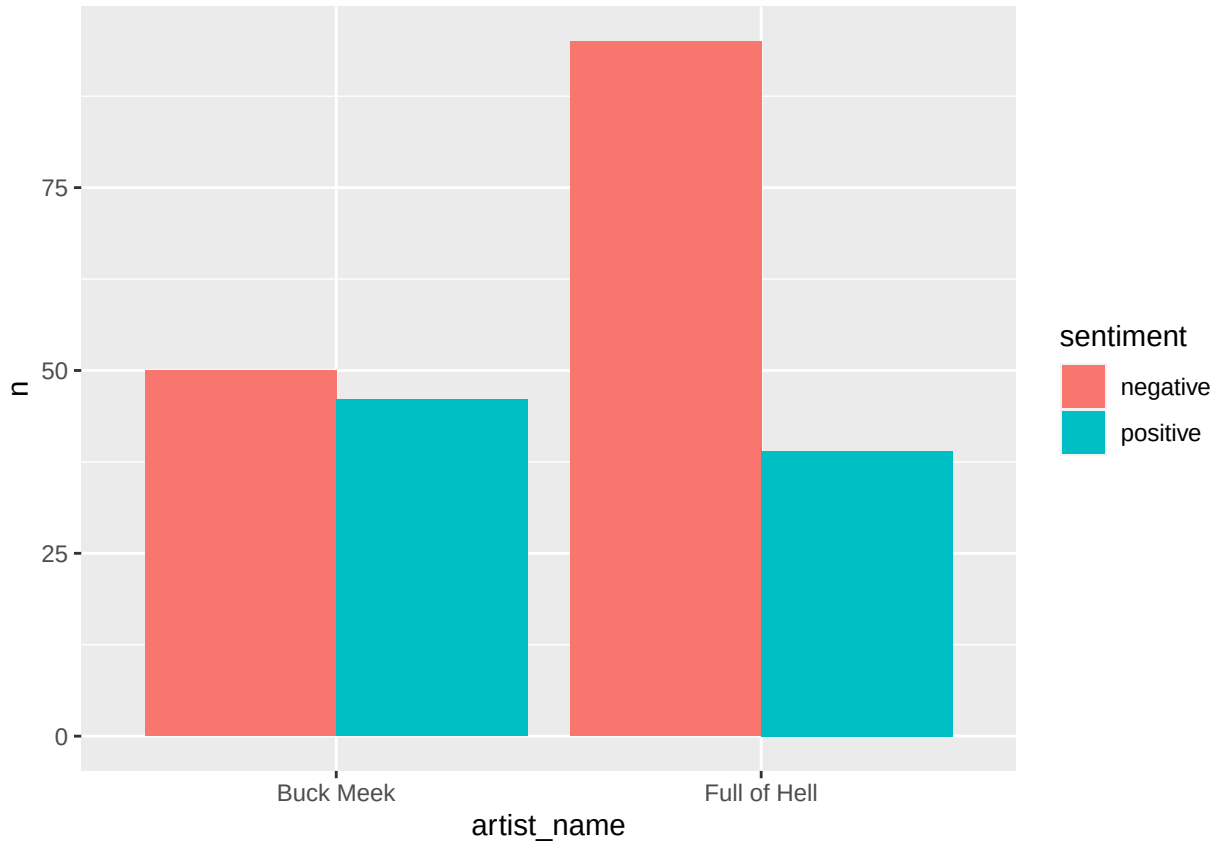
```
sent_count_df <- bing_df %>%
  group_by(artist_name) %>%
  count(sentiment, sort = TRUE) %>%
  group_by(artist_name) %>%
  mutate(prop_sent = round(n / sum(n), 3)) %>%
  ungroup()

sent_count_df
```

```
## # A tibble: 4 x 4
##   artist_name  sentiment     n prop_sent
##   <chr>        <chr>     <int>     <dbl>
## 1 Full of Hell negative     95     0.709
## 2 Buck Meek    negative     50     0.521
## 3 Buck Meek    positive     46     0.479
## 4 Full of Hell positive     39     0.291
```

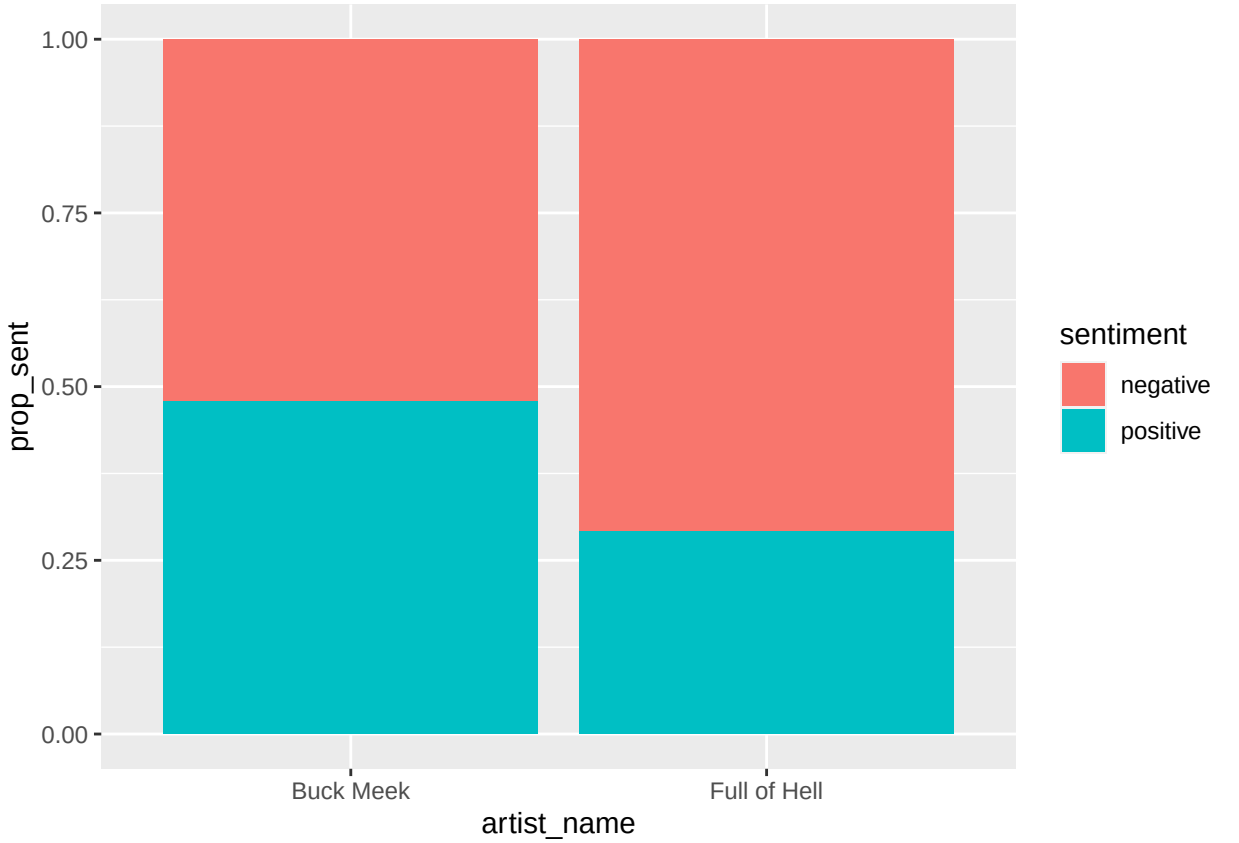
Barplot to visualize counts.

```
sent_count_df %>%
  ggplot(aes(x = artist_name, y = n, fill = sentiment)) +
  geom_col(position = "dodge")
```



Barplot to visualize proportions.

```
sent_count_df %>%
  ggplot(aes(x = artist_name, y = prop_sent, fill = sentiment)) +
  geom_col()
```



More serious data wrangling

This is how text may appear if you don't get to use a fancy API to obtain your data. It's a single block with whitespace characters (`\n`) and extraneous classifiers (`[Verse 1]`, `[Instrumental Break]`). We want to get this into a tidy format, where each word is an observation and we have the line number for each word. To do this, we will use the powerful stringr package.

These are Buck Meek lyrics from the album we analyzed earlier! I scraped these lyrics from the Genius web page, using the rvest package. This is effectively what the Genius API does, but the API does some helpful transformation under the hood that we'll do here!

```
candle_lyrics <- "[Verse 1]\nInnocence is a light beam, you're doing your thing\nWith your arm out your
```

Let's split this into lines. The `\n` whitespace character denotes a line break, so we can delimit the string based on this character! I'll split this into steps for the workshop

```
candle_lyrics %>%
  str_split("\\n") %>%
  unlist() %>%
  str_remove("\\[\\D+\\d+\\]") %>%
  str_remove("\\[\\D+\\]") %>%
```

```
  na_if("") %>%
  na.omit() %>%
  enframe(name = "line_number", value = "line") %>%
  unnest_tokens(word, line)
```

```
## # A tibble: 200 x 2
##    line_number word
##          <int> <chr>
##  1           1 innocence
##  2           1 is
##  3           1 a
##  4           1 light
##  5           1 beam
##  6           1 you're
##  7           1 doing
##  8           1 your
##  9           1 thing
## 10           2 with
## # ... with 190 more rows
```