Text Mining and Sentiment Analysis of Song Lyrics

2023-12-05

Introduction

People passed on their thought and emotion from generation to generation through lyrics of songs, and both laughed at it and cried to it. Lyrics present us with an artist's perspective of life, society, or world. Not only the melody but also the lyrics carry the mood of the times. That's why I want to do the text mining and sentiment analysis of song lyrics.

In my project, there are the English songs in 1980 to 2023. I attempted to analyze the unique characteristics of song lyrics. First, I examined some descriptive statistics about the database, then took a closer look at the lyrics thoughts the decades or different music type, also applied sentiment analysis to them. Next, tried to examine closer the importance of words, by the decades, the types, or specific artists. Finally, performed topic modeling using Latent Dirichlet Allocation (LDA).

Library the functions I need:

```
library(readr)
library(tm)
library(SnowballC)
library(wordcloud)
library(wordcloud2)
library(RColorBrewer)
library(syuzhet)
library(ggplot2)
library(ggraph)
library(igraph)
library(dplyr)
library(tidytext)
library(tidyr)
library(widyr)
library(textTinyR)
library(reshape2)
library(htmltools)
library(htmlwidgets)
library(quanteda)
library(textdata)
library(gridExtra)
library(topicmodels)
library(purrr)
```

Read the data:

```
# roughly processed data
df = read_csv("C:/Users/phoebe/Documents/R file/processed_data3.csv")
# lyrics processed data
df2 = read_csv("C:/Users/phoebe/Documents/R file/processed_data4.csv")
# title word separately data: 1 column
freq_title = read.csv("C:/Users/phoebe/Documents/R file/title_processed_data.csv")
# lyrics word separately data: 1 column
freq_lyrics = read.csv("C:/Users/phoebe/Documents/R file/lyrics_processed_data.csv")
# df2_words = df2 %>% unnest_tokens(word, lyrics)
# lyrics to words data
df2_words = read.csv("C:/Users/phoebe/Documents/R file/lyrics_to_word_clean.csv")
```

I already clean the data set and process it into several types of data set that I need. Did not show the processed at here is because that the original data set is too large for my notebook to knit. The process I do include:

1. delete the lyrics that are not English.
2. delete the views smaller than 100K, which are less important to represent the decades.
3. Cleaning the lyrics, Such as transform all the capital letter to small. Delete the stopwords, marks, onomatopoeia, be-verb, number, etc.

```
summary(df2)
```

```
##     title               tag               artist               year
##  Length:17598       Length:17598       Length:17598       Min.   :1980
##  Class :character   Class :character   Class :character   1st Qu.:2012
##  Mode  :character   Mode  :character   Mode  :character   Median :2016
##                                                           Mean   :2013
##                                                           3rd Qu.:2018
##                                                           Max.   :2023
##      views              lyrics              decade
##  Min.   :  100002   Length:17598       Min.   :1980
##  1st Qu.:  134384   Class :character   1st Qu.:2010
##  Median :  201078   Mode  :character   Median :2010
##  Mean   :  392287                      Mean   :2008
##  3rd Qu.:  376573                      3rd Qu.:2010
##  Max.   :17575634                      Max.   :2020
```

After cleaning, there are 17,598 songs. There are 7 columns:

1. title: record the song name
2. tag: record 6 song types, include Country, Miscellaneous, Pop, Rap, R&B, Rock.
3. artist: record the singer or band names
4. year: show the publish year for each song
5. views: this data set is download from kaggle, which is collected from a lyrics website. The number of the views is how much times people view this song in that website.
6. lyrics: the lyrics for each song
7. decade: I add this column to separate the song by generation, according to the column year

```
summary(df2_words)
```

```
##     title               tag               artist               year
##  Length:2849039     Length:2849039     Length:2849039     Min.   :1980
##  Class :character   Class :character   Class :character   1st Qu.:2012
##  Mode  :character   Mode  :character   Mode  :character   Median :2015
##                                                           Mean   :2013
##                                                           3rd Qu.:2018
##                                                           Max.   :2023
##      views              decade         word
##  Min.   :  100002   Min.   :1980   Length:2849039
##  1st Qu.:  135783   1st Qu.:2010   Class :character
##  Median :  207619   Median :2010   Mode  :character
##  Mean   :  417768   Mean   :2008
##  3rd Qu.:  392836   3rd Qu.:2010
##  Max.   :17575634   Max.   :2020
```

“df2_words” are from “df2”, separate the lyrics to each word. After cleaning the lyrics again, there are 2,849,039 words. In df2_words columns 1 to 6 are title, tag, artist, year, views, decade, all of them are same as df2. In column 7, record words in each song.

Data Visualize

There are million rows in my data set. Visualize it first can help reading it better.

Word Cloud

Top 50 frequency words show in song title

```
word_freq_title = table(freq_title$word)
sorted_title_freq = sort(word_freq_title, decreasing = TRUE)
top_title = head(sorted_title_freq, 50)
title_cloud = wordcloud2(top_title, backgroundColor = "#F8F9F9")
saveWidget(title_cloud, file = "C:/Users/phoebe/Documents/R file/title_cloud_output.html")
html_title_cloud = tags$iframe(src = "C:/Users/phoebe/Documents/R file/title_cloud_output.html", width =
"800", height = "600")
html_title_cloud
```

The word “love” appear the most (435 times). Second and third are “don’t” (205) and “like” (189). Most of the words in the song title are short. Maybe it’s because the artist want people to search and remember their song name easily.

Top 50 frequency words show in song lyrics

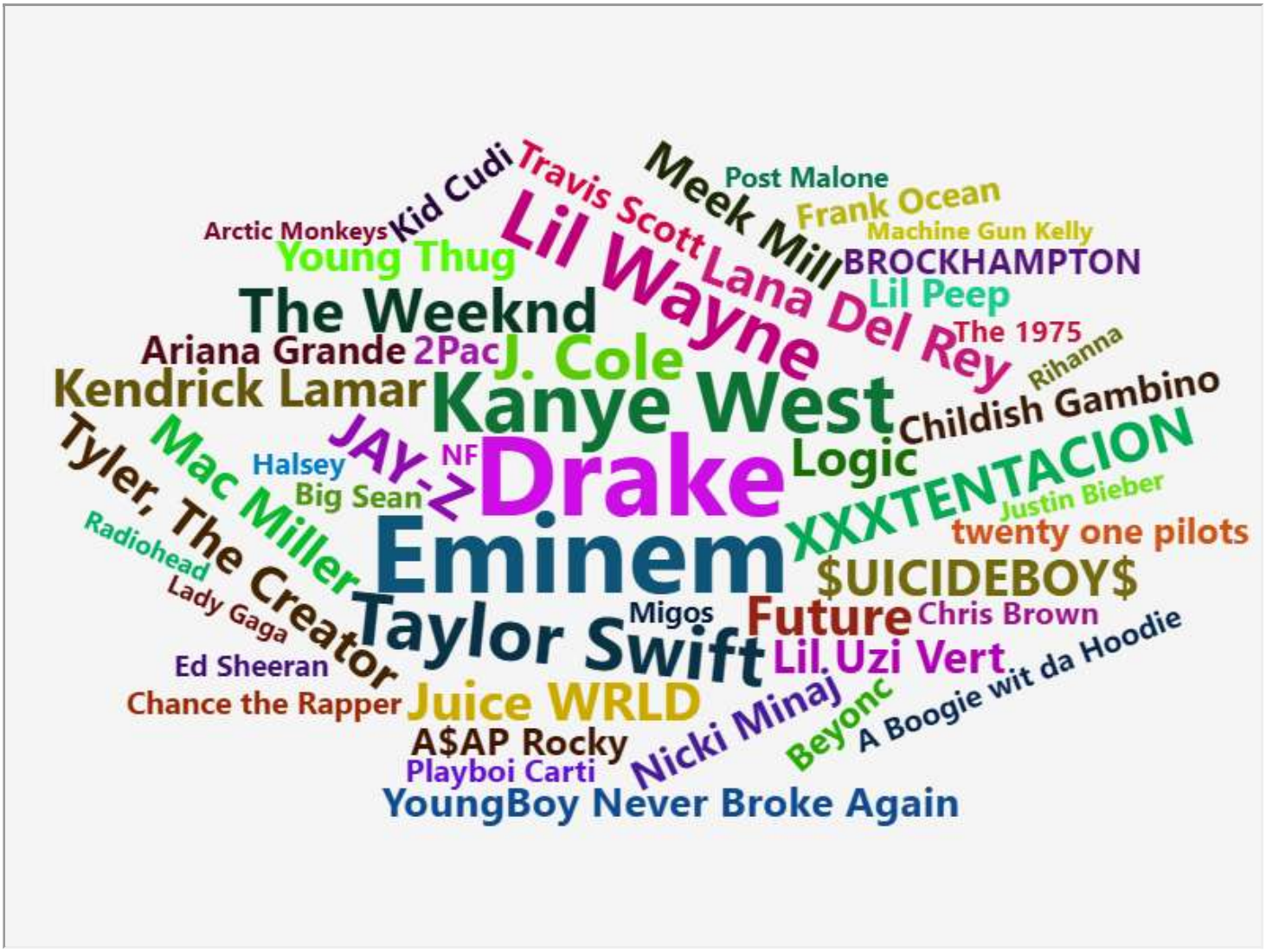
```
word_freq_lyrics = table(freq_lyrics$word)
sorted_lyrics_freq = sort(word_freq_lyrics, decreasing = TRUE)
top_lyrics = head(sorted_lyrics_freq, 150)
lyrics_cloud = wordcloud2(top_lyrics, backgroundColor = "#F8F9F9", size = 1)
saveWidget(lyrics_cloud, file = "C:/Users/phoebe/Documents/R file/lyrics_cloud_output.html")
html_lyrics_cloud = tags$iframe(src = "C:/Users/phoebe/Documents/R file/lyrics_cloud_output.html", width =
"800", height = "600")
html_lyrics_cloud
```

There are some bad words that appear in the song pretty often. Also can find that the words show up the most are all simple, common and short.

Note: In the lyrics, there are lot of similar words like “feelin” and “feeling”, “fuckin” and “fuck” etc. I transform these similar words into one kind of word to represent.

Top 50 artists

```
text = paste(df$artist, collapse = "###")
word_freq_artist = table(strsplit(text, "###"))
top_artist = head(sort(word_freq_artist, decreasing = TRUE), 50)
artist_cloud = wordcloud2(top_artist, size = 0.4, backgroundColor = "#F8F9F9")
saveWidget(artist_cloud, file = "C:/Users/phoebe/Documents/R file/artist_cloud_output.html")
html_artist_cloud = tags$iframe(src = "C:/Users/phoebe/Documents/R file/artist_cloud_output.html", width =
"800", height = "600")
html_artist_cloud
```



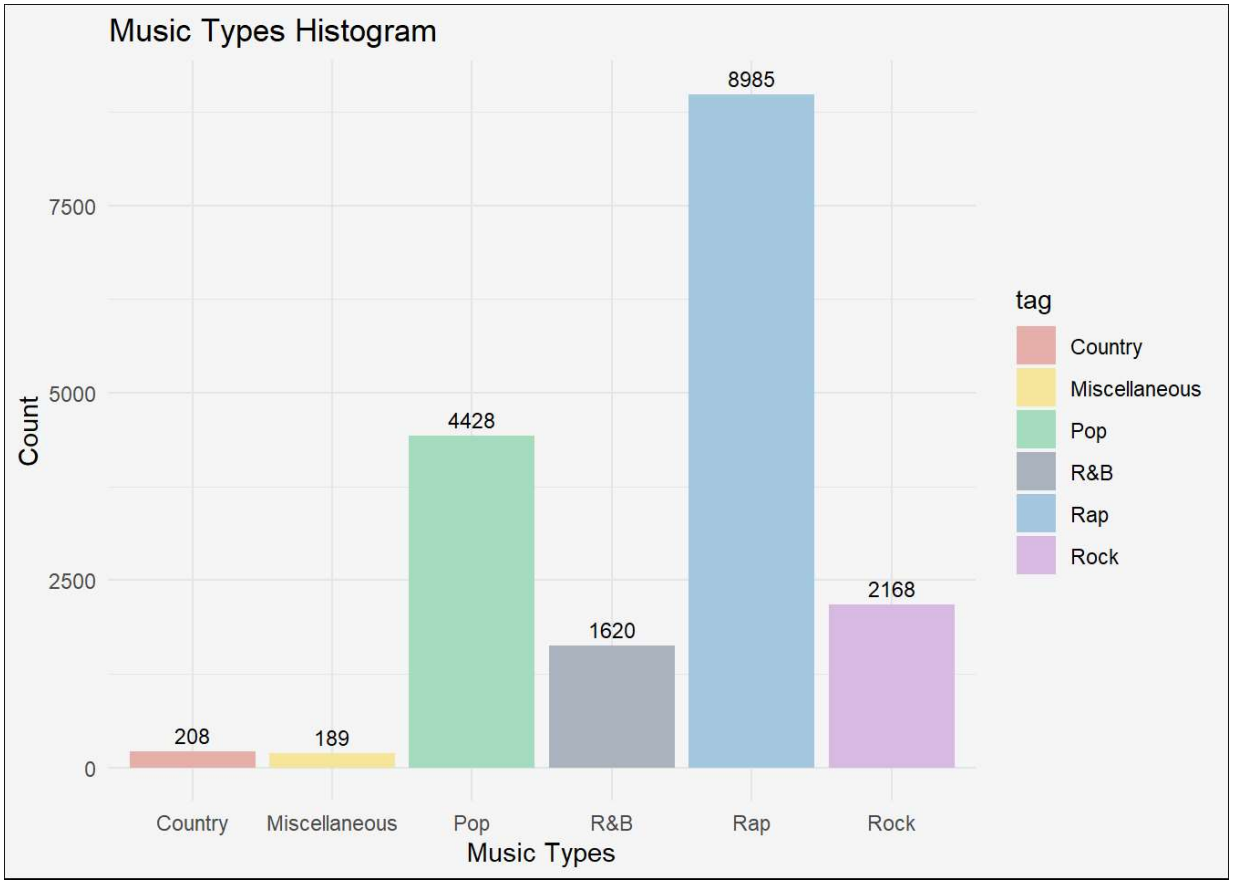
Top 50 artists are select by the number of the songs they published. Probably almost everyone is able to find there at least a few artists they know.

Analysis Music Types

Number of the song in each tag

Visualize the song type such as “pop”, “country”, “rap” etc.

```
tag_counts = table(df$tag)
tag_df = data.frame(tag = c("Country", "Miscellaneous", "Pop", "Rap", "R&B", "Rock"), count = as.numeric(t
ag_counts))

ggplot(data = tag_df, aes(x = tag, y = count, fill = tag)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Music Types Histogram", x = "Music Types", y = "Count") +
  geom_text(aes(label = count), position = position_dodge(width = 0.9), vjust = -0.5, size = 3) +
  theme_minimal() +
  scale_fill_manual(values = c("#E6B0AA", "#F9E79F", "#A9DFBF", "#AEB6BF", "#A9CCE3", "#D7BDE2")) +
  theme(plot.background = element_rect(fill = "#F8F9F9"), panel.grid = element_line(color = "#E5E7E9"))
```

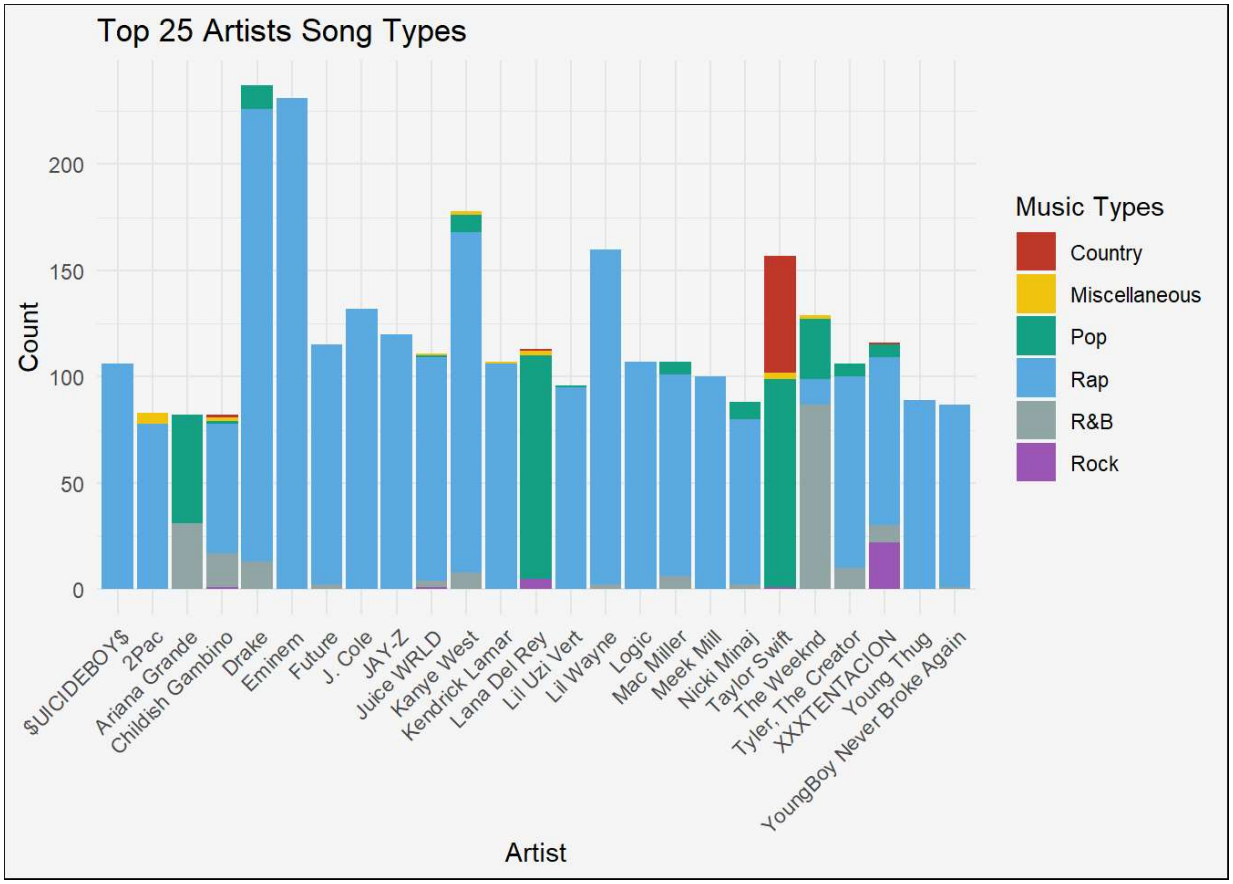


The Histogram shows that “rap” is the largest type, and much larger than the others. “Country” and “Miscellaneous” (remix) are much smaller than the others.

Top 25 artists and their music type

For top 25 artists, analysis their song type.

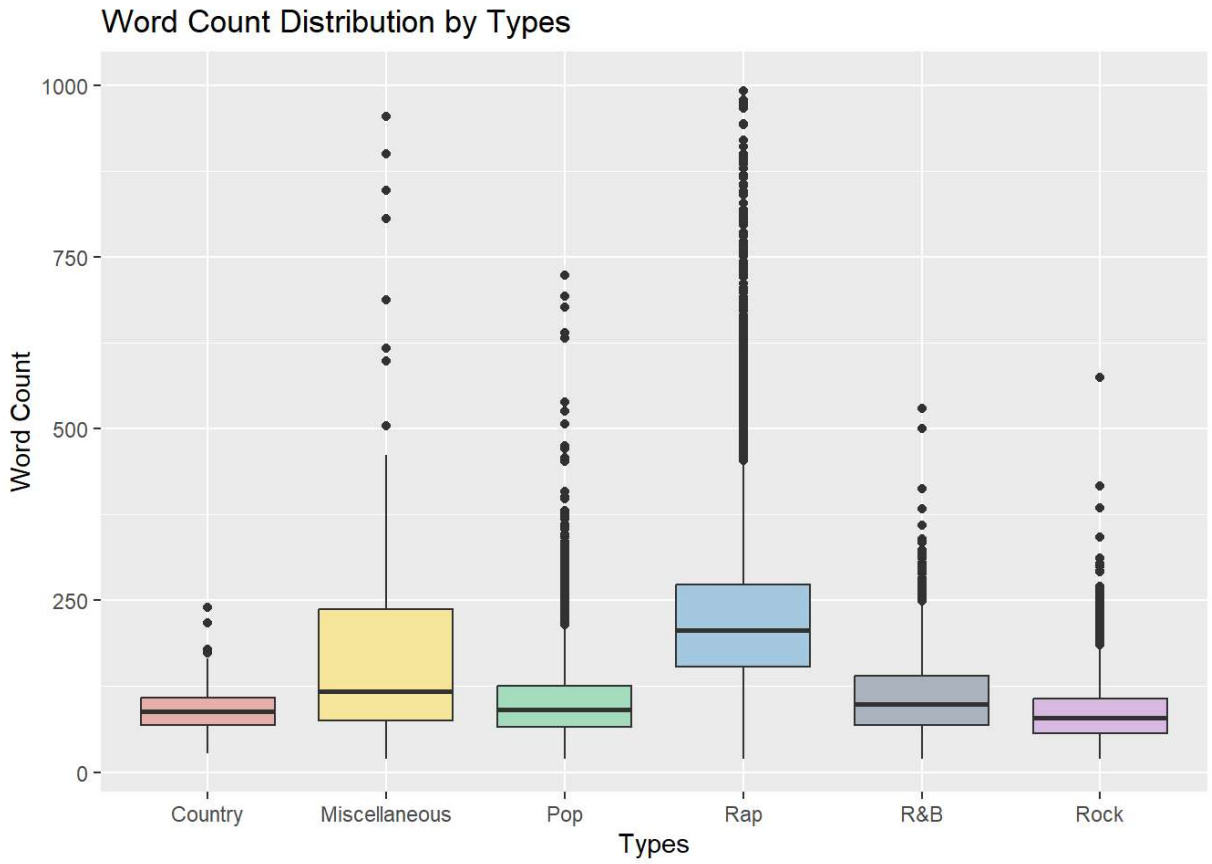
```
top_25 = head(sort(word_freq_artist, decreasing = TRUE), 25)
top_artist = as.character(names(top_25))
top_25_artist = df %>% filter(artist %in% top_artist)
artist_tag_counts = top_25_artist %>% group_by(artist, tag) %>% count()

ggplot(artist_tag_counts, aes(x = artist, y = n, fill = tag)) +
  geom_bar(stat = "identity", position = "stack") +
  labs(title = "Top 25 Artists Song Types", x ="Artist", y ="Count", fill = "Music Types") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_manual(
    values = c("#C0392B", "#F1C40F", "#16A085", "#5DADE2", "#95A5A6", "#9B59B6"),
    breaks = c("country", "misc", "pop", "rap", "rb", "rock"),
    labels = c("Country", "Miscellaneous", "Pop", "Rap", "R&B", "Rock")) +
  theme(plot.background = element_rect(fill = "#F8F9F9"), panel.grid = element_line(color = "#E5E7E9"))
```



From the histogram can see that rapper is the largest group in the top 25 artists. The reason behind might is that rappers publish song quicker and larger than others, or it is due to the population who listen to rap is the biggest group. Another interest thong is that most of the artists are specialization in one kind of music.

Words number in each tag

```
df2_word_count = df2_words %>% group_by(tag, title) %>% summarise(word_count = n())
ggplot(df2_word_count, aes(x = tag, y = word_count, fill = tag)) +
  geom_boxplot(show.legend = FALSE) +
  labs(x = 'Types', y = 'Word Count', title = 'Word Count Distribution by Types') +
   scale_y_continuous(limits = c(20, 1000)) +
  scale_fill_manual(values = c("#E6B0AA", "#F9E79F", "#A9DFBF", "#A9CCE3", "#AEB6BF", "#D7BDE2")) +
  scale_x_discrete(labels = c("Country", "Miscellaneous", "Pop", "Rap", "R&B", "Rock"))
```

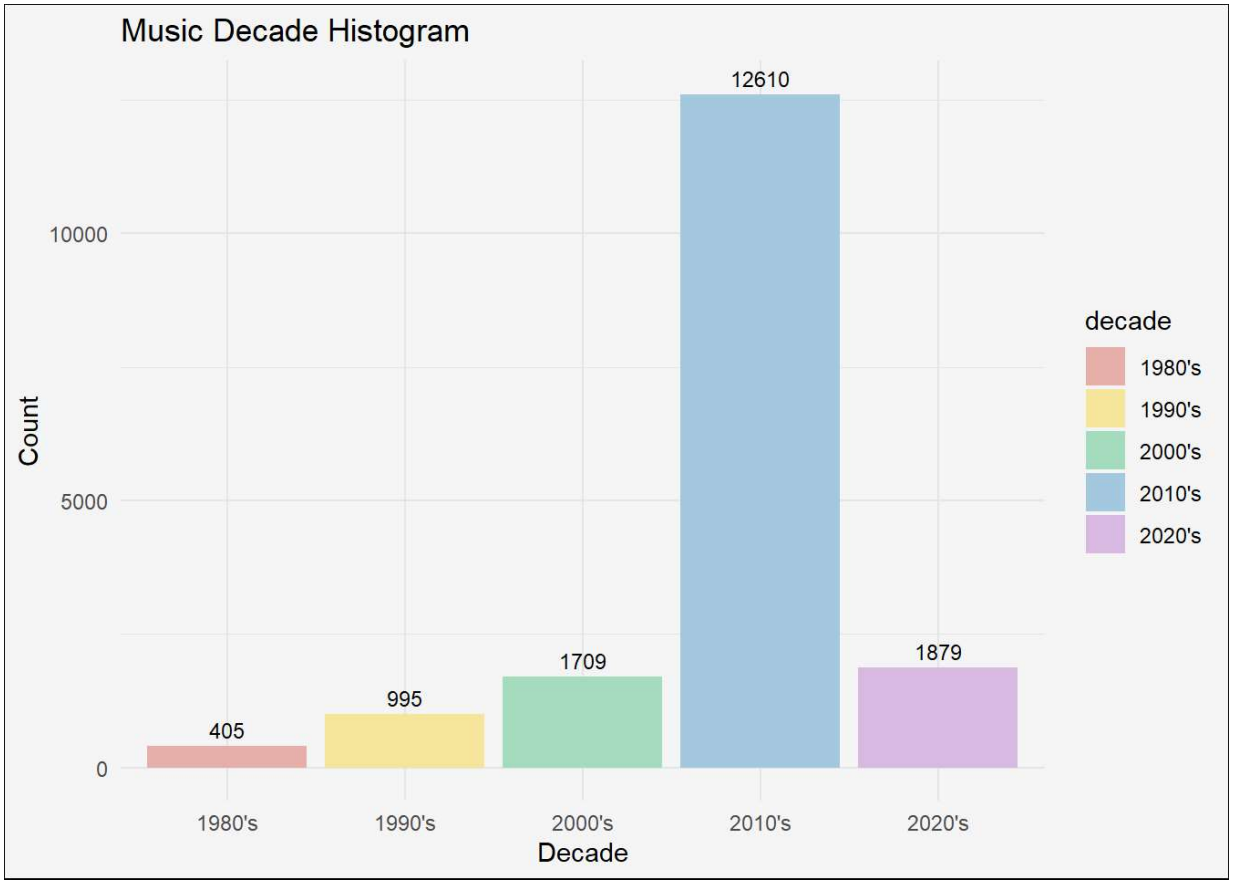


The box plot show that in “rap”, the lyrics are often longer than other types of music. From the extreme values, there are some songs has much longer lyrics in “pop” and “rock”.

Analyze Decade

Number of the song in each decade

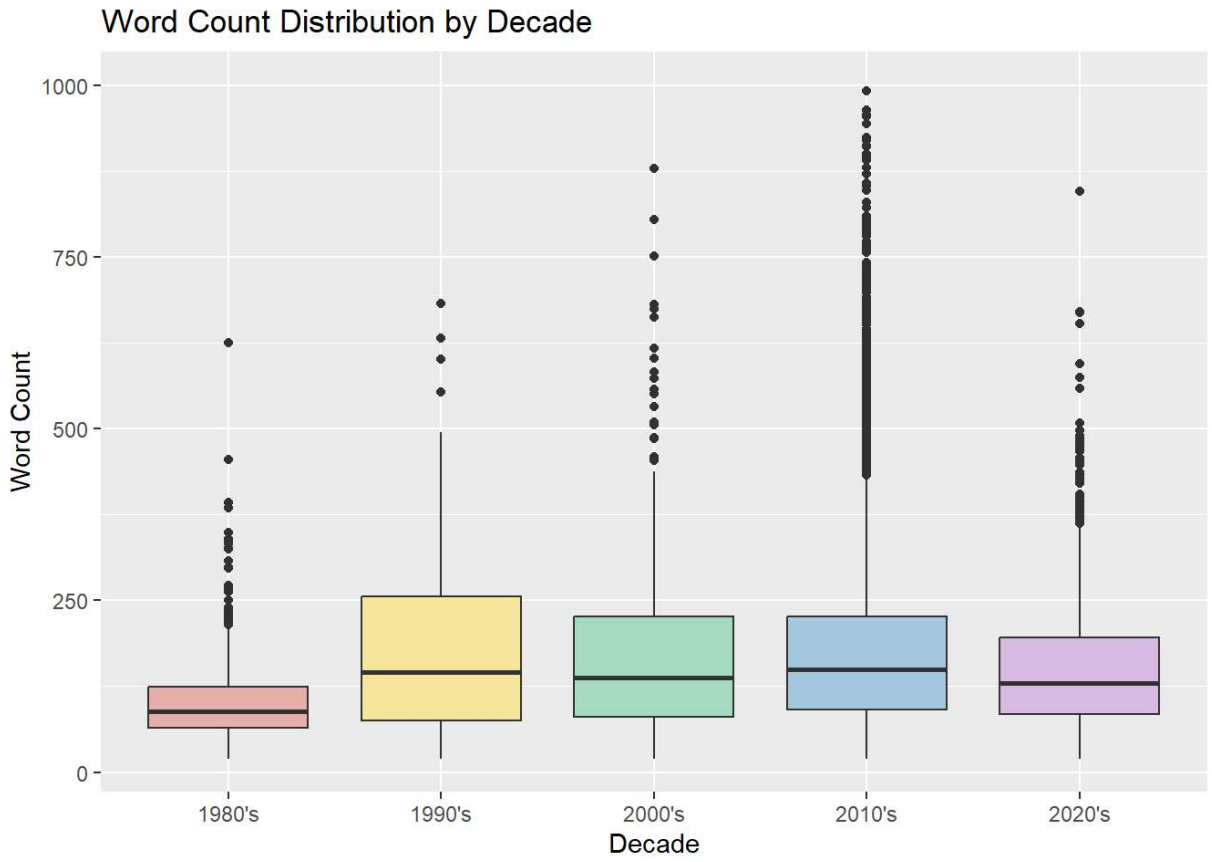
```
decade_counts = table(df$decade)
decade_df = data.frame(decade = c("1980's", "1990's", "2000's", "2010's", "2020's"), count = as.numeric(de
cade_counts))

ggplot(data = decade_df, aes(x = decade, y = count, fill = decade)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Music Decade Histogram", x = "Decade", y = "Count") +
  geom_text(aes(label = count), position = position_dodge(width = 0.9), vjust = -0.5, size = 3) +
  theme_minimal() +
  scale_fill_manual(values = c("#E6B0AA", "#F9E79F", "#A9DFBF", "#A9CCE3", "#D7BDE2")) +
  theme(plot.background = element_rect(fill = "#F8F9F9"), panel.grid = element_line(color = "#E5E7E9"))
```



It's not surprise that there are much more songs in 2010's due to the filter of views. 2020's has only 4 years, but are already become second in this histogram.

Words number in each decade

```
df2_word2_count = df2_words %>% group_by(decade, title) %>% summarise(word_count = n())
df2_word2_count$decade = as.character(df2_word2_count$decade)
ggplot(df2_word2_count, aes(x = decade, y = word_count, fill = decade)) +
  geom_boxplot(show.legend = FALSE) +
  labs(x = 'Decade', y = 'Word Count', title = 'Word Count Distribution by Decade') +
   scale_y_continuous(limits = c(20, 1000)) +
  scale_fill_manual(values = c("#E6B0AA", "#F9E79F", "#A9DFBF", "#A9CCE3", "#D7BDE2")) +
  scale_x_discrete(labels = c("1980's", "1990's", "2000's", "2010's", "2020's"))
```



It's interest to see that the lyrics in 1980's are shorter than other decades obviously. After 1990, the length of the lyrics are almost the same.

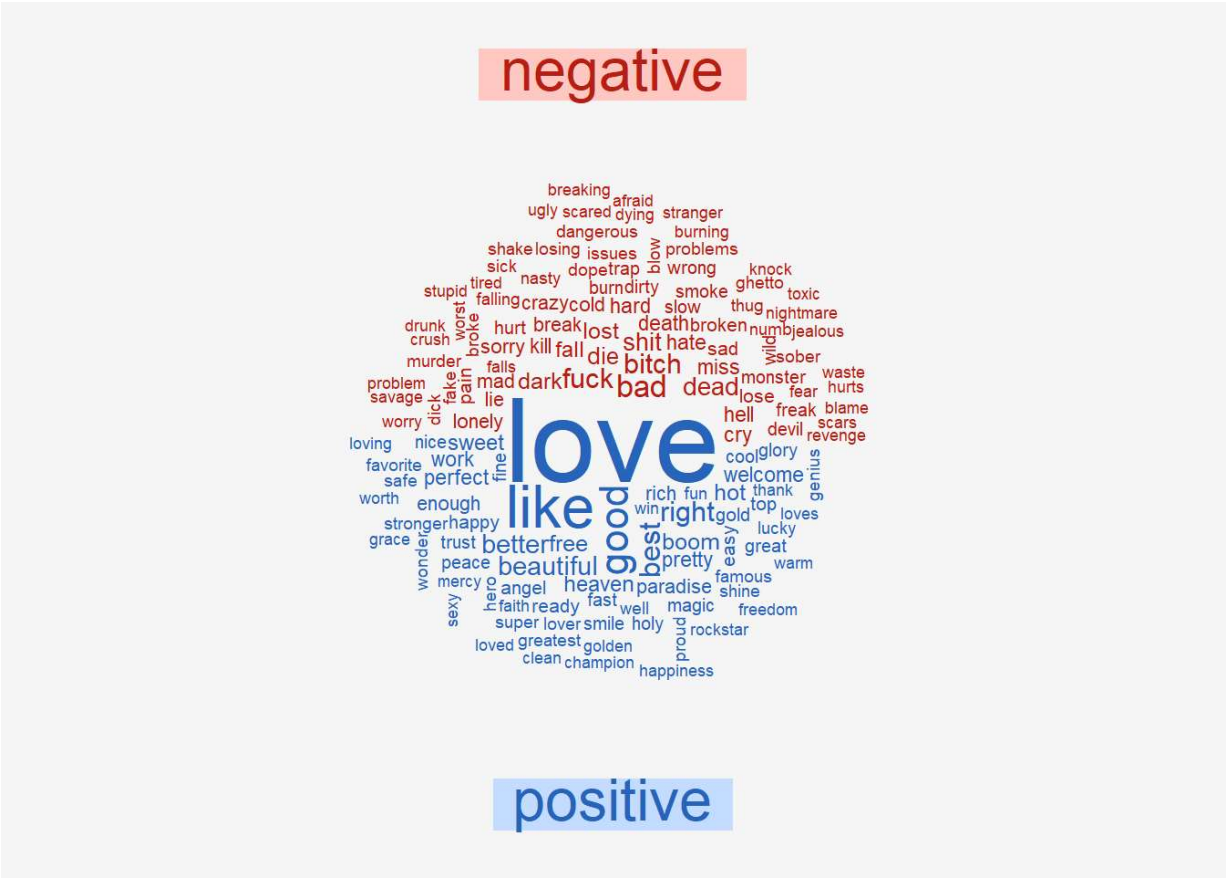
Sentiment Analysis

In this part, I used the tidytext package since it contains sentiment lexicons that are based on single words. The first one, the Bing lexicon, categorizes words into positive and negative. The NRC lexicon, in turn, categorizes them in a binary way into more detailed eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive). Finally, the AFINN lexicon is the one that assigns a score for each word in the range from -5 to 5, where negative scores indicate negative sentiment, while a positive score indicates positive sentiment.

Positive words vs. Negative words in song title

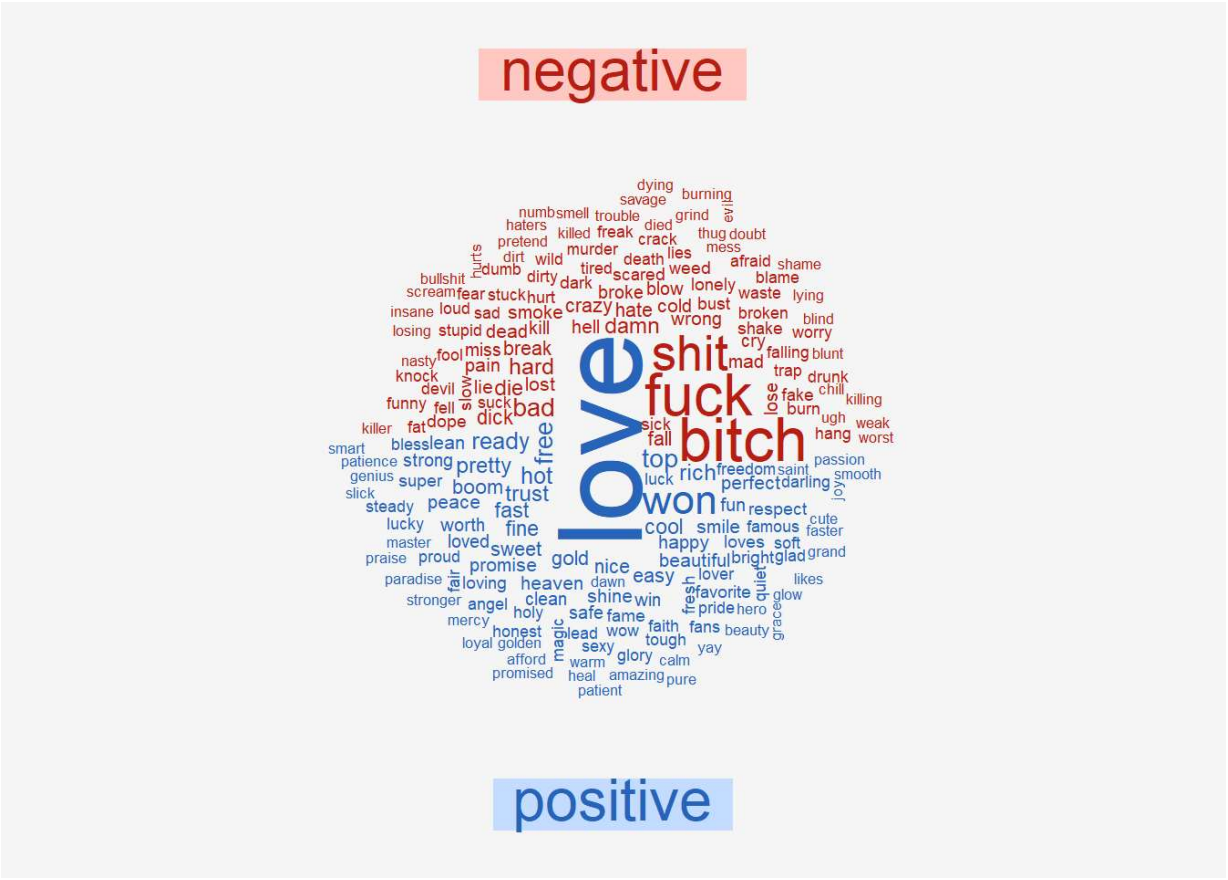
Use Bing lexicon to label the words in song title with positive or negative. Color red and blue represent negative and positive separately. Choose top 150 words to show in the word cloud, note that some of the words won't be show due to there are not in the Bing lexicon.

```
title_freq_bing = freq_title
title_sentiments = inner_join(title_freq_bing, sentiments, by = "word")
par(bg = "#F8F9F9")
title_sentiments %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("#BC2014", "#2866BD"), max.words = 150, title.size = 2, match.colors = TRUE,
title.bg.colors=c("#FFCBC7","#C7DEFF"))
```



Positive words vs. Negative words in lyrics

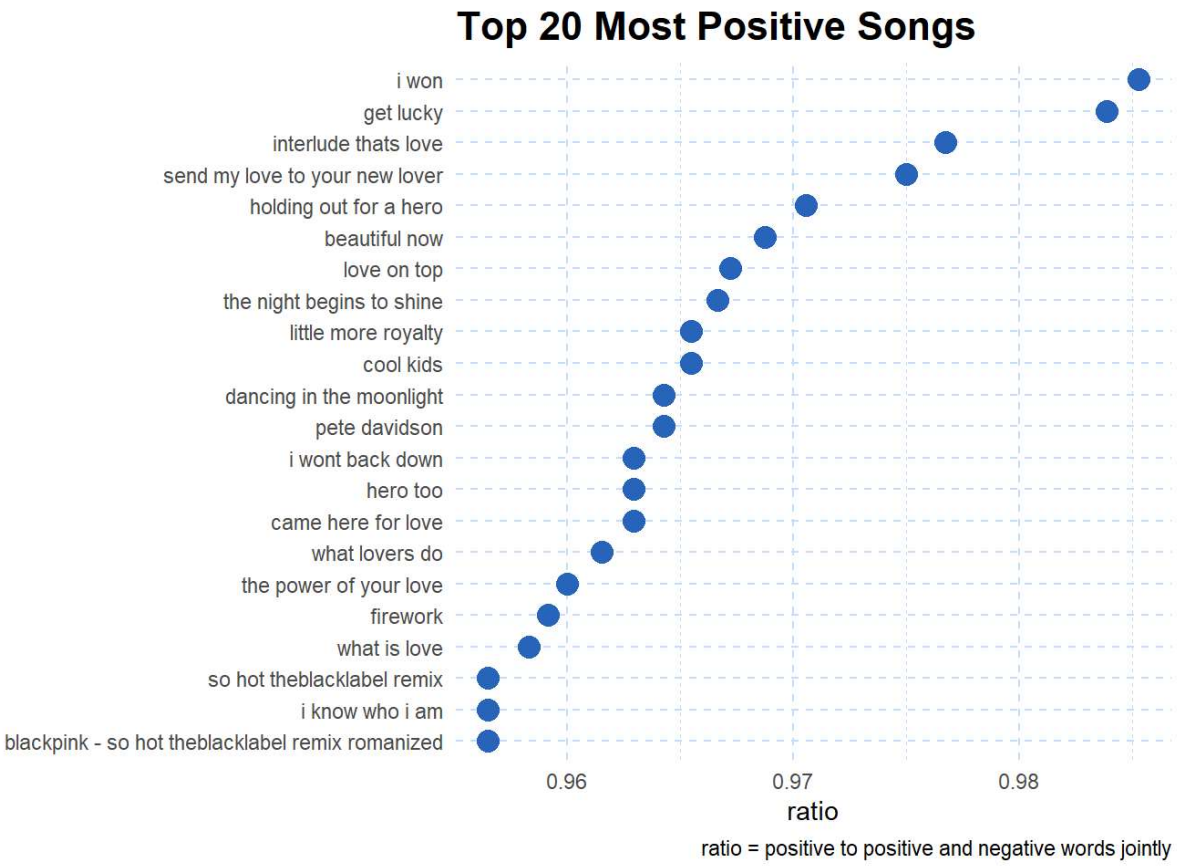
```
lyrics_freq_bing = freq_lyrics
lyrics_sentiments = inner_join(lyrics_freq_bing, sentiments, by = "word")
par(bg = "#F8F9F9")
lyrics_sentiments %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("#BC2014", "#2866BD"), max.words = 200, title.size = 2, match.colors = TRUE,
title.bg.colors=c("#FFCBC7","#C7DEFF"))
```



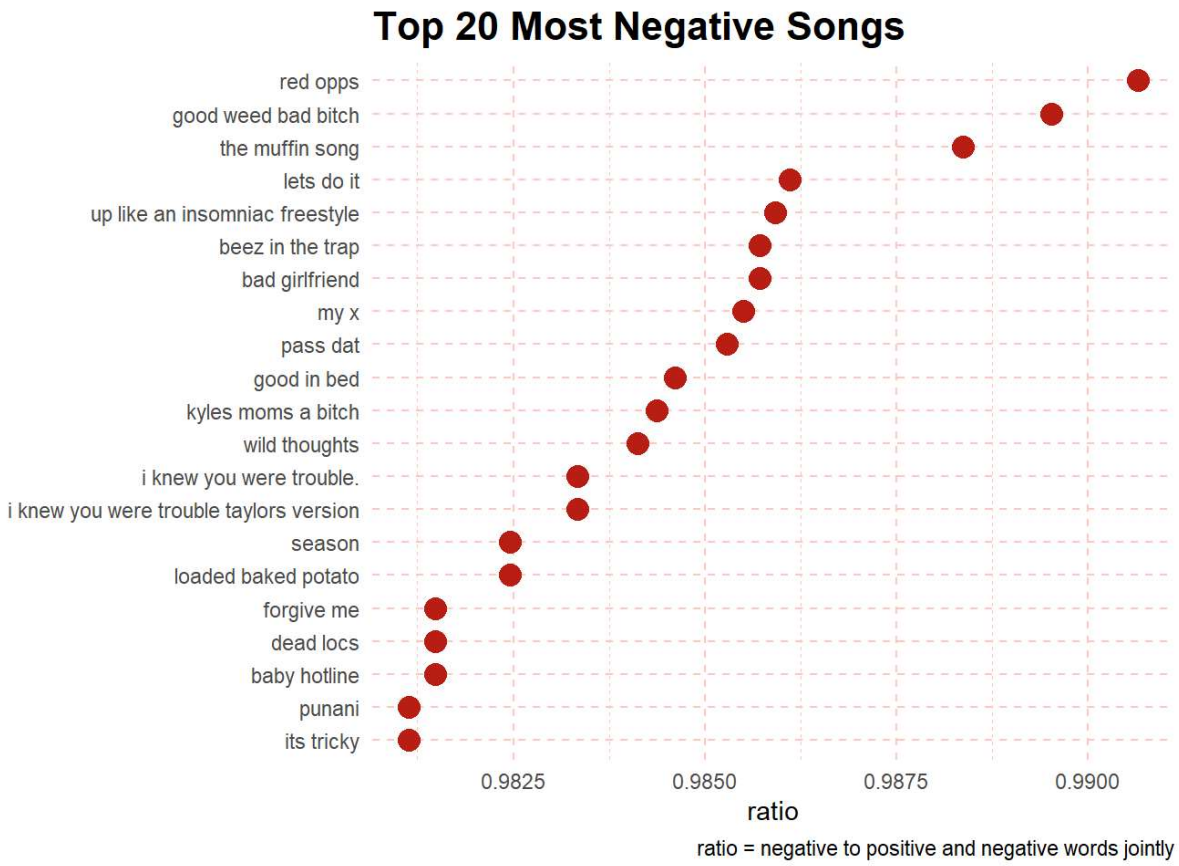
In song title, there are less negative words, but in lyrics, negative words become bigger. Also in the lyrics word cloud can find that some of negative words are much larger than others. This might cause by the song type that “rap” is the largest type in this data set.

Top 20 Most Positive / Negative Songs

```
ratio_sentiment = df2_words %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  group_by(title, sentiment) %>%
  summarize(score = n()) %>%
  ungroup() %>%
  spread(sentiment, score) %>%
  ungroup() %>%
  mutate(ratio = positive / (positive + negative),
         title = reorder(title, ratio))

ratio_sentiment %>%
  top_n(20) %>%
  ggplot(aes(x = title, y = ratio)) +
  geom_point(color = "#2866BD", size = 4) +
  coord_flip() +
  labs(title = "Top 20 Most Positive Songs",
       x = "",
       caption = "ratio = positive to positive and negative words jointly") +
  theme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold"),
        panel.grid = element_line(linetype = "dashed", color = "#C7DEFF", size = 0.5))
```




```
ratio_sentiment %>%
  mutate(ratio = 1 - ratio,
         title = reorder(title, ratio)) %>%
  top_n(20) %>%
  ggplot(aes(x = title, y = ratio)) +
  geom_point(color = "#BC2014", size = 4) +
  coord_flip() +
  labs(title = "Top 20 Most Negative Songs",
       x = "",
       caption = "ratio = negative to positive and negative words jointly") +
  theme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold"),
        panel.grid = element_line(linetype = "dashed", color = "#FFCBC7", size = 0.5))
```

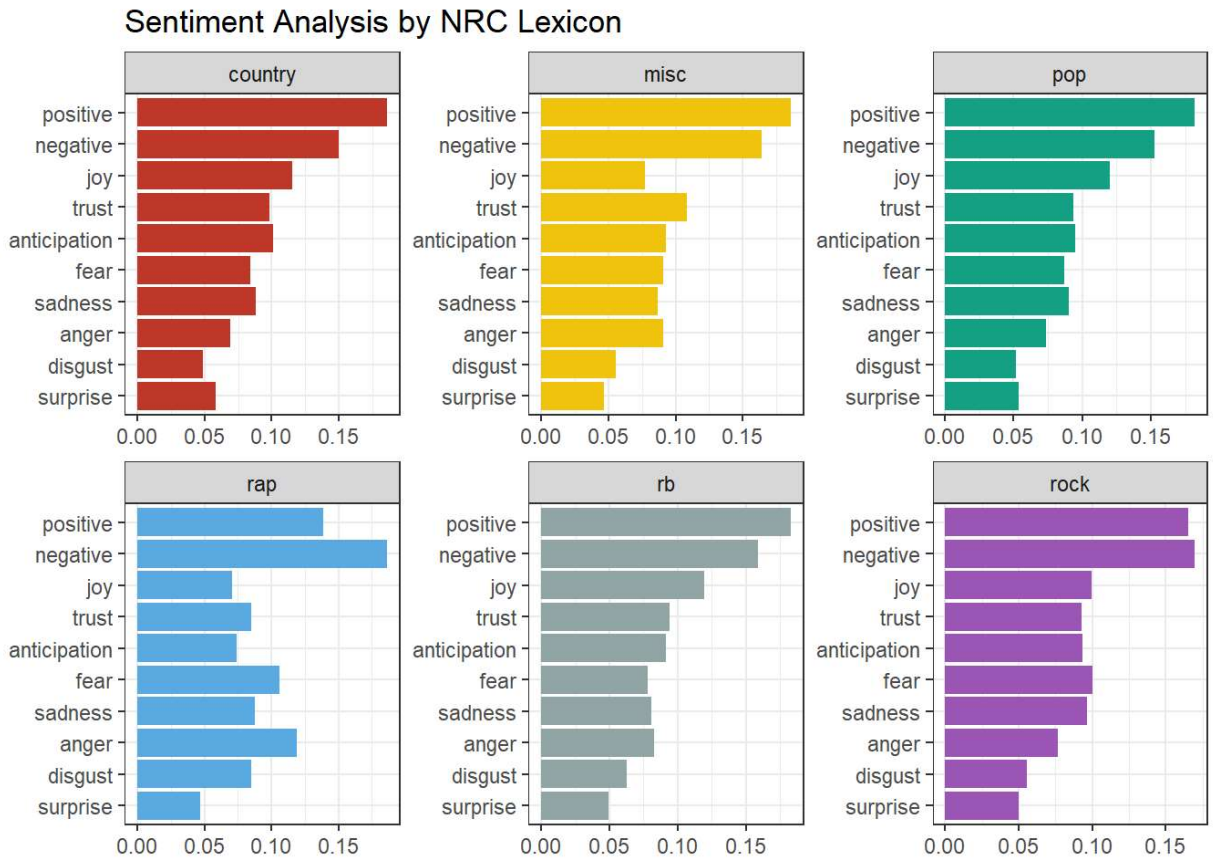


In positive song, the title might not show a pretty positive vibe, but in negative song, can find some bad words in the title before you see the lyrics.

Sentiment analysis with each music type

Analyzed the sentiment of song lyrics in each type using the NRC lexicon

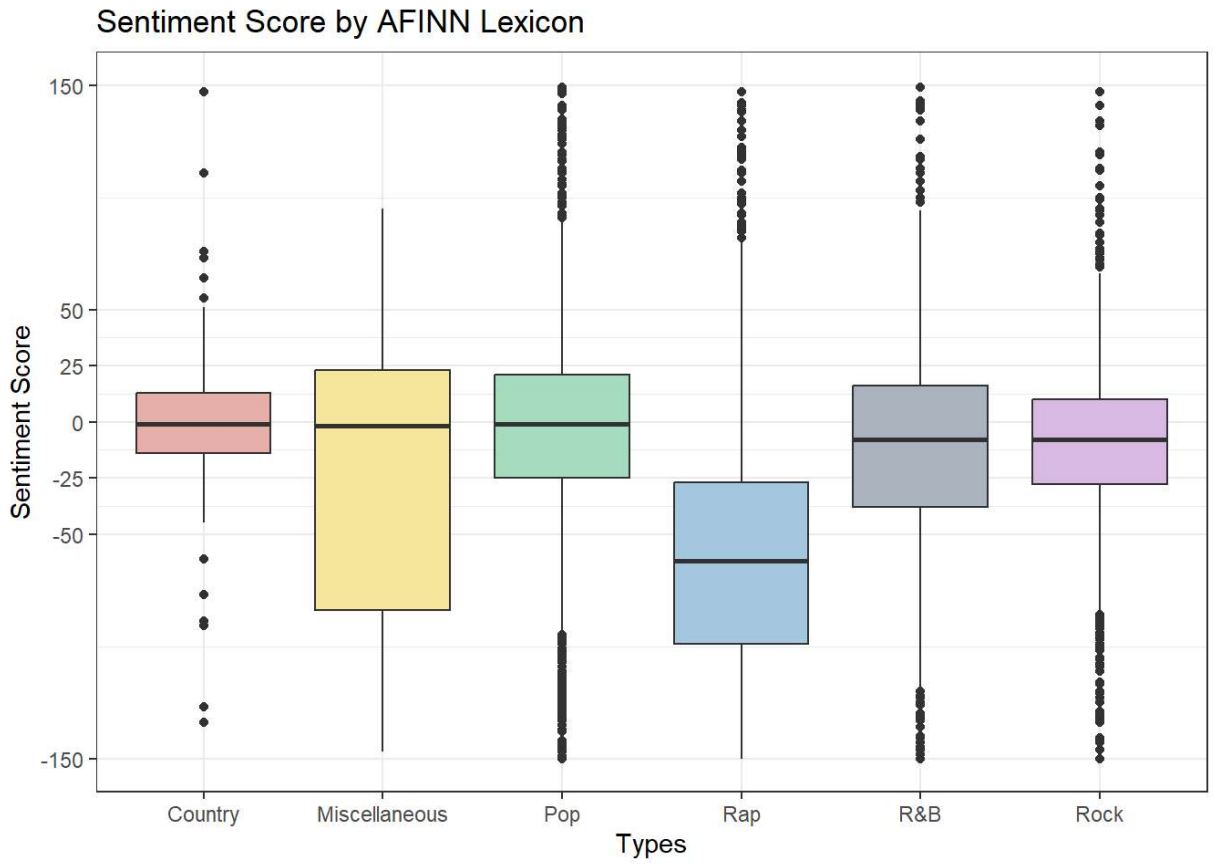
```
nrc = get_sentiments(lexicon = "nrc")

song_nrc = df2_words %>%
  semi_join(nrc, by = "word")
song_nrc = song_nrc %>%
  inner_join(nrc, by = "word")

song_nrc = song_nrc %>%
  group_by(tag, sentiment) %>%
  count() %>%
  ungroup() %>%
  group_by(tag) %>%
  mutate(percentage = n / sum(n)) %>%
  ungroup()

ggplot(song_nrc, aes(x = reorder(sentiment, percentage), y = percentage, fill = tag)) +
  geom_col(show.legend = F) +
  facet_wrap(~tag, scales = "free") +
  coord_flip() +
  labs(x = NULL, y = NULL, title = 'Sentiment Analysis by NRC Lexicon') +
  scale_fill_manual(values = c("#C0392B", "#F1C40F", "#16A085", "#5DADE2", "#95A5A6", "#9B59B6")) +
  theme_bw()
```



"rap" is the only that has more negative words than positive. In "country", "pop", "R&B", positive words are much more than negative.

Analyzed the sentiment of song lyrics in each type using the AFINN lexicon.

```
afinn = get_sentiments(lexicon = 'afinn')

song_afinn <- df2_words %>%
  inner_join(afinn) %>%
  group_by(title) %>%
  mutate(total_score = sum(value)) %>%
  ungroup() %>%
  arrange(desc(value))

ggplot(song_afinn, aes(x = tag, y = total_score, fill = tag)) +
  geom_boxplot(show.legend = F) +
  labs(x = 'Types', y = 'Sentiment Score', title = 'Sentiment Score by AFINN Lexicon') +
  scale_fill_manual(values = c("#E6B0AA", "#F9E79F", "#A9DFBF", "#A9CCE3", "#AEB6BF", "#D7BDE2")) +
  scale_y_continuous(breaks = c(-150, -50, -25, 0, 25, 50, 150), limits = c(-150, 150)) +
  scale_x_discrete(labels = c("Country", "Miscellaneous", "Pop", "Rap", "R&B", "Rock")) +
  theme_bw()
```

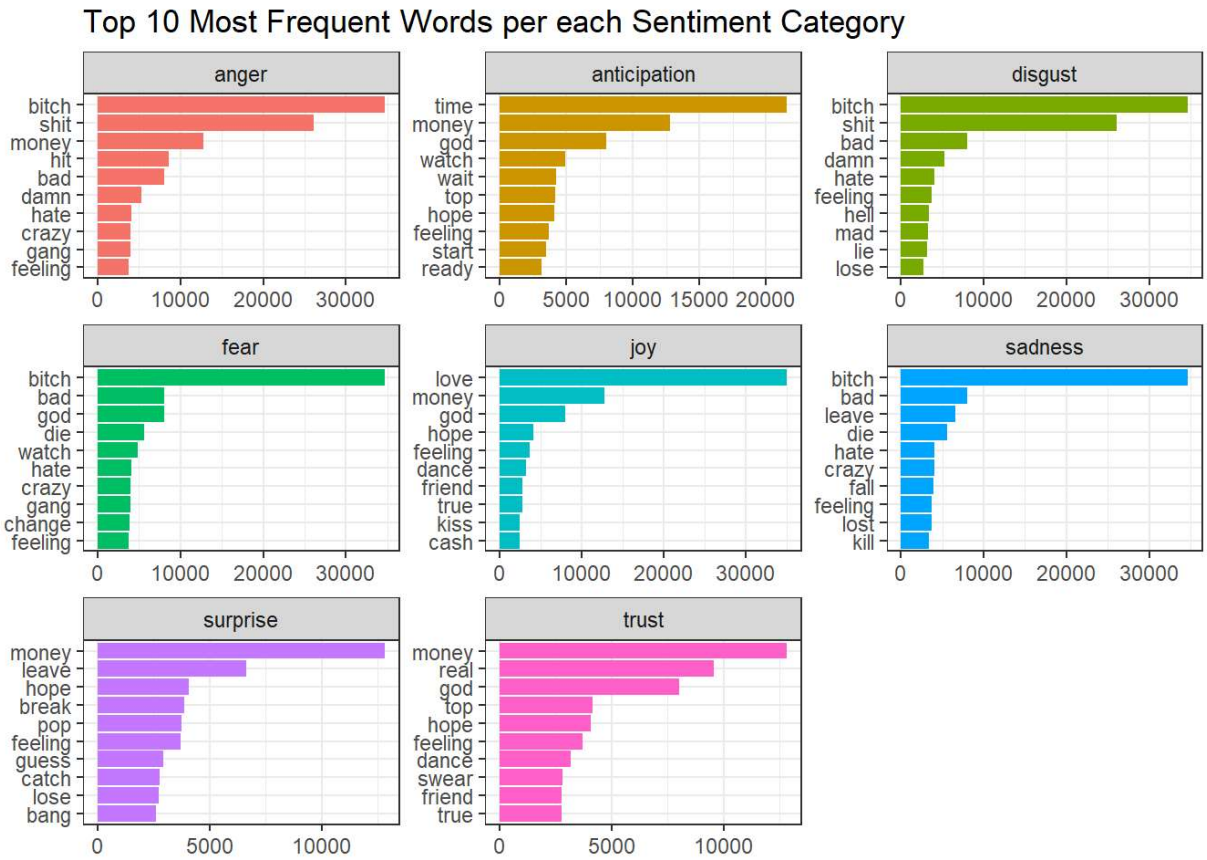


The result are same from the NRC, “rap” is the only type get negative score (by median). Almost 75% of “rap” songs get negative score. For “country”, almost 75% of songs get positive score. For “rock”, is roughly half in positive and half in negative.

Top 10 most frequent words per each sentiment category

Use the NRC lexicon’s eight basic emotions: anger, fear, anticipation, trust, surprise, sadness, joy, and disgust to analyze.

```
lyrics_freq_nrc = freq_lyrics

lyrics_freq_nrc = lyrics_freq_nrc %>%
  semi_join(nrc, by = "word")

lyrics_freq_nrc = lyrics_freq_nrc %>%
  inner_join(nrc, by = "word") %>%
  ungroup() %>%
  filter(!sentiment %in% c("positive", "negative")) %>%
  group_by(word, sentiment) %>%
  count() %>%
  ungroup() %>%
  group_by(sentiment) %>%
  arrange(desc(n)) %>%
  slice(1:10)

ggplot(lyrics_freq_nrc, aes(x = reorder(word, n), y = n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap( ~ sentiment, scales = "free") +
  coord_flip() +
  labs(x = NULL, y = NULL, title = 'Top 10 Most Frequent Words per each Sentiment Category') +
  theme_bw()
```



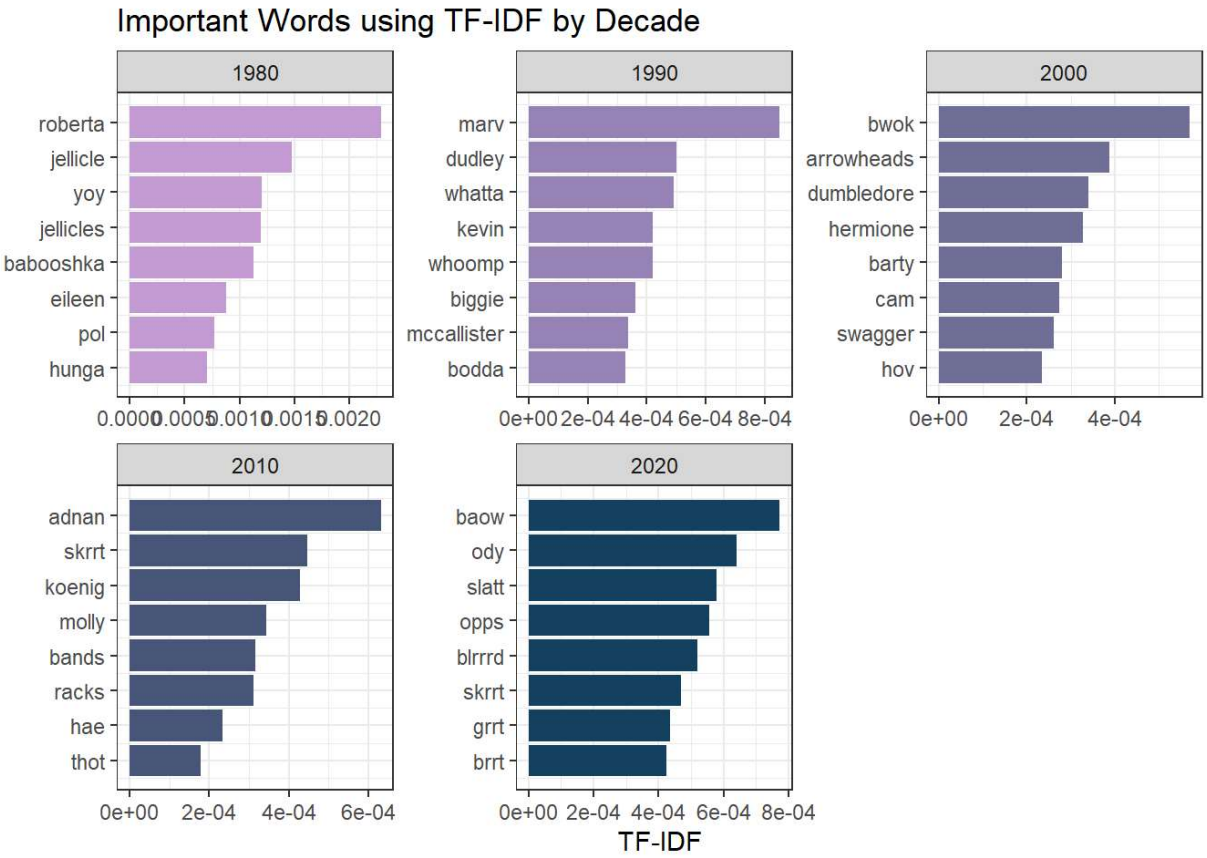
The figure above show the words that represent each sentiment. There are some words show repeatedly in different categories. These frequency words are common, easy to describe and feel.

TF-IDF

I investigated word importance and adjusted for how rarely they were used. To do that, I used the TF-IDF metric. TF-IDF metric scores higher terms that appear more frequently in a document, unless it also occurs in many documents (songs). Carried out our analysis across both decades and music types.

Differentiated by Decade

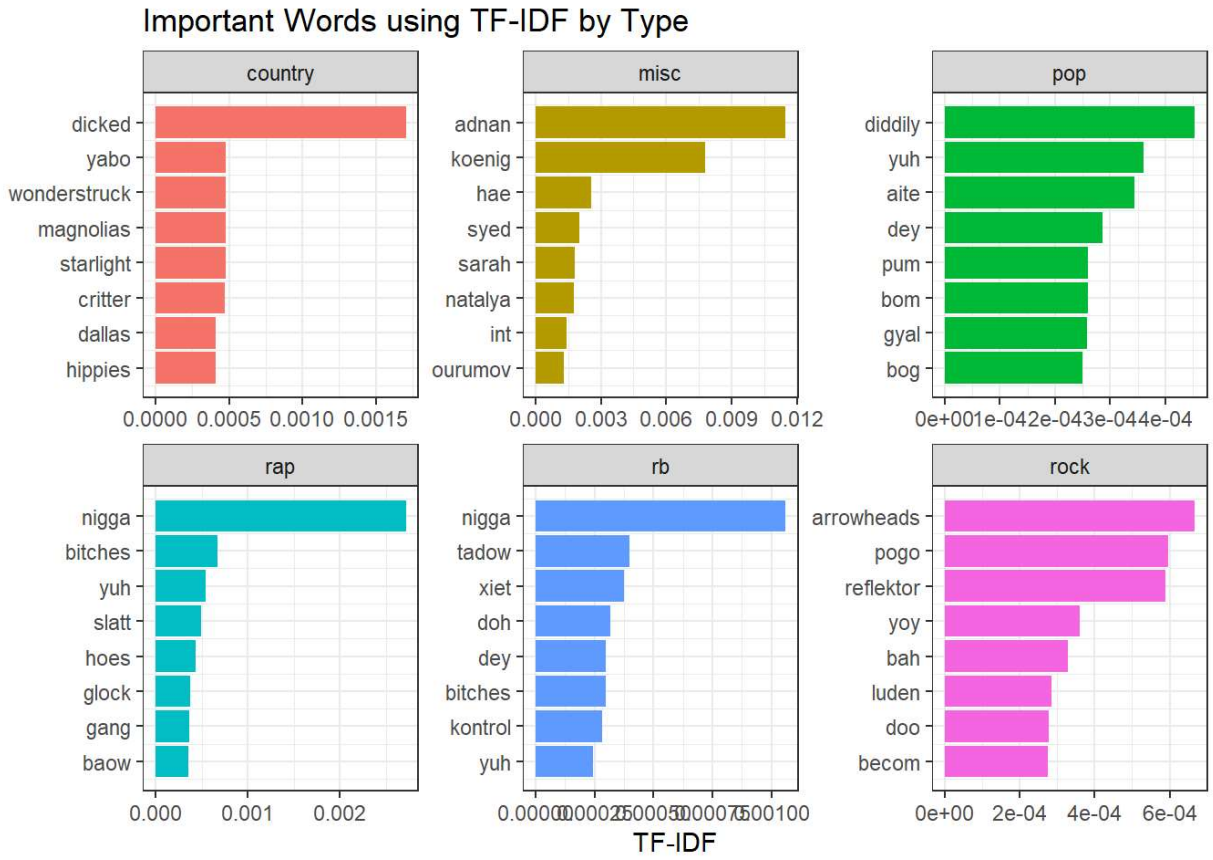
```
tfidf_words_decade = df2_words %>%
  count(decade, word, sort = TRUE) %>%
  ungroup() %>%
  bind_tf_idf(word, decade, n) %>%
  arrange(desc(tf_idf))

top_tfidf_words_decade = tfidf_words_decade %>%
  group_by(decade) %>%
  slice(seq_len(8)) %>%
  ungroup() %>%
  arrange(decade, tf_idf) %>%
  mutate(row = row_number())

ggplot(top_tfidf_words_decade, aes(x = row, tf_idf, fill = decade)) +
  geom_col(show.legend = NULL) +
  labs(x = NULL, y = "TF-IDF") +
  ggtitle("Important Words using TF-IDF by Decade") +
  theme_bw() +
  facet_wrap(~decade,
             ncol = 3, nrow = 2,
             scales = "free") +
  scale_x_continuous(
    breaks = top_tfidf_words_decade$row,
    labels = top_tfidf_words_decade$word) +
  coord_flip() +
  scale_fill_gradient(low = "#C39BD3", high = "#154360")
```



In 2020's, can find some special onomatopoeia. But generally, from the TF-IDF metric scores, there are not significant special words that can represent a decade.

Differentiated by Music Type

```
tfidf_words_tag = df2_words %>%
  count(tag, word, sort = TRUE) %>%
  ungroup() %>%
  bind_tf_idf(word, tag, n) %>%
  arrange(desc(tf_idf))

top_tfidf_words_tag = tfidf_words_tag %>%
  group_by(tag) %>%
  slice(seq_len(8)) %>%
  ungroup() %>%
  arrange(tag, tf_idf) %>%
  mutate(row = row_number())

ggplot(top_tfidf_words_tag, aes(x = row, tf_idf, fill = tag)) +
  geom_col(show.legend = NULL) +
  labs(x = NULL, y = "TF-IDF") +
  ggtitle("Important Words using TF-IDF by Type") +
  theme_bw() +
  facet_wrap(~tag,
             ncol = 3, nrow = 2,
             scales = "free") +
  scale_x_continuous(
    breaks = top_tfidf_words_tag$row,
    labels = top_tfidf_words_tag$word) +
  coord_flip()
```



The bad words such as “nigga” and “bitches” are in “rap” more frequency than other types. That can explain why the word cloud shows these words so big, but only “rap” get negative score in AFINN lexicon.

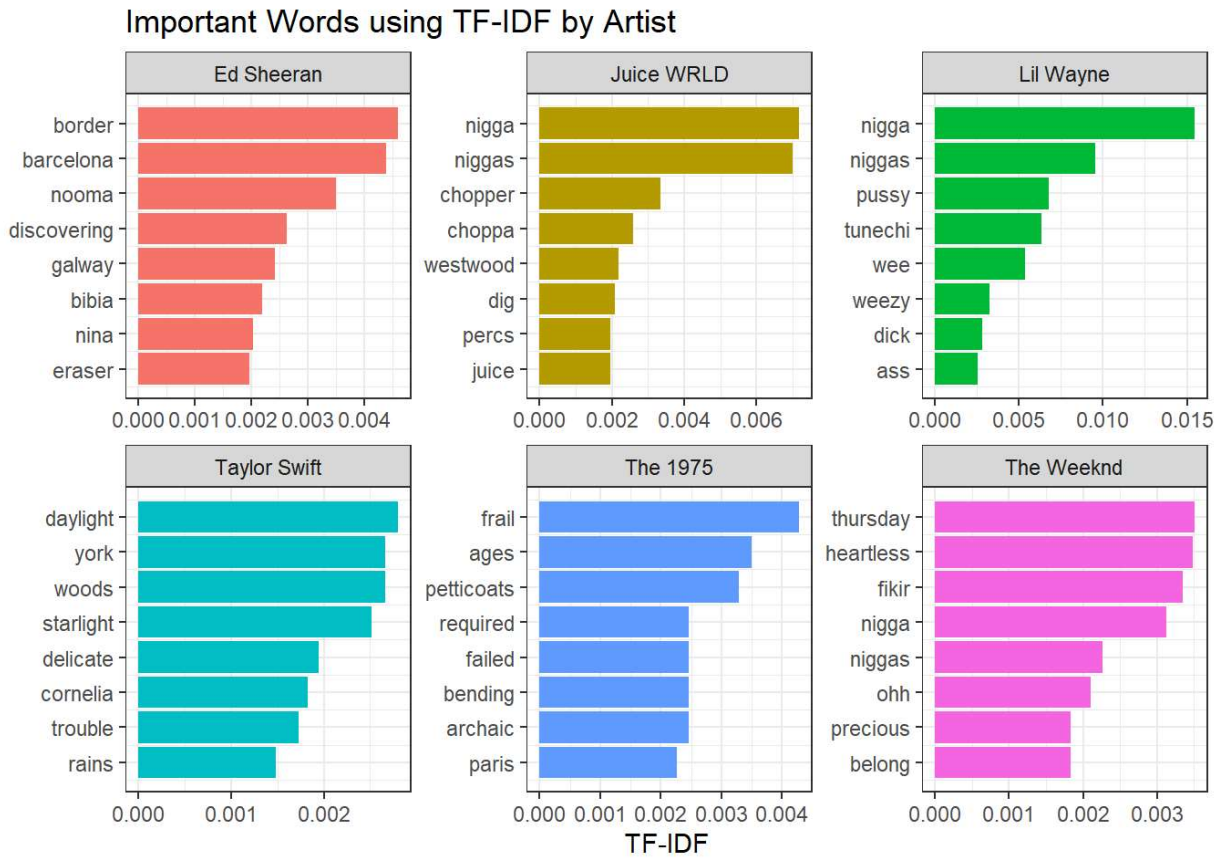
Differentiated by Artists

Choose 6 artists that they are separately represented different music type.

1. Taylor Swift: Country and Pop
2. The Weeknd: R&B and Pop
3. Juice WRLD: Rap
4. The 1975: Rock and Pop
5. Ed Sheeran: Pop
6. Lil Wayne: Rap

Note: choose 2 rappers because there is much more “rap” in my data set.

```
artist_names = c("Taylor Swift", "The Weeknd", "Juice WRLD", "The 1975", "Ed Sheeran", "Lil Wayne")

df_filtered = df2_words %>% filter(artist %in% artist_names)

tfidf_words_artist = df_filtered %>%
  count(artist, word, sort = TRUE) %>%
  ungroup() %>%
  bind_tf_idf(word, artist, n) %>%
  arrange(artist, desc(tf_idf))

top_tfidf_words_artist = tfidf_words_artist %>%
  group_by(artist) %>%
  slice(seq_len(8)) %>%
  ungroup() %>%
  arrange(artist, tf_idf) %>%
  mutate(row = row_number())

ggplot(top_tfidf_words_artist, aes(x = row, tf_idf, fill = artist)) +
  geom_col(show.legend = NULL) +
  labs(x = NULL, y = "TF-IDF") +
  ggtitle("Important Words using TF-IDF by Artist") +
  theme_bw() +
  facet_wrap(~artist,
             ncol = 3, nrow = 2,
             scales = "free") +
  scale_x_continuous(
    breaks = top_tfidf_words_artist$row,
    labels = top_tfidf_words_artist$word) +
  coord_flip()
```



For the rapper, Juice WRLD and Lil Wayne, they use bad word much more than others, especially Lil Wayne. There is another interest thing from Juice WRLD, word “juice” show up in the category, that might due to he write his artist name in his song sometimes. For Taylor Swift and The 1975, most of the words in their categories are not show up before. It seems that their lyrics are quite different from others, and have their own unique vibes.

Networks of Co-occuring Words

After doing the TF-IDF for these 6 artists, I tried to look closer at the relationship between words of their lyrics by examining networks of co-occurring words. I classified their songs into sections and then calculated their pairwise co-occurrence, examining the correlation of words, which does not necessarily exist in the same song.

Taylor Swift

```
section = df2 %>%
  ungroup() %>%
  filter(artist == "Taylor Swift") %>%
  mutate(section = row_number() %/% 5) %>%
  filter(section > 0) %>%
  unnest_tokens(word, lyrics) %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2)

word_corr = section %>%
  group_by(word) %>%
  filter(n() >= 5) %>%
  pairwise_cor(word, section, sort = TRUE)

Taylor = word_corr %>%
  filter(correlation > 0.7) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "kk") +
  geom_edge_link(aes(edge_alpha = correlation), show.legend = FALSE) +
  geom_node_point(color = "#E6B0AA", size = 5) +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void() +
  ggtitle("Taylor Swift", subtitle = "Country & Pop") +
  theme(plot.title = element_text(hjust = 0.5, vjust = 0.5, size = 15, color = "#7B241C")) +
  theme(plot.subtitle = element_text(hjust = 0.5, vjust = 0.5, size = 10, color = "#7B241C"))
```

The Weeknd

```
section2 = df2 %>%
  ungroup() %>%
  filter(artist == "The Weeknd") %>%
  mutate(section = row_number() %/% 5) %>%
  filter(section > 0) %>%
  unnest_tokens(word, lyrics) %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2)

word_corr2 = section2 %>%
  group_by(word) %>%
  filter(n() >= 5) %>%
  pairwise_cor(word, section, sort = TRUE)

word_corr2 = word_corr2 %>%
  filter(across(everything(), ~ !is.infinite(.)))

Weeknd = word_corr2 %>%
  filter(correlation > 0.7) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "kk") +
  geom_edge_link(aes(edge_alpha = correlation), show.legend = FALSE) +
  geom_node_point(color = "#AEB6BF", size = 5) +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void() +
  ggtitle("The Weeknd", subtitle = "R&B & Pop") +
  theme(plot.title = element_text(hjust = 0.5, vjust = 0.5, size = 15, color = "#283747")) +
  theme(plot.subtitle = element_text(hjust = 0.5, vjust = 0.5, size = 10, color = "#283747"))
```

Juice WRLD

```
section3 = df2 %>%
  ungroup() %>%
  filter(artist == "Juice WRLD") %>%
  mutate(section = row_number() %/% 5) %>%
  filter(section > 0) %>%
  unnest_tokens(word, lyrics) %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2)

word_corr3 = section3 %>%
  group_by(word) %>%
  filter(n() >= 5) %>%
  pairwise_cor(word, section, sort = TRUE)

word_corr3 = word_corr3 %>%
  filter(across(everything(), ~ !is.infinite(.)))

Juice = word_corr3 %>%
  filter(correlation > 0.75) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "kk") +
  geom_edge_link(aes(edge_alpha = correlation), show.legend = FALSE) +
  geom_node_point(color = "#A9CCE3", size = 5) +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void() +
  ggtitle("Juice WRLD", subtitle = "Rap") +
  theme(plot.title = element_text(hjust = 0.5, vjust = 0.5, size = 15, color = "#1A5276")) +
  theme(plot.subtitle = element_text(hjust = 0.5, vjust = 0.5, size = 10, color = "#1A5276"))
```

The 1975

```
section4 = df2 %>%
  ungroup() %>%
  filter(artist == "The 1975") %>%
  mutate(section = row_number() %/% 5) %>%
  filter(section > 0) %>%
  unnest_tokens(word, lyrics) %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2)

word_corr4 = section4 %>%
  group_by(word) %>%
  filter(n() >= 5) %>%
  pairwise_cor(word, section, sort = TRUE)

word_corr4 = word_corr4 %>%
  filter(across(everything(), ~ !is.infinite(.)))

The_1975 = word_corr4 %>%
  filter(correlation > 0.7) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "kk") +
  geom_edge_link(aes(edge_alpha = correlation), show.legend = FALSE) +
  geom_node_point(color = "#D7BDE2", size = 5) +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void() +
  ggtitle("The 1975", subtitle = "Rock & Pop") +
  theme(plot.title = element_text(hjust = 0.5, vjust = 0.5, size = 15, color = "#633974")) +
  theme(plot.subtitle = element_text(hjust = 0.5, vjust = 0.5, size = 10, color = "#633974"))
```

Ed Sheeran

```
section5 = df2 %>%
  ungroup() %>%
  filter(artist == "Ed Sheeran") %>%
  mutate(section = row_number() %/% 5) %>%
  filter(section > 0) %>%
  unnest_tokens(word, lyrics) %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2)

word_corr5 = section5 %>%
  group_by(word) %>%
  filter(n() >= 5) %>%
  pairwise_cor(word, section, sort = TRUE)

word_corr5 = word_corr5 %>%
  filter(across(everything(), ~ !is.infinite(.)))

Sheeran = word_corr5 %>%
  filter(correlation > 0.75) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "kk") +
  geom_edge_link(aes(edge_alpha = correlation), show.legend = FALSE) +
  geom_node_point(color = "#A9DFBF", size = 5) +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void() +
  ggtitle("Ed Sheeran", subtitle = "Pop") +
  theme(plot.title = element_text(hjust = 0.5, vjust = 0.5, size = 15, color = "#145A32")) +
  theme(plot.subtitle = element_text(hjust = 0.5, vjust = 0.5, size = 10, color = "#145A32"))
```

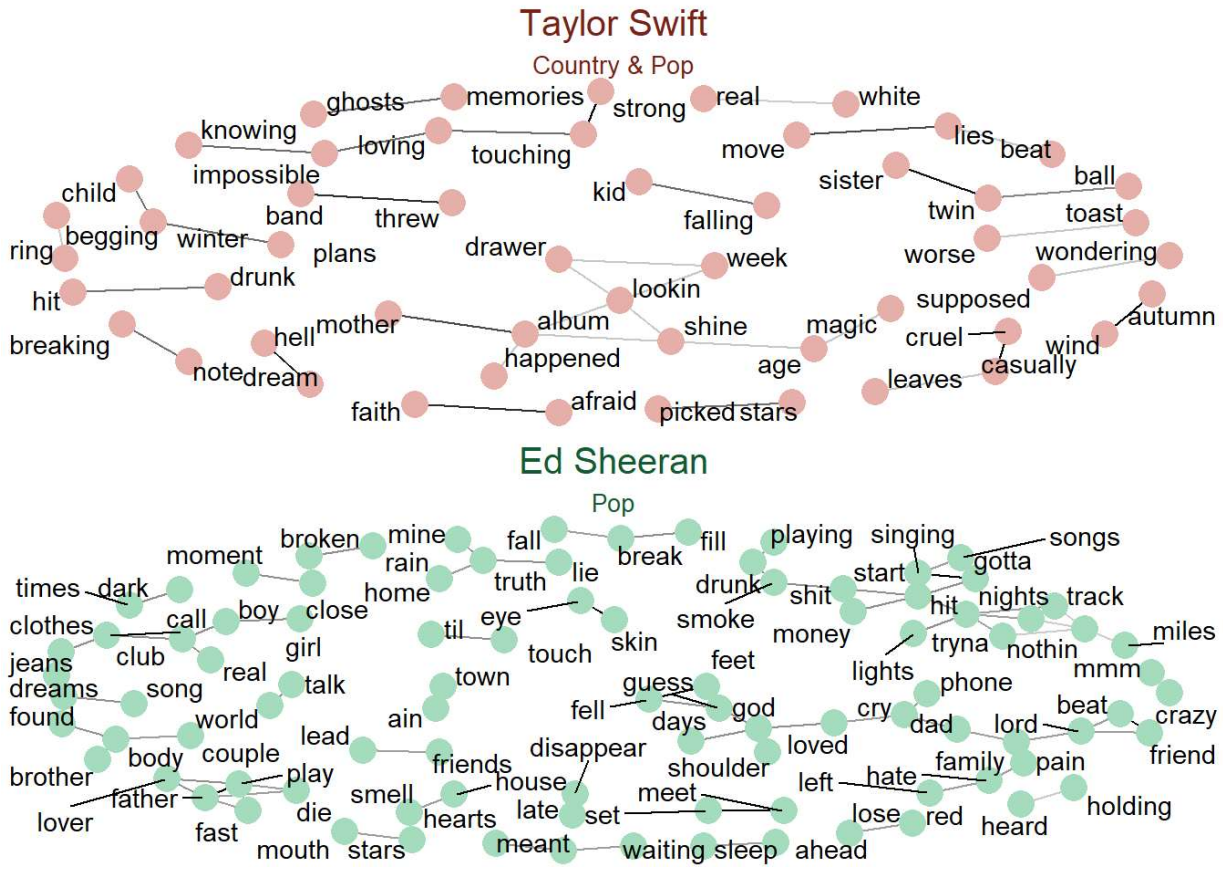
Lil Wayne

```
section6 = df2 %>%
  ungroup() %>%
  filter(artist == "Lil Wayne") %>%
  mutate(section = row_number() %/% 5) %>%
  filter(section > 0) %>%
  unnest_tokens(word, lyrics) %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2)

word_corr6 = section6 %>%
  group_by(word) %>%
  filter(n() >= 5) %>%
  pairwise_cor(word, section, sort = TRUE)

word_corr6 = word_corr6 %>%
  filter(across(everything(), ~ !is.infinite(.)))

Wayne = word_corr6 %>%
  filter(correlation > 0.75) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "kk") +
  geom_edge_link(aes(edge_alpha = correlation), show.legend = FALSE) +
  geom_node_point(color = "#A9CCE3", size = 5) +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void() +
  ggtitle("Lil Wayne", subtitle = "Rap") +
  theme(plot.title = element_text(hjust = 0.5, vjust = 0.5, size = 15, color = "#1A5276")) +
  theme(plot.subtitle = element_text(hjust = 0.5, vjust = 0.5, size = 10, color = "#1A5276"))
```

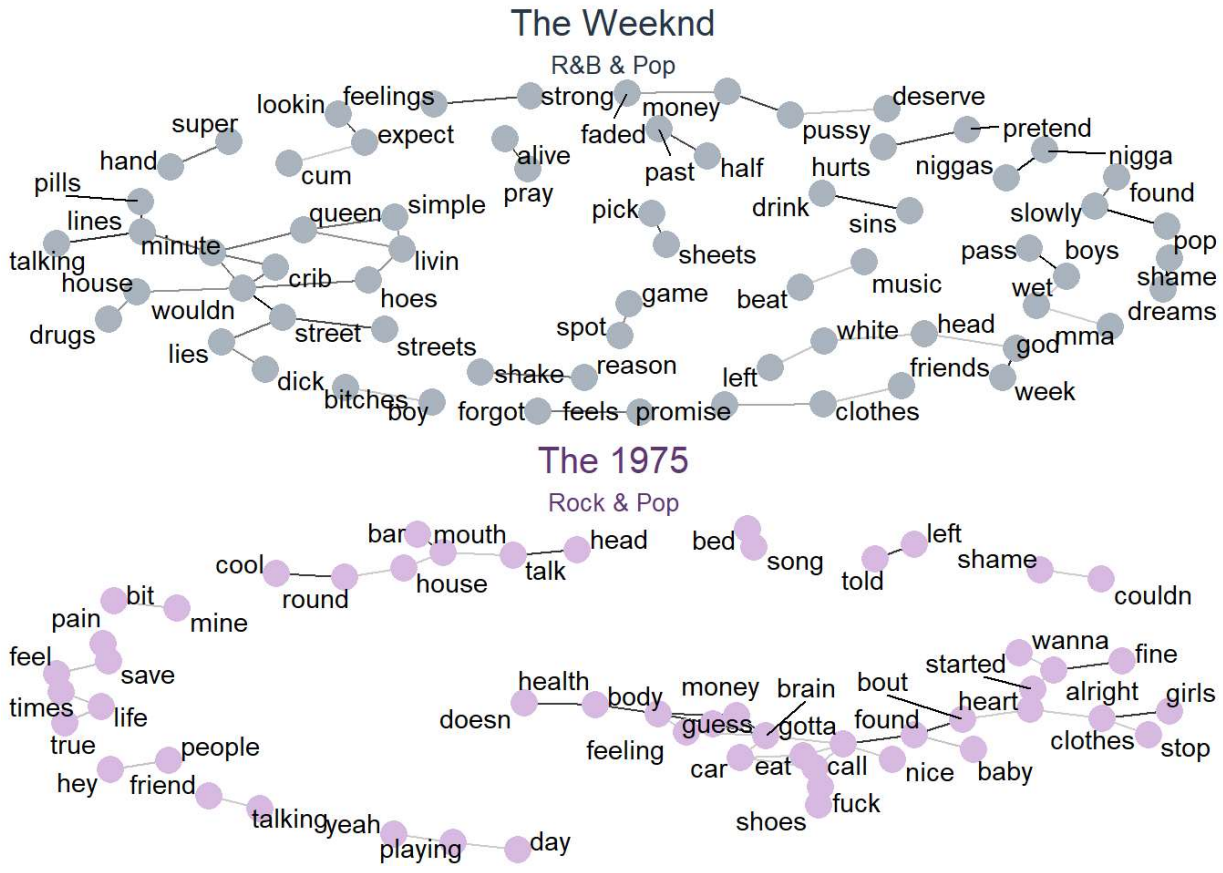
Plot

Plot the results all together.

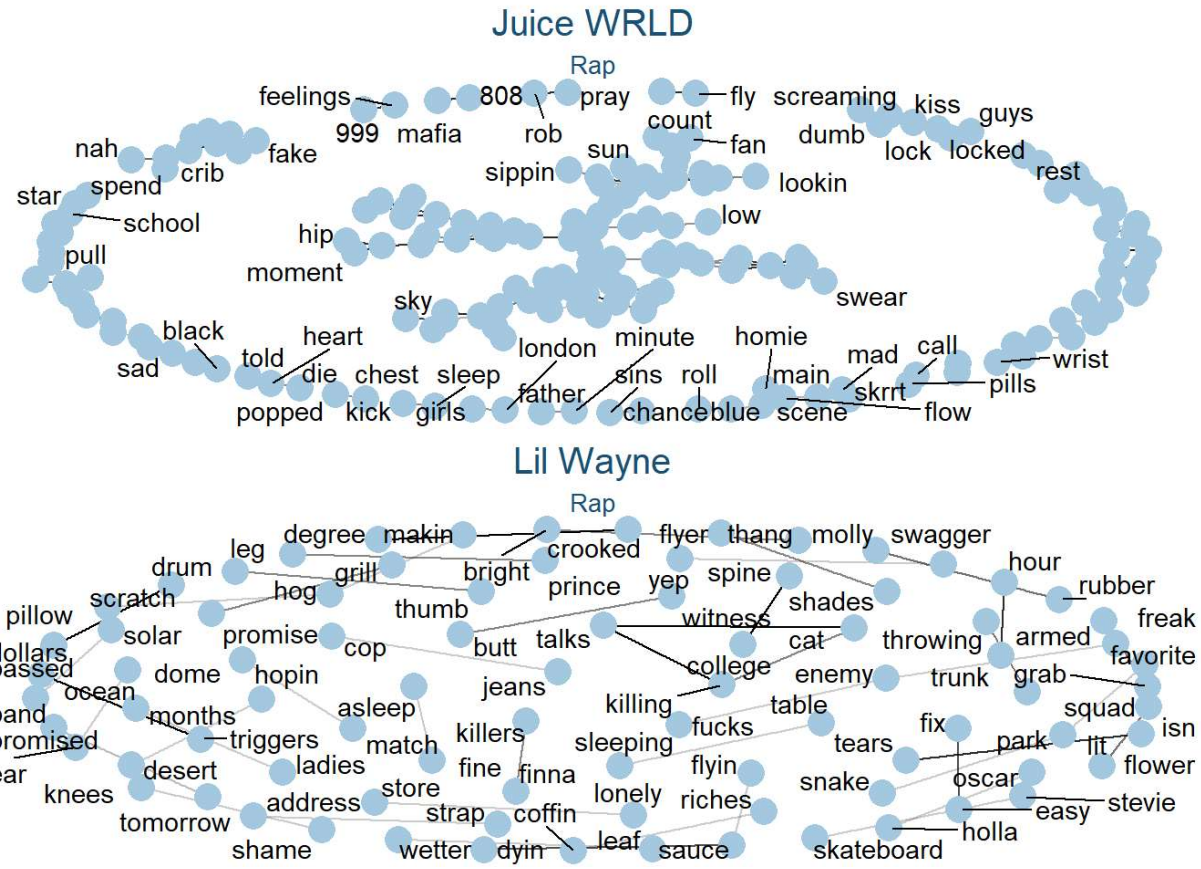
```
grid.arrange(Taylor, Sheeran)
```




```
grid.arrange(Weeknd, The_1975)
```




```
grid.arrange(Juice, Wayne)
```



LDA

Another method of analyzing lyrics data was Latent Dirichlet Allocation (LDA), used for topic modeling. Every document (song) is a mixture of topics. Imagine that each song is containing words from several topics in particular proportions. Every topic is a mixture of words. LDA is a mathematical method for estimating both of these simultaneously: finding the mixture of words associated with each topic. Performed LDA from different decades and music type, specifying 2 topic LDA models.

In each Decade

1980’s

```
df_filtered_1980 = df2_words %>% filter(decade %in% "1980")

tidy_lda = df_filtered_1980 %>%
  ungroup() %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2) %>%
  select(artist, title, tag, word)

topics = LDA(cast_dtm(data = tidy_lda %>%
                        count(artist, word) %>%
                        ungroup(),
                      term = word,
                      document = artist,
                      value = n),
             k = 2, control = list(seed = 42)) %>%
  tidy(matrix = "beta") %>%
  group_by(topic) %>%
  arrange(desc(beta)) %>%
  top_n(12, beta) %>%
  ungroup()

lda_1980 = topics %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = 'free') +
  coord_flip() +
  ggtitle("Topic Modeling using LDA in 1980's")
```

1990’s

```
df_filtered_1990 = df2_words %>% filter(decade %in% "1990")

tidy_lda = df_filtered_1990 %>%
  ungroup() %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2) %>%
  select(artist, title, tag, word)

topics = LDA(cast_dtm(data = tidy_lda %>%
                        count(artist, word) %>%
                        ungroup(),
                      term = word,
                      document = artist,
                      value = n),
             k = 2, control = list(seed = 42)) %>%
  tidy(matrix = "beta") %>%
  group_by(topic) %>%
  arrange(desc(beta)) %>%
  top_n(12, beta) %>%
  ungroup()

lda_1990 = topics %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = 'free') +
  coord_flip() +
  ggtitle("Topic Modeling using LDA in 1990's")
```

2000’s

```
df_filtered_2000 = df2_words %>% filter(decade %in% "2000")

tidy_lda = df_filtered_2000 %>%
  ungroup() %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2) %>%
  select(artist, title, tag, word)

topics = LDA(cast_dtm(data = tidy_lda %>%
                        count(artist, word) %>%
                        ungroup(),
                      term = word,
                      document = artist,
                      value = n),
             k = 2, control = list(seed = 42)) %>%
  tidy(matrix = "beta") %>%
  group_by(topic) %>%
  arrange(desc(beta)) %>%
  top_n(12, beta) %>%
  ungroup()

lda_2000 = topics %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = 'free') +
  coord_flip() +
  ggtitle("Topic Modeling using LDA in 2000's")
```

2010’s

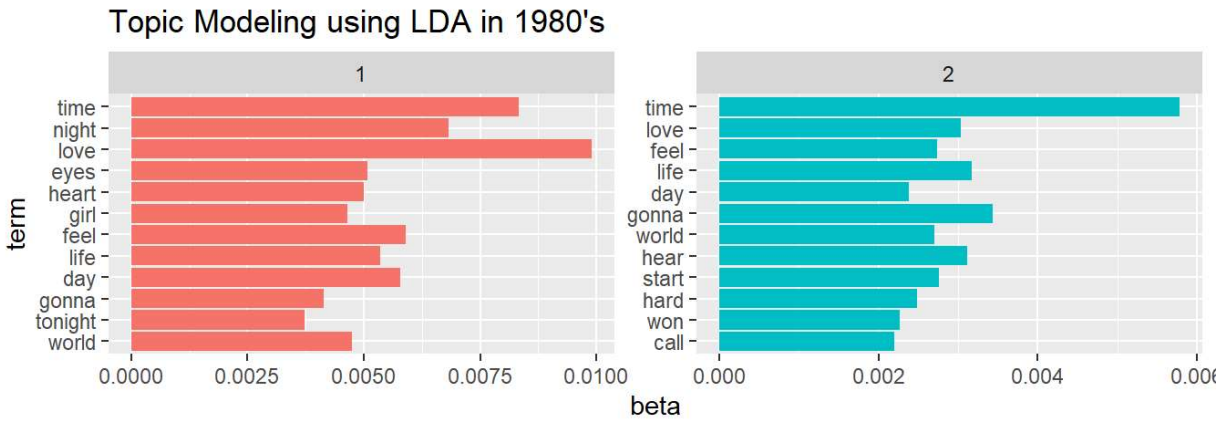
```
df_filtered_2010 = df2_words %>% filter(decade %in% "2010")

tidy_lda = df_filtered_2010 %>%
  ungroup() %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2) %>%
  select(artist, title, tag, word)

topics = LDA(cast_dtm(data = tidy_lda %>%
                        count(artist, word) %>%
                        ungroup(),
                      term = word,
                      document = artist,
                      value = n),
             k = 2, control = list(seed = 42)) %>%
  tidy(matrix = "beta") %>%
  group_by(topic) %>%
  arrange(desc(beta)) %>%
  top_n(12, beta) %>%
  ungroup()

lda_2010 = topics %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = 'free') +
  coord_flip() +
  ggtitle("Topic Modeling using LDA in 2010's")
```

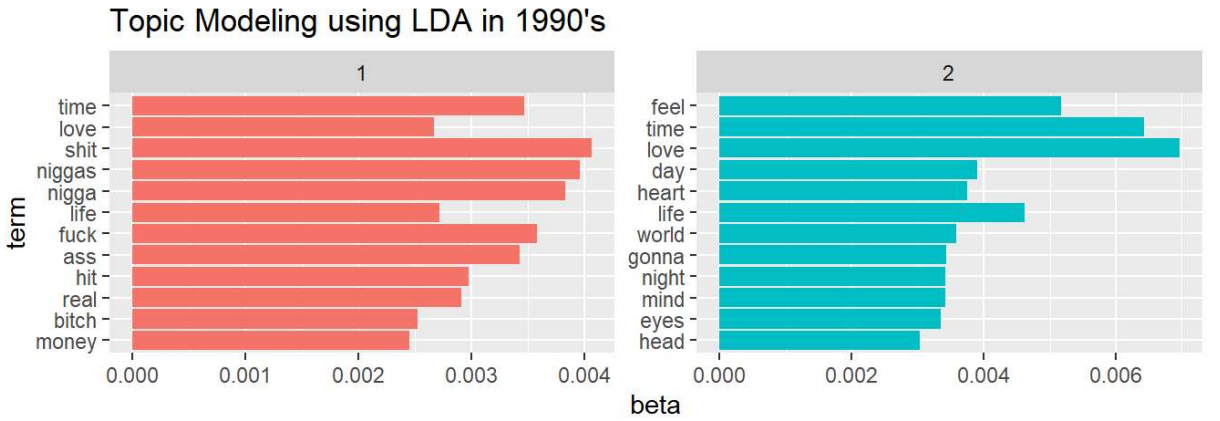
2020’s

```
df_filtered_2020 = df2_words %>% filter(decade %in% "2020")

tidy_lda = df_filtered_2020 %>%
  ungroup() %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2) %>%
  select(artist, title, tag, word)

topics = LDA(cast_dtm(data = tidy_lda %>%
                        count(artist, word) %>%
                        ungroup(),
                      term = word,
                      document = artist,
                      value = n),
             k = 2, control = list(seed = 42)) %>%
  tidy(matrix = "beta") %>%
  group_by(topic) %>%
  arrange(desc(beta)) %>%
  top_n(12, beta) %>%
  ungroup()

lda_2020 = topics %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = 'free') +
  coord_flip() +
  ggtitle("Topic Modeling using LDA in 2020's")
```

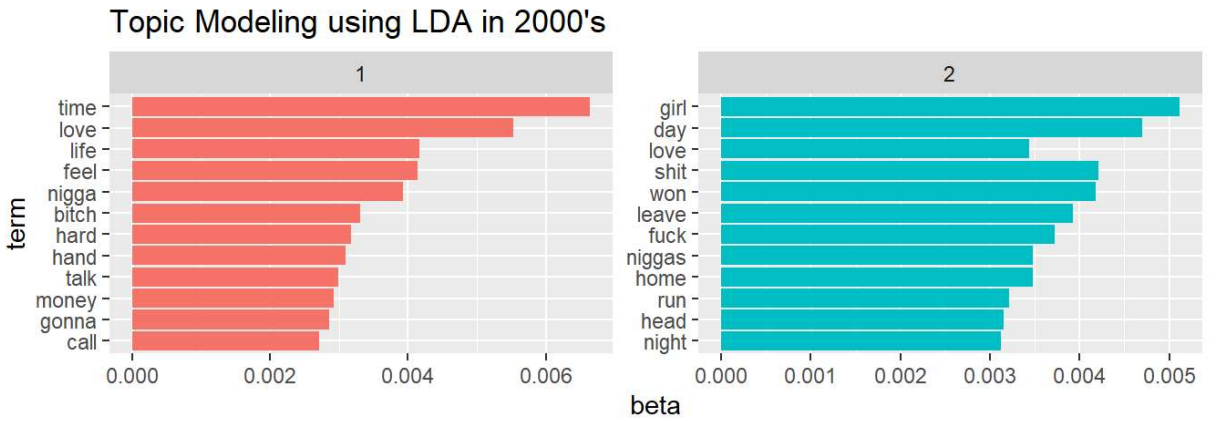
Plot: Decade

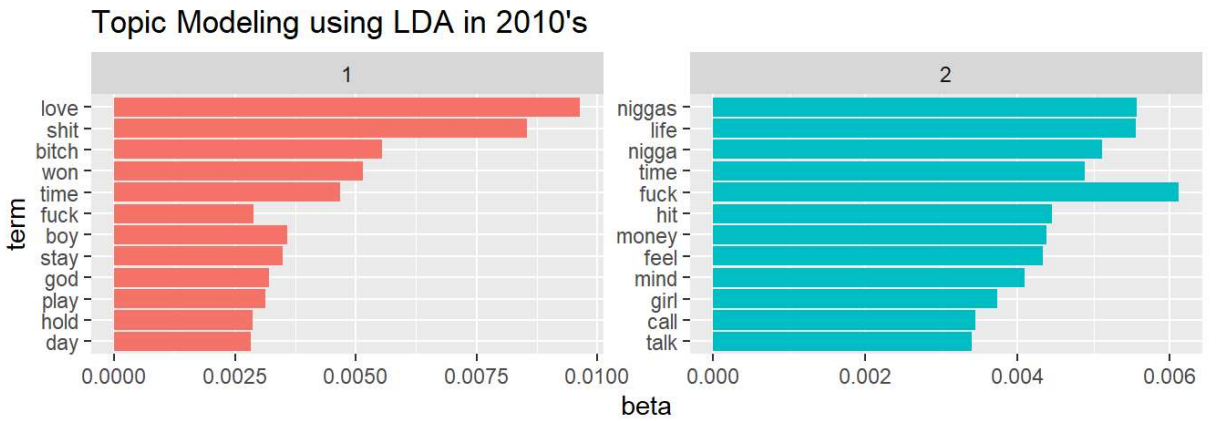
```
grid.arrange(lda_1980, lda_1990)
```



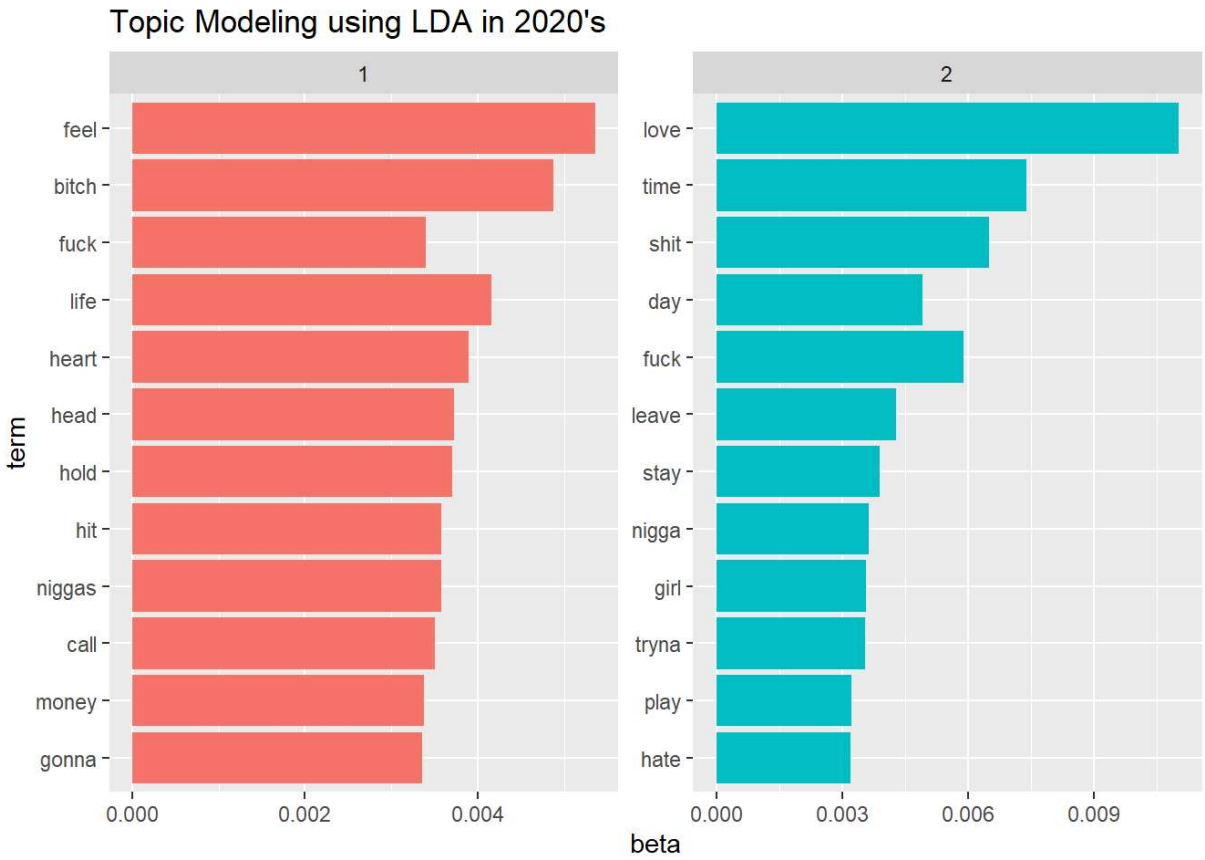



```
grid.arrange(lda_2000, lda_2010)
```






```
lda_2020
```



For 1990's and 2000's, the topics can roughly separate as positive and negative. In 1980's, 2010's, 2020's, it's hard to tell the different between two topics, the topics might are more uniformly distributed.

In each type of music

Country

```
df_filtered_country = df2_words %>% filter(tag %in% "country")

tidy_lda = df_filtered_country %>%
  ungroup() %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2) %>%
  select(artist, title, tag, word)

topics = LDA(cast_dtm(data = tidy_lda %>%
                        count(artist, word) %>%
                        ungroup(),
                      term = word,
                      document = artist,
                      value = n),
             k = 2, control = list(seed = 42)) %>%
  tidy(matrix = "beta") %>%
  group_by(topic) %>%
  arrange(desc(beta)) %>%
  top_n(12, beta) %>%
  ungroup()

lda_country = topics %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = 'free') +
  coord_flip() +
  ggtitle("Topic Modeling using LDA: Country")
```

Miscellaneous

```
df_filtered_misc = df2_words %>% filter(tag %in% "misc")

tidy_lda = df_filtered_misc %>%
  ungroup() %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2) %>%
  select(artist, title, tag, word)

topics = LDA(cast_dtm(data = tidy_lda %>%
                        count(artist, word) %>%
                        ungroup(),
                      term = word,
                      document = artist,
                      value = n),
             k = 2, control = list(seed = 42)) %>%
  tidy(matrix = "beta") %>%
  group_by(topic) %>%
  arrange(desc(beta)) %>%
  top_n(12, beta) %>%
  ungroup()

lda_misc = topics %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = 'free') +
  coord_flip() +
  ggtitle("Topic Modeling using LDA: Miscellaneous")
```

Pop

```
df_filtered_pop = df2_words %>% filter(tag %in% "pop")

tidy_lda = df_filtered_pop %>%
  ungroup() %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2) %>%
  select(artist, title, tag, word)

topics = LDA(cast_dtm(data = tidy_lda %>%
                        count(artist, word) %>%
                        ungroup(),
                      term = word,
                      document = artist,
                      value = n),
             k = 2, control = list(seed = 42)) %>%
  tidy(matrix = "beta") %>%
  group_by(topic) %>%
  arrange(desc(beta)) %>%
  top_n(12, beta) %>%
  ungroup()

lda_pop = topics %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = 'free') +
  coord_flip() +
  ggtitle("Topic Modeling using LDA: Pop")
```

Rap

```
df_filtered_rap = df2_words %>% filter(tag %in% "rap")
df_filtered_rap = df_filtered_rap %>% group_by(title) %>%
  sample_n(size = round(n() * 0.3))

tidy_lda = df_filtered_rap %>%
  ungroup() %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2) %>%
  select(artist, title, tag, word)

topics = LDA(cast_dtm(data = tidy_lda %>%
                        count(artist, word) %>%
                        ungroup(),
                      term = word,
                      document = artist,
                      value = n),
             k = 2, control = list(seed = 42)) %>%
  tidy(matrix = "beta") %>%
  group_by(topic) %>%
  arrange(desc(beta)) %>%
  top_n(12, beta) %>%
  ungroup()

lda_rap = topics %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = 'free') +
  coord_flip() +
  ggtitle("Topic Modeling using LDA: Rap")
```

R&B

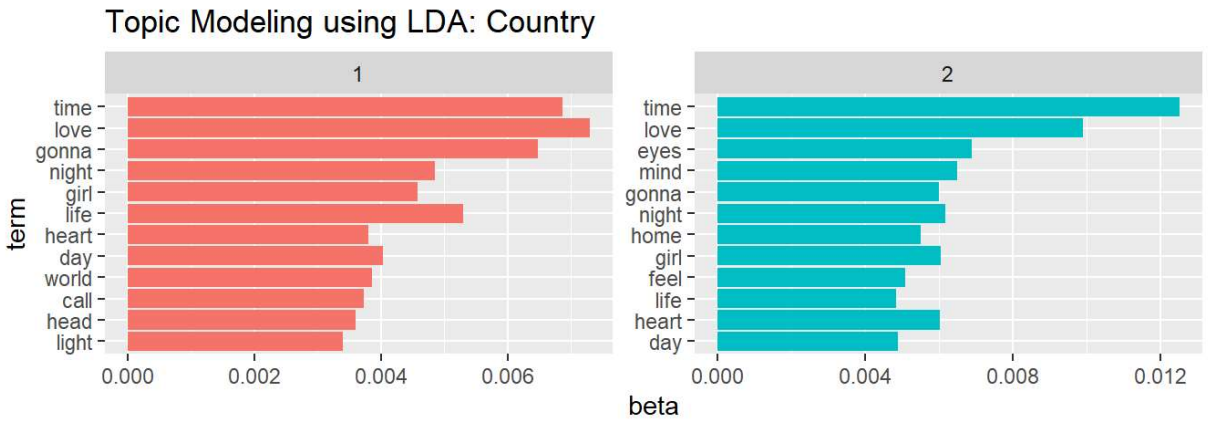
```
df_filtered_rb = df2_words %>% filter(tag %in% "rb")

tidy_lda = df_filtered_rb %>%
  ungroup() %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2) %>%
  select(artist, title, tag, word)

topics = LDA(cast_dtm(data = tidy_lda %>%
                        count(artist, word) %>%
                        ungroup(),
                      term = word,
                      document = artist,
                      value = n),
             k = 2, control = list(seed = 42)) %>%
  tidy(matrix = "beta") %>%
  group_by(topic) %>%
  arrange(desc(beta)) %>%
  top_n(12, beta) %>%
  ungroup()

lda_rb = topics %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = 'free') +
  coord_flip() +
  ggtitle("Topic Modeling using LDA: R&B")
```

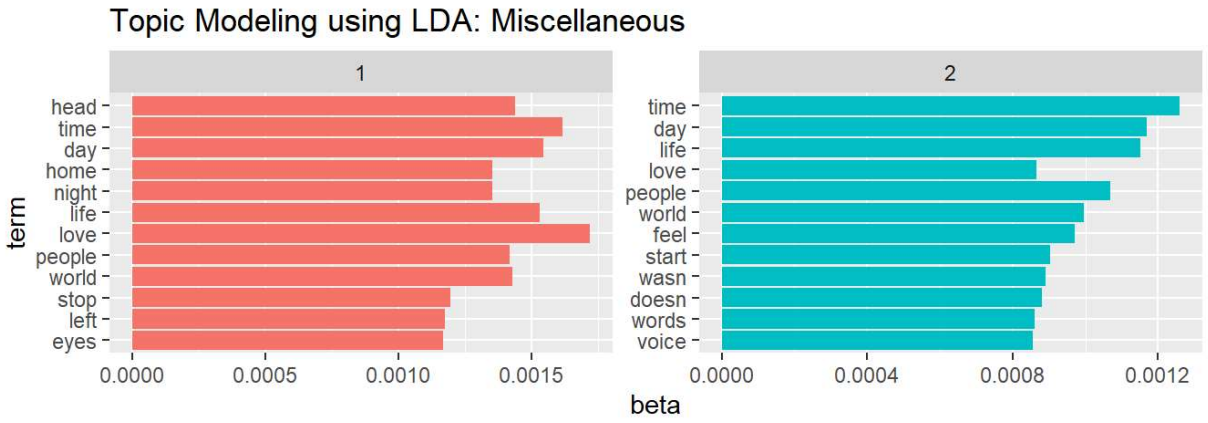
Rock

```
df_filtered_rock = df2_words %>% filter(tag %in% "rock")

tidy_lda = df_filtered_rock %>%
  ungroup() %>%
  distinct() %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 2) %>%
  select(artist, title, tag, word)

topics = LDA(cast_dtm(data = tidy_lda %>%
                        count(artist, word) %>%
                        ungroup(),
                      term = word,
                      document = artist,
                      value = n),
             k = 2, control = list(seed = 42)) %>%
  tidy(matrix = "beta") %>%
  group_by(topic) %>%
  arrange(desc(beta)) %>%
  top_n(12, beta) %>%
  ungroup()

lda_rock = topics %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = 'free') +
  coord_flip() +
  ggtitle("Topic Modeling using LDA: Rock")
```

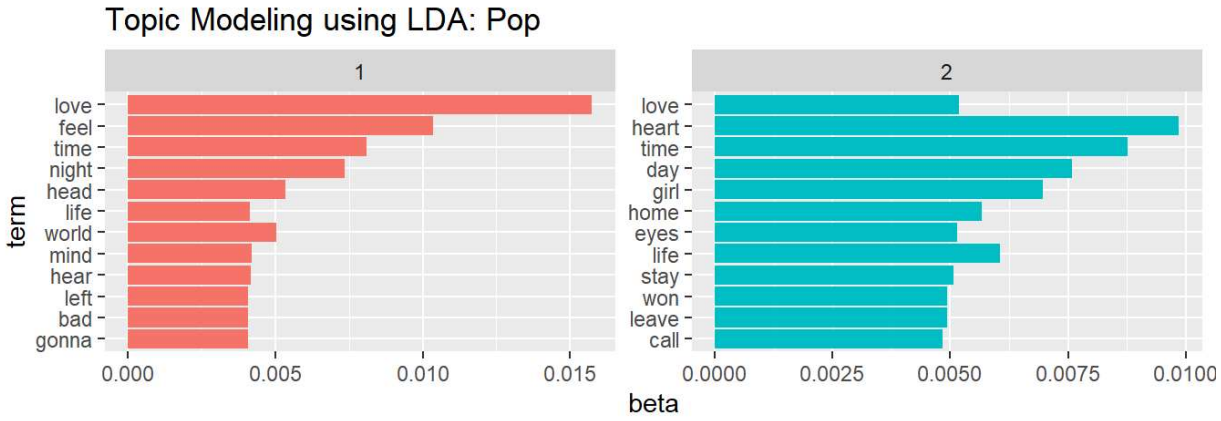
Plot: Music Type

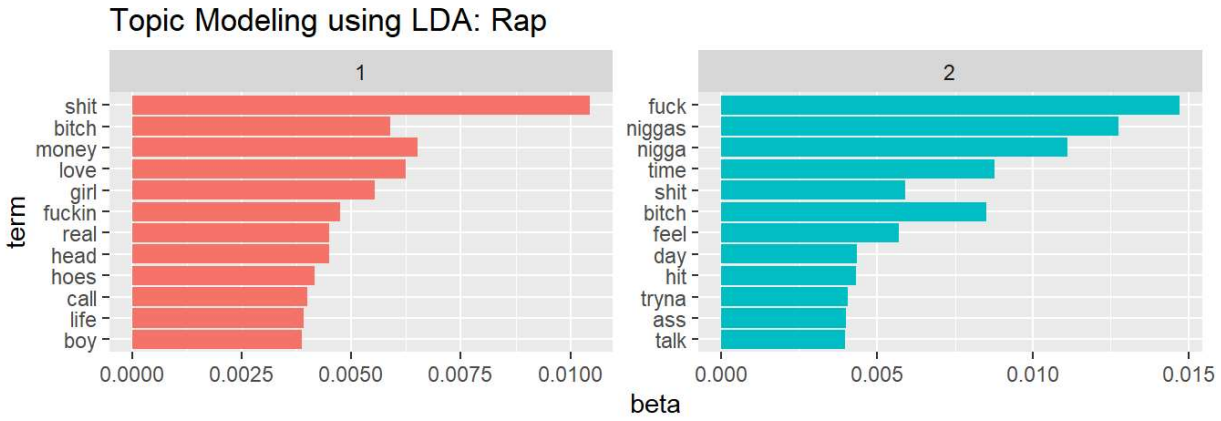
```
grid.arrange(lda_country, lda_misc)
```



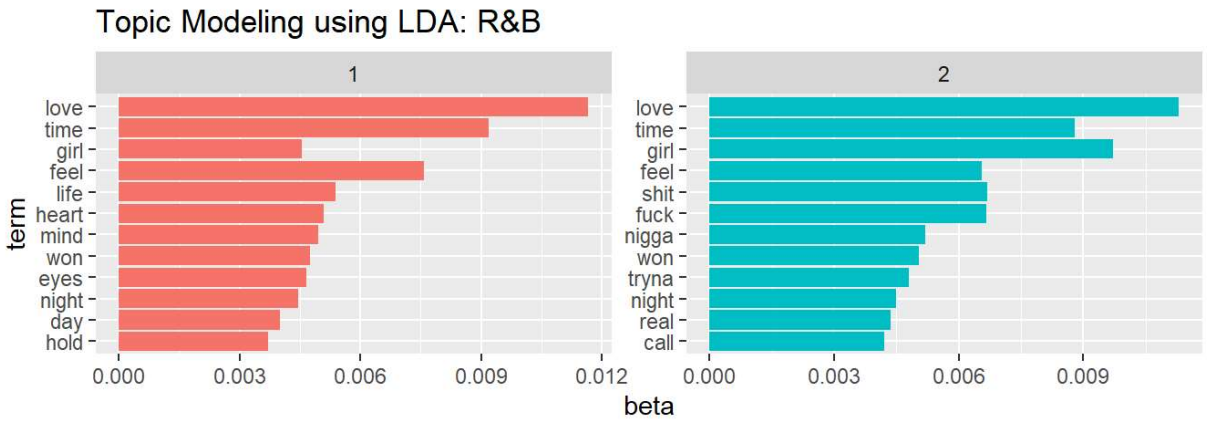


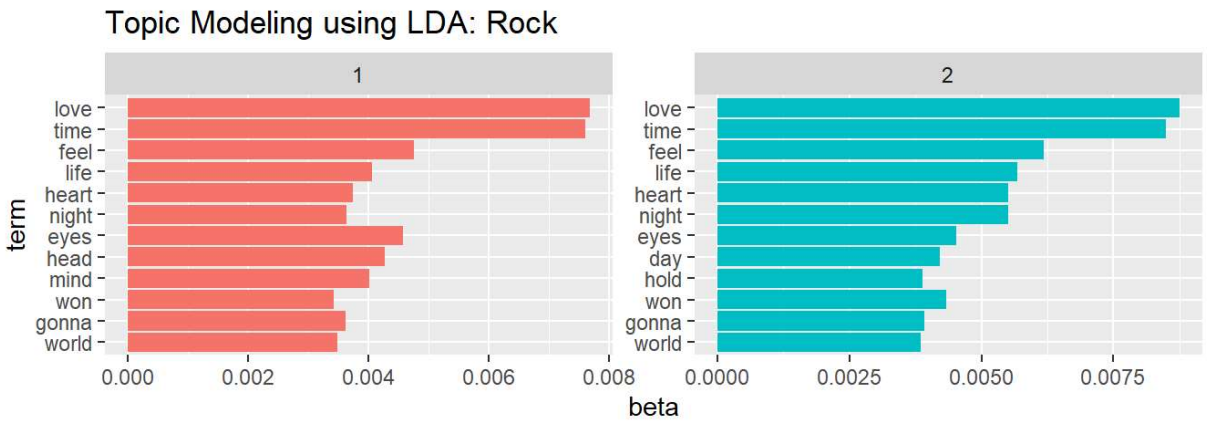

```
grid.arrange(lda_pop, lda_rap)
```






```
grid.arrange(lda_rb, lda_rock)
```





For R&B, the topics can roughly separate as positive and negative. For Rap, both topics have negative and positive words. For other 4 types, both topics are almost all positive words.

Conclusions

Lyrics analysis is not easy for me. It requires a lot of attention during data pre-processing and caution assumptions about this highly unstructured data. Through my analysis, it showed musical lyrics could be very diversified in some aspects, like the positive-negative ratio in certain music types, but at the same time similar in other aspects, like the distribution of different decades. Our analysis also showed that lyrics can be quite different, but some underlying themes like love echo over decades and each types.