The Crusade: Text Mining Trivium Songs

Recle Etino Vibal

Contents

A View of Mining Lyrics

Trivium is my favorite heavy metal band. They also did a wonderful cover of *Master of Puppets*, one of my favorite Metallica songs.

I found this article by HighlandR on the tidy text analysis of Metallica albums. I really enjoyed it, and it also introduced me to the `tidytext` package and text mining.

With my new found passion with R and my fondness of Trivium, I decided to do a tidy text analysis of Trivium song lyrics.

```
library(knitr)
library(tidyverse)
library(geniusR)
library(rebus)
library(tidytext)
library(drlib)
library(wordcloud)
library(extrafont)
library(scales)
library(tinytex)

opts_chunk$set(message = FALSE,
               warning = FALSE,
               fig.align = 'center')
```

Into the Mouth of R We March

A few words on the packages I used here.

`tidyverse` and `knitr` are my most used packages. I use `rebus` so I can avoid regex. `rebus` requires more typing, but it is more intuitive when you want to attacks strings, like song titles and lyrics. I used `wordcloud` to make a wordcloud and `extrafont` to change the default fonts in my plots.

The `tidytext` package, the main tool I will use here, is new to me, but it has become one of my favorites. Another entertaining use of tidytext is Julia Silge's blog post about Sherlock Holmes. I would like to imitate this kind of analysis of a classic text soon.

One gem I found in Silge's Holmes post is David Robinson's personal R package `drlib`. It has several useful functions, but I used the combination of `reorder_within()` and `scale_x_reorder()` to make my plots prettier.

Pull Harder on the Strings of Lyrics with geniusR

One thing I learned from HighlandR's post is the existence of the `geniusR` package by Josiah Parry. `geniusR` gives access to lyrics from Genius as text data.

The first problem I encountered with downloading all Trivium albums is that Snøfall, an instrumental track at the beginning of Silence in the Snow, results to an error. The reason behind this is the presence of the special character "ø" in the title and it's absence in the url (https://genius.com/Trivium-snfall-lyrics).

```
# Get Silence in the Snow Album ------------------------------------------

genius_album("Trivium",
             "Silence in the snow")
```

```
## # A tibble: 458 x 3
##    title                track_n text
##    <chr>                  <int> <chr>
##  1 Silence In The Snow        2 Here we all stand on this canvas of white
##  2 Silence In The Snow        2 Our palette holds but only one shade tonig~
##  3 Silence In The Snow        2 Silence snows in, in her wintery chill
##  4 Silence In The Snow        2 Let's paint the ground red with the blood ~
##  5 Silence In The Snow        2 Kill
##  6 Silence In The Snow        2 The battle goes
##  7 Silence In The Snow        2 Silence in the snow
##  8 Silence In The Snow        2 We must fight till they all die
##  9 Silence In The Snow        2 In their cold blood
## 10 Silence In The Snow        2 Silence in the snow
## # ... with 448 more rows
```

Since Snøfall is an instrumental track, I decided to get the tracklist of Silence in the Snow first using `genius_tracklist()`, then filter out Snøfall before downloading the lyrics with `genius_lyrics()` and the help of `map()` from the `purrr` package.

```
genius_tracklist("Trivium",
                 "Silence in the snow") %>%
  filter(title != "Snøfall") %>%
  mutate(lyrics = map(title,
                      genius_lyrics,
                      artist = "Trivium"),
         album = "Silence in the snow") %>%
```

```
  unnest() %>%
  head(10) %>%
  kable(caption = "Sample Tibbe of Trivium Lyrics Gathered from Genius")
```

Table 1: Sample Tibbe of Trivium Lyrics Gathered from G

title	track_n	track_url	album	title1
Silence In The Snow	2	https://genius.com/Trivium-silence-in-the-snow-lyrics	Silence in the snow	Silence In T
Silence In The Snow	2	https://genius.com/Trivium-silence-in-the-snow-lyrics	Silence in the snow	Silence In T
Silence In The Snow	2	https://genius.com/Trivium-silence-in-the-snow-lyrics	Silence in the snow	Silence In T
Silence In The Snow	2	https://genius.com/Trivium-silence-in-the-snow-lyrics	Silence in the snow	Silence In T
Silence In The Snow	2	https://genius.com/Trivium-silence-in-the-snow-lyrics	Silence in the snow	Silence In T
Silence In The Snow	2	https://genius.com/Trivium-silence-in-the-snow-lyrics	Silence in the snow	Silence In T
Silence In The Snow	2	https://genius.com/Trivium-silence-in-the-snow-lyrics	Silence in the snow	Silence In T
Silence In The Snow	2	https://genius.com/Trivium-silence-in-the-snow-lyrics	Silence in the snow	Silence In T
Silence In The Snow	2	https://genius.com/Trivium-silence-in-the-snow-lyrics	Silence in the snow	Silence In T
Silence In The Snow	2	https://genius.com/Trivium-silence-in-the-snow-lyrics	Silence in the snow	Silence In T

What I did here is similar to how `genius_album()` works. We get a tibble with each song divided into verse lines. I included an album column so I know what album the tracks are from. I used this as a template to create the `get_trivium_album()` function that allowed me to download all lyrics at once in tidy format. I used `map()` again to accomplish this.

```
trivium_albums <-
  c("Ember to Inferno",
    "Ascendancy",
    "The Crusade",
    "Shogun",
    "In Waves",
    "Vengeance Falls",
    "Silence in the Snow",
    "The Sin and the Sentence")

get_trivium_album <- function(trivium_album) {
  genius_tracklist("Trivium",
                   trivium_album) %>%
    filter(title != "Snøfall") %>%
    mutate(lyrics = map(title,
                        genius_lyrics,
                        artist = "Trivium"),
           album = trivium_album) %>%
    unnest()
}

trivium <-
  map_dfr(trivium_albums,
          possibly(get_trivium_album,
                   tibble(title = NA,
                          track_n = NA,
                          album = NA,
                          text = NA,
                          line = NA))) %>%
  drop_na()
```

Prior to this, I tested each album to make sure I will not encounter the same problem with Silence. I still included `possibly()` as an added safety feature then drop all `NA` rows, if any, with `drop_na()`.

```
trivium_covers <- paste(c("Master Of Puppets",
                          "Iron Maiden",
                          "Slave New World",
                          "Skulls... We Are 138",
                          "Losing My Religion"),
                        collapse = "|")
trivium <-
  trivium %>%
  filter(str_detect(title,
                    trivium_covers) == FALSE)
```

After downloading all the Trivium albums, I still had to remove the cover songs because I only wanted to focus on the originals. Trivium's cover of Master of Puppets affected the results in the preliminary analysis. I think it is not proper to allow a cover song to overpower the originals.

Text Dismantled

The tibble `geniusR` gives is not yet very useful, so I had to make some cleaning like removing stop words.

```
SMART_stop_words <-
  stop_words %>%
  filter(lexicon == "SMART")

apostrophe <- paste(c("'" %R% one_or_more(ALPHA),
                      "’" %R% one_or_more(ALPHA)),
                    collapse = "|")
```

One important part of `tidytext` is the `stop_words` data frame. It contains 1149 stop words from three lexicons. I used the SMART lexicon because it contained the most number of stop words.

One annoying thing I discovered is that lyrics from Genius contains curly apostrophes. R detects curly and straight apostrophes differently. Since the `stop_words` data frame contains straight apostrophes only, I had to manually remove curly and straight apostrophes from the text. I checked beforehand, so I know I am only removing apostrophes and letters after it in contractions and possessives.

```
clean_trivium_lyrics <-
  trivium %>%
  mutate(text = str_replace_all(text,
                                "asun-der",
                                "asunder")) %>%
  unnest_tokens(lyrics, text) %>%
  mutate(lyrics = str_replace(lyrics,
                              apostrophe,
                              "")) %>%
  anti_join(SMART_stop_words,
            by = c("lyrics" = "word"))
```

Another annoying string is asun-der. It should be asunder, so I had to change this manually also. After solving the asunder problem, I used `unnest_tokens()` to split the text column into words, or lyrics in my case. I then removed all the apostrophes from the lyrics before using `anti_join()` to remove all the stop words.

Trivium lyrics gathered. Trivium lyrics cleaned. Time to make sense and visualize the data.

Anthem (We are the Functions)

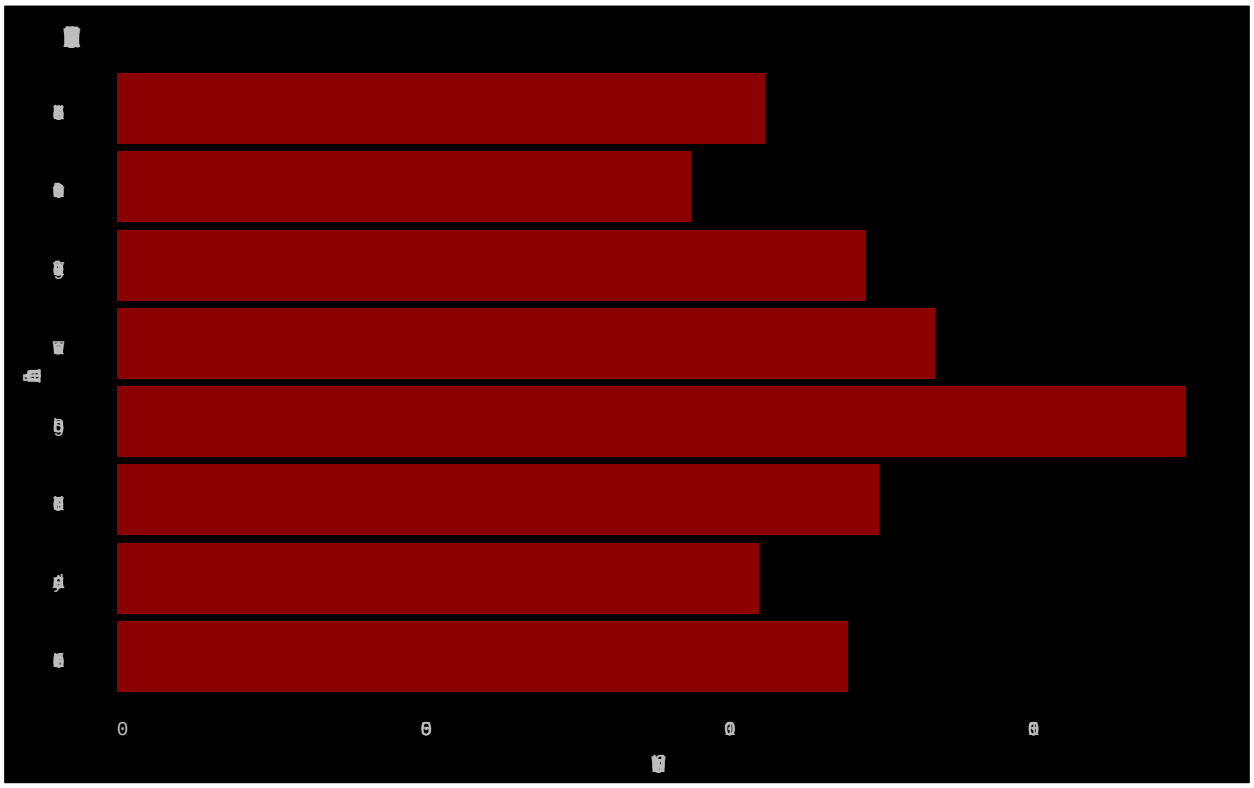
During my preliminaries, I discovered I used some code multiple times. I believe in David Robinson's advice, so I made some of this code into functions, or at least the ones I could. I also made color palette for Trivium inspired by the album covers.

```
trivium_palette <- c("yellow2", # Ember
                     "orange2", # Ascendancy
                     "royalblue4", # Crusade
                     "darkred", # Shogun
                     "darkgrey", # Waves
                     "steelblue4", # Vengeance
                     "whitesmoke", # Silence
                     "saddlebrown") # Sin

theme_trivium <- function(...) {
  theme(panel.background = element_rect(fill = "black"),
        panel.grid = element_blank(),
        plot.background = element_rect(fill = "black"),
        axis.ticks = element_blank(),
        strip.background = element_rect(fill = "black"),
        legend.background = element_rect(fill = "black"),
        legend.key = element_blank(),
        text = element_text(family = "Georgia",
                            color = "grey"),
        axis.text = element_text(color = "grey"),
        strip.text = element_text(color = "grey"),
        ...)
}

scale_fill_trivium <- function(...) {
  scale_fill_manual(...,
                    values = trivium_palette)
}

trivium_album_col <-
  trivium_albums %>%
  length() %>%
  sqrt() %>%
  floor()

facet_trivium_albums <- function(album, trivium_albums, ...) {
  facet_wrap(~factor(album,
                     levels = trivium_albums),
             scales = "free_y",
             ncol = trivium_album_col,
             ...)
}
```

Album That Spawned the Most Words

One thing I wanted to know was which album had the most words, both in count and unique.

```
clean_trivium_lyrics %>%
  count(album) %>%
  ggplot(aes(factor(album,
                    levels = trivium_albums),
             n)) +
  geom_col(fill = "darkred") +
  coord_flip() +
  labs(title = "Number of Words in Each Trivium Album",
       x = "Album",
       y = "Number of Words") +
  theme_trivium()
```

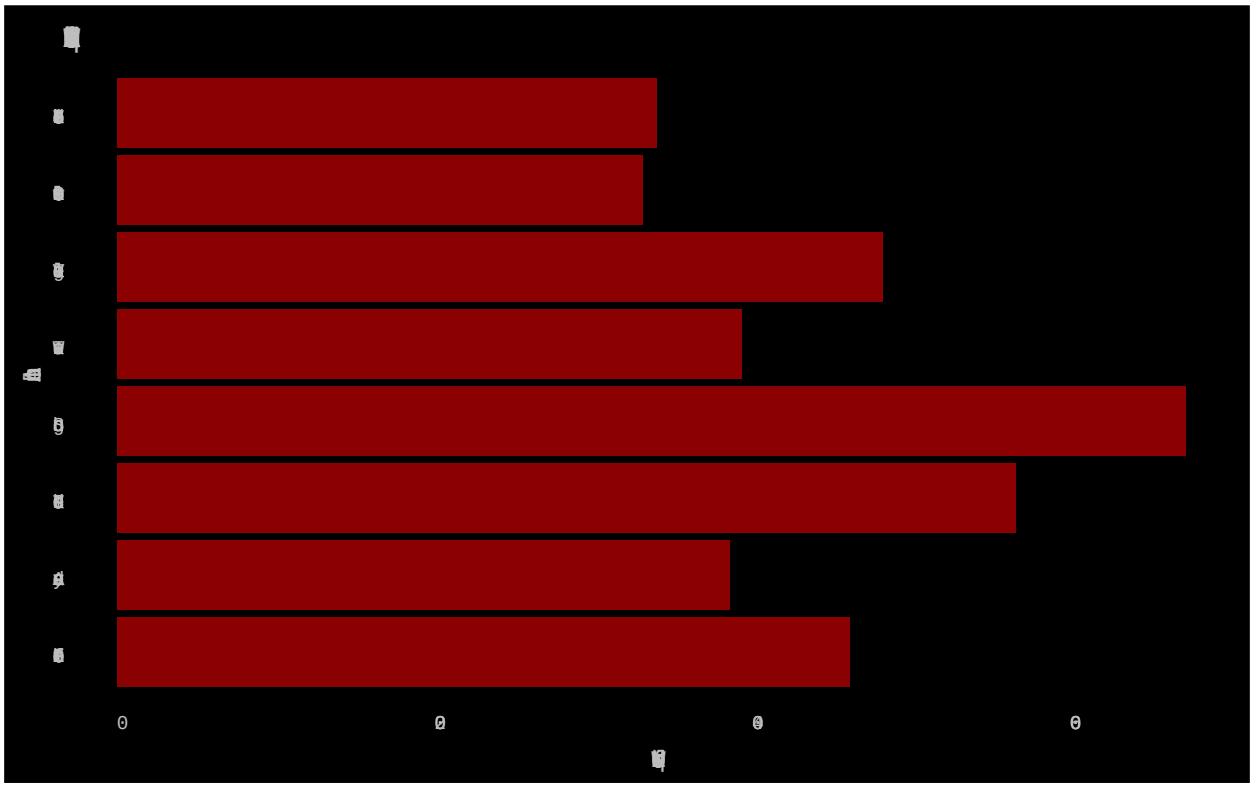


The first three albums had almost the same number of words. Trivium went crazy with the lyrics in Shogun. This is not surprising for an album that features a lot Greek mythology. Trivium toned down the lyrics after Shogun and really took a dip in word count in Silence in the Snow. I think this has something to do with Matt Heafy's change in vocal style. I think the decreased lyrics allowed Matt to relax his vocal cords a little. Fortunately, there is a resurgence in the number of lyrics, but it has not returned to pre-Shogun days.

These are absolute count of words. I also wanted to see how many unique words (a measure of creativity maybe) an album contains.

```
clean_trivium_lyrics %>%
  group_by(album) %>%
  summarise(n = n_distinct(lyrics)) %>%
  ggplot(aes(factor(album,
                    levels = trivium_albums),
```

```
            n)) +
  geom_col(fill = "darkred") +
  coord_flip() +
  labs(title = "Number of Unique Words in Each Trivium Album",
       x = "Album",
       y = "Number of Unique Words") +
  theme_trivium()
```



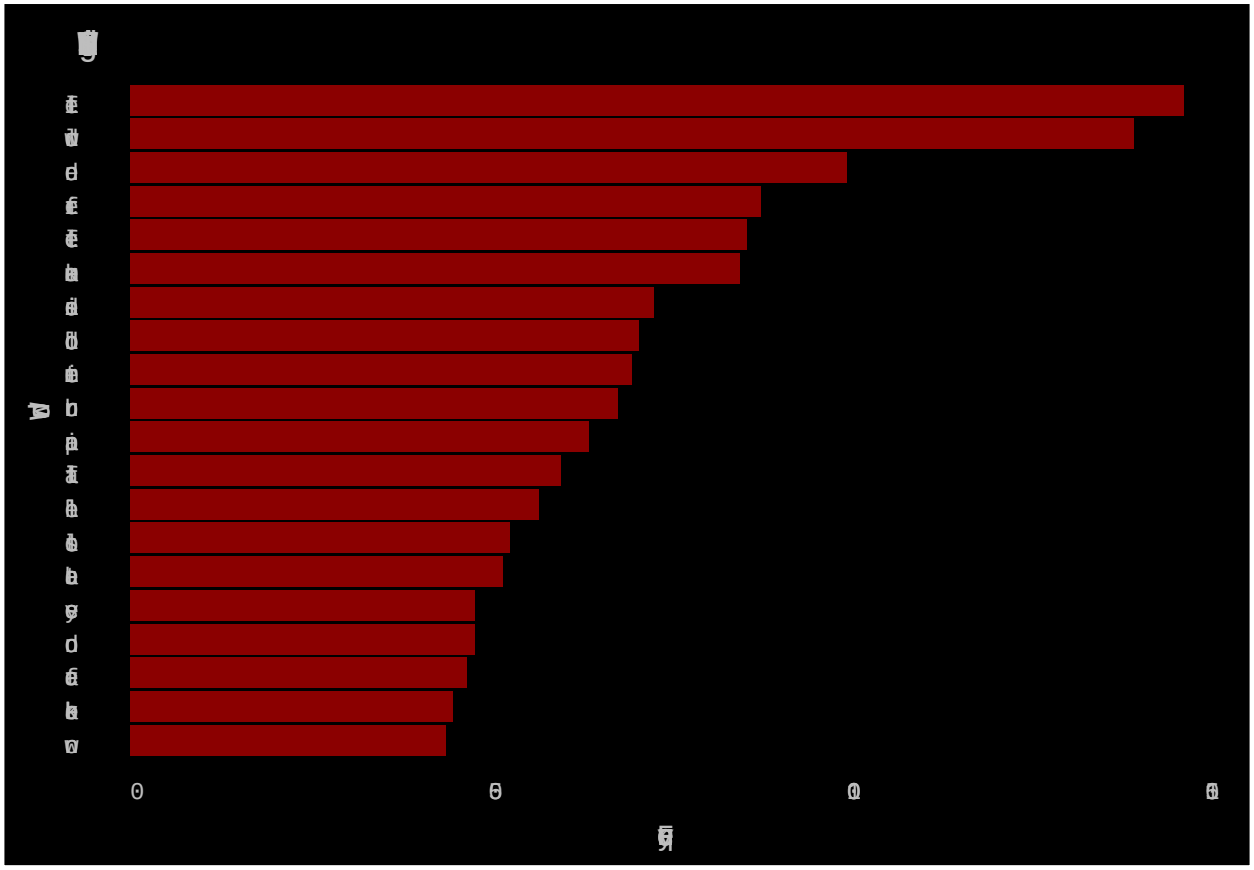
We almost got the same story, but The Crusade and Vengeance Falls also stands out with the number of unique words. Shogun again takes first because of its reference to a lot of Greek mythology. The Crusade is my favorite album and with its reference to a lot of famous killings, I am not surprised it takes second place in unique words. Vengeance is a bit surprising I would have expected In Waves or Ascendancy to be third, but come to think of it, Vengeance is less repetitive than Waves or Ascendancy.

Tread the Words

What is Trivium's most used word?

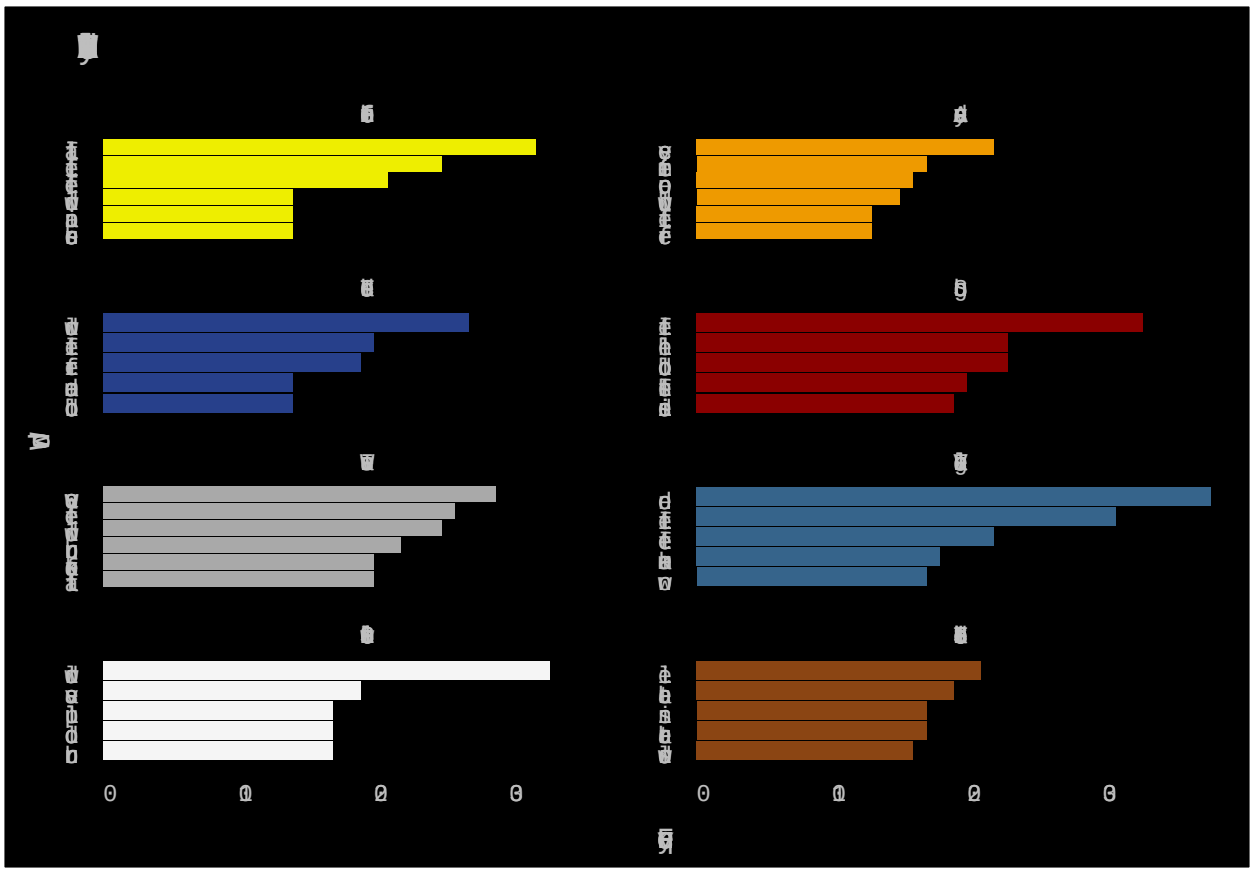
```
clean_trivium_lyrics %>%
  count(lyrics, sort = TRUE) %>%
  top_n(20, n) %>%
  ggplot(aes(reorder(lyrics, n),
             n)) +
  scale_x_reordered() +
  geom_col(fill = "darkred") +
  coord_flip() +
  labs(title = "Most Common Words in Trivium Songs",
       y = "Frequency",
```

```
      x = "Word") +
  theme_trivium() +
  scale_fill_trivium()
```



What are the most common words in each album?

```
clean_trivium_lyrics %>%
  count(album, lyrics, sort = TRUE) %>%
  group_by(album) %>%
  top_n(5, n) %>%
  ggplot(aes(reorder_within(lyrics,
                            n,
                            within = album),
             n,
             fill = factor(album,
                           levels = trivium_albums))) +
  scale_x_reordered() +
  geom_col(show.legend = FALSE) +
  facet_trivium_albums() +
  coord_flip() +
  labs(title = "Most Common Words in Every Trivium Album",
       y = "Frequency",
       x = "Word") +
  theme_trivium() +
  scale_fill_trivium()
```



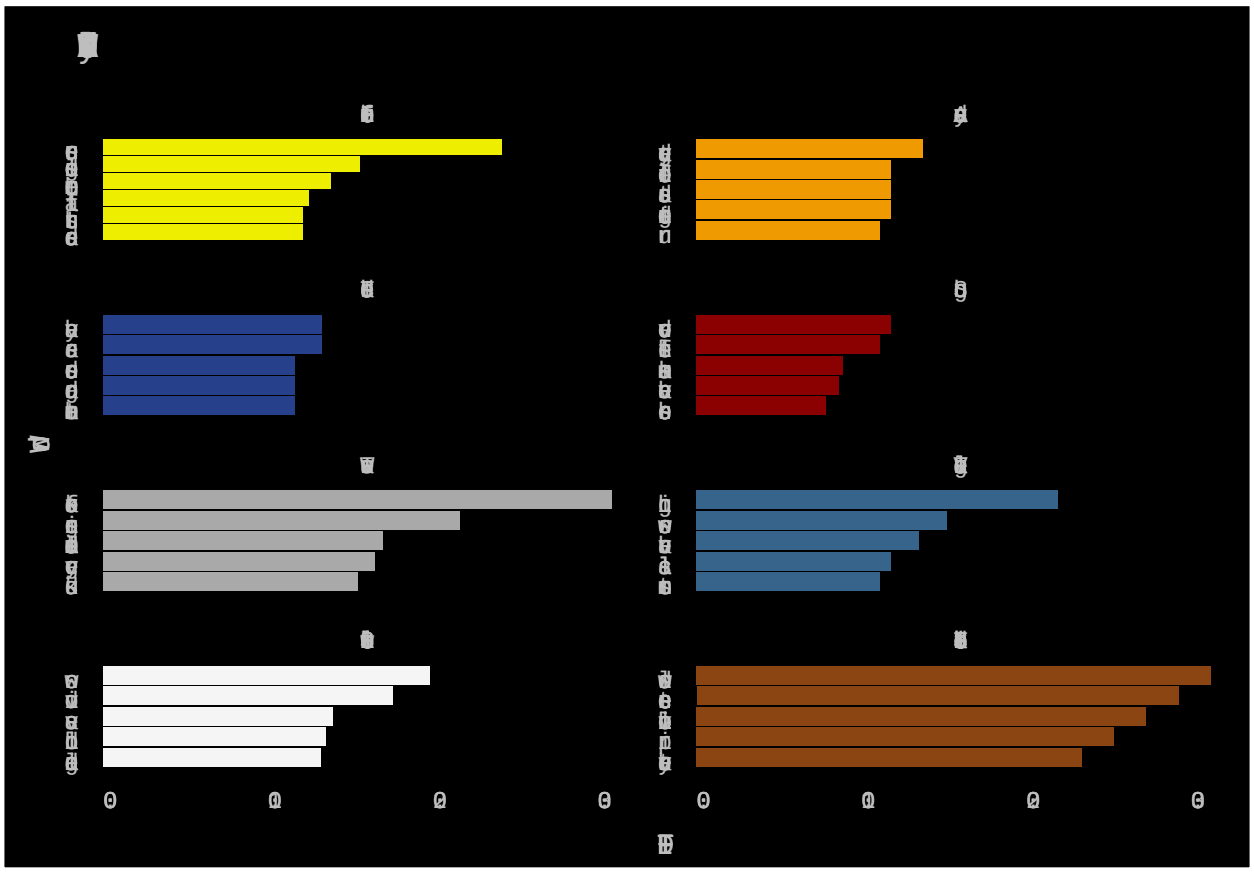
Nothing much to interpret here. It is just nice to see what words are most frequently used. Trivium seems concerned much with World and life more often as both words are quite common across multiple albums. It seems like Trivium shifted focus from a single plane of a world to a multiverse of worlds in The Sin and the Sentence, but the popularity of worlds in this album is mainly because of one song.

Torn Between Term Frequency and Inverse Document Frequency

I also wanted to know how what words are important to each album. The term frequency-inverse document frequency or tf-idf helped me to determine these words. The `tidytext` package also has the `bind_tf_idf()` that makes tf-idf analysis quick and easy. I am still new to tf-idf, but this is how I understand it: we will give points to a word every time it appears in an album, but we will also deduct points to it if it appears in other albums. In the end, the words with the biggest tf-idf are the most important or tied to an album.

```
clean_trivium_lyrics %>%
  count(album, lyrics) %>%
  bind_tf_idf(lyrics,
              album,
              n) %>%
  group_by(album) %>%
  top_n(5, tf_idf) %>%
  ggplot(aes(reorder_within(lyrics,
                            tf_idf,
                            album),
         tf_idf,
         fill = factor(album,
```

```
                      levels = trivium_albums))) +
  scale_x_reordered() +
  geom_col(show.legend = FALSE) +
  facet_trivium_albums() +
  coord_flip() +
  labs(title = "TF-IDF of Words in Every Trivium Album",
       y = "TF-IDF",
       x = "Word") +
  theme_trivium() +
  scale_fill_trivium()
```



I am not sure how to interpret this, but it seems that after Shogun, Trivium is starting to focus on specific words or themes for each album. A topic model would be helpful to verify this, but that can wait for another post.

Negativity Thrives

Another nice feature of `tidytext` is that it allows for sentiment analysis with the `sentiments` data frame. It contains four lexicons that associates a word with a sentiment or a score.

```
bing <-
  sentiments %>%
  filter(lexicon == "bing") %>%
  select(-score)
```

```
trivium_bing <-
  clean_trivium_lyrics %>%
  inner_join(bing,
             by = c("lyrics" = "word"))

trivium_bing %>%
  count(lyrics, sentiment, sort = TRUE) %>%
  head(20) %>%
  kable(caption = "Sample Bing Sentiment of Trivium Lyrics")
```

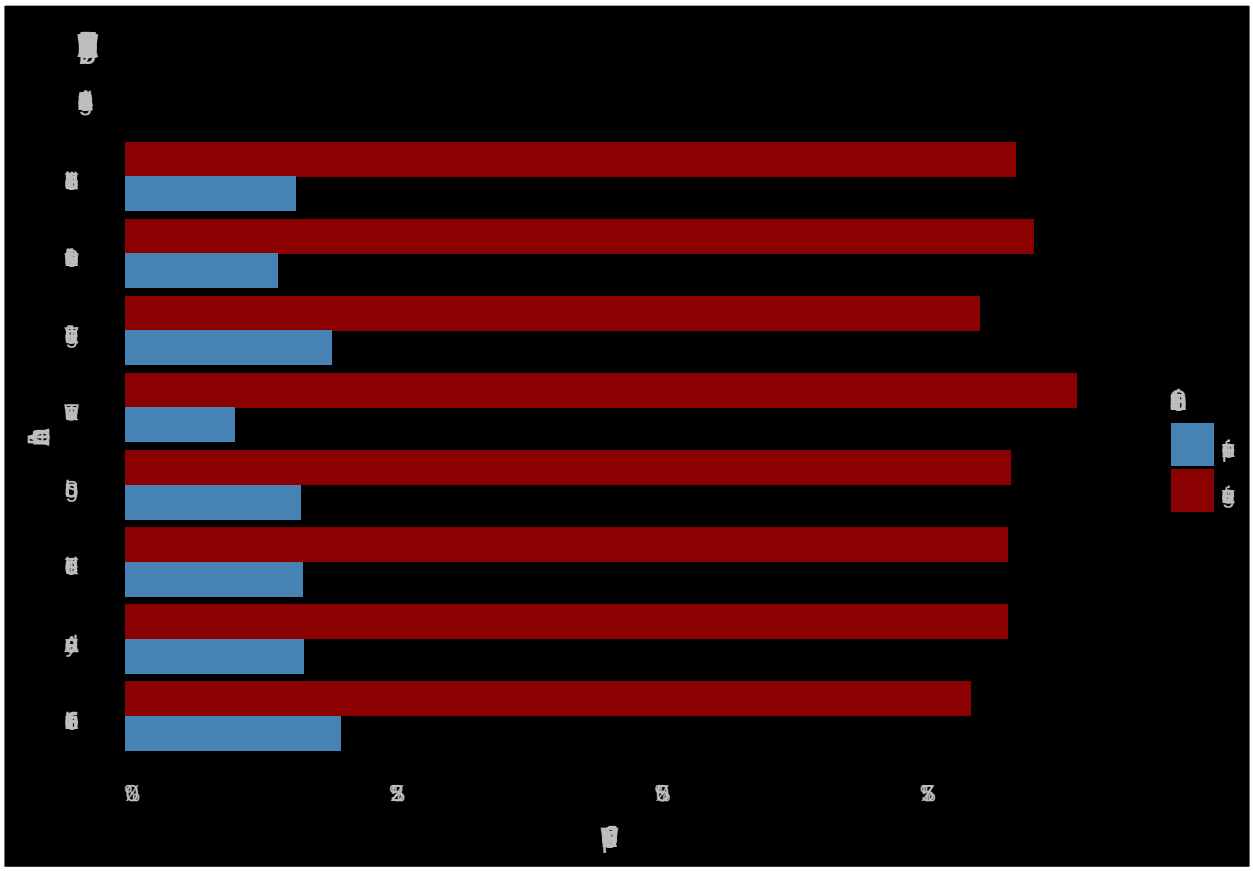
Table 2: Sample Bing Sentiment of Trivium Lyrics

lyrics	sentiment	n
burn	negative	68
pain	negative	64
fall	negative	60
hell	negative	57
lost	negative	53
hate	negative	52
break	negative	45
won	positive	44
die	negative	40
free	positive	38
cold	negative	37
fear	negative	37
dead	negative	35
death	negative	35
bleed	negative	34
lie	negative	34
dark	negative	29
sin	negative	27
burning	negative	26
broken	negative	25

First, I will look into the Bing lexicon. Note that while the Bing lexicon categorizes 6788 words into negative and positive sentiments, only 2707 lyrics out of 9864 were categorized.

```
trivium_bing %>%
  count(album,
        sentiment) %>%
  group_by(album) %>%
  mutate(prop = n / sum(n)) %>%
  ggplot(aes(factor(album,
                    levels = trivium_albums),
             prop,
             fill = factor(sentiment,
                           levels = c("positive",
                                      "negative")))) +
  geom_col(position = "dodge") +
  labs(title = "Proportion of Positive and Negative Words in Every Trivium Album",
       subtitle = "Based on Bing Liu's Sentiment Lexicon",
       x = "Album",
       y = "Proportion of Words",
```

```
      fill = "Sentiment") +
  coord_flip() +
  scale_y_percent() +
  theme_trivium() +
  scale_fill_manual(values = c("steelblue",
                               "darkred"))
```



Trivium is consistent in using more negative words than positive, not surprising for a heavy metal band.

```
afinn <-
  sentiments %>%
  filter(lexicon == "AFINN") %>%
  select(-sentiment)

trivium_afinn <-
  clean_trivium_lyrics %>%
  inner_join(afinn,
             by = c("lyrics" = "word"))

trivium_afinn %>%
  head(20) %>%
  kable(caption = "Sample AFINN Scores of Trivium Lyrics")
```

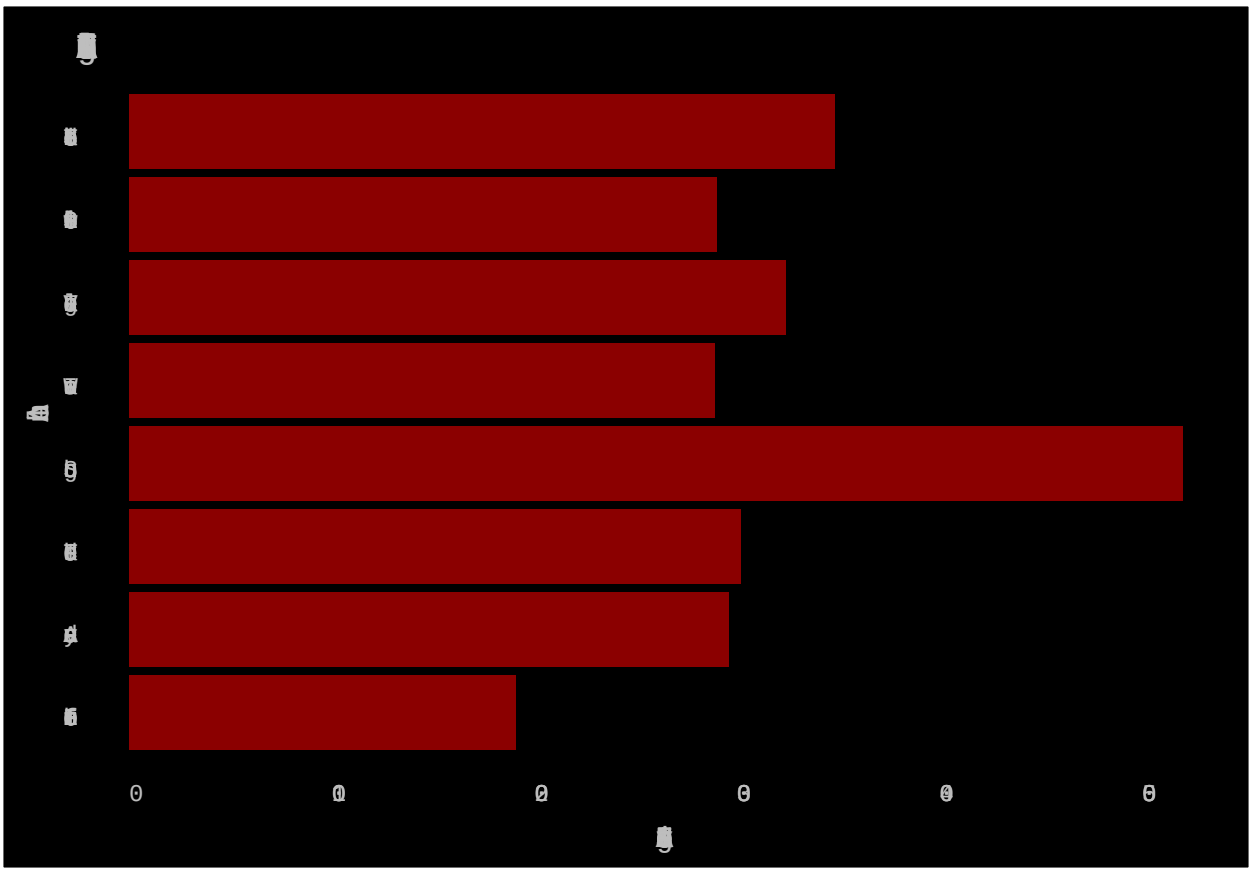
title	track_n	track_url	album

Table 3: Sample AFINN Scores of Trivium Lyrics

title	track_n	track_url	album
Pillars Of Serpents	2	https://genius.com/Trivium-pillars-of-serpents-lyrics	Ember to In
Pillars Of Serpents	2	https://genius.com/Trivium-pillars-of-serpents-lyrics	Ember to In
Pillars Of Serpents	2	https://genius.com/Trivium-pillars-of-serpents-lyrics	Ember to In
Pillars Of Serpents	2	https://genius.com/Trivium-pillars-of-serpents-lyrics	Ember to In
Pillars Of Serpents	2	https://genius.com/Trivium-pillars-of-serpents-lyrics	Ember to In
Pillars Of Serpents	2	https://genius.com/Trivium-pillars-of-serpents-lyrics	Ember to In
Pillars Of Serpents	2	https://genius.com/Trivium-pillars-of-serpents-lyrics	Ember to In
Pillars Of Serpents	2	https://genius.com/Trivium-pillars-of-serpents-lyrics	Ember to In
Pillars Of Serpents	2	https://genius.com/Trivium-pillars-of-serpents-lyrics	Ember to In
Pillars Of Serpents	2	https://genius.com/Trivium-pillars-of-serpents-lyrics	Ember to In
Pillars Of Serpents	2	https://genius.com/Trivium-pillars-of-serpents-lyrics	Ember to In
If I Could Collapse The Masses	3	https://genius.com/Trivium-if-i-could-collapse-the-masses-lyrics	Ember to In
If I Could Collapse The Masses	3	https://genius.com/Trivium-if-i-could-collapse-the-masses-lyrics	Ember to In
If I Could Collapse The Masses	3	https://genius.com/Trivium-if-i-could-collapse-the-masses-lyrics	Ember to In
If I Could Collapse The Masses	3	https://genius.com/Trivium-if-i-could-collapse-the-masses-lyrics	Ember to In
If I Could Collapse The Masses	3	https://genius.com/Trivium-if-i-could-collapse-the-masses-lyrics	Ember to In
If I Could Collapse The Masses	3	https://genius.com/Trivium-if-i-could-collapse-the-masses-lyrics	Ember to In
If I Could Collapse The Masses	3	https://genius.com/Trivium-if-i-could-collapse-the-masses-lyrics	Ember to In
If I Could Collapse The Masses	3	https://genius.com/Trivium-if-i-could-collapse-the-masses-lyrics	Ember to In
If I Could Collapse The Masses	3	https://genius.com/Trivium-if-i-could-collapse-the-masses-lyrics	Ember to In

The AFINN lexicon in the `sentiments` data frame gives integer sentiment scores from -5 to +5 for 2476 words. Note again that AFINN scored only 2261 words out of 9864 Trivium lyrics.

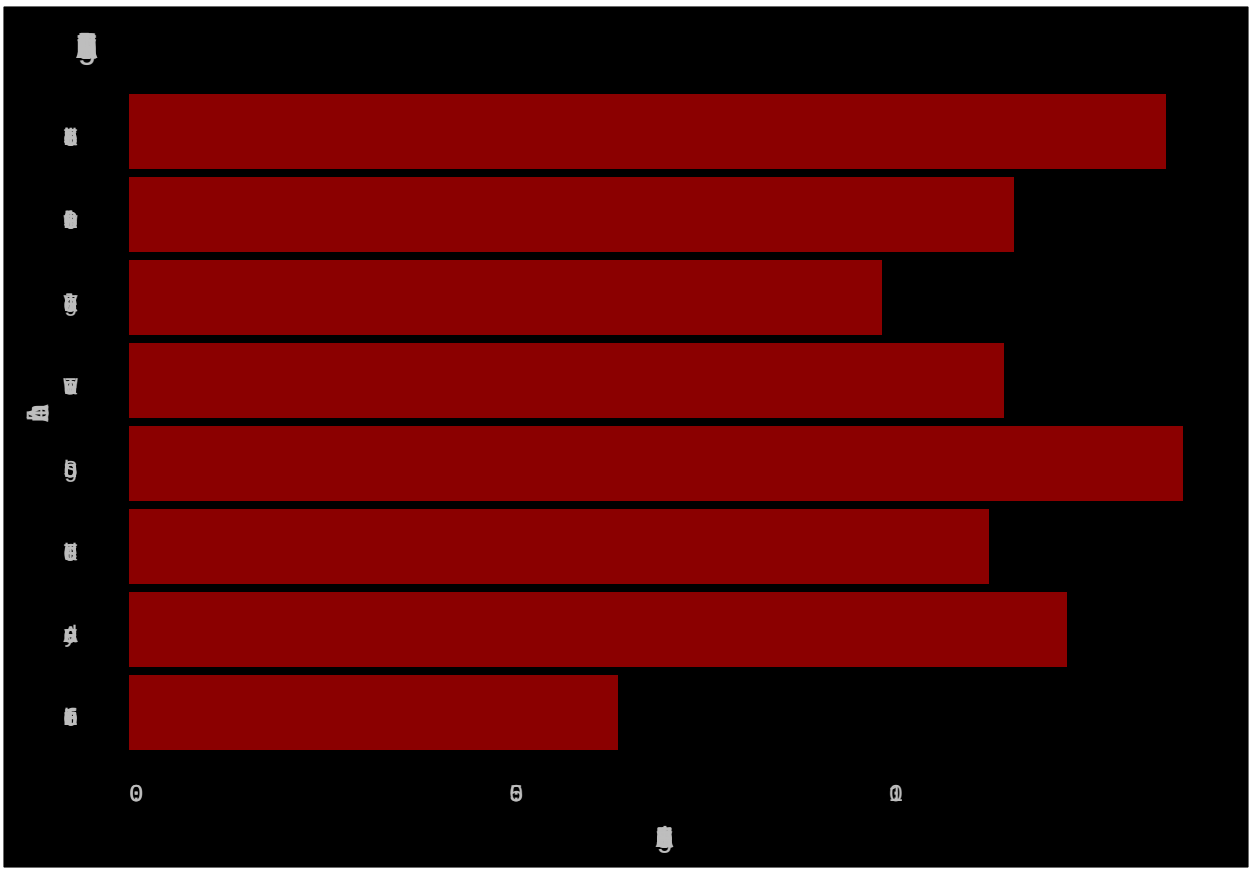
```
trivium_afinn %>%
  group_by(album) %>%
  summarise(total_score = sum(score,
                              na.rm = TRUE)) %>%
  ggplot(aes(factor(album,
                    levels = trivium_albums),
             -total_score)) +
  geom_col(fill = "darkred") +
  labs(title = "Total Negative AFINN Scores of Trivium Albums",
       x = "Album",
       y = "Negative AFINN Score") +
  coord_flip() +
  theme_trivium()
```



Again this much negative sentiment score is not unexpected in a heavy metal band. Shogun takes first again because of its word count.

Shogun has the most word count, so Shogun's big negative score might be misleading. I looked at the mean AFINN scores next.

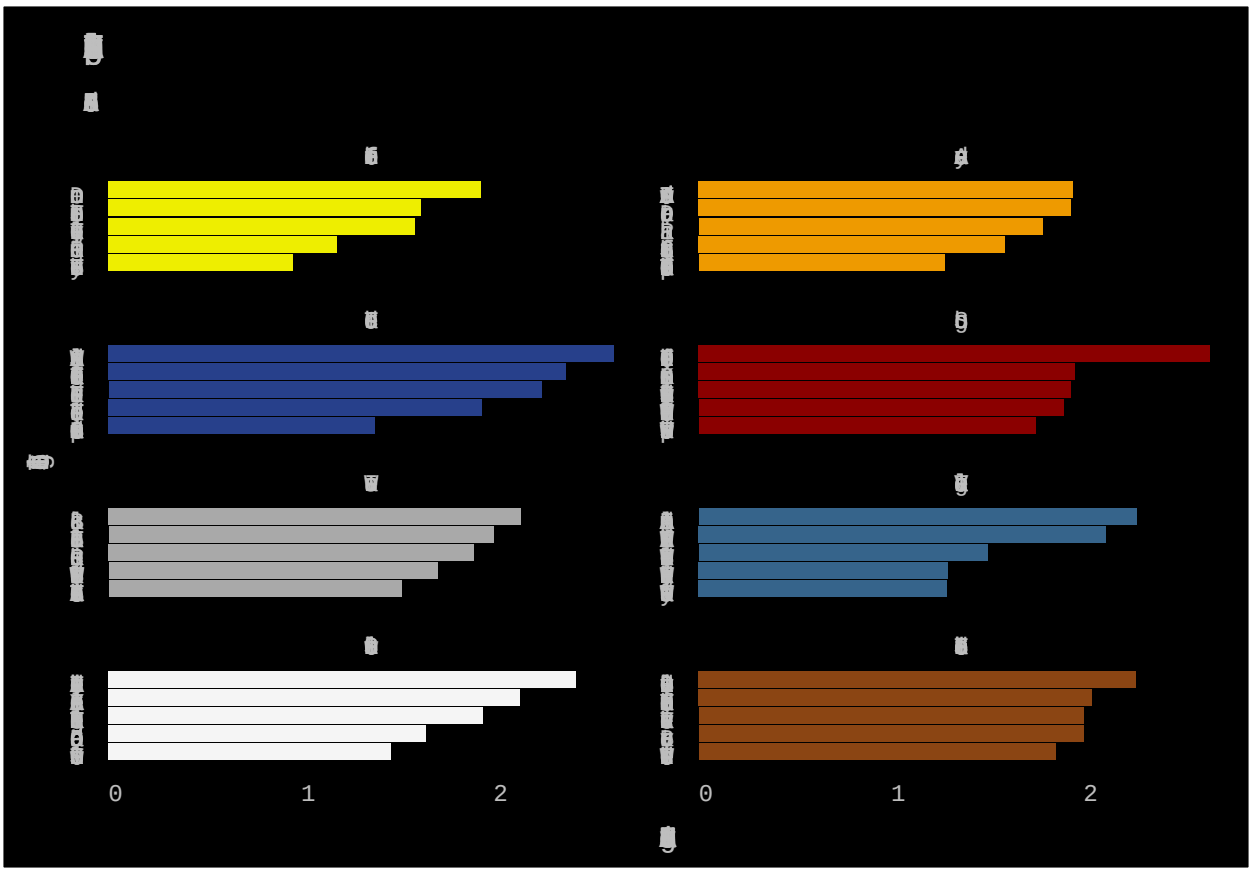
```
trivium_afinn %>%
  group_by(album) %>%
  summarise(mean_score = mean(score,
                              na.rm = TRUE)) %>%
  ggplot(aes(factor(album,
                    levels = trivium_albums),
             -mean_score)) +
  geom_col(fill = "darkred") +
  coord_flip() +
  labs(title = "Negative Mean AFINN Scores of Trivium Albums",
       x = "Album",
       y = "Negative Mean AFINN Score") +
  theme_trivium()
```



Shogun still takes the lead, but Sin is a close second.

How about songs? What Trivium songs are the most negative per album?

```
trivium_afinn %>%
  group_by(album, title) %>%
  summarise(mean_score = mean(score,
                              na.rm = TRUE)) %>%
  top_n(5, -mean_score) %>%
  ggplot(aes(reorder_within(title,
                            -mean_score,
                            album),
             -mean_score,
             fill = factor(album,
                           levels = trivium_albums))) +
  scale_x_reordered() +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  facet_trivium_albums() +
  labs(title = "Top Five Most Negative Trivum Songs per Album",
       subtitle = "Based on AFINN Scores",
       x = "Song Title",
       y = "Negative Mean AFINN Score") +
  theme_trivium() +
  scale_fill_trivium()
```

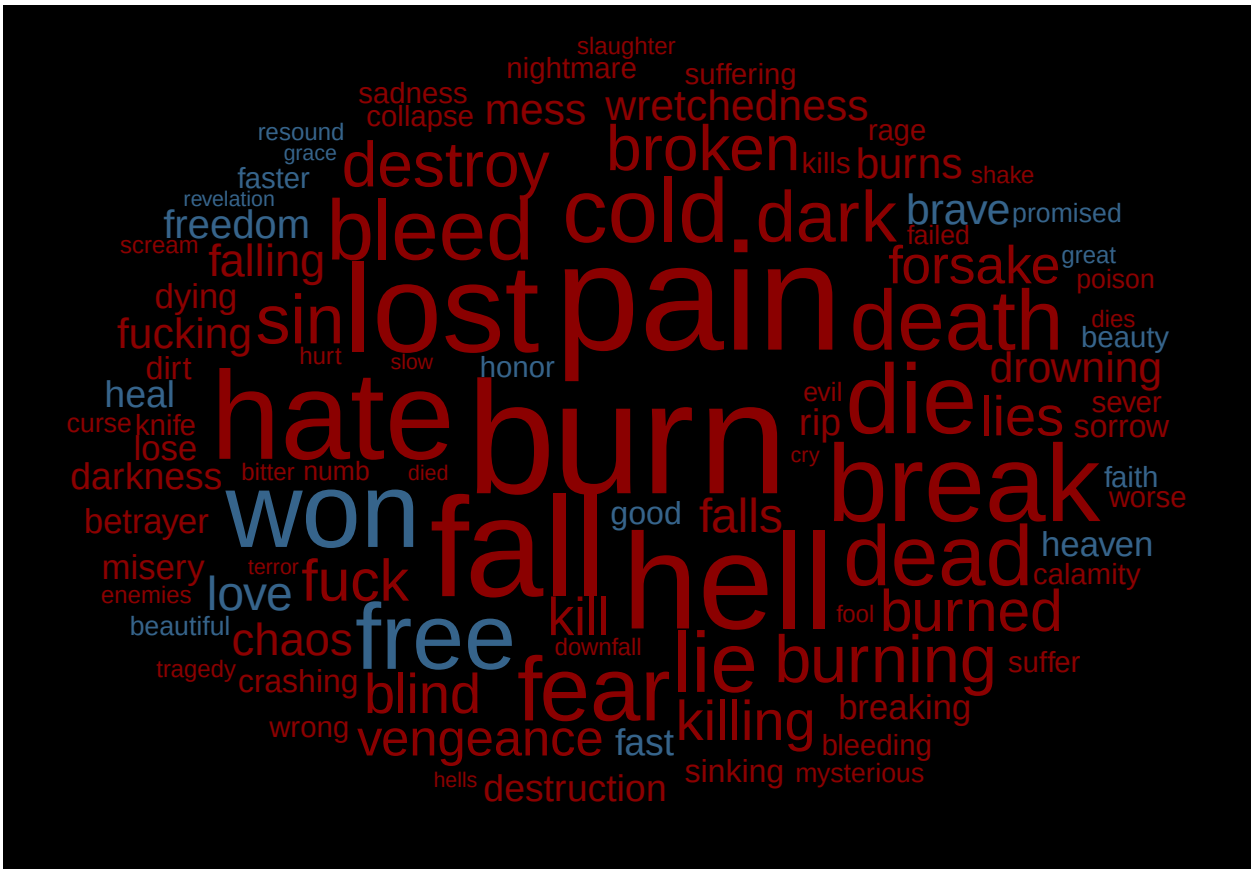


One can actually feel the negative words already just from the song titles.

These Sentiments Can't Tear Us Apart

As a parting image, let's go back to the Bing lexicon and see what negative and positive words appear most frequently in Trivium songs.

```
par(bg = "black")
trivium_bing %>%
  count(lyrics, sentiment, sort = TRUE) %>%
  with(wordcloud(lyrics,
                 n,
                 scale = c(5, 0.25),
                 ordered.colors = TRUE,
                 random.order = FALSE,
                 rot.per = 0,
                 colors = c("darkred", "steelblue4")[factor(sentiment)],
                 max.words = 100,
                 fixed.asp = FALSE))
```


slaughter
nightmare
suffering
sadness
collapse
mess
wretchedness
resound
grace
broken
rage
destroy
kills
burns
shake
faster
revelation
freedom
bleed
cold
dark
brave
promised
scream
falling
failed
forsake
great
poison
dying
sin
lost
pain
death
fucking
dies
beauty
hurt
slow
honor
drowning
dirt
heal
hate
burn
evil
die
lies
sever
curse
knife
rip
sorrow
lose
cry
darkness
bitter
numb
died
break
faith
worse
won
fall
good
falls
betrayer
hell
dead
heaven
misery
terror
fuck
calamity
enemies
love
kill
fool
burned
beautiful
chaos
free
downfall
lie
burning
suffer
tragedy
crashing
blind
fear
killing
breaking
wrong
vengeance
fast
bleeding
sinking
mysterious
hells
destruction