Text analysis

Prof. Maria Tackett

Click for PDF of slides

Packages

In addition to **`tidyverse`** we will be using a few other packages today

```
library(tidyverse)
library(tidytext)
library(genius) # https://github.com/JosiahParry/genius
```

Tidy Data

What makes a data frame tidy?

Tidy Data

What makes a data frame tidy?

1. Each variable must have its own column.
2. Each observation must have its own row.
3. Each value must have its own cell.

Tidytext

- Using tidy data principles can make many text mining tasks easier, more effective, and consistent with tools already in wide use.
- Learn more at **https://www.tidytextmining.com/**.

What is tidy text?

```
text <- c("On your mark ready set let's go",
          "dance floor pro",
          "I know you know I go psycho",
          "When my new joint hit",
          "just can't sit",
          "Got to get jiggy wit it",
          "ooh, that's it")

text
```

```
## [1] "On your mark ready set let's go" "dance floor pro"
## [3] "I know you know I go psycho"     "When my new joint hit"
## [5] "just can't sit"                  "Got to get jiggy wit it"
## [7] "ooh, that's it"
```

What is tidy text?

```
text_df <- tibble(line = 1:7, text = text)

text_df
```

```
## # A tibble: 7 x 2
##    line text
##   <int> <chr>
## 1     1 On your mark ready set let's go
## 2     2 dance floor pro
## 3     3 I know you know I go psycho
## 4     4 When my new joint hit
## 5     5 just can't sit
## 6     6 Got to get jiggy wit it
## 7     7 ooh, that's it
```

What is tidy text?

```
text_df %>%
  unnest_tokens(word, text)
```

```
## # A tibble: 34 x 2
##     line word
##    <int> <chr>
##  1     1 on
##  2     1 your
##  3     1 mark
##  4     1 ready
##  5     1 set
##  6     1 let's
##  7     1 go
##  8     2 dance
##  9     2 floor
## 10     2 pro
## # ... with 24
```

Let's get some data

We'll use the **`genius`** package to get song lyric data from Genius.

- **`genius_album()`** allows you to download the lyrics for an entire album in a tidy format.
- Input: Two arguments: artist and album. Supply the quoted name of artist and the album (if it gives you issues check that you have the album name and artists as specified on Genius).
- Output: A tidy data frame with three columns corresponding to the track name, the track number, and lyrics

Let's get some data

```
tswift <- genius_album(
  artist = "Taylor Swift",
  album = "Lover"
  )

tswift
```

```
## # A tibble: 913 x 4
##    track_n  line lyric                                    track_title
##      <int> <int> <chr>                                    <chr>
##  1       1     1 How many days did I spend thinking       I Forgot That You
##  2       1     2 'Bout how you did me wrong, wrong, wron… I Forgot That You
##  3       1     3 Lived in the shade you were throwing     I Forgot That You
##  4       1     4 'Til all of my sunshine was gone, gone,… I Forgot That You
##  5       1     5 And I couldn't get away from ya          I Forgot That You
##  6       1     6 In my feelings more than Drake, so yeah  I Forgot That You
##  7       1     7 Your name on my lips, tongue-tied        I Forgot That You
```

What songs are in the album?

```
tswift %>%
  distinct(track_title)
```

```
## # A tibble: 18 x 1
##    track_title
##    <chr>
##  1 I Forgot That You Existed
##  2 Cruel Summer
##  3 Lover
##  4 The Man
##  5 The Archer
##  6 I Think He Knows
##  7 Miss Americana & The Heartbreak Prince
##  8 Paper Rings
##  9 Cornelia Street
## 10 Death by a Thousand Cuts
```

How long are the songs?

Length is measured by number of lines

```
tswift %>%
  count(track_title, sort = TRUE)
```

```
## # A tibble: 18 x 2
##    track_title                                  n
##    <chr>                                    <int>
##  1 I Think He Knows                            65
##  2 Paper Rings                                 65
##  3 Cruel Summer                                62
##  4 Miss Americana & The Heartbreak Prince      62
##  5 Death by a Thousand Cuts                    59
##  6 Daylight                                    58
##  7 London Boy                                  57
##  8 ME! (Ft. Brendon Urie)                          53
```

Tidy up your lyrics!

```
tswift_lyrics <- tswift %>%
  unnest_tokens(word, lyric)

tswift_lyrics
```

```
## # A tibble: 6,844 x 4
##    track_n  line track_title                word
##      <int> <int> <chr>                      <chr>
##  1       1     1 I Forgot That You Existed how
##  2       1     1 I Forgot That You Existed many
##  3       1     1 I Forgot That You Existed days
##  4       1     1 I Forgot That You Existed did
##  5       1     1 I Forgot That You Existed i
##  6       1     1 I Forgot That You Existed spend
##  7       1     1 I Forgot That You Existed thinking
##  8       1     2 I Forgot That You Existed bout
##  9       1     2 I Forgot That You Existed how
```

What are the most common words?

```
tswift_lyrics %>%
  count(word) %>%
  arrange(desc(n))
```

```
## # A tibble: 1,029 x 2
##    word      n
##    <chr> <int>
##  1 i       396
##  2 you     263
##  3 the     243
##  4 and     155
##  5 my      148
##  6 me      132
##  7 a       117
##  8 to      115
##  9 oh      102
## 10 i        96
```

Stop words

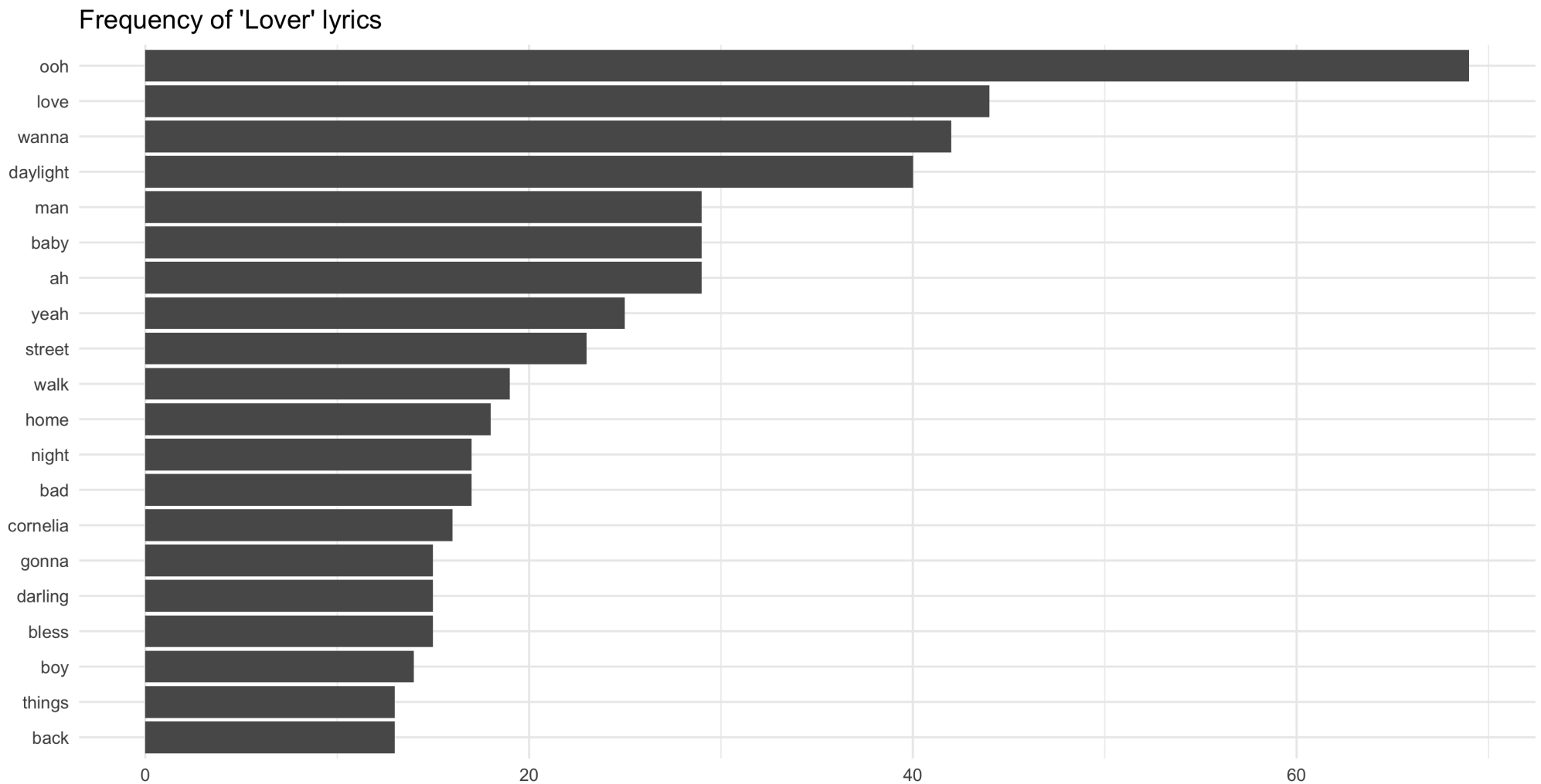
- In computing, stop words are words which are filtered out before or after processing of natural language data (text).
- They usually refer to the most common words in a language, but there is not a single list of stop words used by all natural language processing tools.

What are the most common words?

```
tswift_lyrics %>%
  anti_join(stop_words) %>%
  count(word) %>%
  arrange(desc(n))
```

```
## # A tibble: 759 x 2
##    word          n
##    <chr>     <int>
##  1 ooh          69
##  2 love         44
##  3 wanna        42
##  4 daylight     40
##  5 ah           29
##  6 baby         29
##  7 yeah         25
##  8 street       23
##  9 all          19
```

What are the most common words?



...the code

```
tswift_lyrics %>%
  anti_join(get_stopwords(source = "smart")) %>%
  count(word) %>%
  arrange(desc(n)) %>%
  top_n(20) %>%
  ggplot(aes(fct_reorder(word, n), n)) +
    geom_col() +
    coord_flip() +
    theme_minimal() +
    labs(title = "Frequency of 'Lover' lyrics",
         y = "",
         x = "")
```

Sentiment analysis

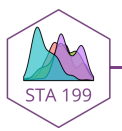
Sentiment analysis

- One way to analyze the sentiment of a text is to consider the text as a combination of its individual words
- The sentiment content of the whole text as the sum of the sentiment content of the individual words
- The sentiment attached to each word is given by a *lexicon*, which may be downloaded from external sources

Sentiment lexicons

```
get_sentiments("afinn")
```

```
## # A tibble: 2,477 x 2
##    word       value
##    <chr>      <dbl>
##  1 abandon       -2
##  2 abandoned     -2
##  3 abandons      -2
##  4 abducted      -2
##  5 abduction     -2
##  6 abductions    -2
##  7 abhor         -3
##  8 abhorred      -3
##  9 abhorrent     -3
## 10 abhors        -3
## #   with 2,467 more rows
```



Sentiment lexicons

```
get_sentiments("afinn")
```

```
## # A tibble: 2,477 x 2
##    word       value
##    <chr>      <dbl>
##  1 abandon       -2
##  2 abandoned     -2
##  3 abandons      -2
##  4 abducted      -2
##  5 abduction     -2
##  6 abductions    -2
##  7 abhor         -3
##  8 abhorred      -3
##  9 abhorrent     -3
## 10 abhors        -3
## #   with 2,467 more rows
```

```
get_sentiments("bing")
```

```
## # A tibble: 6,786 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 2-faces     negative
##  2 abnormal    negative
##  3 abolish     negative
##  4 abominable  negative
##  5 abominably  negative
##  6 abominate   negative
##  7 abomination negative
##  8 abort       negative
##  9 aborted     negative
## 10 aborts      negative
## #   with 6,776 more rows
```

Sentiment lexicons

```
get_sentiments("nrc")
```

```
## # A tibble: 13,901 x 2
##    word         sentiment
##    <chr>        <chr>
##  1 abacus       trust
##  2 abandon      fear
##  3 abandon      negative
##  4 abandon      sadness
##  5 abandoned    anger
##  6 abandoned    fear
##  7 abandoned    negative
##  8 abandoned    sadness
##  9 abandonment  anger
## 10 abandonment  fear
## #   with 13,891 more rows
```

Sentiment lexicons

```
get_sentiments("nrc")
```

```
## # A tibble: 13,901 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 abacus      trust
##  2 abandon     fear
##  3 abandon     negative
##  4 abandon     sadness
##  5 abandoned   anger
##  6 abandoned   fear
##  7 abandoned   negative
##  8 abandoned   sadness
##  9 abandonment anger
## 10 abandonment fear
## #   with 13,891 more rows
```

```
get_sentiments("loughran")
```

```
## # A tibble: 4,150 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 abandon     negative
##  2 abandoned   negative
##  3 abandoning  negative
##  4 abandonment negative
##  5 abandonments negative
##  6 abandons    negative
##  7 abdicated   negative
##  8 abdicates   negative
##  9 abdicating  negative
## 10 abdication  negative
## #   with 4,140 more rows
```

Notes about sentiment lexicons

- Not every word is in a lexicon!

```
get_sentiments("bing") %>%
  filter(word == "data")
```

```
## # A tibble: 0 x 2
## # … with 2 variables: word <chr>, sentiment <chr>
```

Notes about sentiment lexicons

- Not every word is in a lexicon!

```
get_sentiments("bing") %>%
  filter(word == "data")
```

```
## # A tibble: 0 x 2
## # … with 2 variables: word <chr>, sentiment <chr>
```

- Lexicons do not account for qualifiers before a word (e.g., "not happy") because they were constructed for one-word tokens only

Notes about sentiment lexicons

- Not every word is in a lexicon!

```
get_sentiments("bing") %>%
  filter(word == "data")
```

```
## # A tibble: 0 x 2
## # … with 2 variables: word <chr>, sentiment <chr>
```

- Lexicons do not account for qualifiers before a word (e.g., "not happy") because they were constructed for one-word tokens only

- Summing up each word's sentiment may result in a neutral sentiment, even if there are strong positive and negative sentiments in the body

Sentiments in lyrics

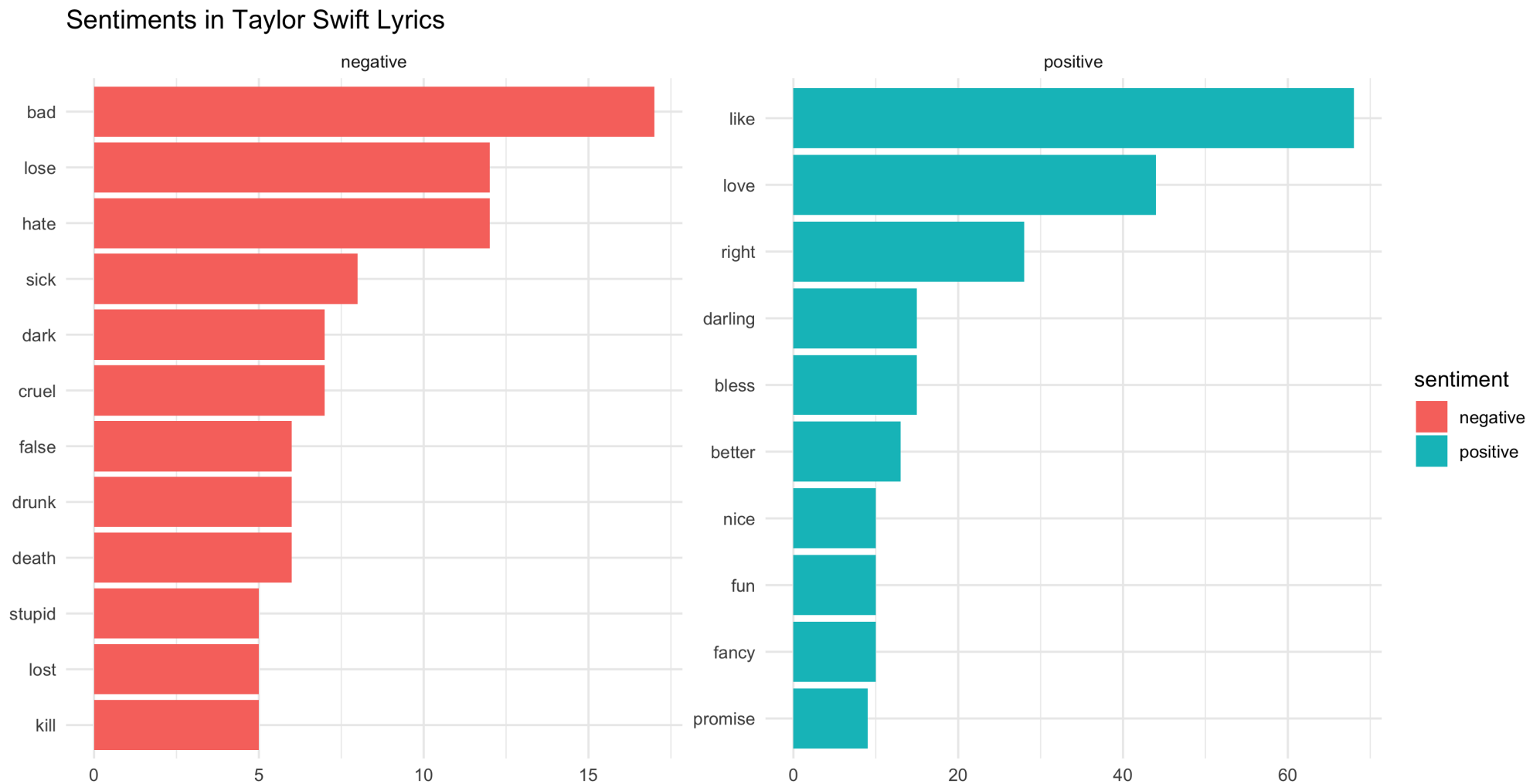
```
tswift_lyrics %>%
  inner_join(get_sentiments("bing")) %>%
  count(sentiment, word, sort = TRUE)
```

```
## # A tibble: 165 x 3
##    sentiment word        n
##    <chr>     <chr>   <int>
##  1 positive  like       68
##  2 positive  love       44
##  3 positive  right      28
##  4 negative  bad        17
##  5 positive  bless      15
##  6 positive  darling    15
##  7 positive  better     13
##  8 negative  hate       12
##  9 negative  lose       12
```

Let's visualize T.Swift's top 10 sentiments

```
tswift_top10 <- tswift_lyrics %>%
  inner_join(get_sentiments("bing")) %>%
  count(sentiment, word) %>%
  arrange(desc(n)) %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup()
```

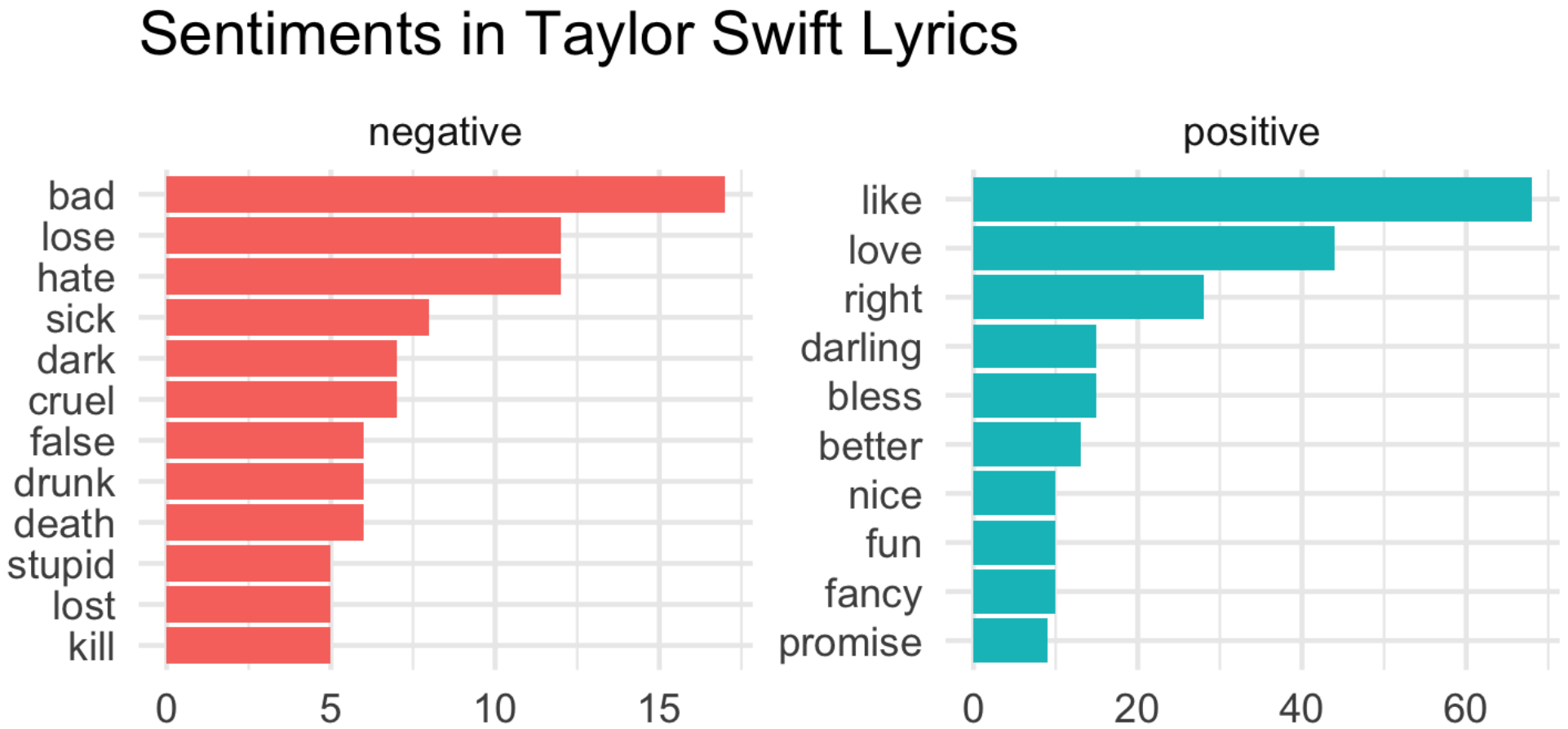
Visualizing the top 10



Let's remove the redundant legend

```
ggplot(tswift_top10, aes(fct_reorder(word, n), n, fill = sentiment)) +
  geom_col() +
  coord_flip() +
  facet_wrap(~ sentiment, scales = "free") +
  theme_minimal() +
  labs(title = "Sentiments in Taylor Swift Lyrics", x = "", y = "") +
  guides(fill = FALSE)
```

Let's remove the redundant legend



Scoring sentiments

```
tswift_lyrics %>%
  anti_join(stop_words) %>%
  left_join(get_sentiments("afinn"))
```

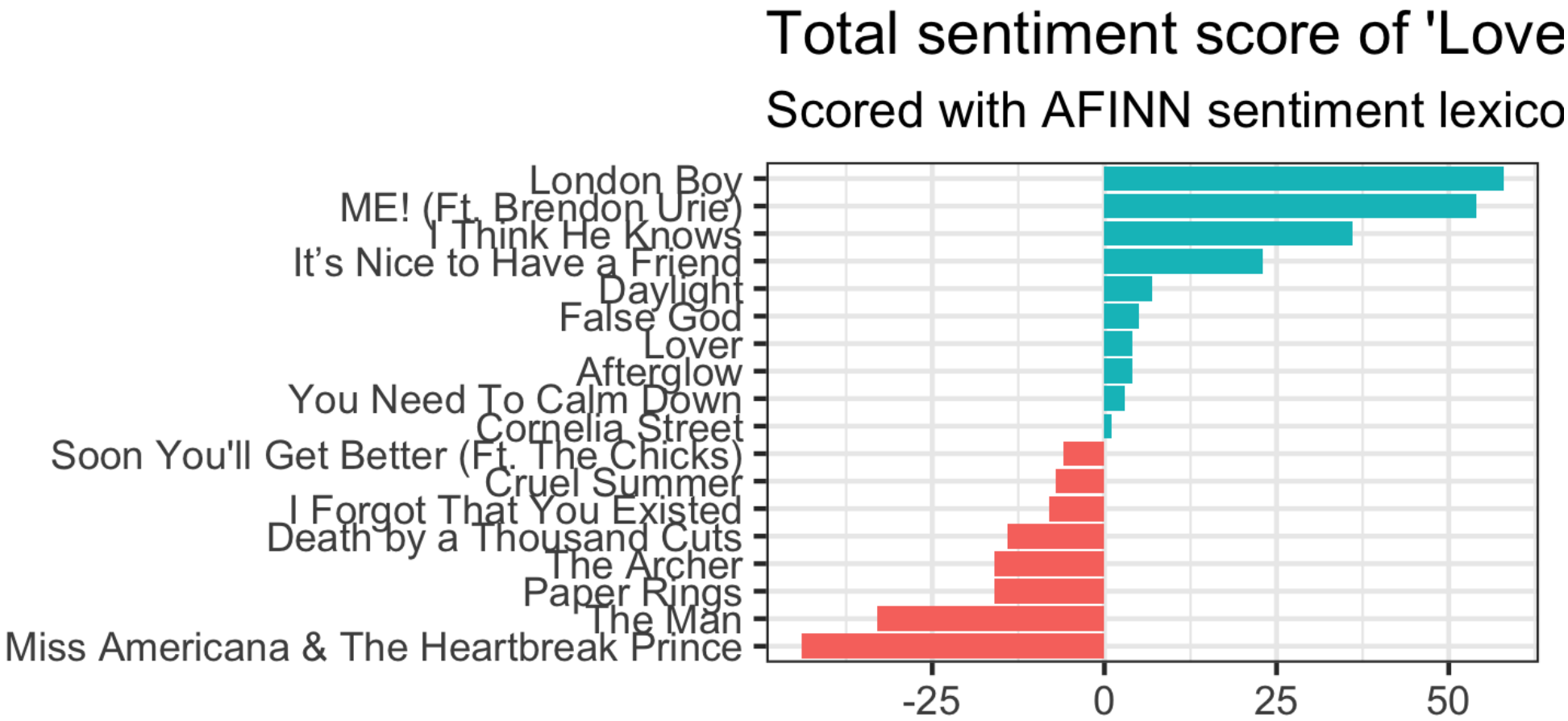
```
## # A tibble: 2,047 x 5
##    track_n  line track_title                word     value
##      <int> <int> <chr>                      <chr>    <dbl>
##  1       1     1 I Forgot That You Existed days        NA
##  2       1     1 I Forgot That You Existed spend       NA
##  3       1     1 I Forgot That You Existed thinking    NA
##  4       1     2 I Forgot That You Existed bout        NA
##  5       1     2 I Forgot That You Existed wrong       -2
##  6       1     2 I Forgot That You Existed wrong       -2
##  7       1     2 I Forgot That You Existed wrong       -2
##  8       1     3 I Forgot That You Existed lived       NA
##  9       1     3 I Forgot That You Existed shade       NA
## 10       1     3 I Forgot That You Existed throwing    NA
## # … with 2,037 more rows
```

Assigning a sentiment score

```
tswift_lyrics %>%
  anti_join(stop_words) %>%
  left_join(get_sentiments("afinn")) %>%
  filter(!is.na(value)) %>%
  group_by(track_title) %>%
  summarise(total_sentiment = sum(value)) %>%
  arrange(total_sentiment)
```

```
## # A tibble: 18 x 2
##    track_title                                  total_sentiment
##    <chr>                                                  <dbl>
##  1 Miss Americana & The Heartbreak Prince                   -44
##  2 The Man                                                  -33
##  3 Paper Rings                                              -16
##  4 The Archer                                               -16
##  5 Death by a Thousand Cuts                                 -14
##  6 I Forgot That You Existed                                 -8
##  7 Cruel Summer                                              -7
##  8 Soon You'll Get Better (Ft. The Chicks)                 -6
##  9 Cornelia Street                                            1
```

Visualizing sentiment scores



...the code

```
tswift_lyrics %>%
  anti_join(stop_words) %>%
  left_join(get_sentiments("afinn")) %>%
  filter(!is.na(value)) %>%
  group_by(track_title) %>%
  summarise(total_sentiment = sum(value)) %>%
  ungroup() %>%
  arrange(total_sentiment) %>%
  mutate(
    total_sentiment_sign = if_else(total_sentiment < 0, "negative", "positive")
  ) %>%
  ggplot(aes(x = reorder(track_title, total_sentiment), y = total_sentiment,
             fill = total_sentiment_sign)) +
  geom_col() +
  guides(fill = FALSE) +
  coord_flip() +
  labs(x = "", y = "",
    title = "Total sentiment score of 'Lover' tracks",
    subtitle = "Scored with AFINN sentiment lexicon")
```

Additional resources

Text Mining with R

- Chapter 1: The tidy text format
- Chapter 2: Sentiment analysis with tidy data