

# 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.



# Tidyttext

- 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
```



# 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.&nbsp;Brendon&nbsp;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 .     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.



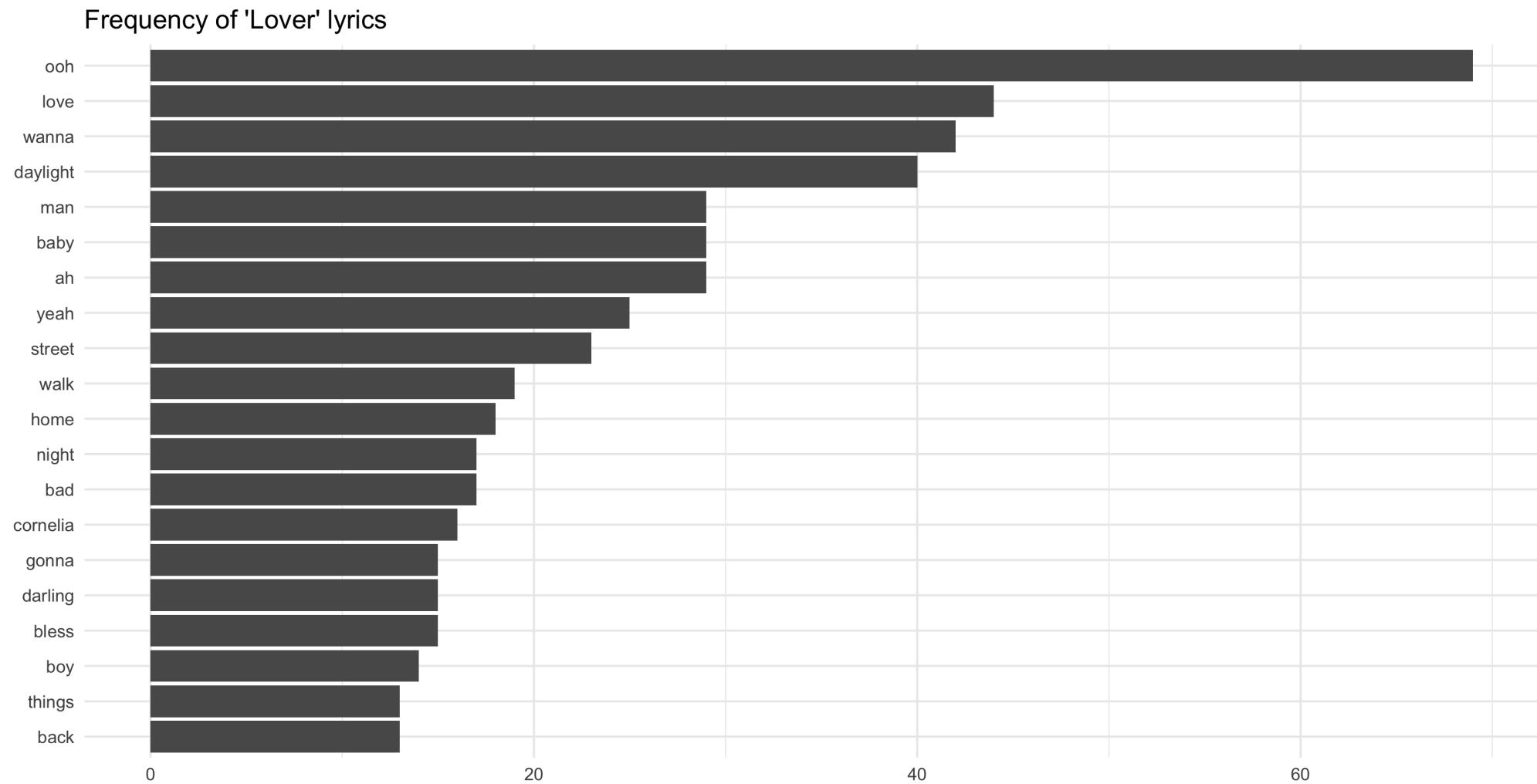
# What are the most common words?

```
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 ...
```



# 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  
## 10 negative  break    12  
## 11 neutral   you     12  
## 12 neutral   baby    12  
## 13 neutral   love    12  
## 14 neutral   heart   12  
## 15 neutral   dream   12  
## 16 neutral   smile   12  
## 17 neutral   kiss    12  
## 18 neutral   tears   12  
## 19 neutral   heartbreak 12  
## 20 neutral   heartache 12  
## 21 neutral   heartbreak 12  
## 22 neutral   heartache 12  
## 23 neutral   heartbreak 12  
## 24 neutral   heartache 12  
## 25 neutral   heartbreak 12  
## 26 neutral   heartache 12  
## 27 neutral   heartbreak 12  
## 28 neutral   heartache 12  
## 29 neutral   heartbreak 12  
## 30 neutral   heartache 12  
## 31 neutral   heartbreak 12  
## 32 neutral   heartache 12  
## 33 neutral   heartbreak 12  
## 34 neutral   heartache 12  
## 35 neutral   heartbreak 12  
## 36 neutral   heartache 12  
## 37 neutral   heartbreak 12  
## 38 neutral   heartache 12  
## 39 neutral   heartbreak 12  
## 40 neutral   heartache 12  
## 41 neutral   heartbreak 12  
## 42 neutral   heartache 12  
## 43 neutral   heartbreak 12  
## 44 neutral   heartache 12  
## 45 neutral   heartbreak 12  
## 46 neutral   heartache 12  
## 47 neutral   heartbreak 12  
## 48 neutral   heartache 12  
## 49 neutral   heartbreak 12  
## 50 neutral   heartache 12  
## 51 neutral   heartbreak 12  
## 52 neutral   heartache 12  
## 53 neutral   heartbreak 12  
## 54 neutral   heartache 12  
## 55 neutral   heartbreak 12  
## 56 neutral   heartache 12  
## 57 neutral   heartbreak 12  
## 58 neutral   heartache 12  
## 59 neutral   heartbreak 12  
## 60 neutral   heartache 12  
## 61 neutral   heartbreak 12  
## 62 neutral   heartache 12  
## 63 neutral   heartbreak 12  
## 64 neutral   heartache 12  
## 65 neutral   heartbreak 12  
## 66 neutral   heartache 12  
## 67 neutral   heartbreak 12  
## 68 neutral   heartache 12  
## 69 neutral   heartbreak 12  
## 70 neutral   heartache 12  
## 71 neutral   heartbreak 12  
## 72 neutral   heartache 12  
## 73 neutral   heartbreak 12  
## 74 neutral   heartache 12  
## 75 neutral   heartbreak 12  
## 76 neutral   heartache 12  
## 77 neutral   heartbreak 12  
## 78 neutral   heartache 12  
## 79 neutral   heartbreak 12  
## 80 neutral   heartache 12  
## 81 neutral   heartbreak 12  
## 82 neutral   heartache 12  
## 83 neutral   heartbreak 12  
## 84 neutral   heartache 12  
## 85 neutral   heartbreak 12  
## 86 neutral   heartache 12  
## 87 neutral   heartbreak 12  
## 88 neutral   heartache 12  
## 89 neutral   heartbreak 12  
## 90 neutral   heartache 12  
## 91 neutral   heartbreak 12  
## 92 neutral   heartache 12  
## 93 neutral   heartbreak 12  
## 94 neutral   heartache 12  
## 95 neutral   heartbreak 12  
## 96 neutral   heartache 12  
## 97 neutral   heartbreak 12  
## 98 neutral   heartache 12  
## 99 neutral   heartbreak 12  
## 100 neutral  heartache 12  
## 101 neutral  heartbreak 12  
## 102 neutral  heartache 12  
## 103 neutral  heartbreak 12  
## 104 neutral  heartache 12  
## 105 neutral  heartbreak 12  
## 106 neutral  heartache 12  
## 107 neutral  heartbreak 12  
## 108 neutral  heartache 12  
## 109 neutral  heartbreak 12  
## 110 neutral  heartache 12  
## 111 neutral  heartbreak 12  
## 112 neutral  heartache 12  
## 113 neutral  heartbreak 12  
## 114 neutral  heartache 12  
## 115 neutral  heartbreak 12  
## 116 neutral  heartache 12  
## 117 neutral  heartbreak 12  
## 118 neutral  heartache 12  
## 119 neutral  heartbreak 12  
## 120 neutral  heartache 12  
## 121 neutral  heartbreak 12  
## 122 neutral  heartache 12  
## 123 neutral  heartbreak 12  
## 124 neutral  heartache 12  
## 125 neutral  heartbreak 12  
## 126 neutral  heartache 12  
## 127 neutral  heartbreak 12  
## 128 neutral  heartache 12  
## 129 neutral  heartbreak 12  
## 130 neutral  heartache 12  
## 131 neutral  heartbreak 12  
## 132 neutral  heartache 12  
## 133 neutral  heartbreak 12  
## 134 neutral  heartache 12  
## 135 neutral  heartbreak 12  
## 136 neutral  heartache 12  
## 137 neutral  heartbreak 12  
## 138 neutral  heartache 12  
## 139 neutral  heartbreak 12  
## 140 neutral  heartache 12  
## 141 neutral  heartbreak 12  
## 142 neutral  heartache 12  
## 143 neutral  heartbreak 12  
## 144 neutral  heartache 12  
## 145 neutral  heartbreak 12  
## 146 neutral  heartache 12  
## 147 neutral  heartbreak 12  
## 148 neutral  heartache 12  
## 149 neutral  heartbreak 12  
## 150 neutral  heartache 12  
## 151 neutral  heartbreak 12  
## 152 neutral  heartache 12  
## 153 neutral  heartbreak 12  
## 154 neutral  heartache 12  
## 155 neutral  heartbreak 12  
## 156 neutral  heartache 12  
## 157 neutral  heartbreak 12  
## 158 neutral  heartache 12  
## 159 neutral  heartbreak 12  
## 160 neutral  heartache 12  
## 161 neutral  heartbreak 12  
## 162 neutral  heartache 12  
## 163 neutral  heartbreak 12  
## 164 neutral  heartache 12  
## 165 neutral  heartbreak 12
```

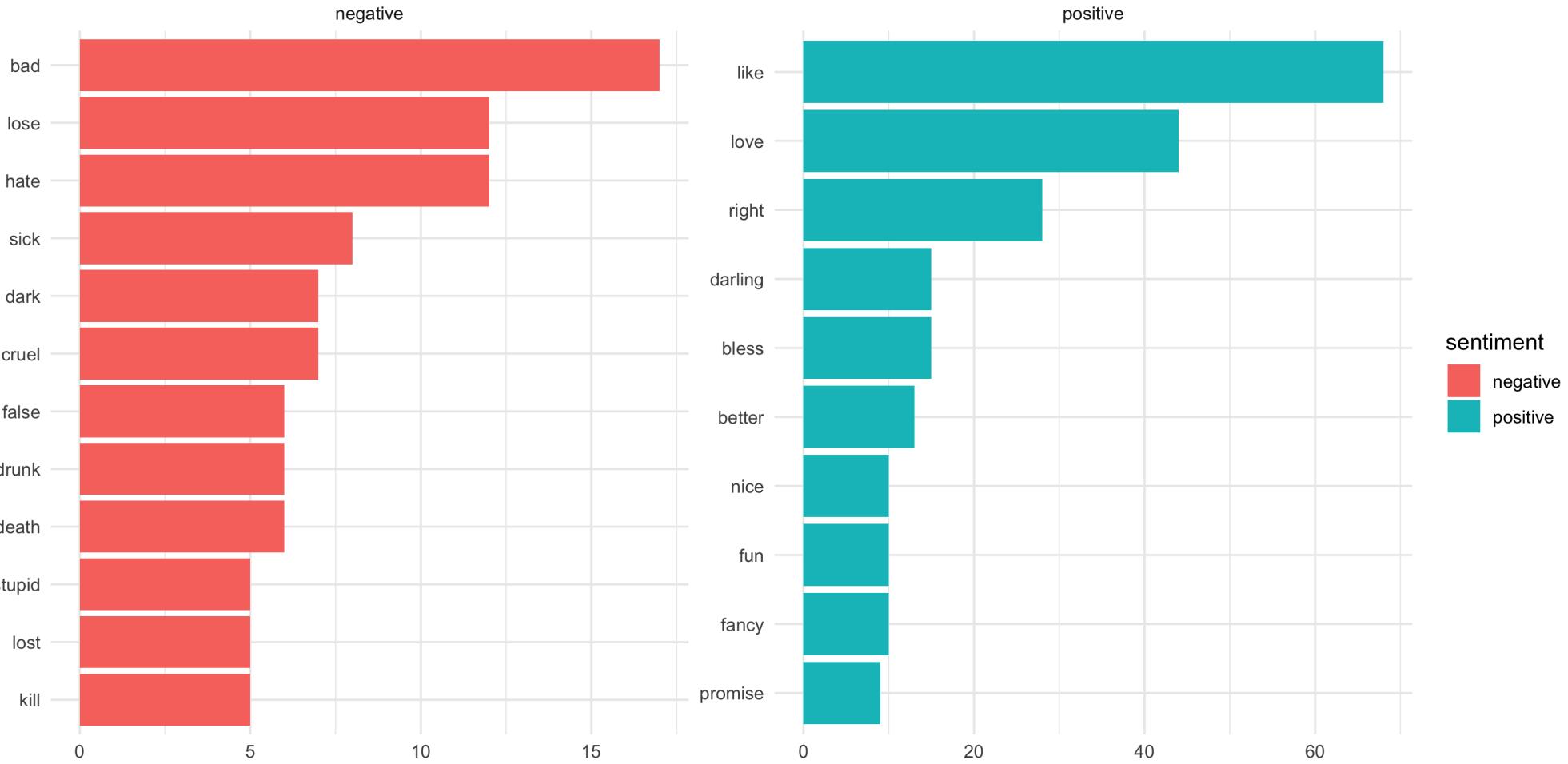
# Let's visualize T.Swift's top 10 sentiments

```
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

Sentiments in Taylor Swift Lyrics



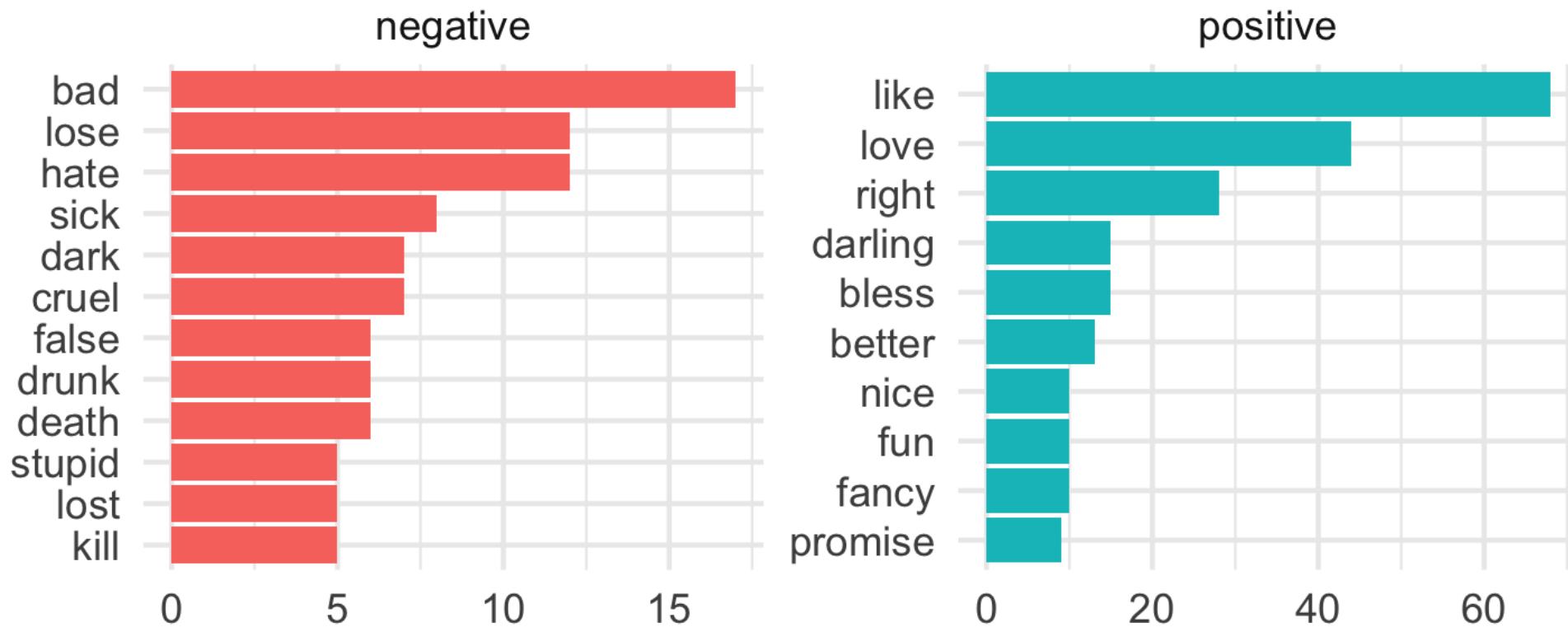
# Let's remove the redundant legend

```
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

## Sentiments in Taylor Swift Lyrics



# 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
```



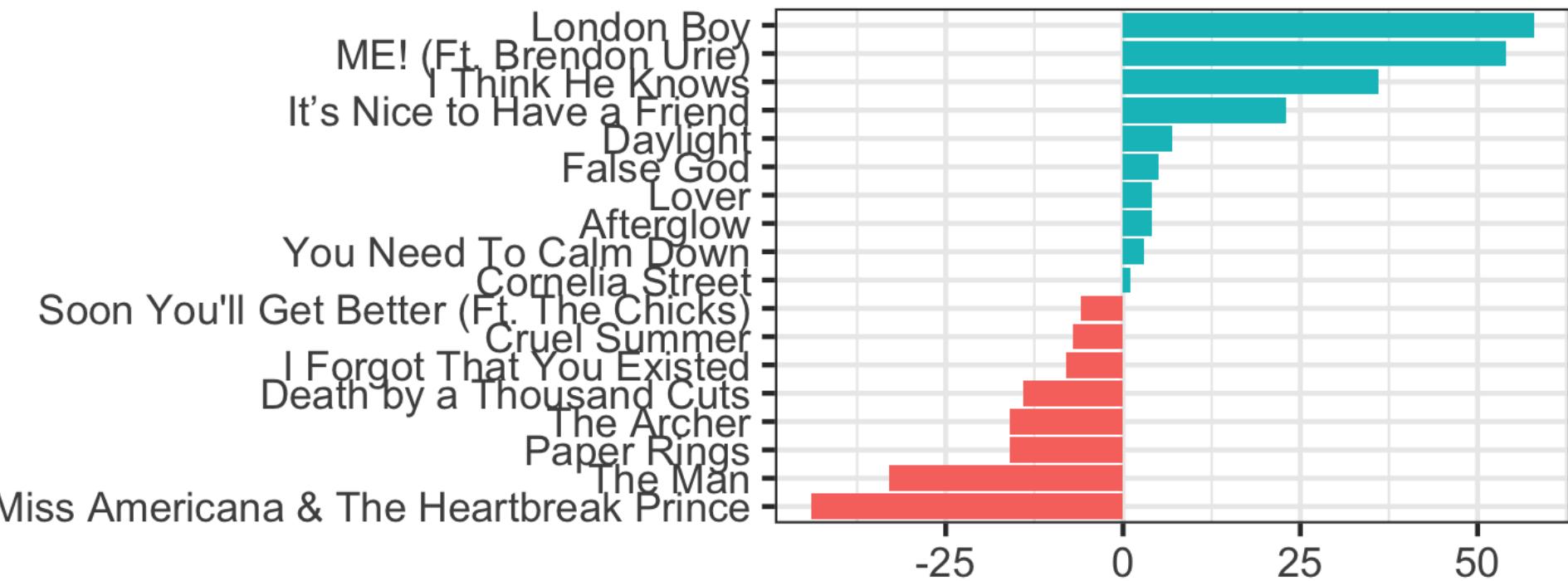
# Assigning a sentiment score

```
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.&nbsp;The&nbsp;Chicks) -6  
## 9 Cornelia Street
```

# Visualizing sentiment scores

Total sentiment score of 'Love  
Scored with AFINN sentiment lexico



# ...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

