Week 10 / Extra credit

Alina Vikhnevich

2025-05-15

Introduction

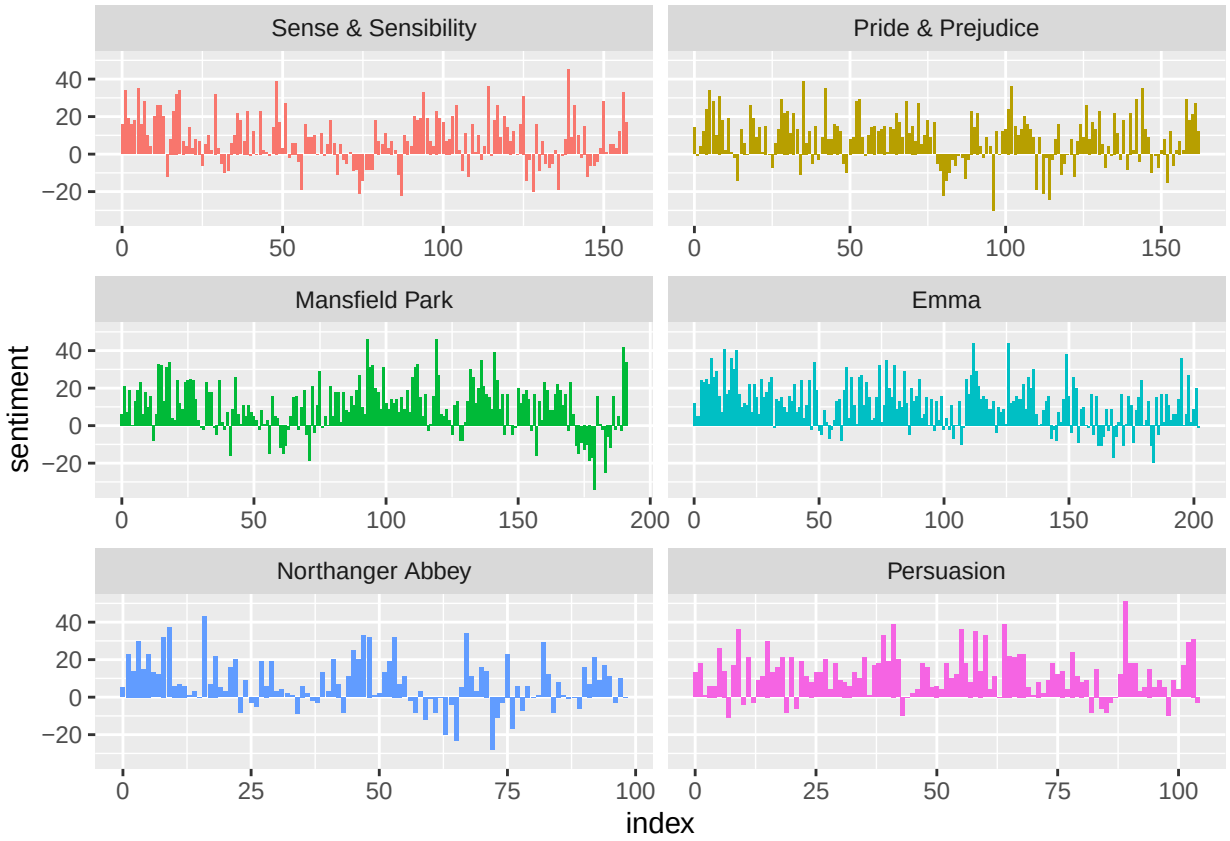
This assignment explores sentiment analysis using tidy data principles, as introduced in Text Mining with R by Julia Silge and David Robinson (https://www.tidytextmining.com/sentiment.html). The primary objective is to analyze how sentiment evolves across literary text by applying tidytext's sentiment lexicons. After replicating the provided example with Jane Austen's novels using the Bing lexicon, I extend the analysis to a new corpus classic song lyrics and introduce an additional lexicon using the sentimentr package to gain deeper insights into emotional tone with contextual nuances.

Replicating Chapter 2 Example: Jane Austen's Novels

Here, I replicate the example from *Text Mining with R* using Jane Austen's novels. The text is transformed into a tidy format, and sentiment scores are calculated using the Bing lexicon. This helps visualize how emotional tone changes across different sections of each novel.

```
tidy_books <- austen_books() %>%
  group_by(book) %>%
  mutate(
    linenumber = row_number(),
    chapter = cumsum(str_detect(text, regex("^chapter [\\divxlc]", ignore_case = TRUE)))
  ) %>%
  ungroup() %>%
  unnest_tokens(word, text)
```

```
bing_sentiment <- tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(book, index = linenumber %/% 80, sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(sentiment = positive - negative)

ggplot(bing_sentiment, aes(index, sentiment, fill = book)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~book, ncol = 2, scales = "free_x")
```



New Corpus: Classic Song Lyrics

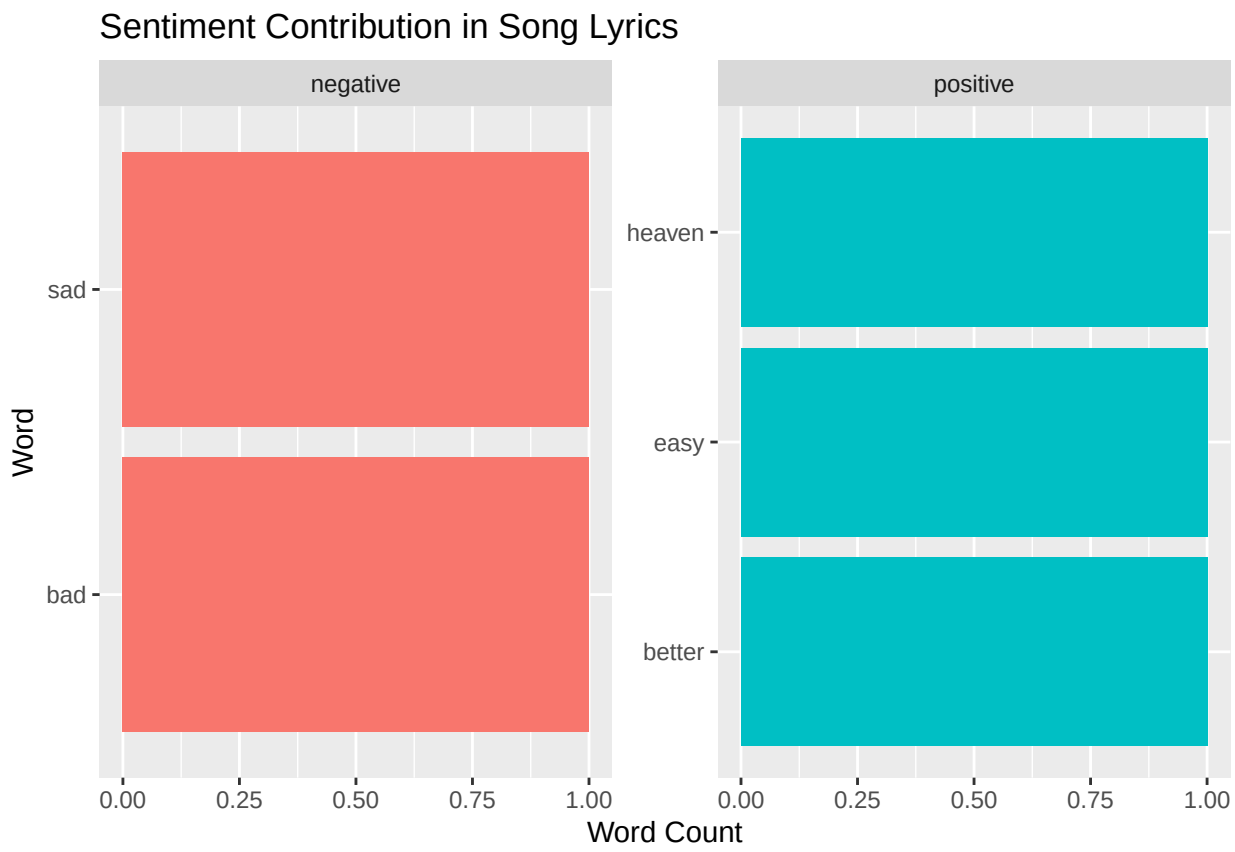
To personalize the analysis, I selected excerpts from iconic Beatles and Bob Dylan lyrics. This smaller, curated corpus lets us explore how tidy sentiment analysis performs on lyrical, metaphor-rich text quite different from prose.

```
lyrics_text <- data.frame(line = 1:6, text = c(
  "Hey Jude, don't make it bad",
  "Take a sad song and make it better",
  "Imagine there's no heaven",
  "It's easy if you try",
  "The answer, my friend, is blowin' in the wind",
  "You may say I'm a dreamer, but I'm not the only one"
))

lyrics_words <- lyrics_text %>%
  unnest_tokens(word, text)
```

Sentiment Analysis Using Bing Lexicon

I applied the Bing sentiment lexicon to the lyrics corpus. This binary lexicon labels each word as either positive or negative. The bar plot reveals which words most strongly contribute to the overall tone of the lyrics.

```r
bing_lyrics <- lyrics_words %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE)

ggplot(bing_lyrics, aes(n, reorder(word, n), fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(title = "Sentiment Contribution in Song Lyrics", x = "Word Count", y = "Word")
```



Extended Analysis with sentimentr

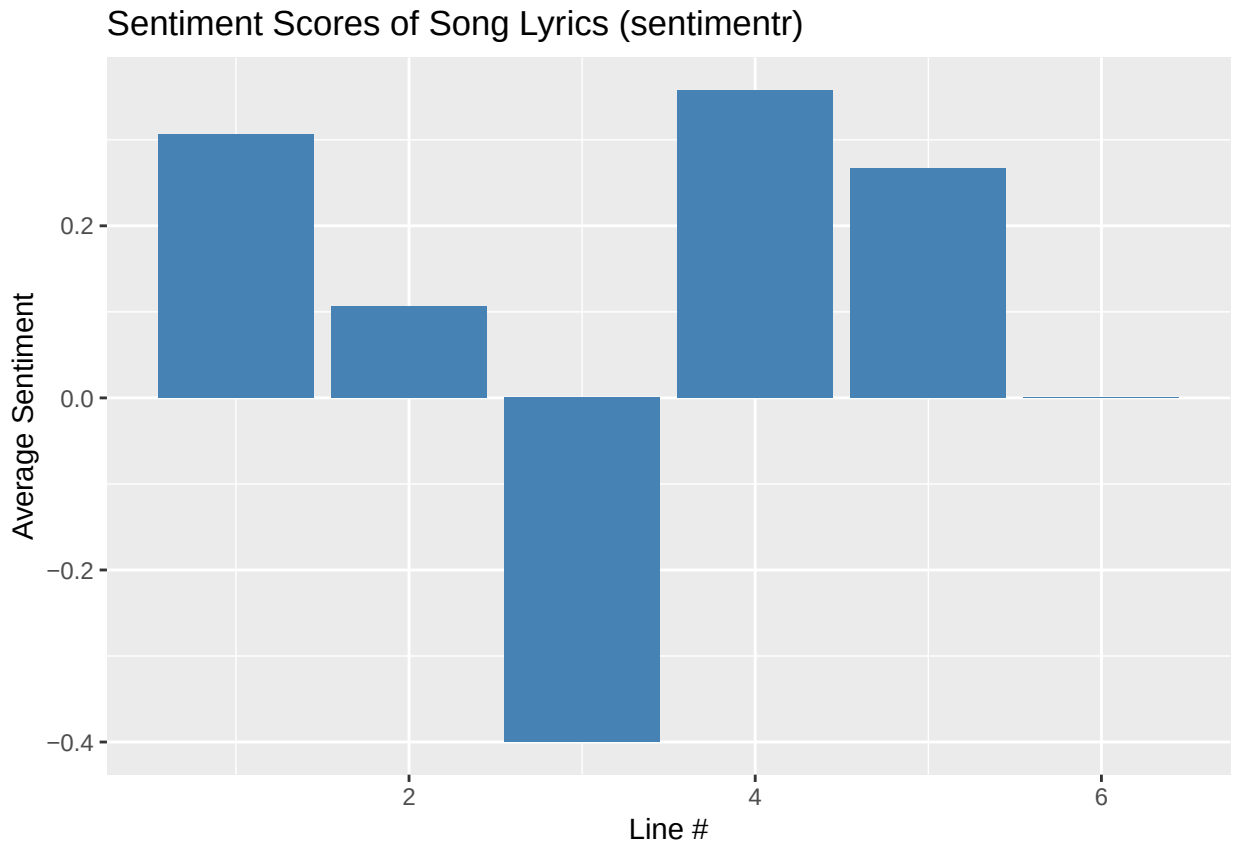
While tidytext sentiment lexicons are useful, they don't handle word context (e.g., "not good"). Here, I use the `sentimentr` package to compute sentiment at the sentence level, accounting for negators and amplifiers. The plot shows average sentiment by line, offering a nuanced emotional profile of the lyrics.

```r
sentiment_result <- sentiment_by(lyrics_text$text)
sentiment_result
```

```
## Key: <element_id>
##    element_id word_count    sd ave_sentiment
##         <int>      <int> <num>         <num>
## 1:          1          6    NA     0.3061862
## 2:          2          8    NA     0.1060660
## 3:          3          4    NA    -0.4000000
```

```
## 4:           4           5    NA     0.3577709
## 5:           5           9    NA     0.2666667
## 6:           6          12    NA     0.0000000
```

```
ggplot(sentiment_result, aes(element_id, ave_sentiment)) +
  geom_col(fill = "steelblue") +
  labs(title = "Sentiment Scores of Song Lyrics (sentimentr)", x = "Line #", y = "Average Sentiment")
```

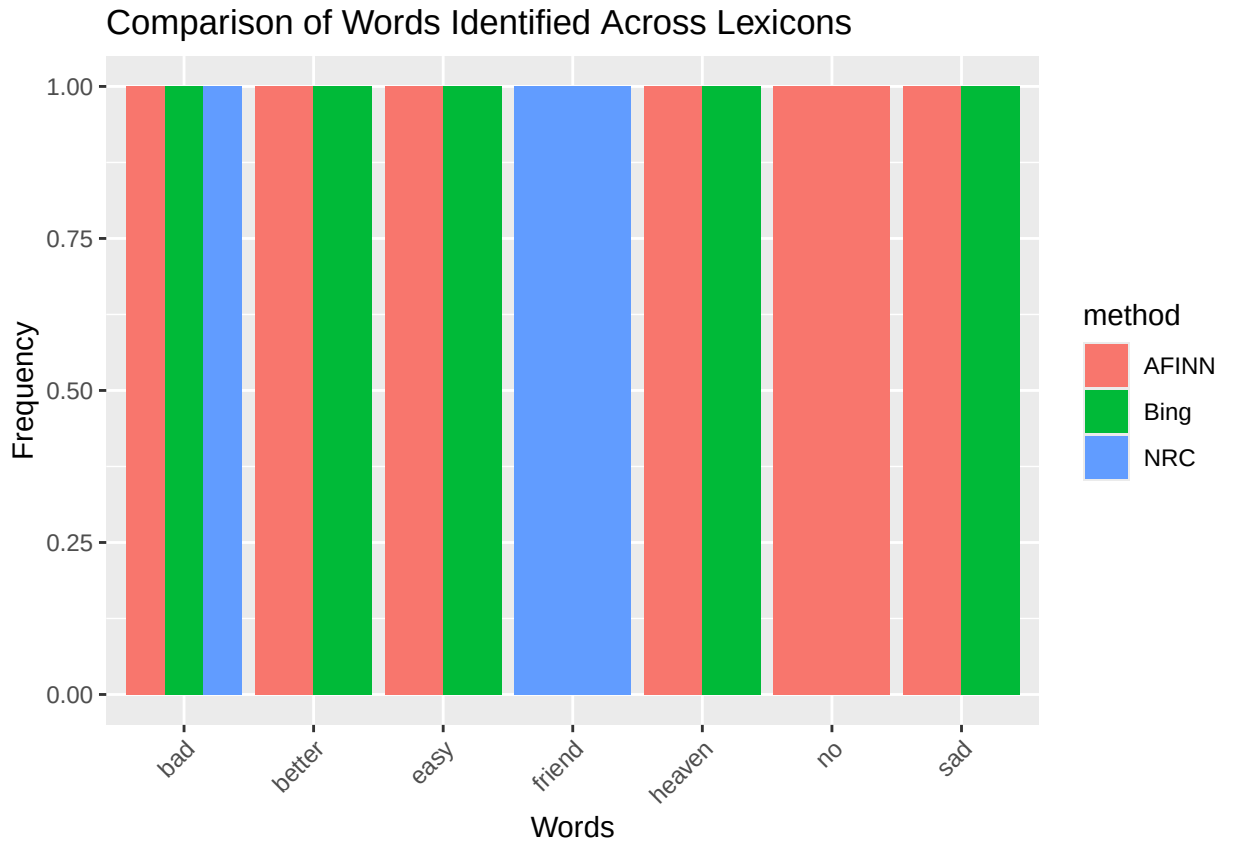


Lexicon Comparison on Song Lyrics

Different lexicons interpret sentiment differently. This section compares AFINN, Bing, and NRC results on the same lyrics. It highlights how some words are recognized by all lexicons, while others are unique to one. This helps demonstrate why lexicon choice matters in text analysis.

```
lyrics_afinn <- lyrics_words %>%
  inner_join(get_sentiments("afinn")) %>%
  mutate(method = "AFINN")

lyrics_bing <- lyrics_words %>%
  inner_join(get_sentiments("bing")) %>%
  mutate(method = "Bing")

lyrics_nrc <- lyrics_words %>%
  inner_join(get_sentiments("nrc") %>% filter(sentiment %in% c("positive", "negative"))) %>%
```

```
  mutate(method = "NRC")

combined_sentiment <- bind_rows(lyrics_afinn, lyrics_bing, lyrics_nrc)

ggplot(combined_sentiment, aes(x = word, fill = method)) +
  geom_bar(position = "dodge") +
  labs(title = "Comparison of Words Identified Across Lexicons", x = "Words", y = "Frequency") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

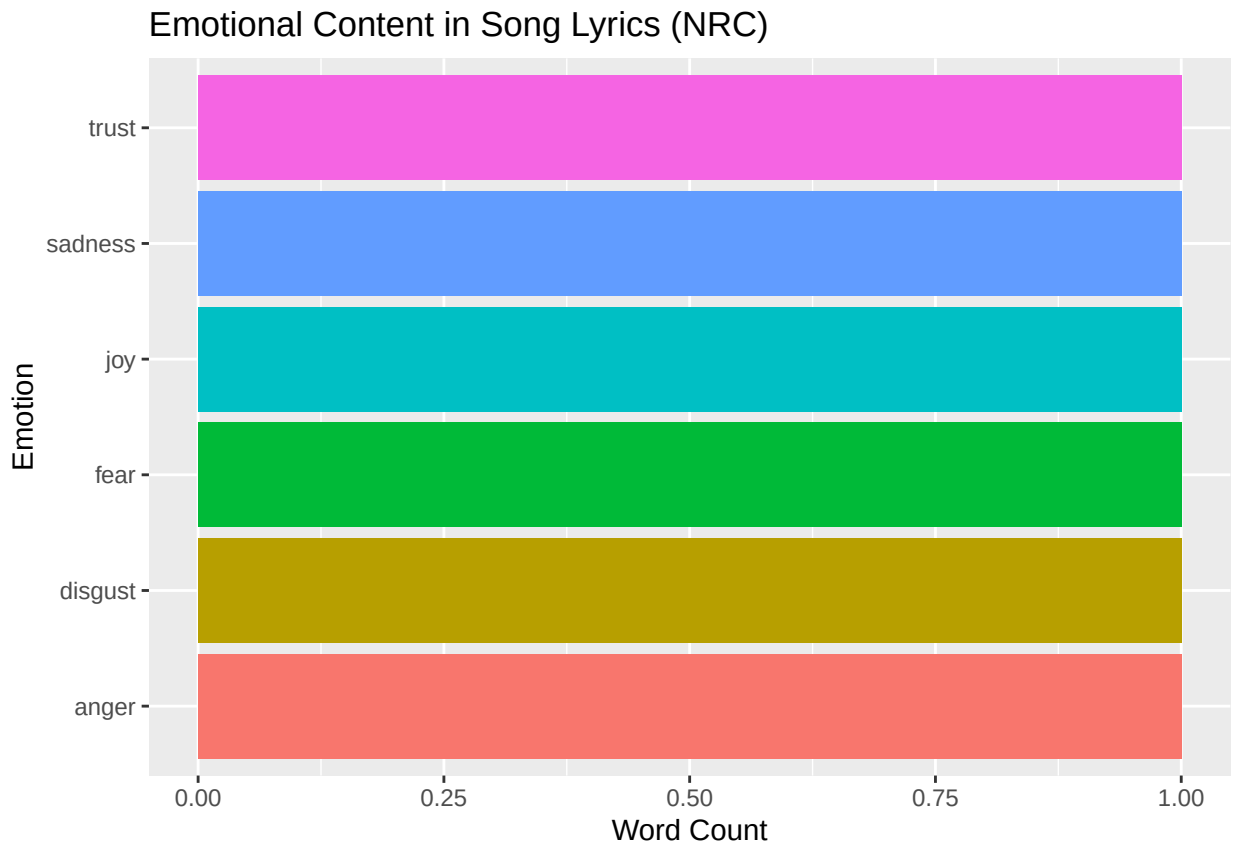


NRC Emotion Distribution

Beyond positive/negative polarity, the **NRC** lexicon captures emotional categories like *joy*, *anger*, *fear*, and *trust*. The bar chart shows how frequently each emotion appears in the lyrics, revealing deeper layers of emotional meaning.

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

emotion_counts <- lyrics_words %>%
  inner_join(nrc_emotions) %>%
  count(sentiment, sort = TRUE)

ggplot(emotion_counts, aes(x = reorder(sentiment, n), y = n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
```

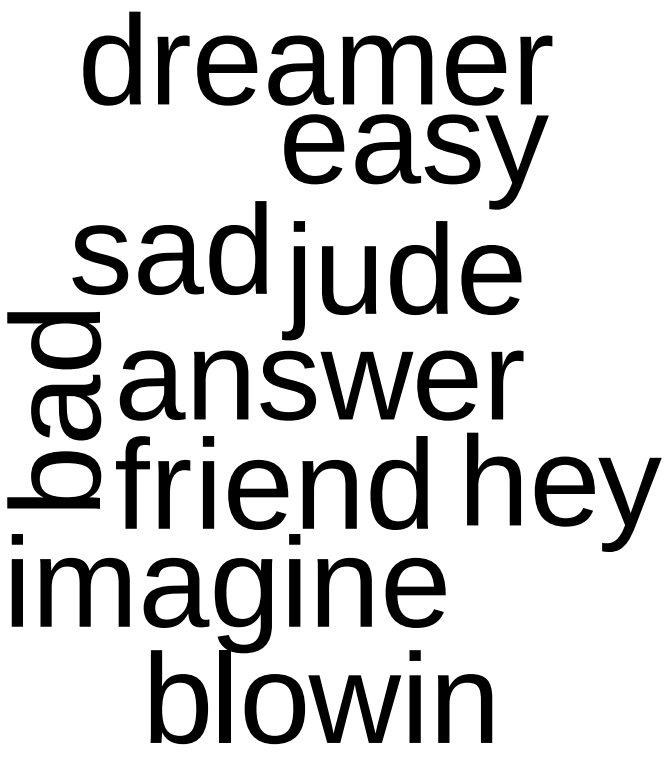
```
  coord_flip() +
  labs(title = "Emotional Content in Song Lyrics (NRC)", x = "Emotion", y = "Word Count")
```



Wordcloud of Most Common Words

A wordcloud is a simple but effective way to visualize the most frequent non-stopwords in the corpus. This helps identify key themes or repeated motifs, especially useful in song lyrics where repetition is common.

```
lyrics_words %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100))
```



Most Positive and Negative Lines (sentimentr)

Using the detailed output from `sentimentr`, I identified the most emotionally intense lines. This can be useful in applications like lyric recommendations, mood-based playlist curation, or narrative editing.

```
sentiment_scores <- sentiment(lyrics_text$text)

most_positive <- lyrics_text$text[which.max(sentiment_scores$sentiment)]
most_negative <- lyrics_text$text[which.min(sentiment_scores$sentiment)]

cat("Most Positive Line:", most_positive, "\\n")
```

```
## Most Positive Line: It's easy if you try \n
```

```
cat("Most Negative Line:", most_negative, "\\n")
```

```
## Most Negative Line: Imagine there's no heaven \n
```

Conclusion

Through this analysis, I demonstrated how tidy sentiment techniques can be adapted to different types of text from long-form novels to short lyrical excerpts. Replicating the Jane Austen example established a

strong foundation, while the custom song lyric analysis allowed for creative exploration beyond the original scope.

The use of multiple lexicons (AFINN, Bing, NRC) revealed that sentiment results can vary based on lexicon structure and word coverage. The NRC emotion categories helped uncover nuanced emotional tones such as trust, sadness, or joy, while 'sentimentr' added contextual depth, capturing line-level sentiment with greater subtlety. Identifying the most emotionally charged lines and generating a wordcloud made the analysis more interpretive and visually engaging.

These findings underscore the importance of selecting the right tools based on text structure, context, and intended insight and show how sentiment analysis can be both a quantitative and interpretive process.