Ed Sheeran Lyrics Analysis

Chi Ting Low

Introduction

After learning sentiment analysis and completed sentiment analysis course and tutorial from DataCamp. I am eager to apply what I have learned. I am always interested in text analysis and music. Therefore, in this project I am going to analyze music lyrics.

Ed Sheeran

Edward Christopher Sheeran (Ed Sheeran) is an English singer, songwriter, guitarist, record producer, actor and rapper. He began his career from 2004 till present. Ed Sheeran has produced many top hit songs such as Thinking Out Loud, Perfect, Shape of You and many other songs which frequently appear in many countries billboard list. It was estimated that he has sold more than 26 million alboums and 100 million singles worldwide, which making him one of the world's best-selling music artists.

In this project, it was aimed to discover the secret of his sucess by utilize text mining techniques on his song's lyrics. This project aims to explore word frequency, term frequency inverse document frequency and sentiment lexicon.

Data Preperation and exploration

The data is collected from different website put into dataframe as below:

```
#most of the libraries needed
library(dplyr) #data manipulation
library(tidyr) #Spread, separate, unite, text mining (also included in the tidyverse package)
library(ggplot2) #visualizations
library(ggrepel) #`geom_label_repel`
library(gridExtra) #viewing multiple plots together
library(tidytext) #text mining
library(wordcloud2) #creative visualizations
library(readxl) #read xlsx files
library(knitr) #Create nicely formatted output tables
library(kableExtra) #Create nicely formatted output tables
library(formattable) #For the color_tile function
library(ggthemes) #for ggplot theme

ed_original <- read_xlsx("Lyrics.xlsx")

glimpse(ed_original)
```

```
## Observations: 108
## Variables: 4
## $ Album     <chr> "EP Orange Room", "EP Orange Room", "EP Orange Room"...
## $ Song_name <chr> "I Love you", "Addicted", "Typical Average", "Misery...
## $ Lyrics    <chr> "I'm standing on a mountain\r\nWaiting for you to co...
## $ Year      <dbl> 2005, 2005, 2005, 2005, 2005, 2006, 2006, 2006, 2006...
```

As presented above, the data have 4 variables which are Album, song's name, lyrics from the song and year where the song published. Overall there are 108 songs published by Ed Sheeran.

Data Cleaning

Prior the data analysis, we have to clean the data. Due to the R chracter required and easy interpretation of the lyrics, we have to deal with the contraction. In addition, we also need to remove any special chracter and lowercase of song's lyrics.

```
# function to expand contractions in an English-language source
contractions <- function(doc) {
  # "won't" is a special case as it does not expand to "wo not"
  doc <- gsub("won't", "will not", doc)
  doc <- gsub("can't", "can not", doc)
  doc <- gsub("n't", " not", doc)
  doc <- gsub("'ll", " will", doc)
  doc <- gsub("'re", " are", doc)
  doc <- gsub("'ve", " have", doc)
  doc <- gsub("'m", " am", doc)
  doc <- gsub("'d", " would", doc)
  # 's could be 'is' or could be possessive: it has no expansion
  doc <- gsub("'s", "", doc)
  return(doc)
}

# fix (expand) contractions
ed_original$Lyrics <- sapply(ed_original$Lyrics, contractions)

#function to remove any special chracter
specialChars <- function(x) gsub("[^a-zA-Z0-9 ]", " ", x)

#remove special chracter from data
ed_original$Lyrics <- sapply(ed_original$Lyrics, specialChars)

#convert to lower case
ed_original$Lyrics <- sapply(ed_original$Lyrics, tolower)
```

Text Mining

To begin the analysis, you need tokenized the text to break out the lyrics into individual words and begin mining for insights.

```
#unnest and remove stop, undesirable and short words
words_filtered <- ed_original %>%
  unnest_tokens(word, Lyrics) %>%
  anti_join(stop_words) %>%
  distinct()

dim(words_filtered)
```

```
## [1] 7081    4
```

After tokenized the songs lyrics, there were 7081 words and 4 columns.

Word Frequency

Next we are going to examine the word frequency from the song.

```r
#Factor the album
ed_original$Album <- as.factor(ed_original$Album)

full_word_count <- ed_original %>%
  unnest_tokens(word, Lyrics) %>%
  group_by(Song_name,Album) %>%
  summarise(num_words = n()) %>%
  arrange(desc(num_words))

full_word_count[1:10,] %>%
  ungroup(num_words, Song_name) %>%
  mutate(num_words = color_bar("lightblue")(num_words)) %>%
  kable("html", escape = FALSE, align = "c", caption = "Songs With Highest Word Count") %>%
  kable_styling(bootstrap_options =
                  c("striped", "condensed", "bordered"),
                  full_width = FALSE)
```

Songs With Highest Word Count

Song_name

Album

num_words

You Need Me, I Don't Need You

You Need Me

1103

Take It Back

X

1033

Goodbye To You

No. 5 Collaborations Project

1003

Little Lady

No. 5 Collaborations Project

921

Lately

No. 5 Collaborations Project

831

You Don't Know

The Slumdon Bridge

811

Nina

X

770

London Bridge

The Slumdon Bridge

765

The Man

X

762

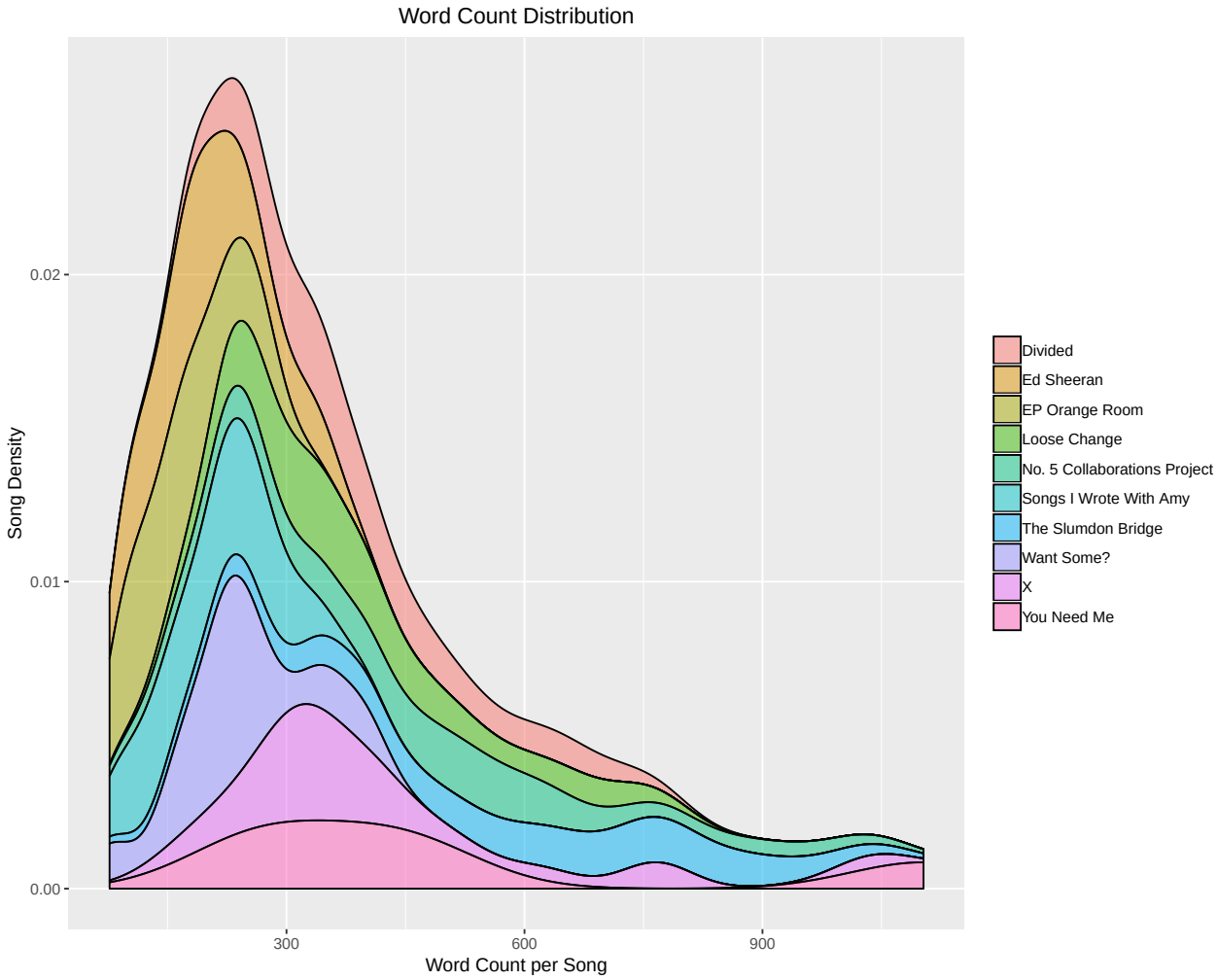
Shape Of You

Divided

701

As you can see, the song 'You Nee Me, I Don't Need You', 'Take It Back' and 'Goodbye To You' is the top three songs with the highest word count.

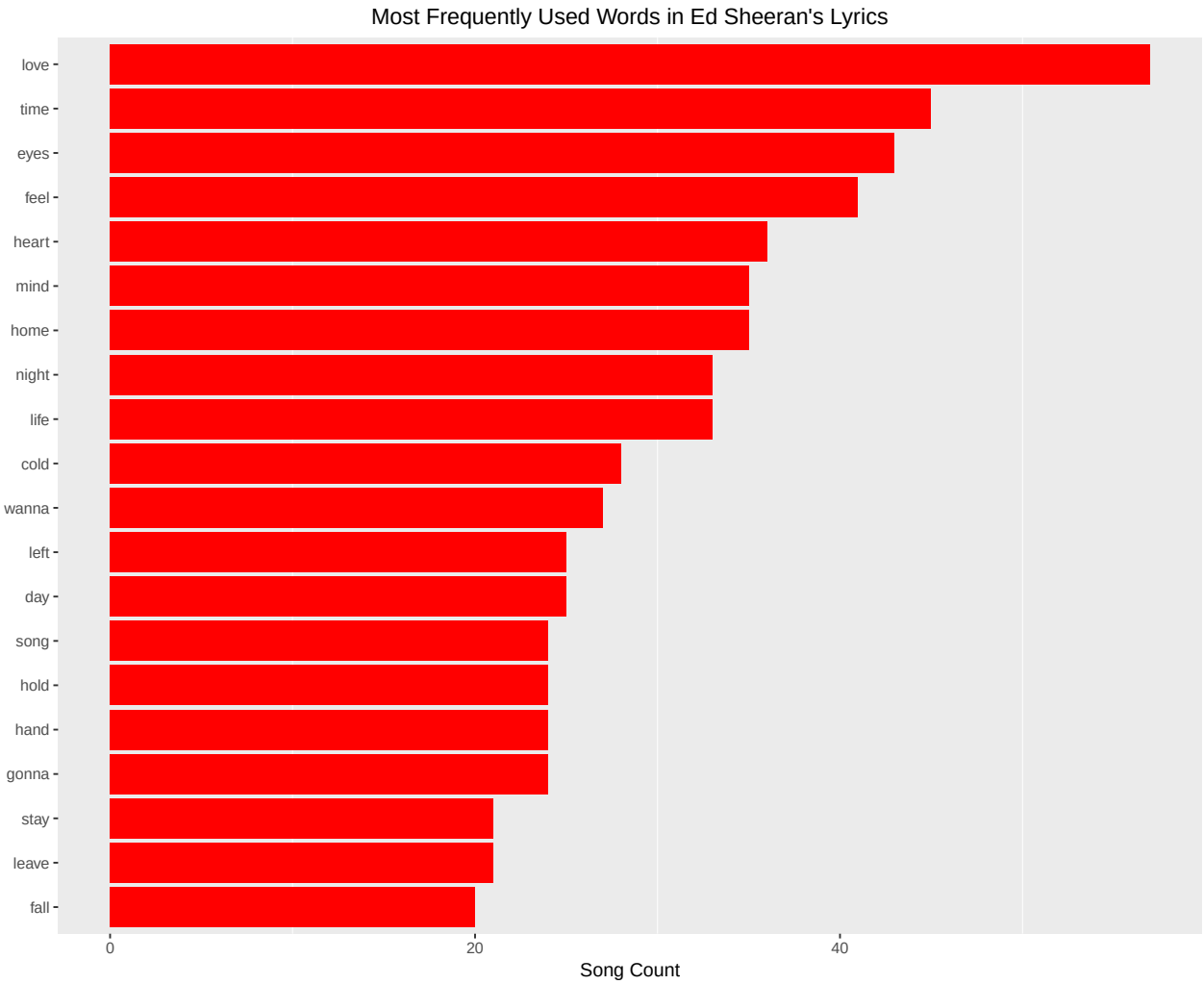
```
full_word_count %>%
  ggplot() +
    geom_density(aes(x = num_words, fill = Album), alpha = 0.5, position = 'stack') +
    ylab("Song Density") +
    xlab("Word Count per Song") +
    ggtitle("Word Count Distribution") +
    theme(plot.title = element_text(hjust = 0.5),
          legend.title = element_blank(),
          panel.grid.minor.y = element_blank())
```



Top Words

Next we are going to examin the top words used by Ed Sheeran when produced his songs. We are going to examine the top 20 words which is most frequently used in the song.

```
words_filtered %>%
  count(word, sort = TRUE) %>%
  top_n(20) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot() +
    geom_col(aes(word, n), fill = 'red') +
    theme(legend.position = "none",
          plot.title = element_text(hjust = 0.5),
          panel.grid.major = element_blank()) +
    xlab("") +
    ylab("Song Count") +
    ggtitle("Most Frequently Used Words in Ed Sheeran's Lyrics") +
    coord_flip()
```



As you can see, **Love**, **time**, **eyes**, **feel**, **heart** are the most frequent word used in Ed Sheeran's lyrics. This is not surprising as he is known for writing love songs.

Words Cloud

Next, we are going to visualize it using wordcloud as presented below;

```
words_counts <- words_filtered %>%
  count(word, sort = TRUE)

wordcloud2(words_counts[1:300, ], size = .5)
```

Words Across Year

In this section we are going to examine is there any changes of the words he used to write his song across the year.

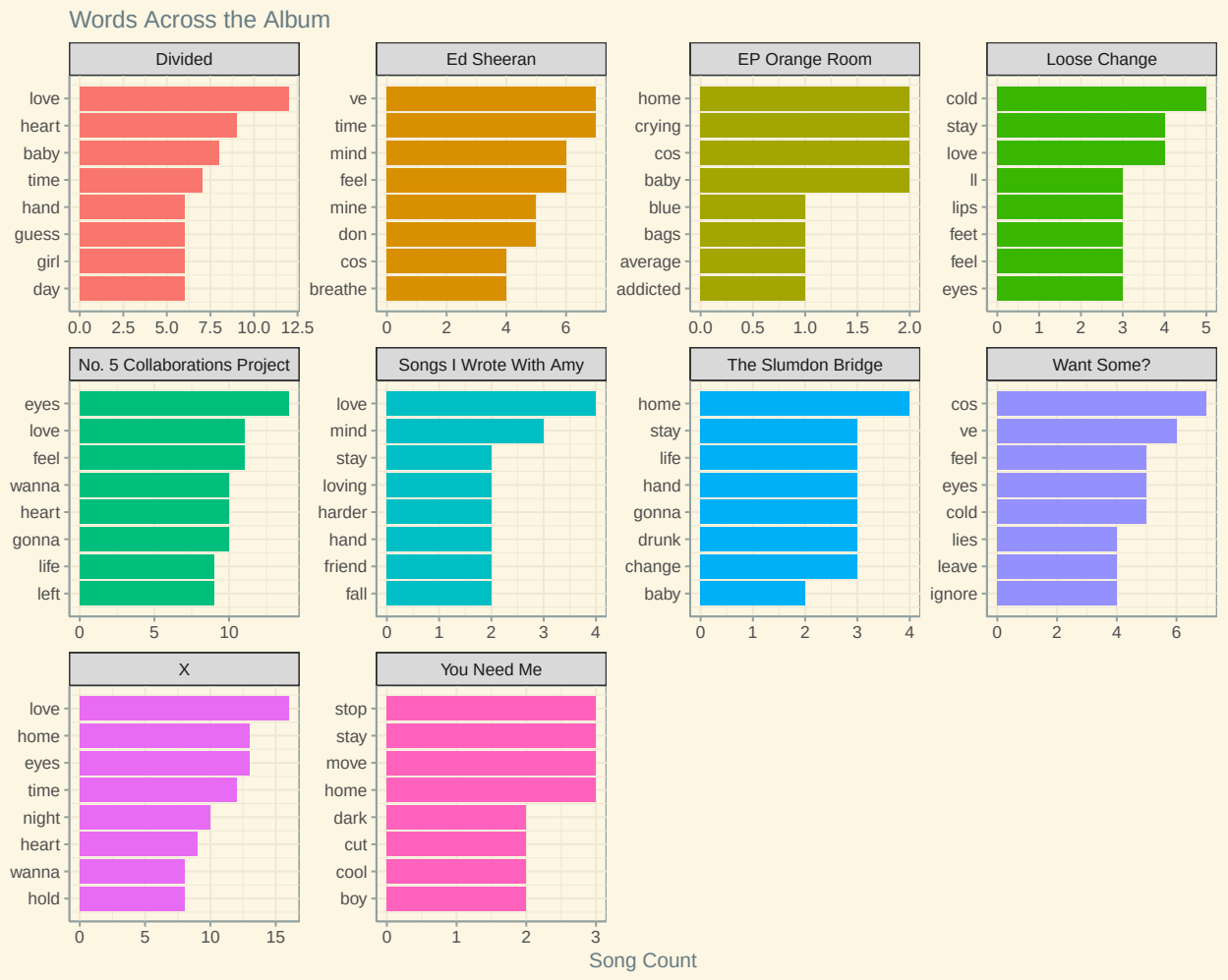
```
words <- words_filtered %>%
  group_by(Year) %>%
  count(word, Year, sort = TRUE) %>%
  slice(seq_len(8)) %>%
  ungroup() %>%
  arrange(Year, n) %>%
  mutate(row = row_number())

words %>%
  ggplot(aes(row, n, fill = Year)) +
    geom_col(show.legend = NULL) +
    labs(x = NULL, y = "Song Count") +
    ggtitle("Words Across the Year") +
    theme_solarized() +
    facet_wrap(~Year, scales = "free") +
```

```
    scale_x_continuous(  # This handles replacement of row
      breaks = words$row, # notice need to reuse data frame
      labels = words$word) +
    coord_flip()
```

There is a different words used across the year. How about we examine is there a different across the album.

```
words <- words_filtered %>%
  group_by(Album) %>%
  count(word, Album, sort = TRUE) %>%
  slice(seq_len(8)) %>%
  ungroup() %>%
  arrange(Album, n) %>%
  mutate(row = row_number())

words %>%
  ggplot(aes(row, n, fill = Album)) +
    geom_col(show.legend = NULL) +
    labs(x = NULL, y = "Song Count") +
    ggtitle("Words Across the Album") +
    theme_solarized() +
```

```
  facet_wrap(~Album, scales = "free") +
  scale_x_continuous(  # This handles replacement of row
    breaks = words$row, # notice need to reuse data frame
    labels = words$word) +
  coord_flip()
```



As you can see, there is a slight difference where the words used across different albums.

Term Freqeuncy Inverse Document Frequency (TF-IDF)

In this section we are going to examin the term freqeuncy of the song. This approach is different from the word frequency as it attaches a lower weight for commonly used words and a higher weight for words that are not used much in a collection of text.

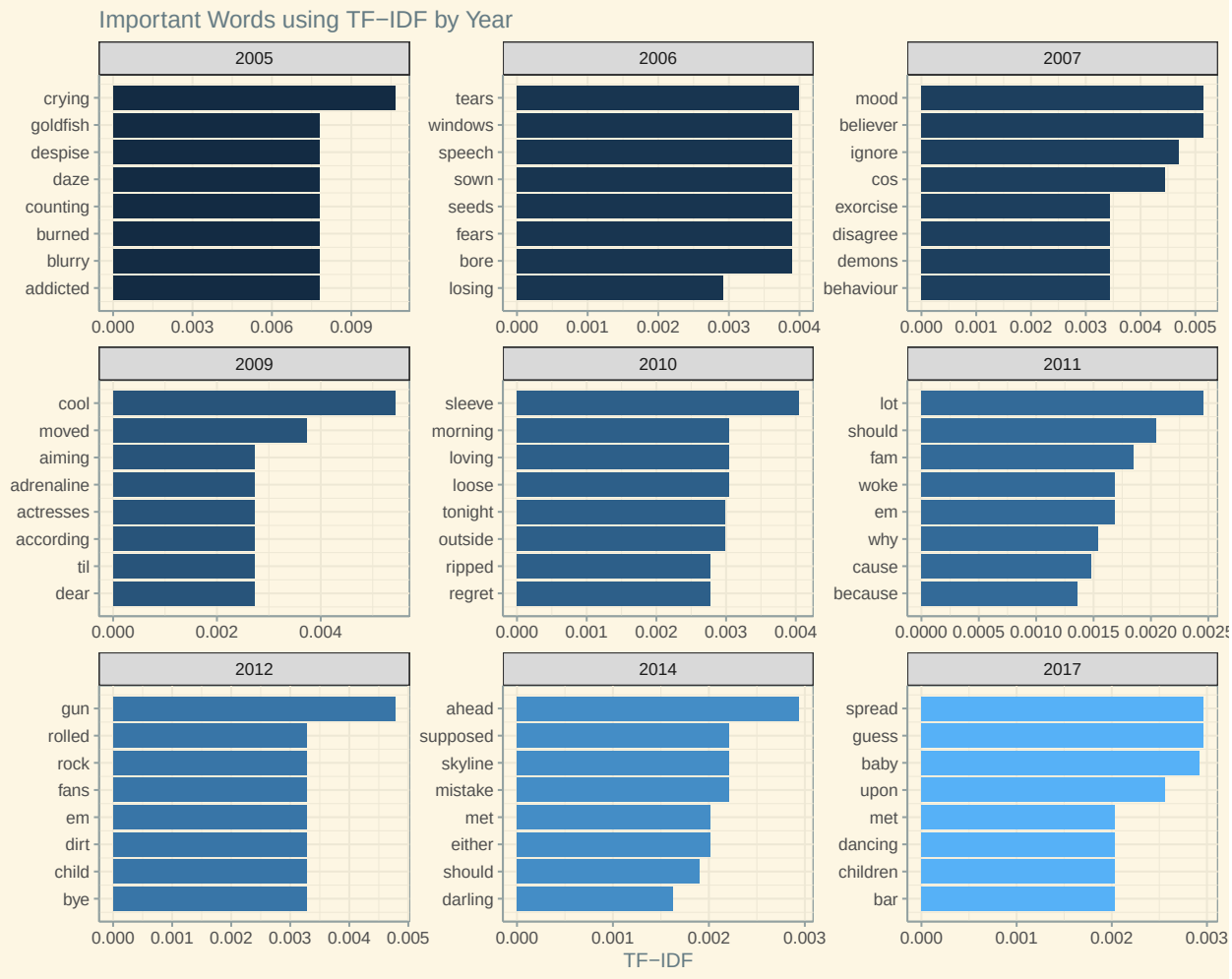
```
tfidf_words <- ed_original %>%
  unnest_tokens(word, Lyrics) %>%
  distinct() %>%
  count(Album, word, sort = TRUE) %>%
  ungroup() %>%
  bind_tf_idf(word, Album, n)
```

```
head(tfidf_words)
```

```
## # A tibble: 6 x 6
##   Album                        word      n      tf   idf tf_idf
##   <fct>                        <chr> <int>   <dbl> <dbl>  <dbl>
## 1 No. 5 Collaborations Project and      21 0.00588     0      0
## 2 No. 5 Collaborations Project i        21 0.00588     0      0
## 3 No. 5 Collaborations Project the      21 0.00588     0      0
## 4 No. 5 Collaborations Project to       21 0.00588     0      0
## 5 No. 5 Collaborations Project will     21 0.00588     0      0
## 6 No. 5 Collaborations Project you      21 0.00588     0      0
```

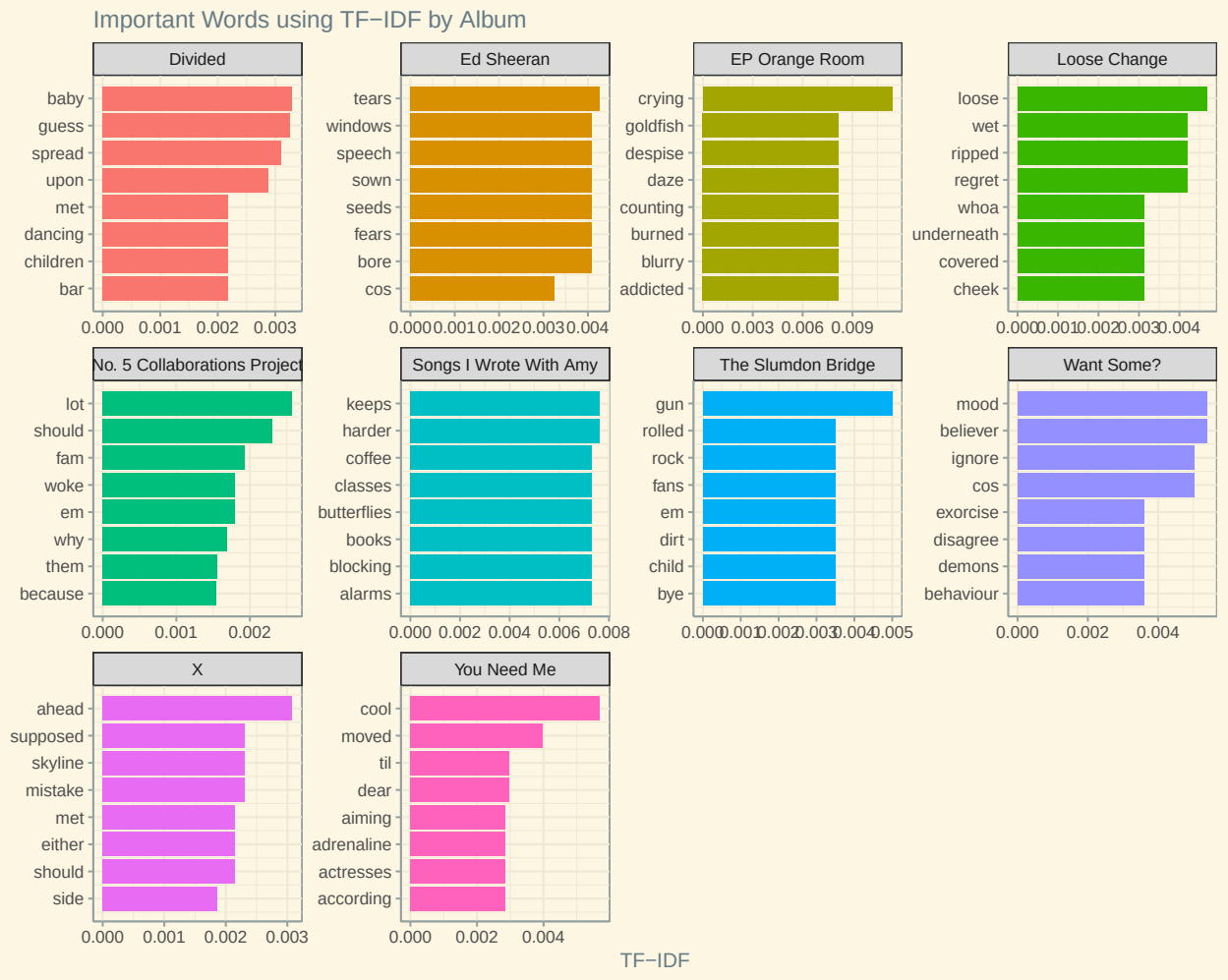
Next we are going to visualize the TF-IDF across different years and album.

```
tfidf_words_year <- ed_original %>%
  unnest_tokens(word, Lyrics) %>%
  distinct() %>%
  count(Year, word, sort = TRUE) %>%
  ungroup() %>%
  bind_tf_idf(word, Year, n) %>%
  arrange(desc(tf_idf))

top_tfidf_words_year <- tfidf_words_year %>%
  group_by(Year) %>%
  slice(seq_len(8)) %>%
  ungroup() %>%
  arrange(Year, tf_idf) %>%
  mutate(row = row_number())

top_tfidf_words_year %>%
  ggplot(aes(x = row, tf_idf, fill = Year)) +
    geom_col(show.legend = NULL) +
    labs(x = NULL, y = "TF-IDF") +
    ggtitle("Important Words using TF-IDF by Year") +
    theme_solarized() +
    facet_wrap(~Year,
               scales = "free") +
    scale_x_continuous(  # this handles replacement of row
      breaks = top_tfidf_words_year$row, # notice need to reuse data frame
      labels = top_tfidf_words_year$word) +
    coord_flip()
```




```
tfidf_words_Album <- ed_original %>%
  unnest_tokens(word, Lyrics) %>%
  distinct() %>%
  count(Album, word, sort = TRUE) %>%
  ungroup() %>%
  bind_tf_idf(word, Album, n) %>%
  arrange(desc(tf_idf))

top_tfidf_words_Album <- tfidf_words_Album %>%
  group_by(Album) %>%
  slice(seq_len(8)) %>%
  ungroup() %>%
  arrange(Album, tf_idf) %>%
  mutate(row = row_number())

top_tfidf_words_Album %>%
  ggplot(aes(x = row, tf_idf, fill = Album)) +
  geom_col(show.legend = NULL) +
  labs(x = NULL, y = "TF-IDF") +
  ggtitle("Important Words using TF-IDF by Album") +
  theme_solarized() +
  facet_wrap(~Album,
```

```
            scales = "free") +
  scale_x_continuous(  # this handles replacement of row
    breaks = top_tfidf_words_Album$row, # notice need to reuse data frame
    labels = top_tfidf_words_Album$word) +
  coord_flip()
```



As you can see, we can gain different insight of the term frequency across different album and years.

Mood of the songs

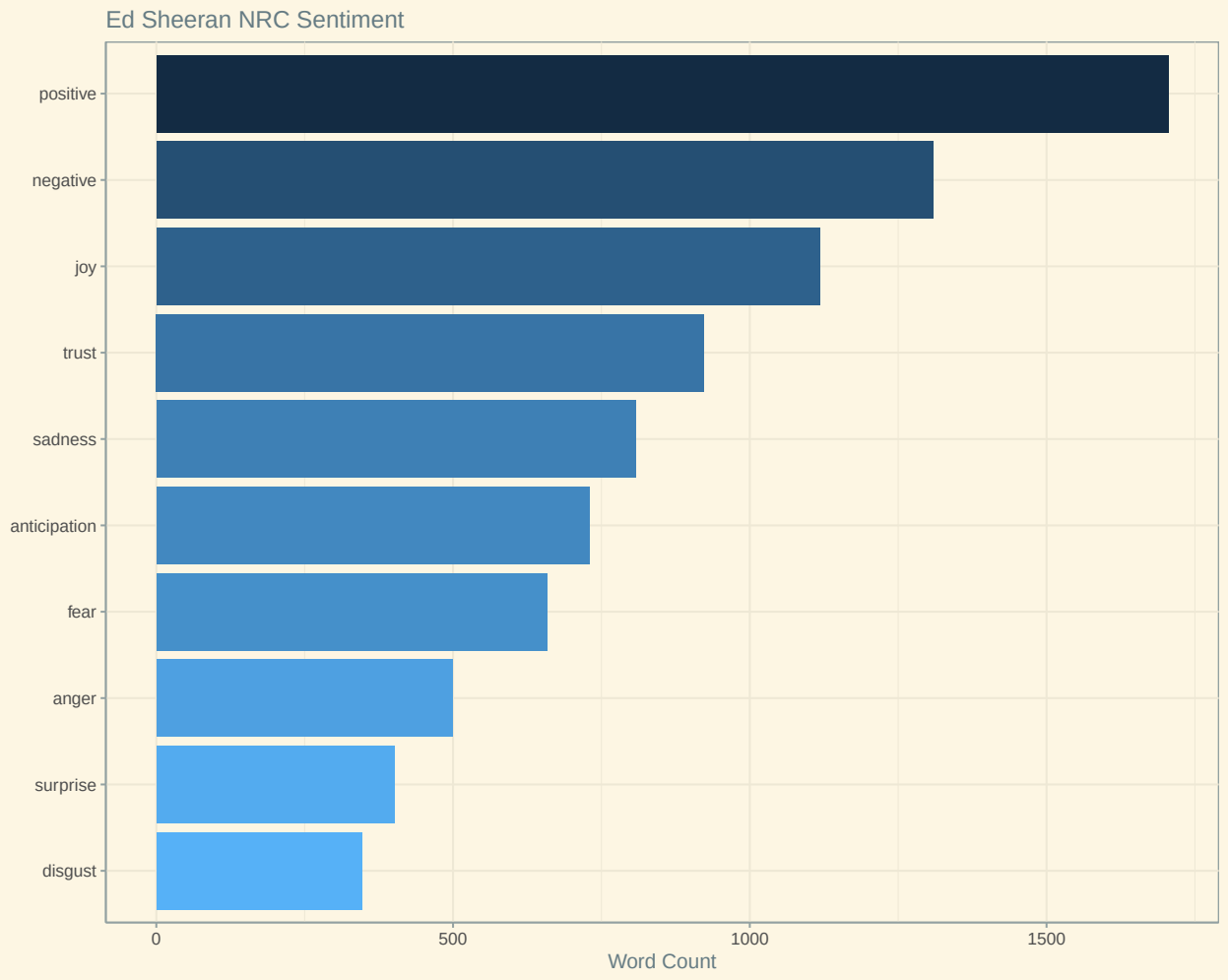
There are three different approaches in tidytext packages use to analyze sentiment lexicons. These are:

- AFINN: assigns words with a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment
- Bing: assigns words into positive and negative categories
- NRC: assigns words into one or more of the following ten categories: positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust

In this project, we are going to use Bing and NRC for analysis.

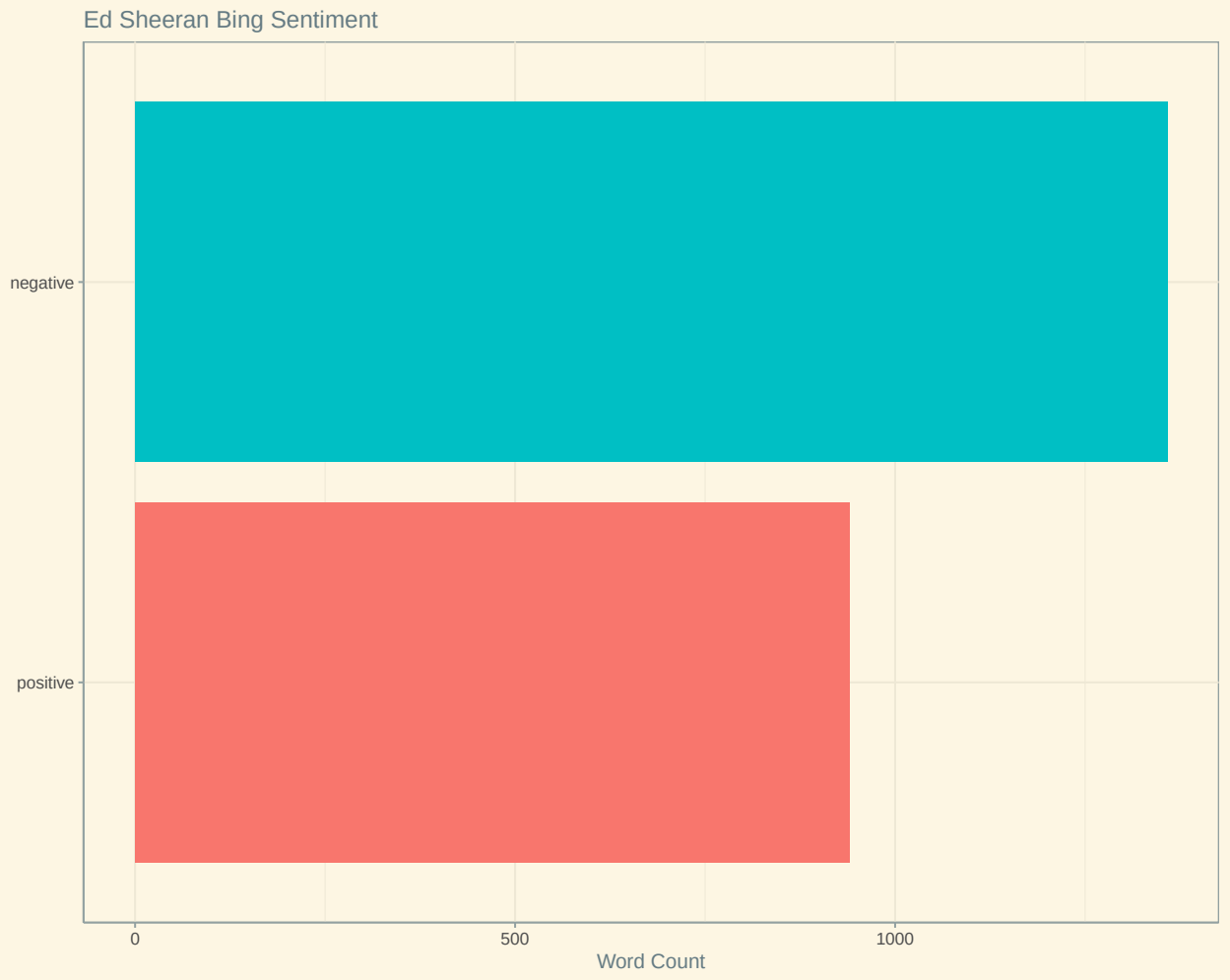
```r
ed_tidy <- ed_original %>%
  unnest_tokens(word, Lyrics) %>% #Break the lyrics into individual words
  anti_join(stop_words) #Data provided by the tidytext package

ed_bing <- ed_tidy %>%
  inner_join(get_sentiments("bing"))

ed_nrc <- ed_tidy %>%
  inner_join(get_sentiments("nrc"))

ed_nrc_sub <- ed_tidy %>%
  inner_join(get_sentiments("nrc")) %>%
  filter(!sentiment %in% c("positive", "negative"))
```

```r
nrc_plot <- ed_nrc %>%
  group_by(sentiment) %>%
  summarise(word_count = n()) %>%
  ungroup() %>%
  mutate(sentiment = reorder(sentiment, word_count)) %>%
  #Use `fill = -word_count` to make the larger bars darker
  ggplot(aes(sentiment, word_count, fill = -word_count)) +
  geom_col() +
  theme_solarized() +
  labs(x = NULL, y = "Word Count") +
  ggtitle("Ed Sheeran NRC Sentiment") +
  coord_flip()

nrc_plot + guides(fill=FALSE)
```



As you can see majority of Ed Sheeran lyrics are categorized as postive.

```
bing_plot <- ed_bing %>%
  group_by(sentiment) %>%
  summarise(word_count = n()) %>%
  ungroup() %>%
  mutate(sentiment = reorder(sentiment, word_count)) %>%
  ggplot(aes(sentiment, word_count, fill = sentiment)) +
  geom_col() +
  guides(fill = FALSE) +
  theme_solarized() +
  labs(x = NULL, y = "Word Count") +
  ggtitle("Ed Sheeran Bing Sentiment") +
  coord_flip()

bing_plot
```



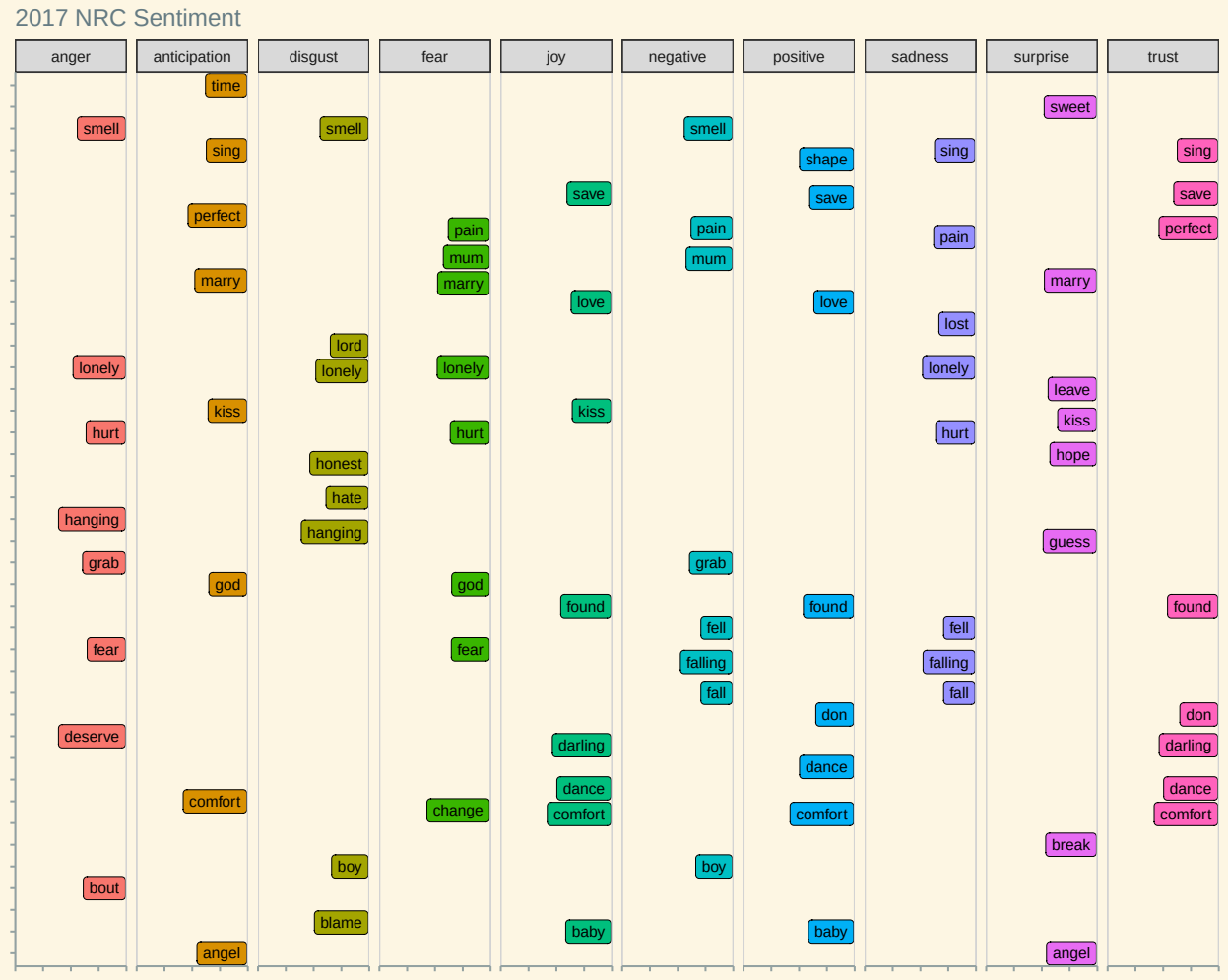
This also agreed with bing analysis.

NRC Sentiment

In this section we are going to examine the mood sentiment used in Ed Sheeran songs from the most recent album, Divide in 2017.

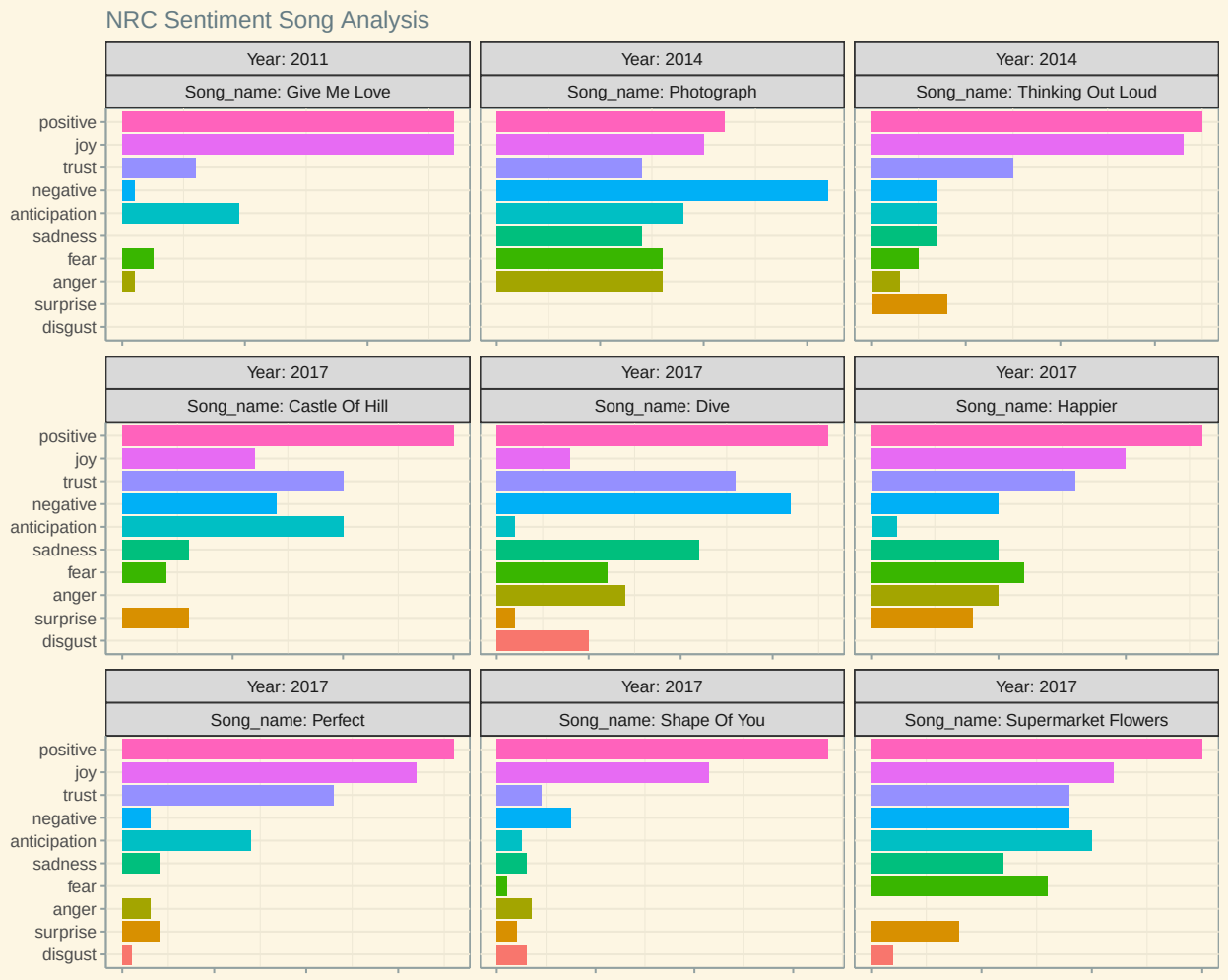
```
plot_words <- ed_nrc %>%
  filter(Year == '2017') %>%
  group_by(sentiment) %>%
  count(word, sort = TRUE) %>%
  arrange(desc(n)) %>%
  slice(seq_len(8)) %>% #consider top_n() from dplyr also
  ungroup()

plot_words %>%
  #Set `y = 1` to just plot one variable and use word as the label
  ggplot(aes(word, 1, label = word, fill = sentiment)) +
  #You want the words, not the points
  geom_point(color = "transparent") +
  #Make sure the labels don't overlap
```

```
geom_label_repel(force = 1,nudge_y = .5,
                 direction = "y",
                 box.padding = 0.04,
                 segment.color = "transparent",
                 size = 3) +
facet_grid(~sentiment) +
theme_solarized() +
theme(axis.text.y = element_blank(), axis.text.x = element_blank(),
      axis.title.x = element_text(size = 6),
      panel.grid = element_blank(), panel.background = element_blank(),
      panel.border = element_rect("lightgray", fill = NA),
      strip.text.x = element_text(size = 9)) +
xlab(NULL) + ylab(NULL) +
ggtitle("2017 NRC Sentiment") +
guides(fill = FALSE) +
coord_flip()
```



As you can see the plot shows the different sentiment presented in different categories. Next we are going to look at the top 9 song by Ed Sheeran and examine the NRC sentiment of each song.

```
ed_nrc %>%
   filter(Song_name %in% c("Perfect", "Shape Of You", "Thinking Out Loud",
                           "Happier", "Castle Of Hill", "Dive", "Give Me Love", 'Photograph', "Supermarket Fl
  count(Song_name, sentiment, Year) %>%
  mutate(sentiment = reorder(sentiment, n), song = reorder(Song_name, n)) %>%
  ggplot(aes(sentiment, n, fill = sentiment)) +
  geom_col() +
  facet_wrap(Year ~ Song_name, scales = "free_x", labeller = label_both) +
  theme_solarized() +
  theme(panel.grid.major.x = element_blank(),
        axis.text.x = element_blank()) +
  labs(x = NULL, y = NULL) +
  ggtitle("NRC Sentiment Song Analysis") +
  coord_flip() +
  guides(fill = FALSE)
```

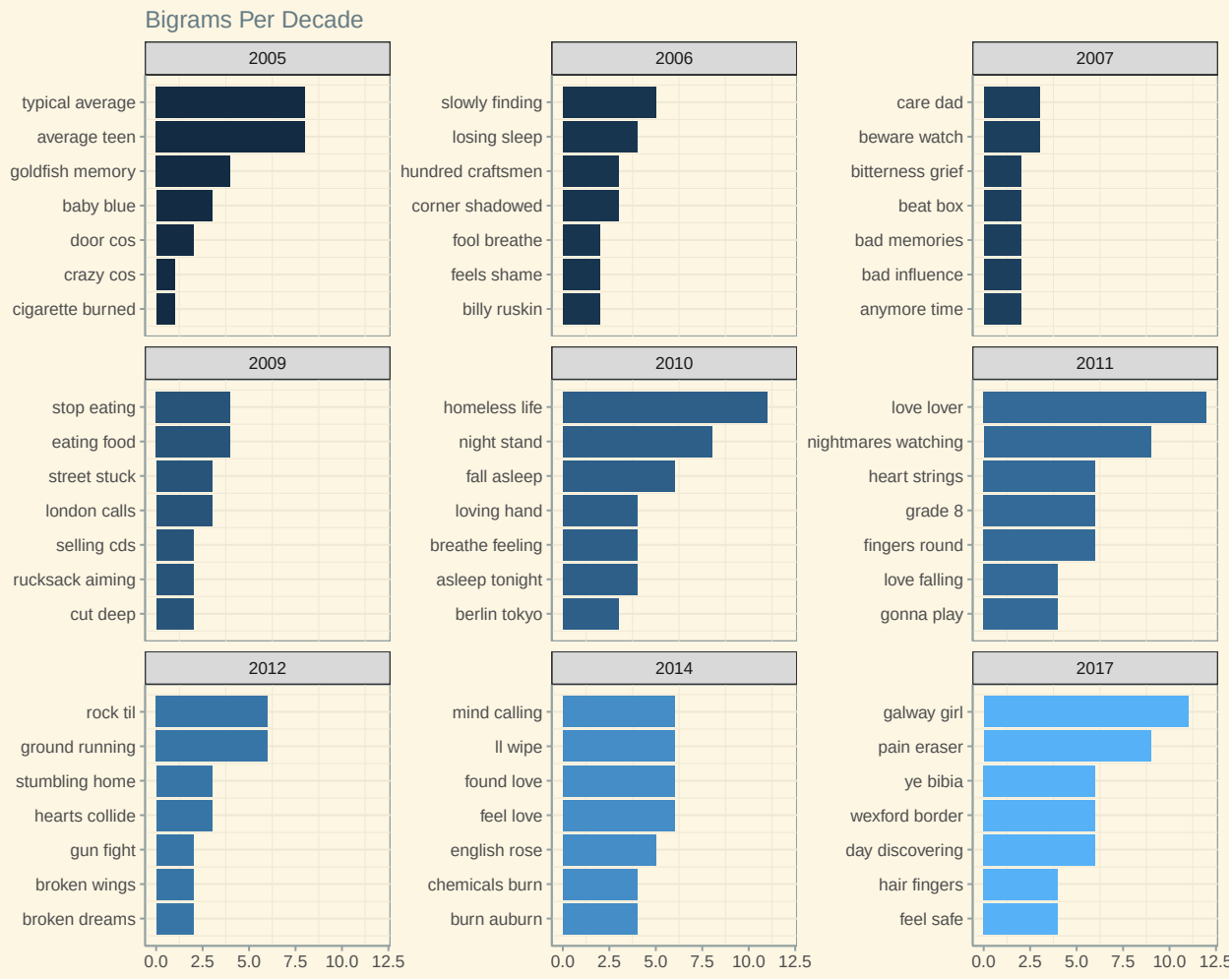


As you can see, love song demostrate more positive and sad song demostrate more anger, sadness and fear.

Bigrams

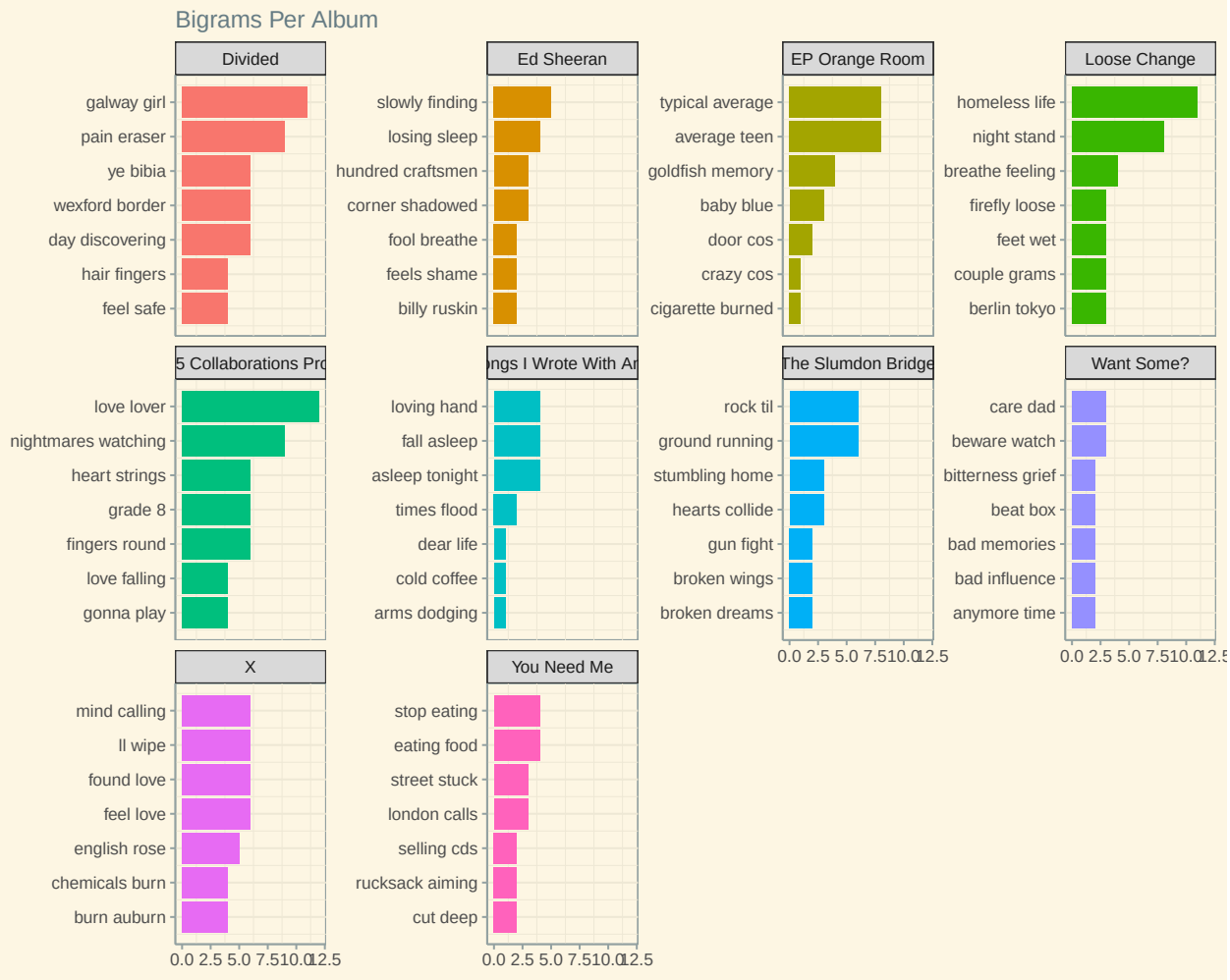
So far we only examine single words section, how about we examine two most frequent words used by utilizing bigram.

```
ed_bigrams <- ed_original %>%
  unnest_tokens(bigram, Lyrics, token = "ngrams", n = 2)

bigrams_separated <- ed_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")

bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

#Because there is so much repetition in music, also filter out the cases where the two words are the sa
bigram_decade <- bigrams_filtered %>%
  filter(word1 != word2) %>%
  unite(bigram, word1, word2, sep = " ") %>%
  inner_join(ed_original) %>%
  count(bigram, Year, sort = TRUE) %>%
  group_by(Year) %>%
  slice(seq_len(7)) %>%
  ungroup() %>%
  arrange(Year, n) %>%
  mutate(row = row_number())

bigram_decade %>%
  ggplot(aes(row, n, fill = Year)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~Year, scales = "free_y") +
  xlab(NULL) + ylab(NULL) +
  scale_x_continuous(  # This handles replacement of row
      breaks = bigram_decade$row, # Notice need to reuse data frame
      labels = bigram_decade$bigram) +
  theme_solarized() +
  theme(panel.grid.major.x = element_blank()) +
  ggtitle("Bigrams Per Decade") +
  coord_flip()
```




```
bigram_album <- bigrams_filtered %>%
  filter(word1 != word2) %>%
  unite(bigram, word1, word2, sep = " ") %>%
  inner_join(ed_original) %>%
  count(bigram, Album, sort = TRUE) %>%
  group_by(Album) %>%
  slice(seq_len(7)) %>%
  ungroup() %>%
  arrange(Album, n) %>%
  mutate(row = row_number())

bigram_album %>%
  ggplot(aes(row, n, fill = Album)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~Album, scales = "free_y") +
  xlab(NULL) + ylab(NULL) +
  scale_x_continuous(  # This handles replacement of row
    breaks = bigram_album$row, # Notice need to reuse data frame
    labels = bigram_album$bigram) +
  theme_solarized() +
  theme(panel.grid.major.x = element_blank()) +
  ggtitle("Bigrams Per Album") +
```

```
coord_flip()
```



As you can see, we can gain different insight on bigram by examine Year and Album from Ed Sheeran.

Conclusion

In conclusion, I am satisfied with what I have learned and applied it using different dataset. In the future, I hope I can applied topic modeling and machine learning and natural language processing method to classifiy Ed Sheeran song genre. You can check out my github