TEXT MINING WITH

tidy data principles
+
count-based methods

Julia Silge
@juliasilge
https://juliasilge.com/

TEXT DATA IS INCREASINGLY IMPORTANT

🤔

NLP TRAINING IS SCARCE ON THE GROUND

😩

TIDY DATA PRINCIPLES

+

COUNT-BASED METHODS

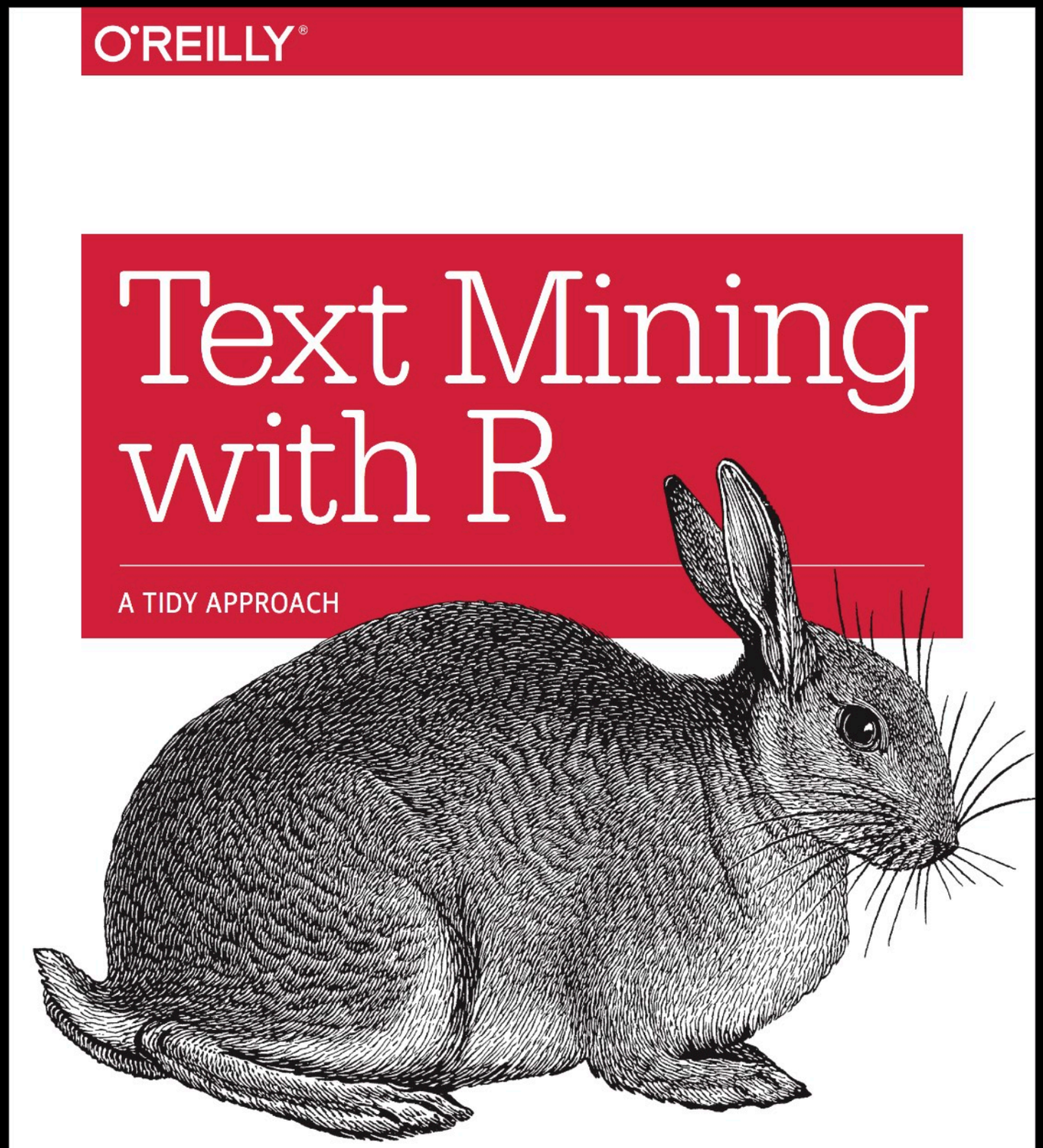
=

tidytext: Text mining using dplyr, ggplot2, and other tidy tools

Authors: Julia Silge, David Robinson

Using tidy data principles can make many text mining tasks easier, more effective, and consistent with tools already in wide use. Much of the infrastructure needed for text mining with tidy data frames already exists in packages like dplyr, broom, tidyr and ggplot2. In this package, we provide functions and supporting data sets to allow conversion of text to and from tidy formats, and to switch seamlessly between tidy tools and existing text mining packages.

https://github.com/juliasilge/tidytext



http://tidytextmining.com/

What do we mean by tidy text?

What do we mean by tidy text?

```
> text <- c("Because I could not stop for Death -",
            "He kindly stopped for me -",
            "The Carriage held but just Ourselves -",
            "and Immortality")
>
> text

## [1] "Because I could not stop for Death -"   "He kindly stopped for me -"
## [3] "The Carriage held but just Ourselves -" "and Immortality"
```

What do we mean by tidy text?

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

## # A tibble: 20 × 2
##     line    word
##    <int>   <chr>
## 1      1 because
## 2      1       i
## 3      1   could
## 4      1     not
## 5      1    stop
## 6      1     for
## 7      1   death
## 8      2      he
## 9      2  kindly
## 10     2 stopped
## # ... with 10 more rows
```

- Other columns have been retained
- Punctuation has been stripped
- Words have been converted to lowercase

Tidying the works of Jane Austen

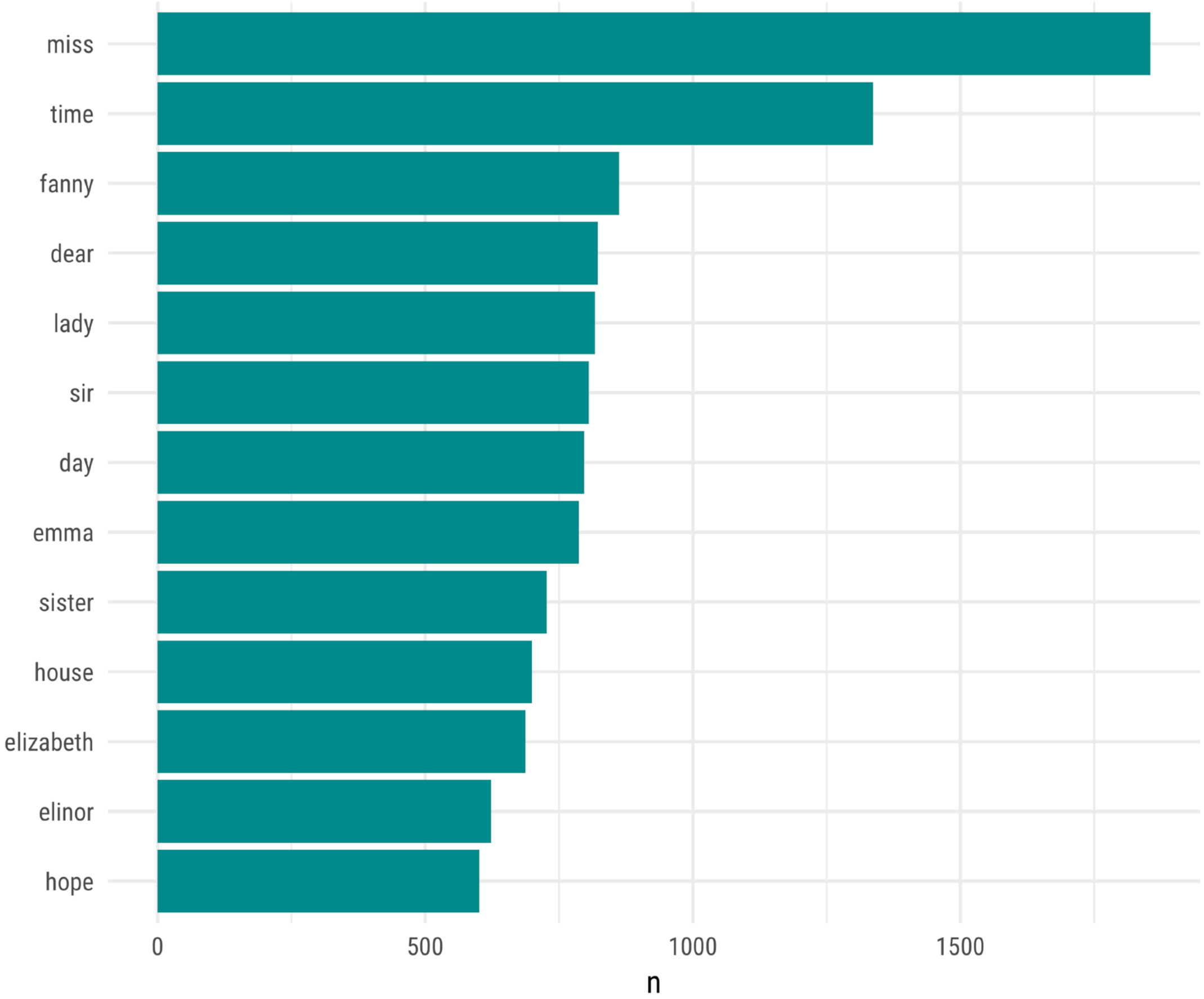
```
> tidy_books <- original_books %>%
    unnest_tokens(word, text)
>
> tidy_books
# A tibble: 725,054 × 4
                   book linenumber chapter        word
                 <fctr>      <int>   <int>       <chr>
1  Sense & Sensibility          1       0       sense
2  Sense & Sensibility          1       0         and
3  Sense & Sensibility          1       0 sensibility
4  Sense & Sensibility          3       0          by
5  Sense & Sensibility          3       0        jane
6  Sense & Sensibility          3       0      austen
7  Sense & Sensibility          5       0        1811
8  Sense & Sensibility         10       1     chapter
9  Sense & Sensibility         10       1           1
10 Sense & Sensibility         13       1         the
# ... with 725,044 more rows
```

OUR TEXT IS TIDY NOW

WHAT NEXT?

REMOVING STOP WORDS

```
> data(stop_words)
>
> tidy_books <- tidy_books %>%
      anti_join(stop_words)
>
> tidy_books %>%
      count(word, sort = TRUE)
```


miss
time
fanny
dear
lady
sir
day
emma
sister
house
elizabeth
elinor
hope
0
500
1000
1500
n

Sentiment analysis

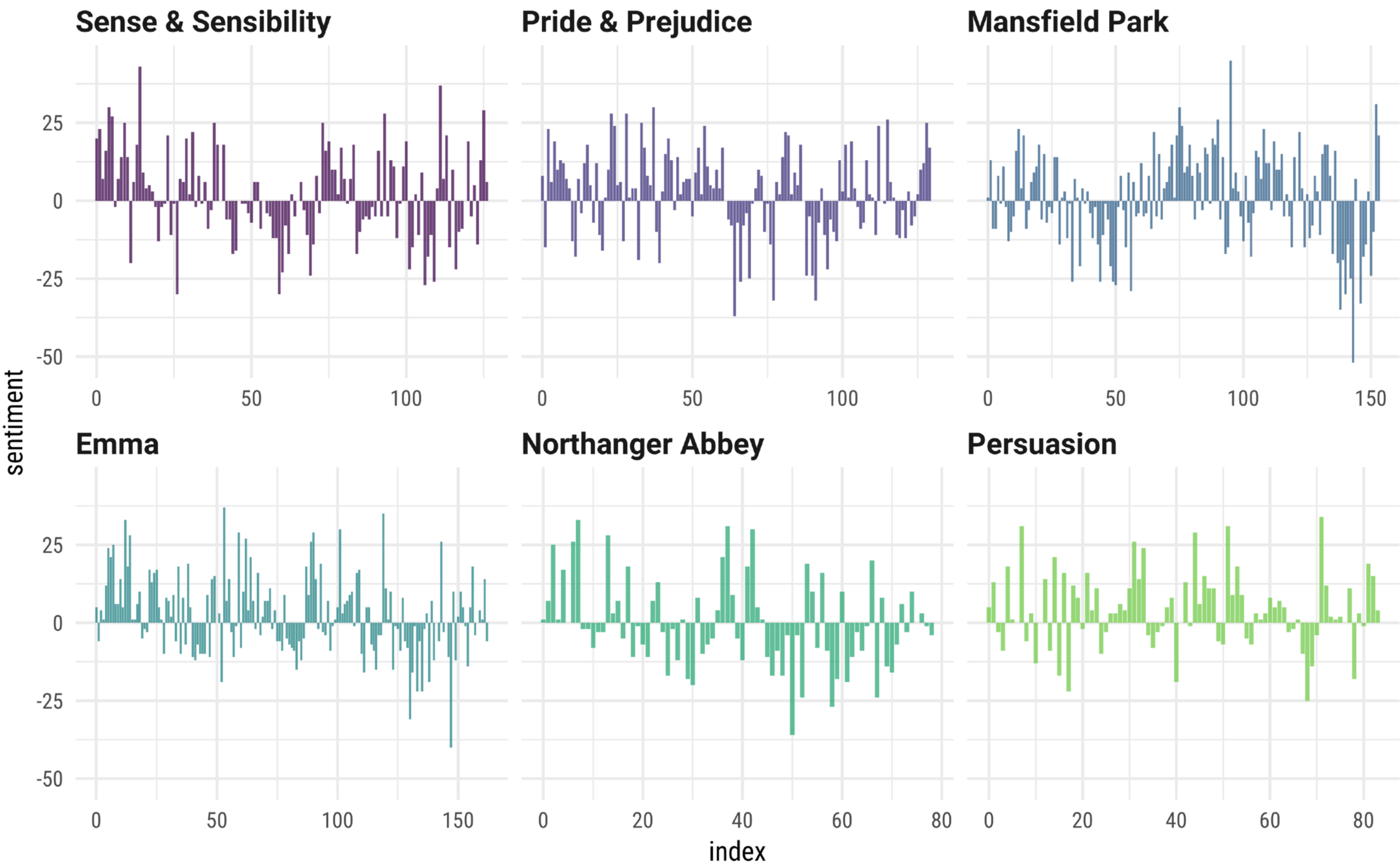
```
> get_sentiments("afinn")
# A tibble: 2,476 × 2
         word score
        <chr> <int>
1     abandon    -2
2   abandoned    -2
3    abandons    -2
4    abducted    -2
5   abduction    -2
6  abductions    -2
7       abhor    -3
8    abhorred    -3
9   abhorrent    -3
10     abhors    -3
# ... with 2,466 more rows
```

```
> get_sentiments("bing")
# A tibble: 6,788 × 2
          word sentiment
         <chr>     <chr>
1     2-faced  negative
2     2-faces  negative
3          a+  positive
4    abnormal  negative
5     abolish  negative
6   abominable  negative
7   abominably  negative
8    abominate  negative
9  abomination  negative
10       abort  negative
# ... with 6,778 more rows
```

```
> get_sentiments("nrc")
# A tibble: 13,901 × 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 analysis

```
> library(tidyr)
>
> janeaustensentiment <- tidy_books %>%
inner_join(get_sentiments("bing")) %>%
count(book, index = linenumber %/% 100, sentiment) %>%
spread(sentiment, n, fill = 0) %>%
mutate(sentiment = positive - negative)
```


Sense & Sensibility
Pride & Prejudice
Mansfield Park
Emma
Northanger Abbey
Persuasion
sentiment
index
25
0
-25
-50
0
50
100
150
20
40
60
80

Sentiment analysis

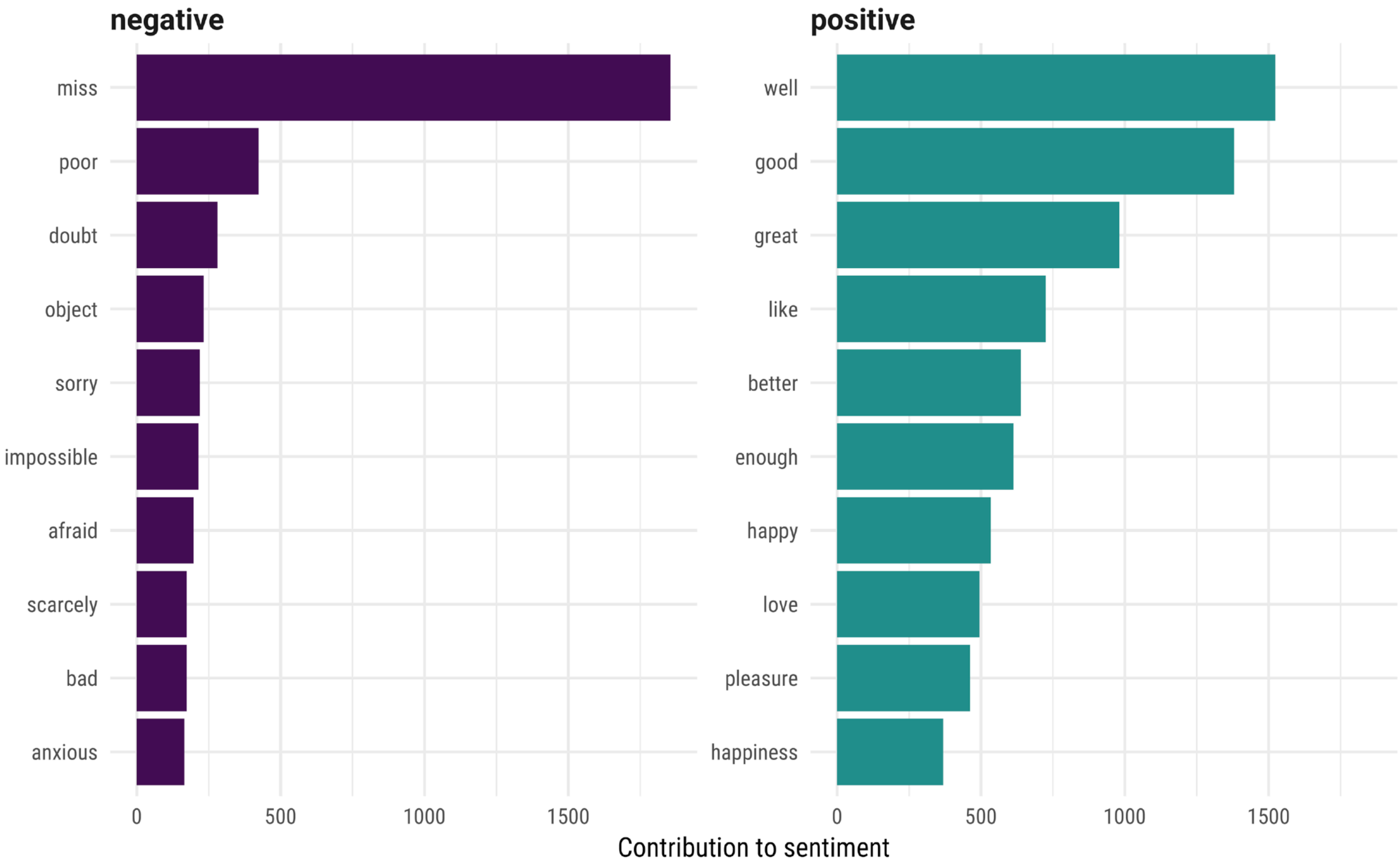
```
> bing_word_counts <- austen_books() %>%
    unnest_tokens(word, text) %>%
    inner_join(get_sentiments("bing")) %>%
    count(word, sentiment, sort = TRUE) %>%
    ungroup()
```

Which words contribute to each sentiment?

Sentiment analysis

```
> bing_word_counts
# A tibble: 2,585 × 3
       word sentiment     n
      <chr>     <chr> <int>
1      miss  negative  1855
2      well  positive  1523
3      good  positive  1380
4     great  positive   981
5      like  positive   725
6    better  positive   639
7    enough  positive   613
8     happy  positive   534
9      love  positive   495
10 pleasure  positive   462
# ... with 2,575 more rows
```

Which words contribute to each sentiment?


negative
miss
poor
doubt
object
sorry
impossible
afraid
scarcely
bad
anxious
0
500
1000
1500
positive
well
good
great
like
better
enough
happy
love
pleasure
happiness
0
500
1000
1500
Contribution to sentiment

What is a document about?

TERM FREQUENCY

INVERSE DOCUMENT FREQUENCY

$$idf(\text{term}) = \ln\left(\frac{n_{\text{documents}}}{n_{\text{documents containing term}}}\right)$$

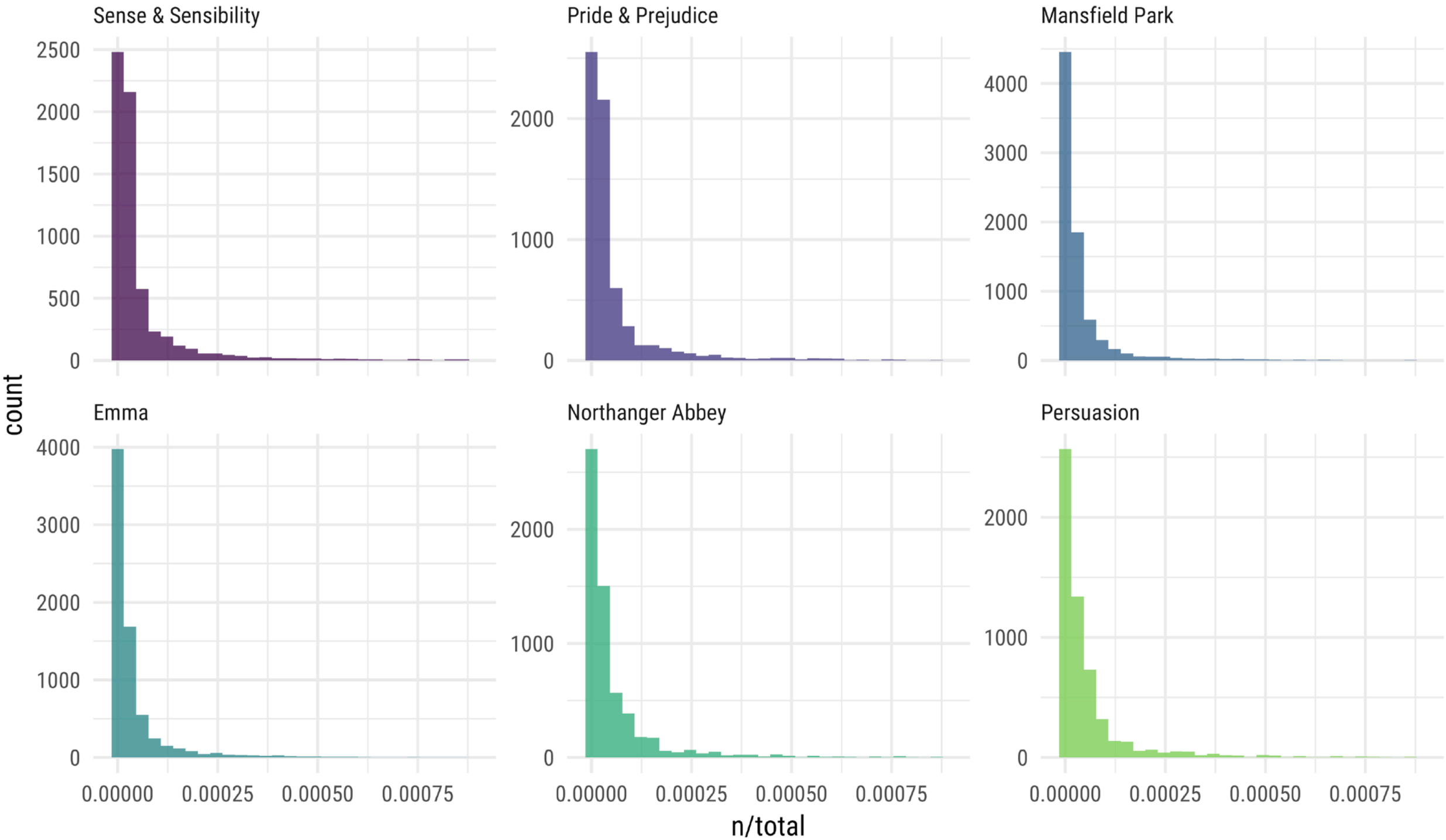
TF-IDF

```
> book_words <- austen_books() %>%
      unnest_tokens(word, text) %>%
      count(book, word, sort = TRUE) %>%
      ungroup()
>
> total_words <- book_words %>%
      group_by(book) %>%
      summarize(total = sum(n))
>
> book_words <- left_join(book_words, total_words)
```

TF-IDF

```
> book_words
# A tibble: 40,379 × 4
                 book  word     n  total
               <fctr> <chr> <int>  <int>
1     Mansfield Park   the  6206 160460
2     Mansfield Park    to  5475 160460
3     Mansfield Park   and  5438 160460
4               Emma    to  5239 160996
5               Emma   the  5201 160996
6               Emma   and  4896 160996
7     Mansfield Park    of  4778 160460
8  Pride & Prejudice   the  4331 122204
9               Emma    of  4291 160996
10 Pride & Prejudice    to  4162 122204
# ... with 40,369 more rows
```

Term Frequency Distribution in Jane Austen's Novels



TF-IDF

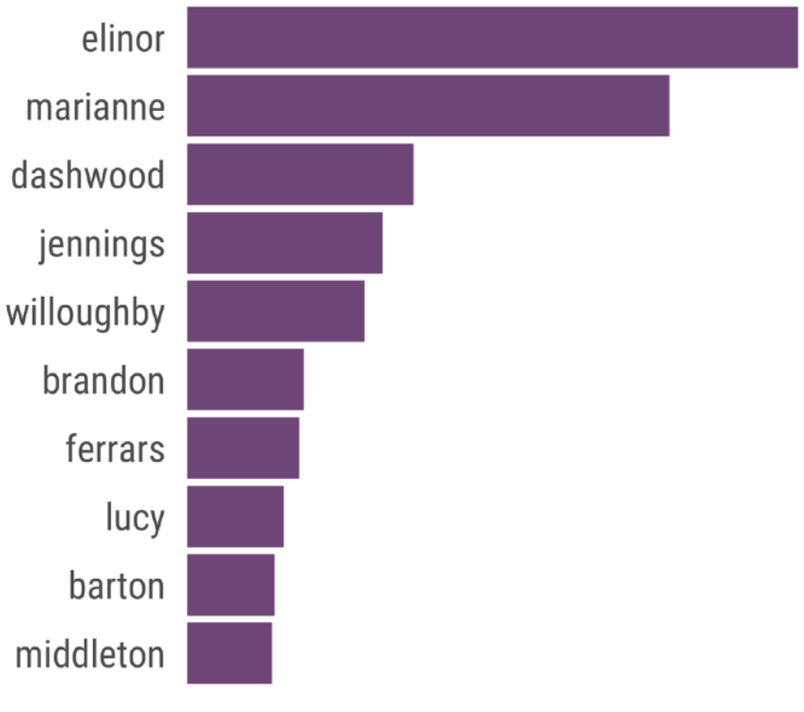
```
> book_words <- book_words %>%
     bind_tf_idf(word, book, n)
> book_words
# A tibble: 40,379 × 7
                book  word     n  total         tf   idf tf_idf
              <fctr> <chr> <int>  <int>      <dbl> <dbl>  <dbl>
1     Mansfield Park   the  6206 160460 0.03867631     0      0
2     Mansfield Park    to  5475 160460 0.03412065     0      0
3     Mansfield Park   and  5438 160460 0.03389007     0      0
4               Emma    to  5239 160996 0.03254118     0      0
5               Emma   the  5201 160996 0.03230515     0      0
6               Emma   and  4896 160996 0.03041069     0      0
7     Mansfield Park    of  4778 160460 0.02977689     0      0
8  Pride & Prejudice   the  4331 122204 0.03544074     0      0
9               Emma    of  4291 160996 0.02665284     0      0
10 Pride & Prejudice    to  4162 122204 0.03405780     0      0
# ... with 40,369 more rows
```

TF-IDF

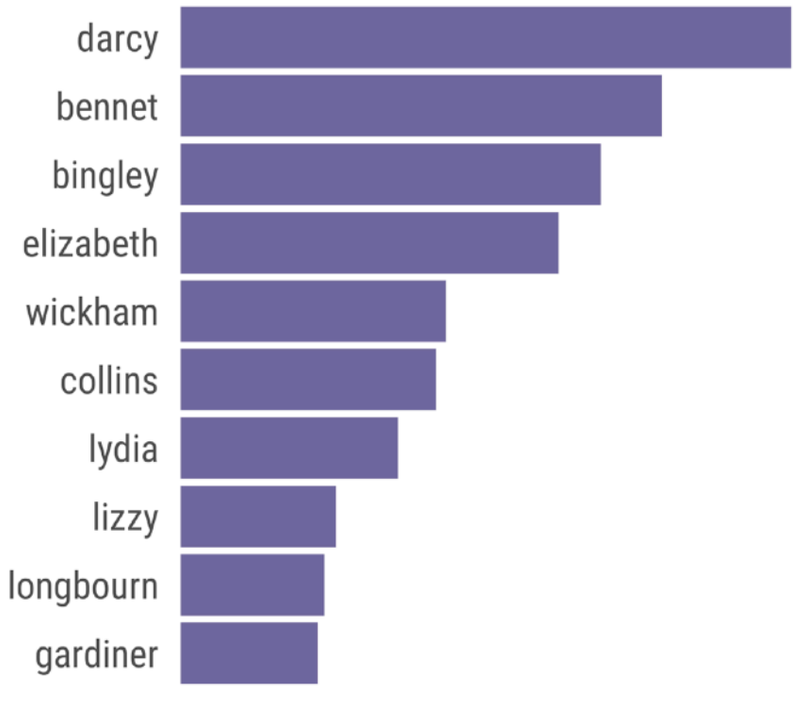
```
> book_words %>%
+     select(-total) %>%
+     arrange(desc(tf_idf))
# A tibble: 40,379 × 6
                 book      word     n          tf      idf      tf_idf
               <fctr>     <chr> <int>       <dbl>    <dbl>       <dbl>
1  Sense & Sensibility    elinor   623 0.005193528 1.791759 0.009305552
2  Sense & Sensibility  marianne   492 0.004101470 1.791759 0.007348847
3       Mansfield Park  crawford   493 0.003072417 1.791759 0.005505032
4    Pride & Prejudice     darcy   373 0.003052273 1.791759 0.005468939
5            Persuasion    elliot   254 0.003036207 1.791759 0.005440153
6                  Emma      emma   786 0.004882109 1.098612 0.005363545
7     Northanger Abbey    tilney   196 0.002519928 1.791759 0.004515105
8                  Emma    weston   389 0.002416209 1.791759 0.004329266
9    Pride & Prejudice    bennet   294 0.002405813 1.791759 0.004310639
10           Persuasion wentworth   191 0.002283132 1.791759 0.004090824
# ... with 40,369 more rows
```

Highest tf-idf words in Jane Austen's Novels

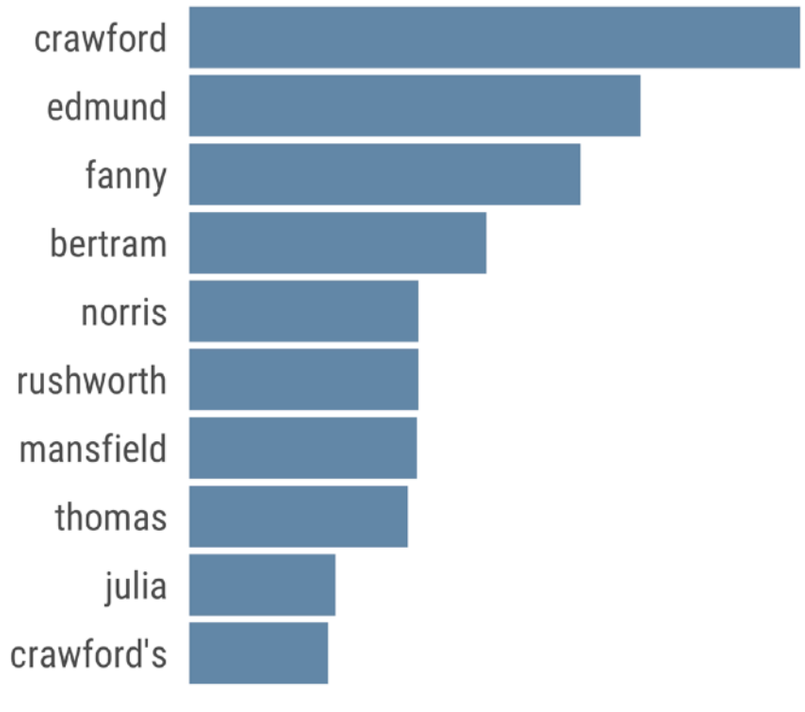
Sense & Sensibility



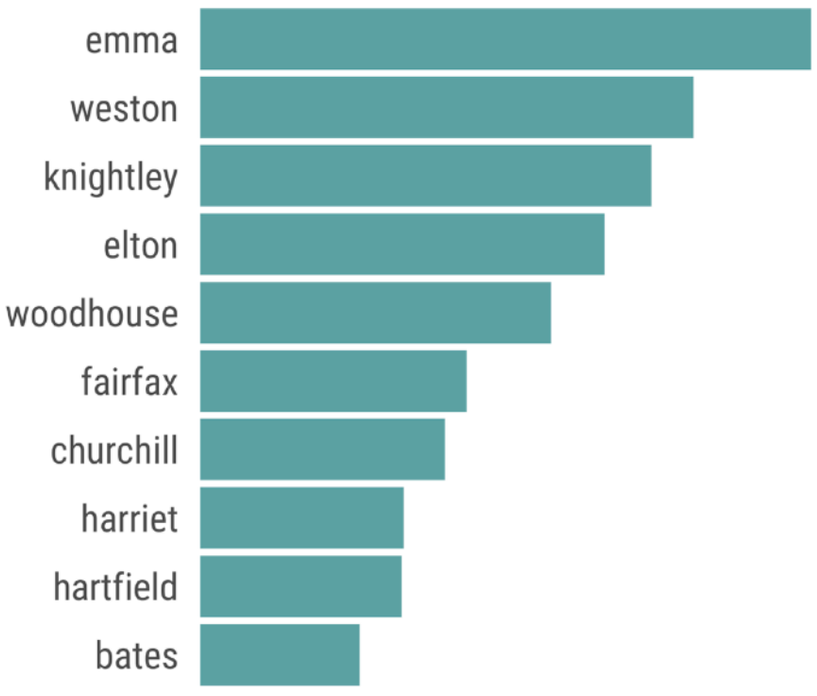
Pride & Prejudice



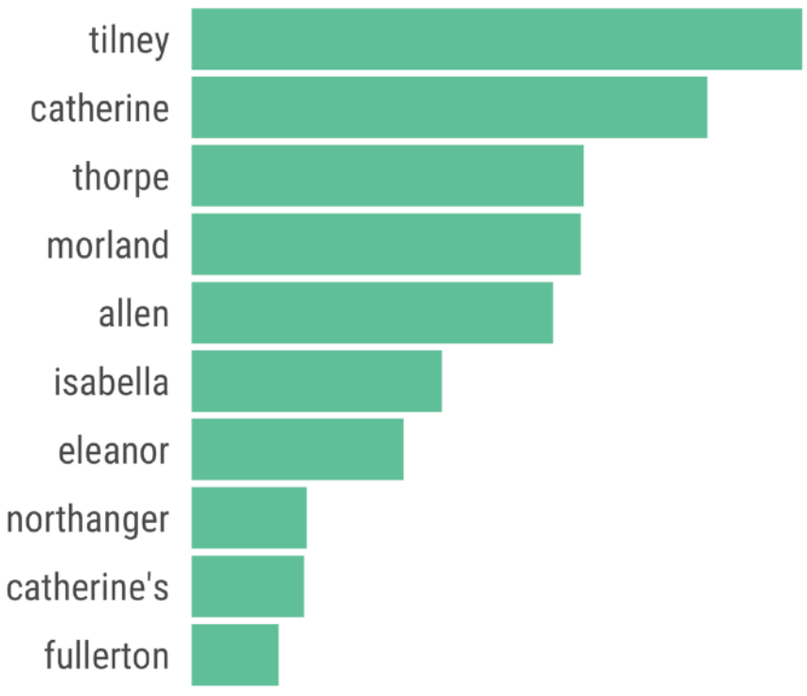
Mansfield Park



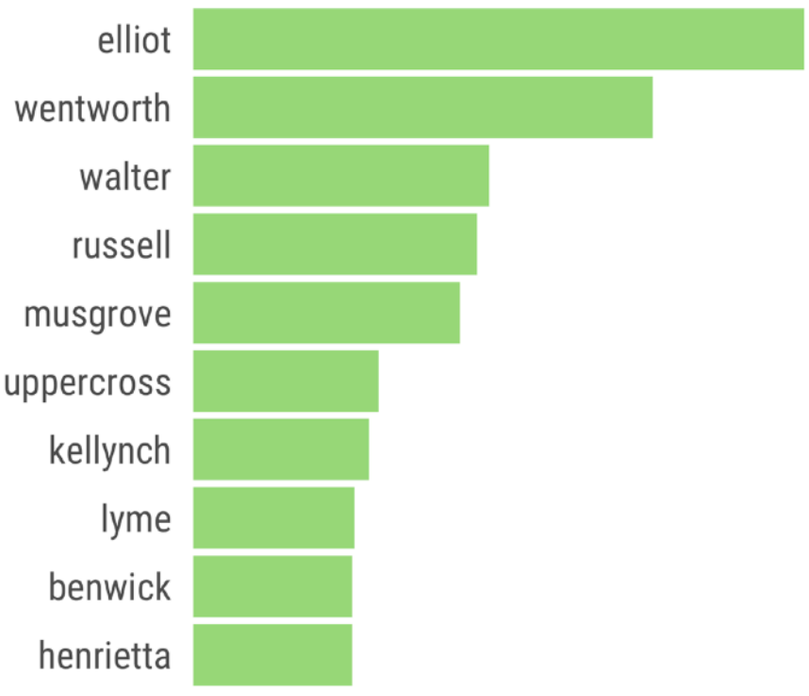
Emma



Northanger Abbey



Persuasion

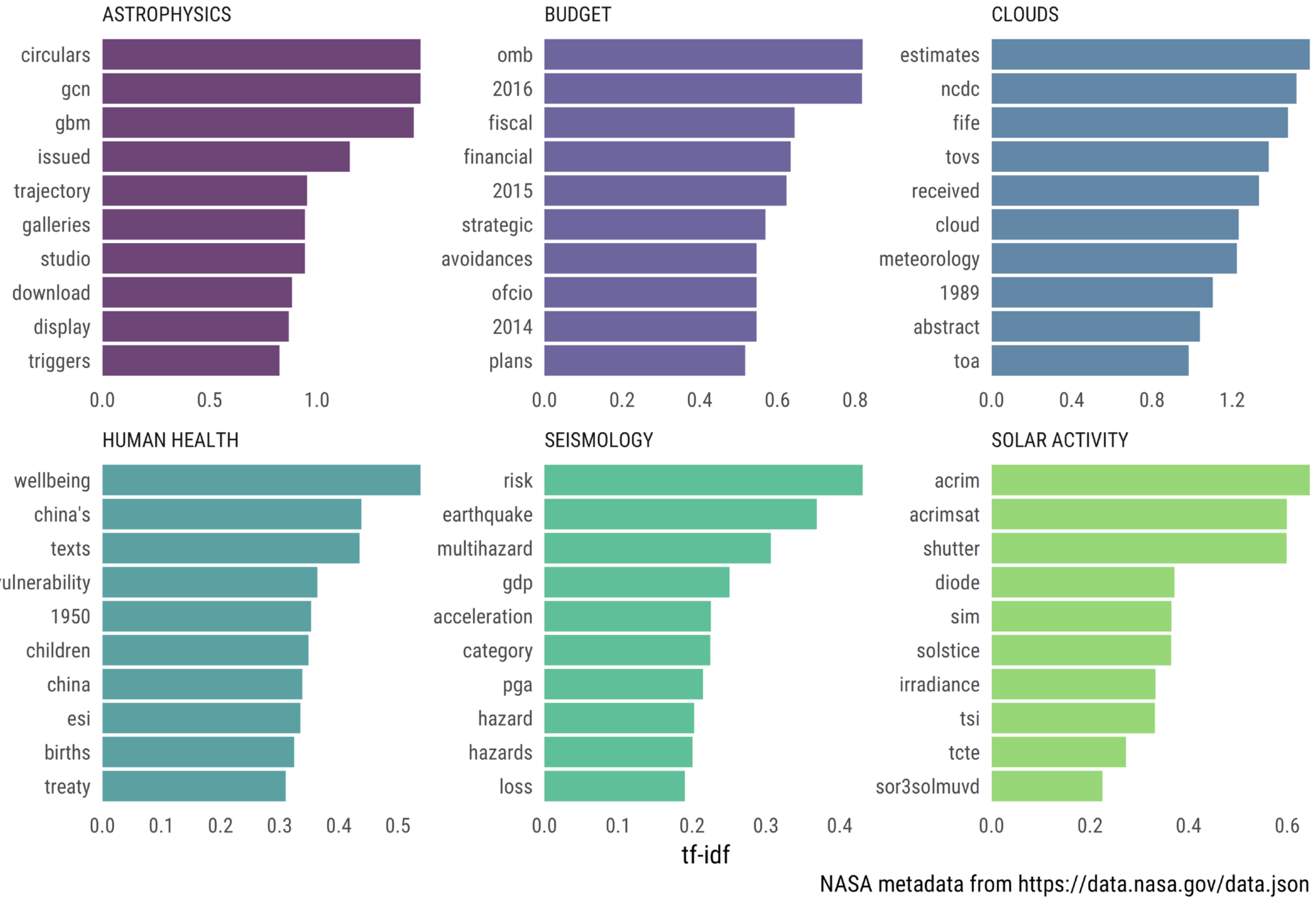


tf-idf

- As part of the NASA Datanauts program, I am working on a project to understand NASA datasets
- Metadata includes title, description, keywords, etc

Highest tf-idf words in NASA Metadata Description Fields

Distribution of tf-idf for words from datasets labeled with select keywords



NASA metadata from https://data.nasa.gov/data.json

TAKING TIDY TEXT TO THE NEXT LEVEL

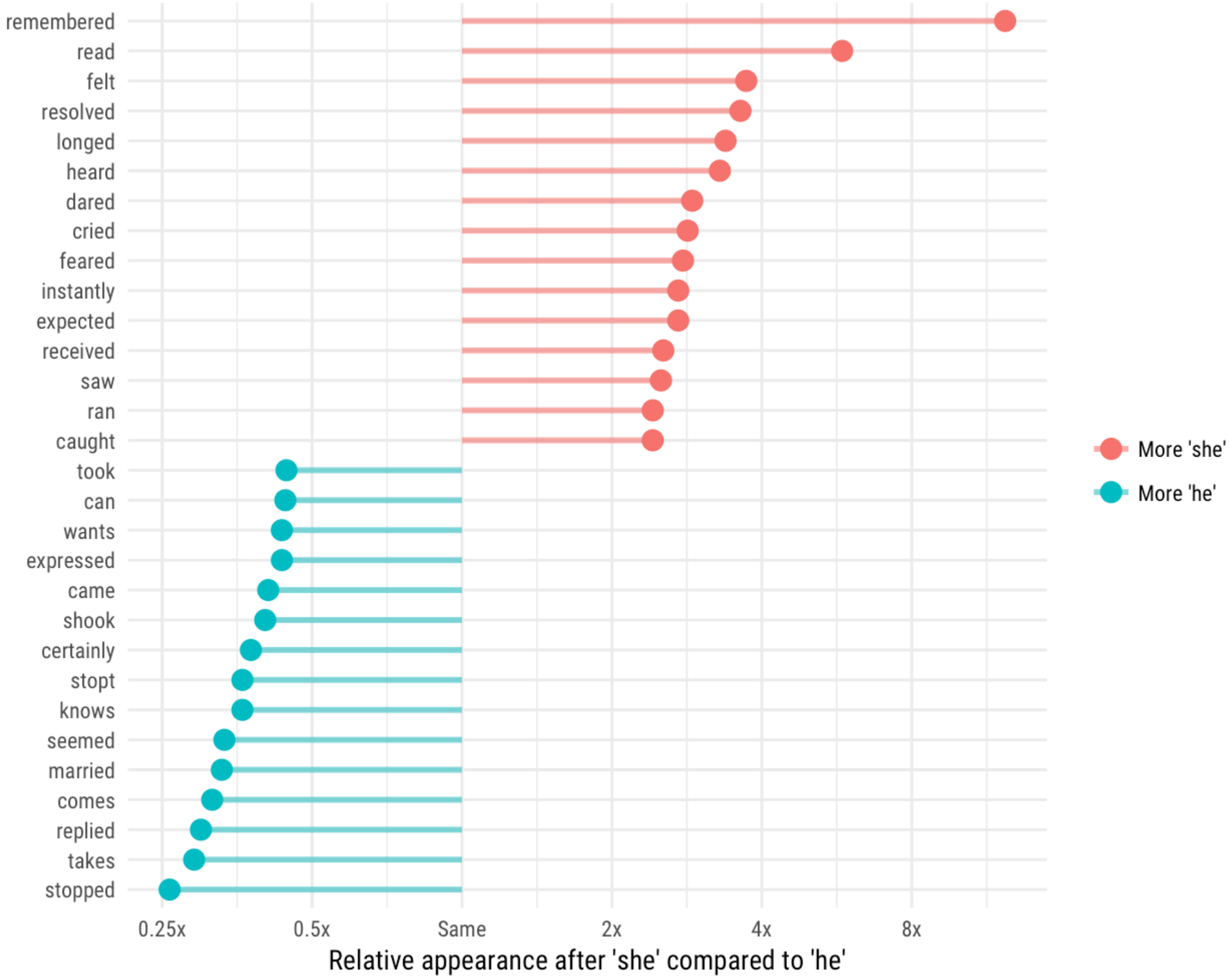
n-grams,

networks, &

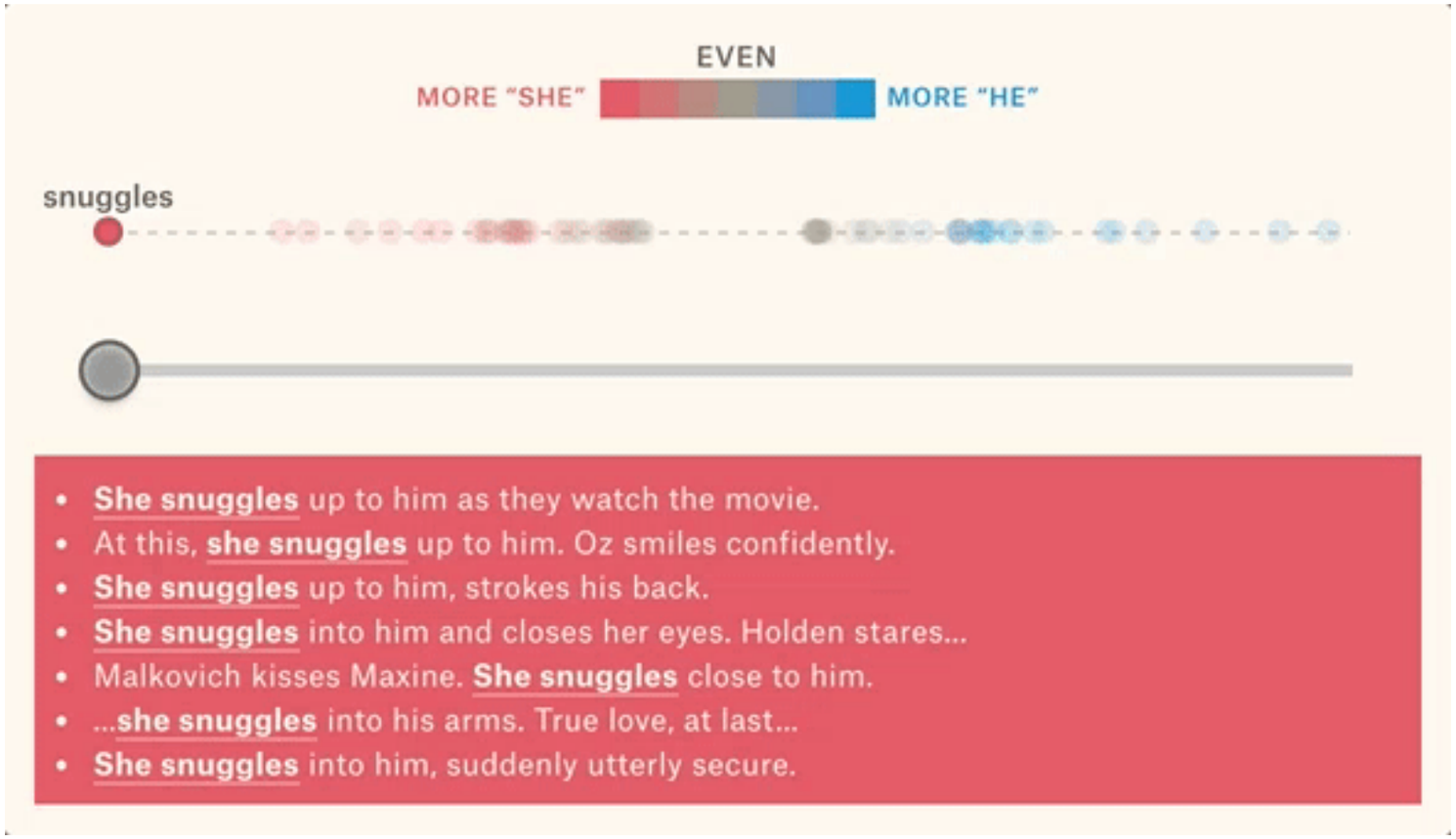
negation

Words paired with 'he' and 'she' in Jane Austen's novels

Women remember, read, and feel while men stop, take, and reply



SHE GIGGLES, HE GALLOPS



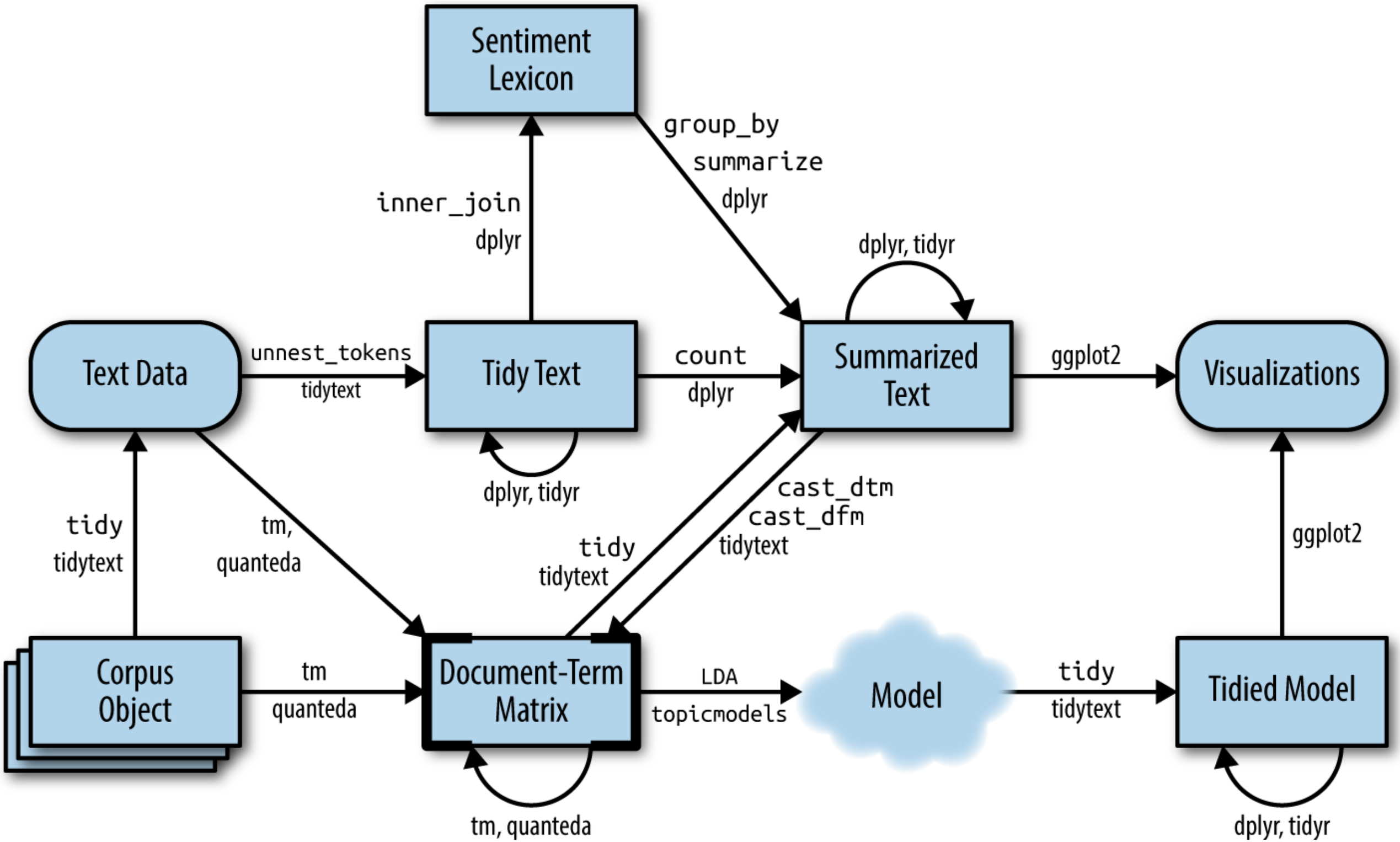
from The Pudding

TAKING TIDY TEXT TO THE NEXT LEVEL

tidying

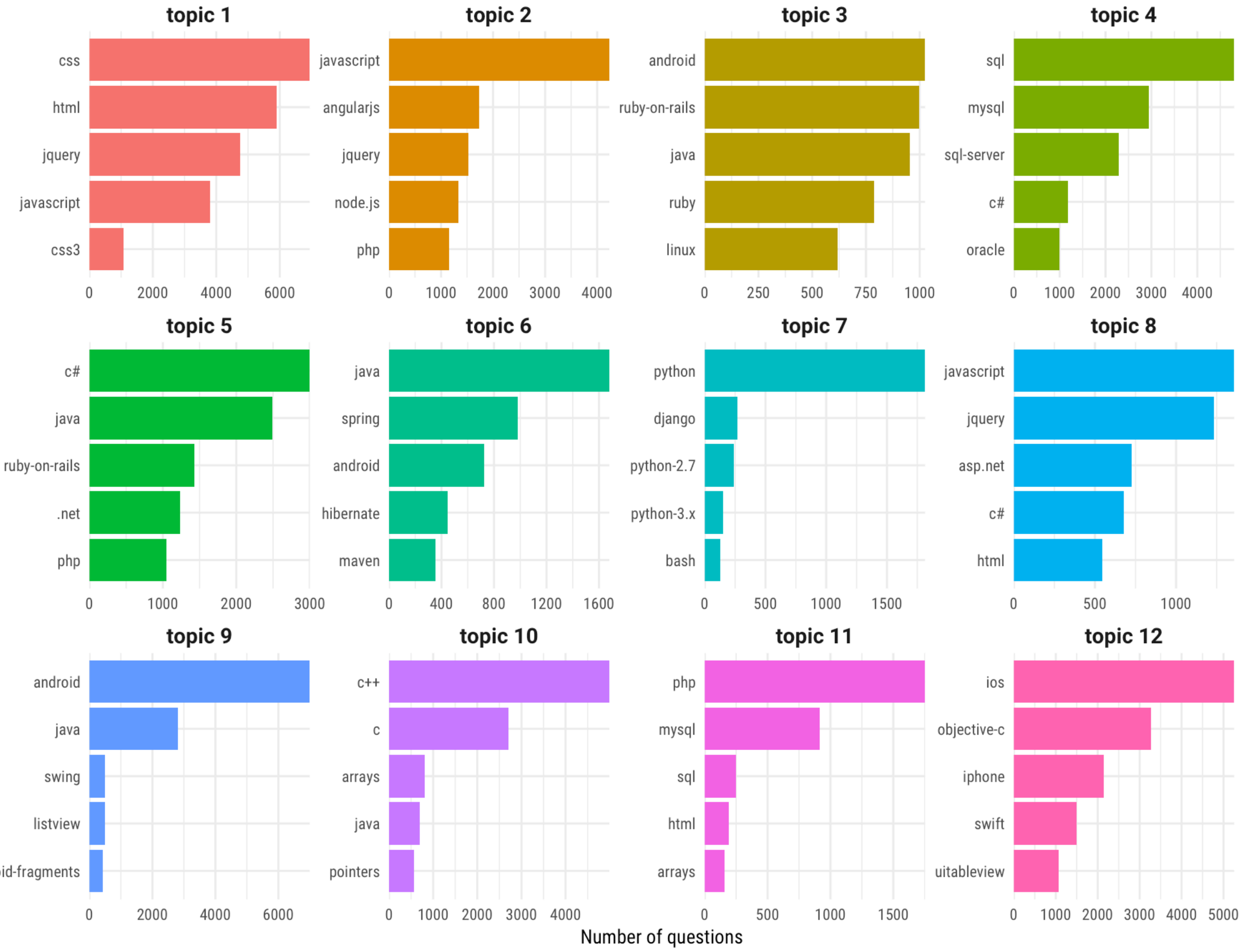
&

casting


Sentiment Lexicon
group_by
summarize
dplyr
inner_join
dplyr
dplyr, tidyr
Text Data
unnest_tokens
tidytext
Tidy Text
count
dplyr
Summarized Text
ggplot2
Visualizations
dplyr, tidyr
cast_dtm
cast_dfm
tidytext
tidy
tidytext
tm,
quanteda
tidy
tidytext
ggplot2
Corpus Object
tm
quanteda
Document-Term Matrix
LDA
topicmodels
Model
tidy
tidytext
Tidied Model
tm, quanteda
dplyr, tidyr

Top tags for each LDA topic

For questions with >80% probability for that topic



TAKING TIDY TEXT TO THE NEXT LEVEL

finding word vectors

WORD VECTORS

```
> tidy_pmi <- hacker_news_text %>%
    unnest_tokens(word, text) %>%
    add_count(word) %>%
    filter(n >= 20) %>%
    select(-n) %>%
    slide_windows(quo(postID), 8) %>%
    pairwise_pmi(word, window_id)

> tidy_word_vectors <- tidy_pmi %>%
    widely_svd(item1, item2, pmi, nv = 256, maxit = 1000)
```

WORD VECTORS

```
> tidy_word_vectors %>%
    nearest_synonyms("python")

## # A tibble: 27,267 x 2
##    item1       value
##    <chr>       <dbl>
##  1 python     0.0533
##  2 ruby       0.0309
##  3 java       0.0250
##  4 php        0.0241
##  5 c          0.0229
##  6 perl       0.0222
##  7 javascript 0.0203
##  8 django     0.0202
##  9 libraries  0.0184
## 10 languages  0.0180
## # ... with 27,257 more rows
```

WORD VECTORS

```
> tidy_word_vectors %>%
    nearest_synonyms("bitcoin")

## # A tibble: 27,267 x 2
##    item1          value
##    <chr>          <dbl>
##  1 bitcoin      0.0626
##  2 currency     0.0328
##  3 btc          0.0320
##  4 coins        0.0300
##  5 blockchain   0.0285
##  6 bitcoins     0.0258
##  7 mining       0.0252
##  8 transactions 0.0241
##  9 transaction  0.0235
## 10 currencies   0.0228
## # ... with 27,257 more rows
```

WORD VECTORS

```
> tidy_word_vectors %>%
    nearest_synonyms("women")

## # A tibble: 27,267 x 2
##    item1   value
##    <chr>   <dbl>
##  1 women  0.0648
##  2 men    0.0508
##  3 male   0.0345
##  4 female 0.0319
##  5 gender 0.0274
##  6 sex    0.0256
##  7 woman  0.0241
##  8 sexual 0.0226
##  9 males  0.0197
## 10 girls  0.0195
## # ... with 27,257 more rows
```

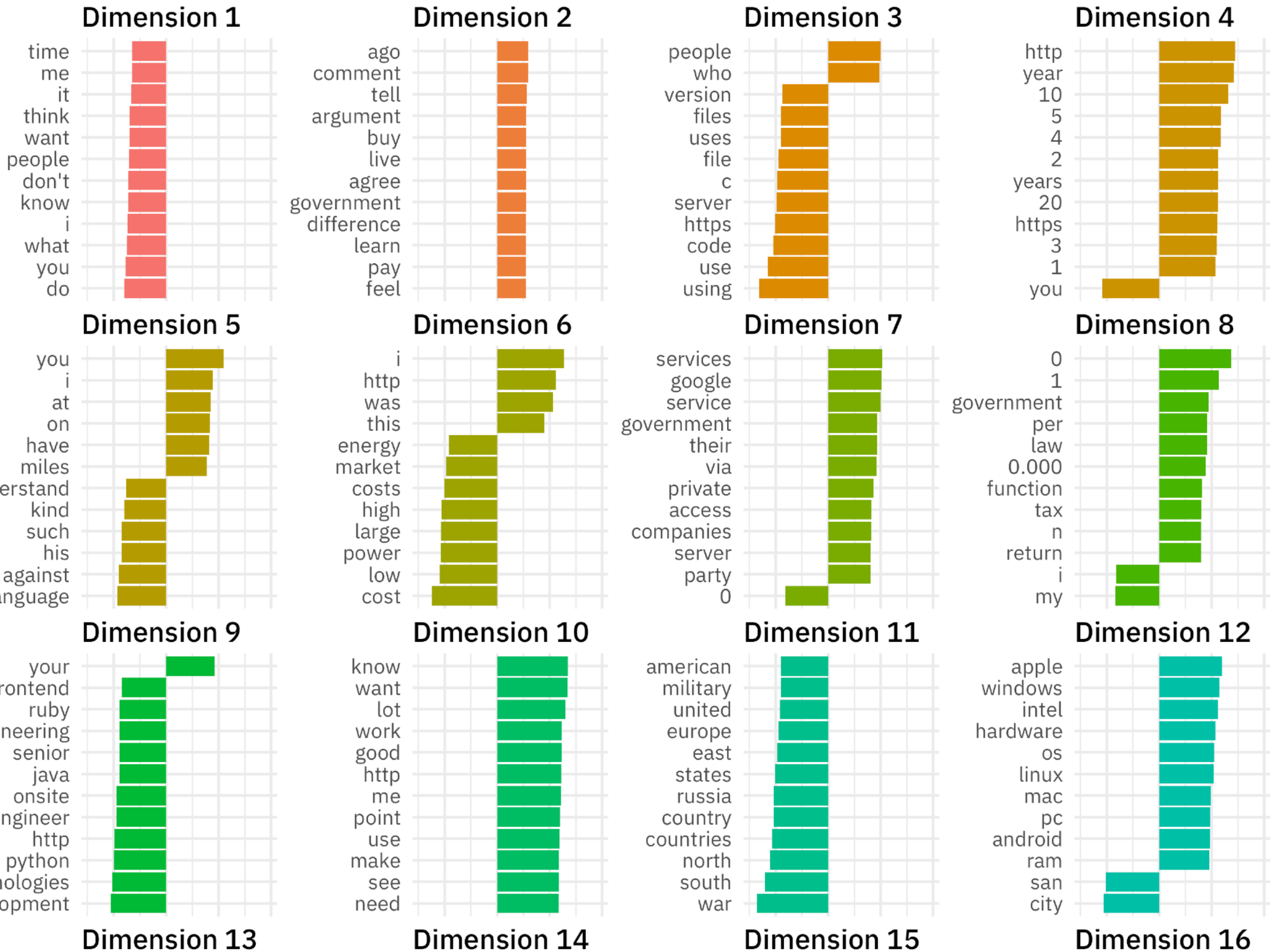
WORD VECTORS

```
> tidy_word_vectors %>%
    analogy("osx", "apple", "microsoft")

## # A tibble: 27,267 x 2
##    item1      value
##    <chr>      <dbl>
##  1 windows   0.0357
##  2 microsoft 0.0281
##  3 ms        0.0245
##  4 visual    0.0195
##  5 linux     0.0188
##  6 studio    0.0178
##  7 net       0.0171
##  8 desktop   0.0164
##  9 xp        0.0163
## 10 office    0.0147
## # ... with 27,257 more rows
```

First 24 principal components of the Hacker News corpus

Top words contributing to the components that explain the most variation



Thank You

Julia Silge
@juliasilge
https://juliasilge.com/