Text Mining

USING TIDY PRINCIPLES

Julia Silge | SDSS | 29 May 2019

Let's install some packages

```
install.packages(c("tidyverse",
                   "tidytext",
                   "gutenbergr"))
```

Find us at...

@juliasilge
@juliasilge
juliasilge.com

Find us at...

@dataandme
@batpigandme
maraaverick.rbind.io

Text in the real world

- Text data is increasingly important 📚
- NLP training is scarce on the ground 😱


Oh you take delight in vexing me

TIDY DATA PRINCIPLES + TEXT MINING = 🎉

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. Check out our book to learn more about text mining using tidy data principles.

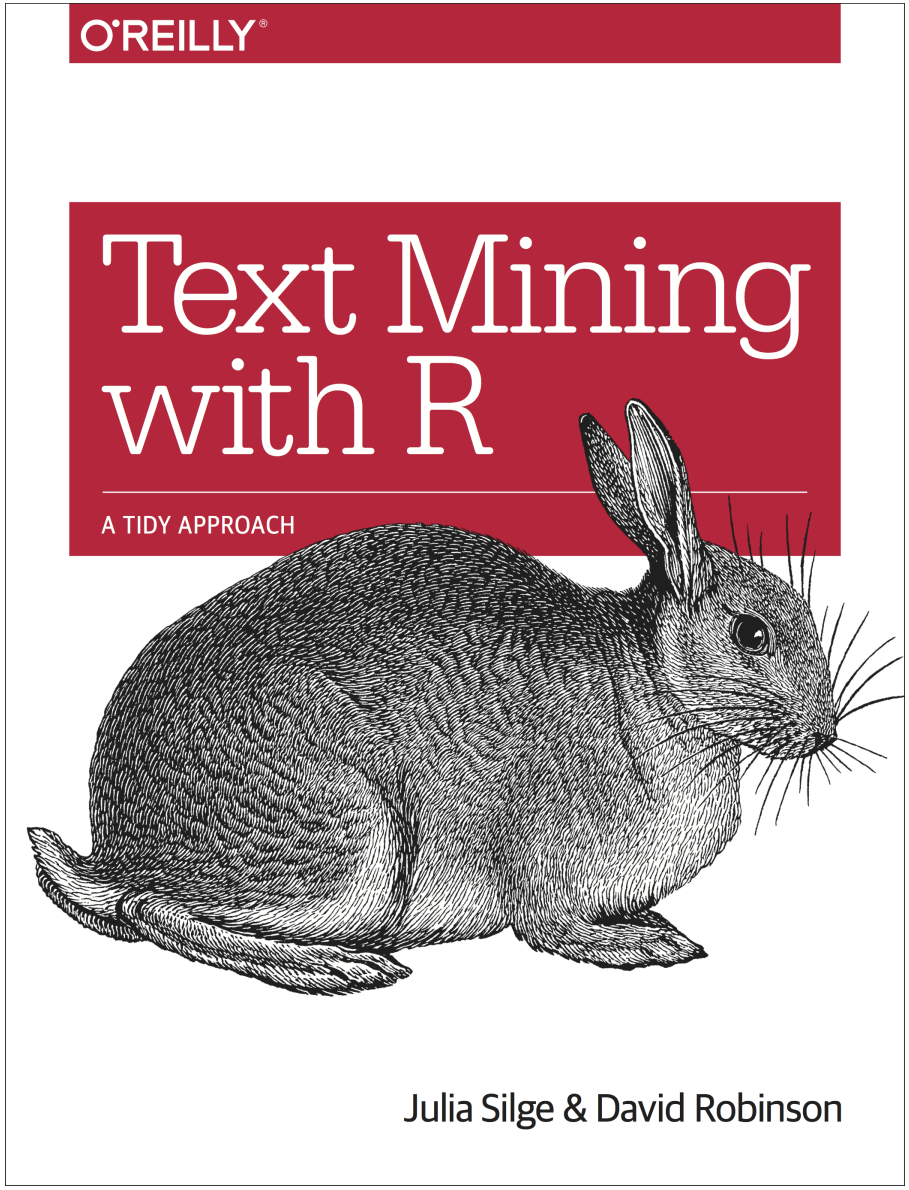
Installation

You can install this package from CRAN:

```
install.packages("tidytext")
```

https://github.com/juliasilge/tidytext

http://tidytextmining.com/


O'REILLY
Text Mining with R
A TIDY APPROACH
Julia Silge & David Robinson

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 -"
## [2] "He kindly stopped for me -"
## [3] "The Carriage held but just Ourselves -"
## [4] "and Immortality"
```

What do we mean by tidy text?

```
library(tidyverse)
text_df <- data_frame(line = 1:4, text = text)

text_df
```

```
## # A tibble: 4 x 2
##    line text
##   <int> <chr>
## 1     1 Because I could not stop for Death -
## 2     2 He kindly stopped for me -
## 3     3 The Carriage held but just Ourselves -
## 4     4 and Immortality
```

What do we mean by tidy text?

```
library(tidytext)

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

```
## # A tibble: 20 x 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
## 11     2 for
## 12     2 me
## 13     3 the
## 14     3 carriage
## 15     3 held
## 16     3 but
## 17     3 just
## 18     3 ourselves
## 19     4 and
## 20     4 immortality
```

Gathering more data

You can access the full text of many public domain works from Project Gutenberg using the gutenbergr package.

```
library(gutenbergr)

full_text <- gutenberg_download(1342)
```

What book do *you* want to analyze today? 📖 👯 📖

Time to tidy your text!

```
tidy_book <- full_text %>%
  mutate(line = row_number()) %>%
  unnest_tokens(word, text)

tidy_book
```

```
## # A tibble: 122,204 x 3
##    gutenberg_id  line word
##           <int> <int> <chr>
##  1         1342     1 pride
##  2         1342     1 and
##  3         1342     1 prejudice
##  4         1342     3 by
##  5         1342     3 jane
##  6         1342     3 austen
##  7         1342     7 chapter
##  8         1342     7 1
##  9         1342    10 it
## 10         1342    10 is
## # … with 122,194 more rows
```

What are the most common words?

```
tidy_book %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 6,538 x 2
##    word      n
##    <chr> <int>
##  1 the    4331
##  2 to     4162
##  3 of     3610
##  4 and    3585
##  5 her    2203
##  6 i      2065
##  7 a      1954
##  8 in     1880
##  9 was    1843
## 10 she    1695
## # … with 6,528 more rows
```

Stop words

Stop words

```
get_stopwords()
```

```
## # A tibble: 175 x 2
##    word      lexicon
##    <chr>     <chr>
##  1 i         snowball
##  2 me        snowball
##  3 my        snowball
##  4 myself    snowball
##  5 we        snowball
##  6 our       snowball
##  7 ours      snowball
##  8 ourselves snowball
##  9 you       snowball
## 10 your      snowball
## # … with 165 more rows
```

Stop words

```
get_stopwords(language = "es")
```

```
## # A tibble: 308 x 2
##    word  lexicon
##    <chr> <chr>
##  1 de    snowball
##  2 la    snowball
##  3 que   snowball
##  4 el    snowball
##  5 en    snowball
##  6 y     snowball
##  7 a     snowball
##  8 los   snowball
##  9 del   snowball
## 10 se    snowball
## # … with 298 more rows
```

Stop words

```
get_stopwords(language = "pt")
```

```
## # A tibble: 203 x 2
##    word  lexicon
##    <chr> <chr>
##  1 de    snowball
##  2 a     snowball
##  3 o     snowball
##  4 que   snowball
##  5 e     snowball
##  6 do    snowball
##  7 da    snowball
##  8 em    snowball
##  9 um    snowball
## 10 para  snowball
## # … with 193 more rows
```

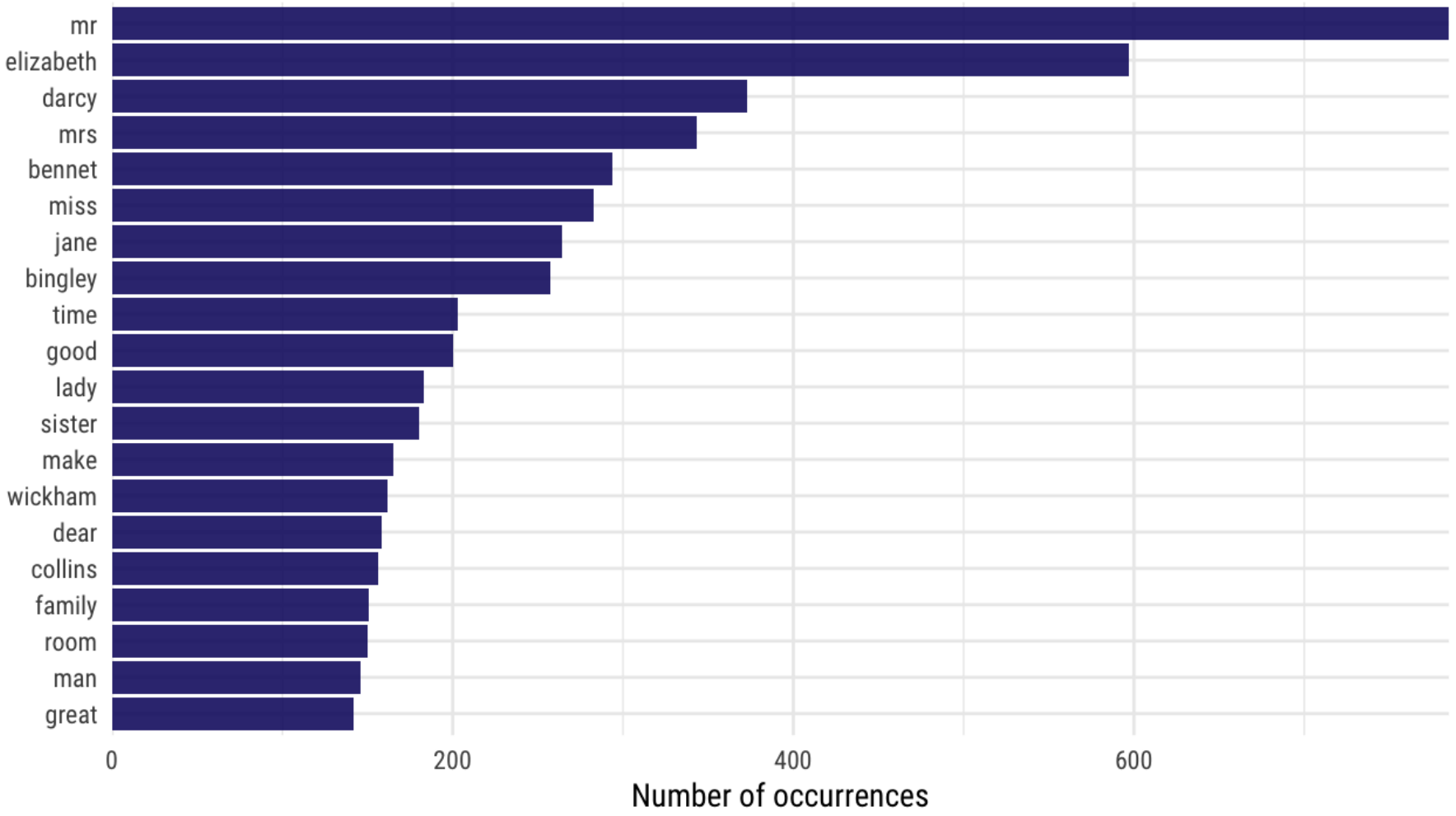
Stop words

```
get_stopwords(source = "smart")
```

```
## # A tibble: 571 x 2
##    word        lexicon
##    <chr>       <chr>
##  1 a           smart
##  2 a's         smart
##  3 able        smart
##  4 about       smart
##  5 above       smart
##  6 according   smart
##  7 accordingly smart
##  8 across      smart
##  9 actually    smart
## 10 after       smart
## # … with 561 more rows
```

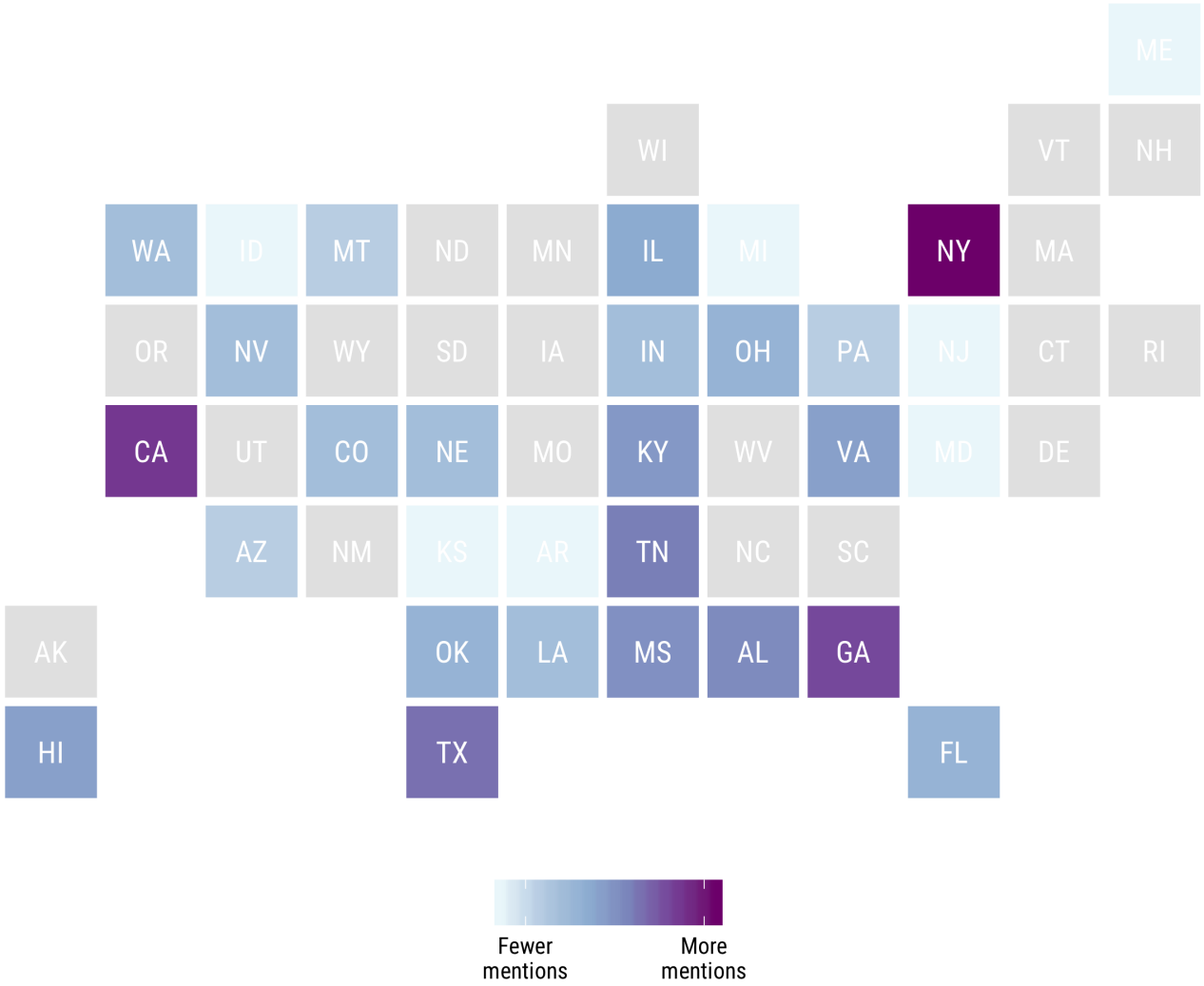
What are the most common words?

```
tidy_book %>%
  anti_join(get_stopwords(source = "smart")) %>%
  count(word, sort = TRUE) %>%
  top_n(20) %>%
  ggplot(aes(fct_reorder(word, n), n)) +
  geom_col() +
  coord_flip()
```


mr
elizabeth
darcy
mrs
bennet
miss
jane
bingley
time
good
lady
sister
make
wickham
dear
collins
family
room
man
great
0
200
400
600
Number of occurrences

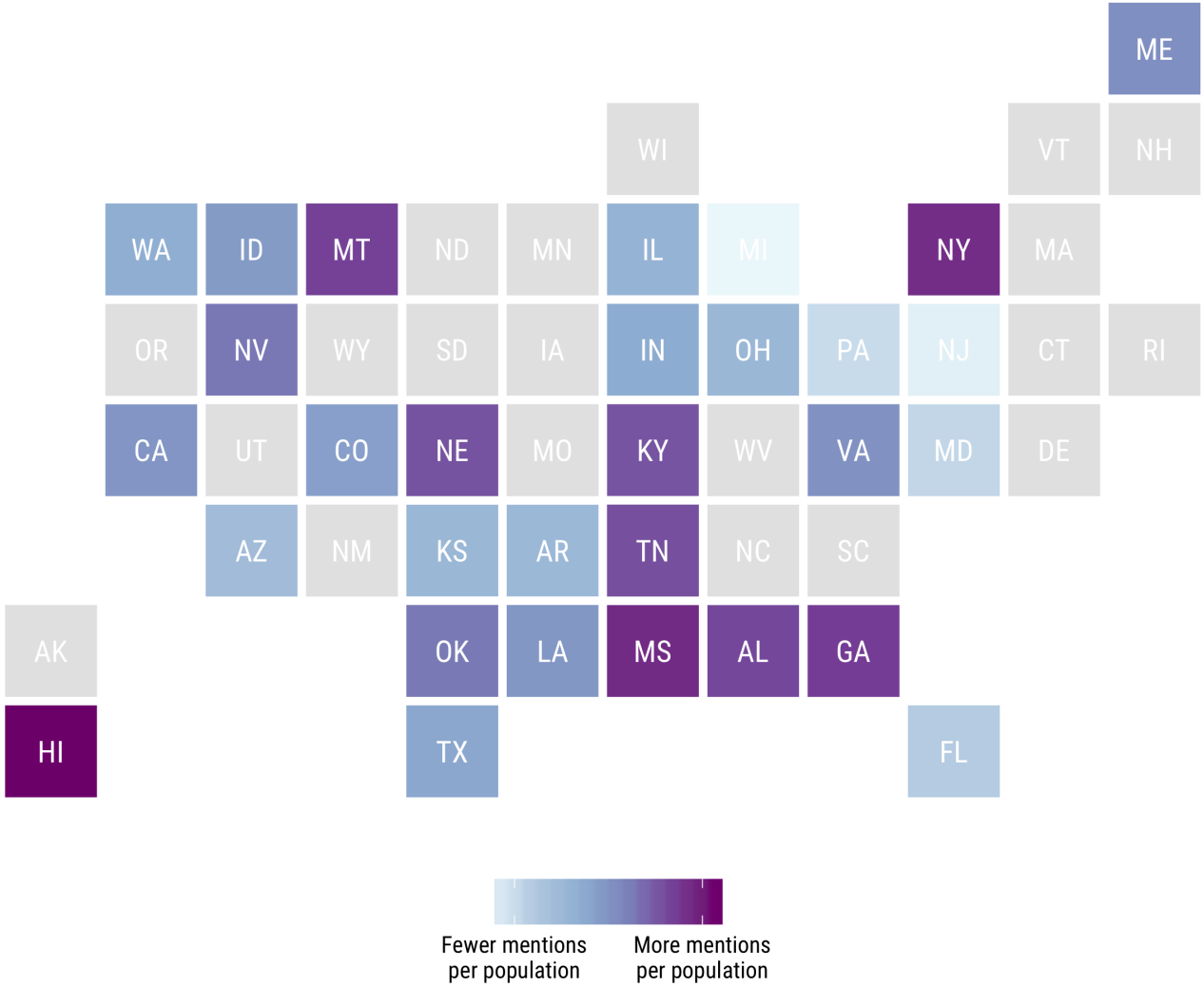
Which States Are Mentioned Most in Song Lyrics?

States like California are mentioned most often



Which States Are Mentioned Most in Song Lyrics?

States like Hawaii and Montana are mentioned more often relative to their population



SENTIMENT ANALYSIS 😄 😢 😠

Sentiment lexicons

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

```
## # A tibble: 2,476 x 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
```

Sentiment lexicons

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

```
## # A tibble: 6,788 x 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
```

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("loughran")
```

```
## # A tibble: 4,149 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 4,139 more rows
```

Implementing sentiment analysis

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

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 positive   5052
## 2 negative   3652
```

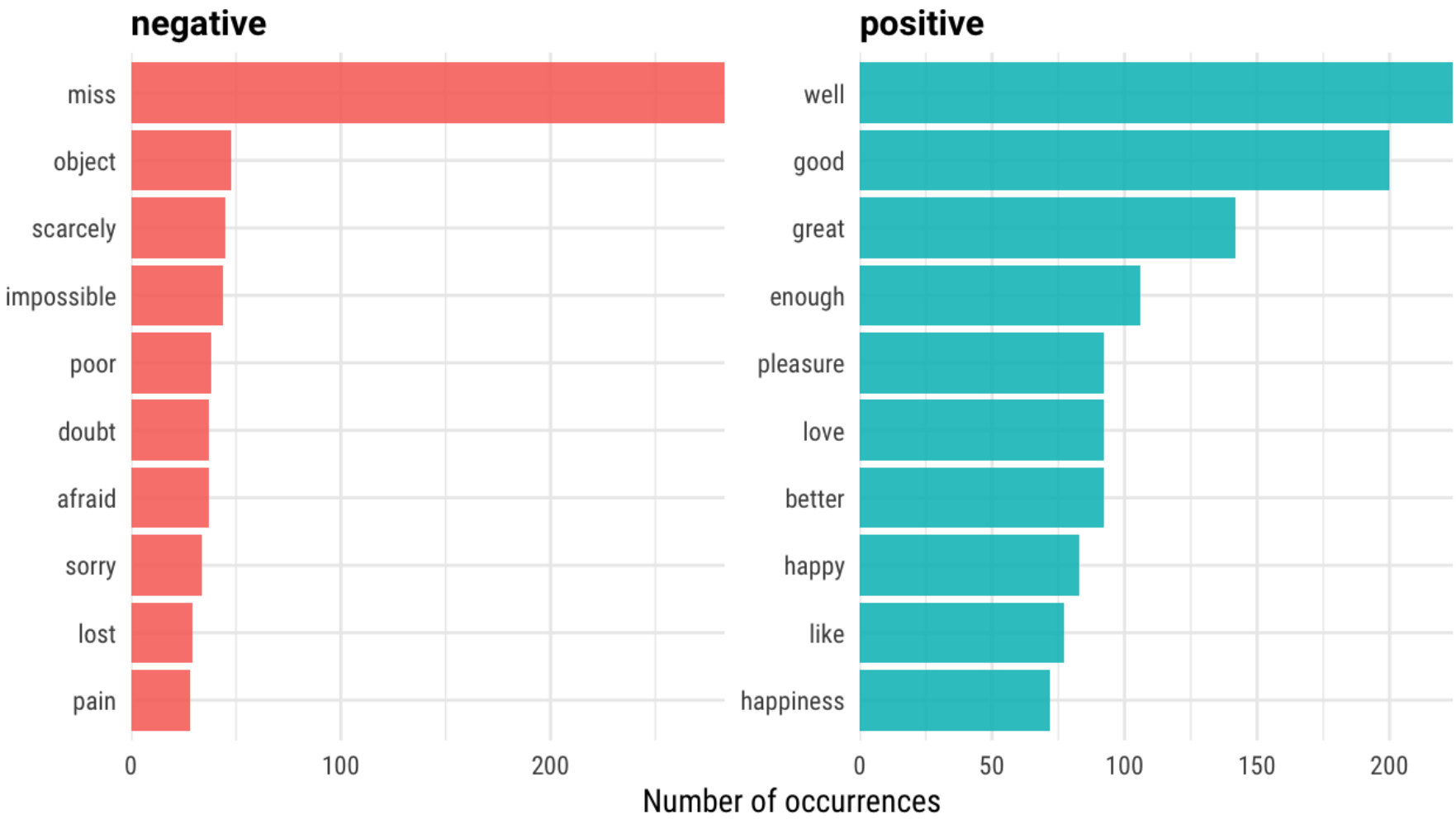
Implementing sentiment analysis

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

```
## # A tibble: 1,430 x 3
##    sentiment word          n
##    <chr>     <chr>     <int>
##  1 negative  miss        283
##  2 positive  well        224
##  3 positive  good        200
##  4 positive  great       142
##  5 positive  enough      106
##  6 positive  better       92
##  7 positive  love         92
##  8 positive  pleasure     92
##  9 positive  happy        83
## 10 positive  like         77
## # … with 1,420 more rows
```

Implementing sentiment analysis

```
tidy_book %>%
  inner_join(get_sentiments("bing")) %>%
  count(sentiment, word, sort = TRUE) %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup %>%
  ggplot(aes(fct_reorder(word, n),
             n,
             fill = sentiment)) +
  geom_col() +
  coord_flip() +
  facet_wrap(~ sentiment, scales = "free")
```


negative
miss
object
scarcely
impossible
poor
doubt
afraid
sorry
lost
pain
0
100
200
positive
well
good
great
enough
pleasure
love
better
happy
like
happiness
0
50
100
150
200
Number of occurrences

WHAT IS A DOCUMENT ABOUT? 🤔

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 is about comparing documents within a collection.

Understanding tf-idf

Make a collection (*corpus*) for yourself! 💅

```
full_collection <- gutenberg_download(c(1342, 158, 161, 141),
                                      meta_fields = "title")

full_collection
```

```
## # A tibble: 57,251 x 3
##    gutenberg_id text            title
##           <int> <chr>           <chr>
##  1          141 MANSFIELD PARK Mansfield Park
##  2          141 ""             Mansfield Park
##  3          141 (1814)         Mansfield Park
##  4          141 ""             Mansfield Park
##  5          141 ""             Mansfield Park
##  6          141 By Jane Austen Mansfield Park
##  7          141 ""             Mansfield Park
##  8          141 ""             Mansfield Park
##  9          141 ""             Mansfield Park
## 10          141 ""             Mansfield Park
## # … with 57,241 more rows
```

Counting word frequencies in your collection

```
book_words <- full_collection %>%
  unnest_tokens(word, text) %>%
  count(title, word, sort = TRUE)

book_words
```

```
## # A tibble: 28,395 x 3
##    title                word      n
##    <chr>                <chr> <int>
##  1 Mansfield Park       the    6206
##  2 Mansfield Park       to     5475
##  3 Mansfield Park       and    5438
##  4 Emma                 to     5239
##  5 Emma                 the    5201
##  6 Emma                 and    4896
##  7 Mansfield Park       of     4778
##  8 Pride and Prejudice the    4331
##  9 Emma                 of     4291
## 10 Pride and Prejudice to     4162
## # … with 28,385 more rows
```

Calculating tf-idf

That's... super exciting???

```
book_tfidf <- book_words %>%
  bind_tf_idf(word, title, n)

book_tfidf
```

```
## # A tibble: 28,395 x 6
##    title               word      n     tf   idf tf_idf
##    <chr>               <chr> <int>  <dbl> <dbl>  <dbl>
##  1 Mansfield Park      the    6206 0.0387     0      0
##  2 Mansfield Park      to     5475 0.0341     0      0
##  3 Mansfield Park      and    5438 0.0339     0      0
##  4 Emma                to     5239 0.0325     0      0
##  5 Emma                the    5201 0.0323     0      0
##  6 Emma                and    4896 0.0304     0      0
##  7 Mansfield Park      of     4778 0.0298     0      0
##  8 Pride and Prejudice the    4331 0.0354     0      0
##  9 Emma                of     4291 0.0267     0      0
## 10 Pride and Prejudice to     4162 0.0341     0      0
## # … with 28,385 more rows
```

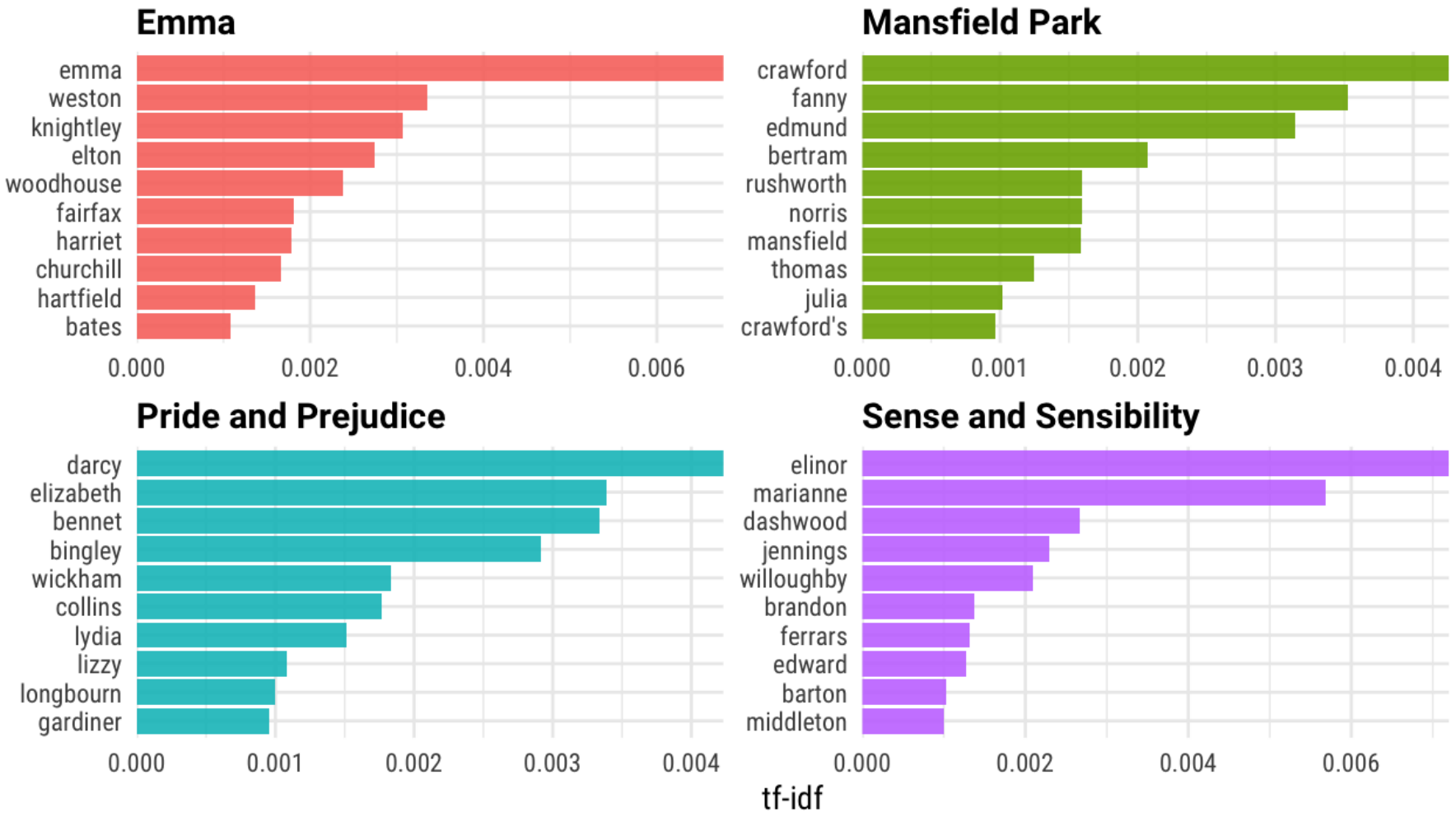
Calculating tf-idf

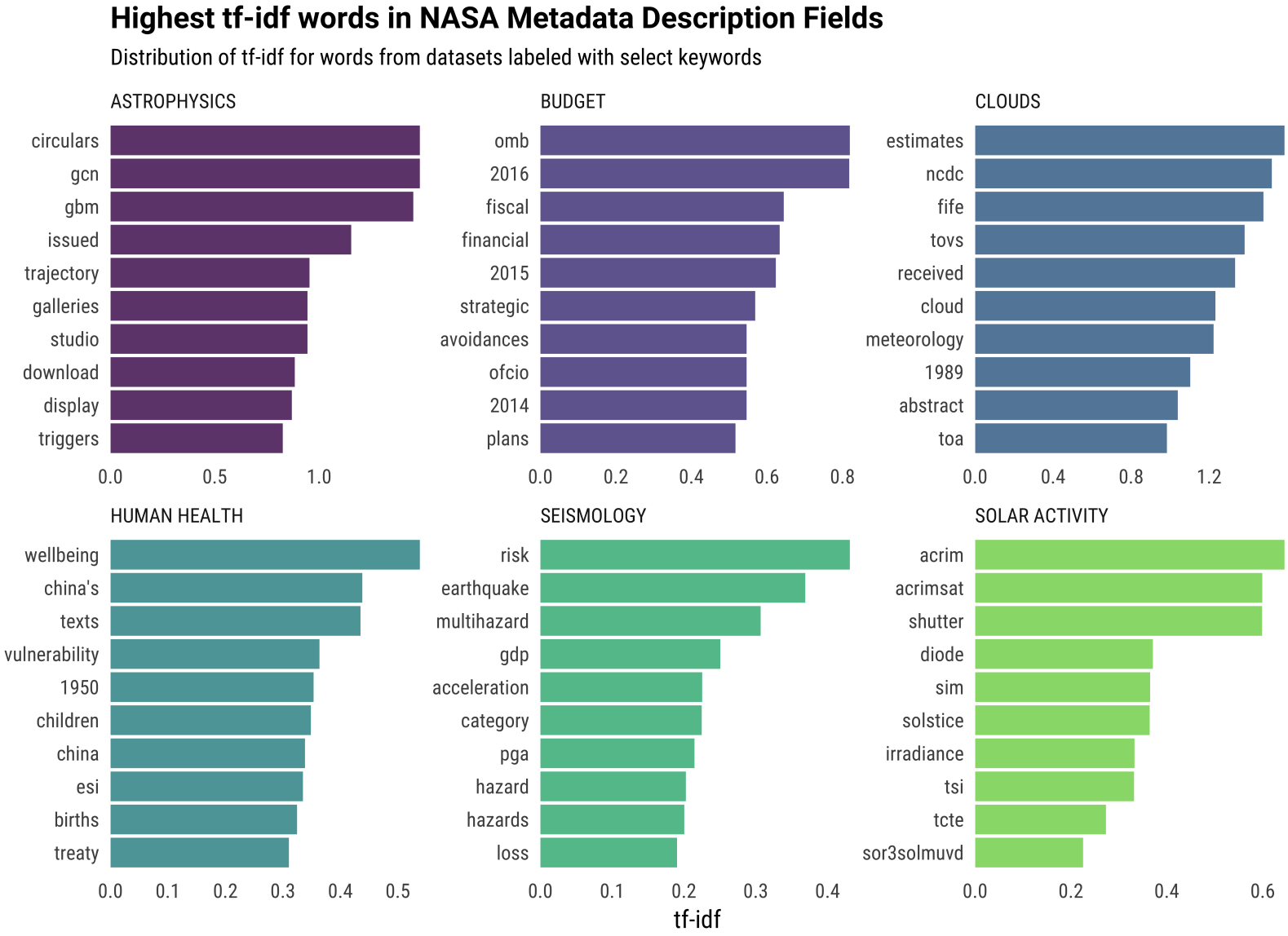
```
book_tfidf %>%
  arrange(-tf_idf)
```

```
## # A tibble: 28,395 x 6
##    title                word          n      tf   idf  tf_idf
##    <chr>                <chr>     <int>   <dbl> <dbl>   <dbl>
##  1 Sense and Sensibility elinor      623 0.00519 1.39  0.00720
##  2 Emma                 emma         786 0.00488 1.39  0.00677
##  3 Sense and Sensibility marianne    492 0.00410 1.39  0.00569
##  4 Mansfield Park       crawford     493 0.00307 1.39  0.00426
##  5 Pride and Prejudice  darcy        373 0.00305 1.39  0.00423
##  6 Mansfield Park       fanny        816 0.00509 0.693 0.00352
##  7 Pride and Prejudice  elizabeth    597 0.00489 0.693 0.00339
##  8 Emma                 weston       389 0.00242 1.39  0.00335
##  9 Pride and Prejudice  bennet       294 0.00241 1.39  0.00334
## 10 Mansfield Park       edmund       364 0.00227 1.39  0.00314
## # … with 28,385 more rows
```

Calculating tf-idf

```
book_tfidf %>%
  group_by(title) %>%
  top_n(10) %>%
  ungroup %>%
  ggplot(aes(fct_reorder(word, tf_idf),
             tf_idf,
             fill = title)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  facet_wrap(~title, scales = "free")
```


Emma
emma
weston
knightley
elton
woodhouse
fairfax
harriet
churchill
hartfield
bates
0.000
0.002
0.004
0.006
Mansfield Park
crawford
fanny
edmund
bertram
rushworth
norris
mansfield
thomas
julia
crawford's
0.000
0.001
0.002
0.003
0.004
Pride and Prejudice
darcy
elizabeth
bennet
bingley
wickham
collins
lydia
lizzy
longbourn
gardiner
0.000
0.001
0.002
0.003
0.004
Sense and Sensibility
elinor
marianne
dashwood
jennings
willoughby
brandon
ferrars
edward
barton
middleton
0.000
0.002
0.004
0.006
tf-idf


Highest tf-idf words in NASA Metadata Description Fields
Distribution of tf-idf for words from datasets labeled with select keywords
ASTROPHYSICS
circulars
gcn
gbm
issued
trajectory
galleries
studio
download
display
triggers
0.0
0.5
1.0
BUDGET
omb
2016
fiscal
financial
2015
strategic
avoidances
ofcio
2014
plans
0.0
0.2
0.4
0.6
0.8
CLOUDS
estimates
ncdc
fife
tovs
received
cloud
meteorology
1989
abstract
toa
0.0
0.4
0.8
1.2
HUMAN HEALTH
wellbeing
china's
texts
vulnerability
1950
children
china
esi
births
treaty
0.0
0.1
0.2
0.3
0.4
0.5
SEISMOLOGY
risk
earthquake
multihazard
gdp
acceleration
category
pga
hazard
hazards
loss
0.0
0.1
0.2
0.3
0.4
SOLAR ACTIVITY
acrim
acrimsat
shutter
diode
sim
solstice
irradiance
tsi
tcte
sor3solmuvd
0.0
0.2
0.4
0.6
tf-idf
NASA metadata from https://data.nasa.gov/data.json

N-grams... and beyond! 🚀

```
tidy_ngram <- full_text %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

tidy_ngram
```

```
## # A tibble: 122,203 x 2
##    gutenberg_id bigram
##           <int> <chr>
##  1         1342 pride and
##  2         1342 and prejudice
##  3         1342 prejudice by
##  4         1342 by jane
##  5         1342 jane austen
##  6         1342 austen chapter
##  7         1342 chapter 1
##  8         1342 1 it
##  9         1342 it is
## 10         1342 is a
## # … with 122,193 more rows
```

N-grams... and beyond! 🚀

```
tidy_ngram %>%
  count(bigram, sort = TRUE)
```

```
## # A tibble: 54,998 x 2
##    bigram       n
##    <chr>    <int>
##  1 of the     464
##  2 to be      443
##  3 in the     382
##  4 i am       302
##  5 of her     260
##  6 to the     252
##  7 it was     251
##  8 mr darcy   243
##  9 of his     234
## 10 she was    209
## # … with 54,988 more rows
```

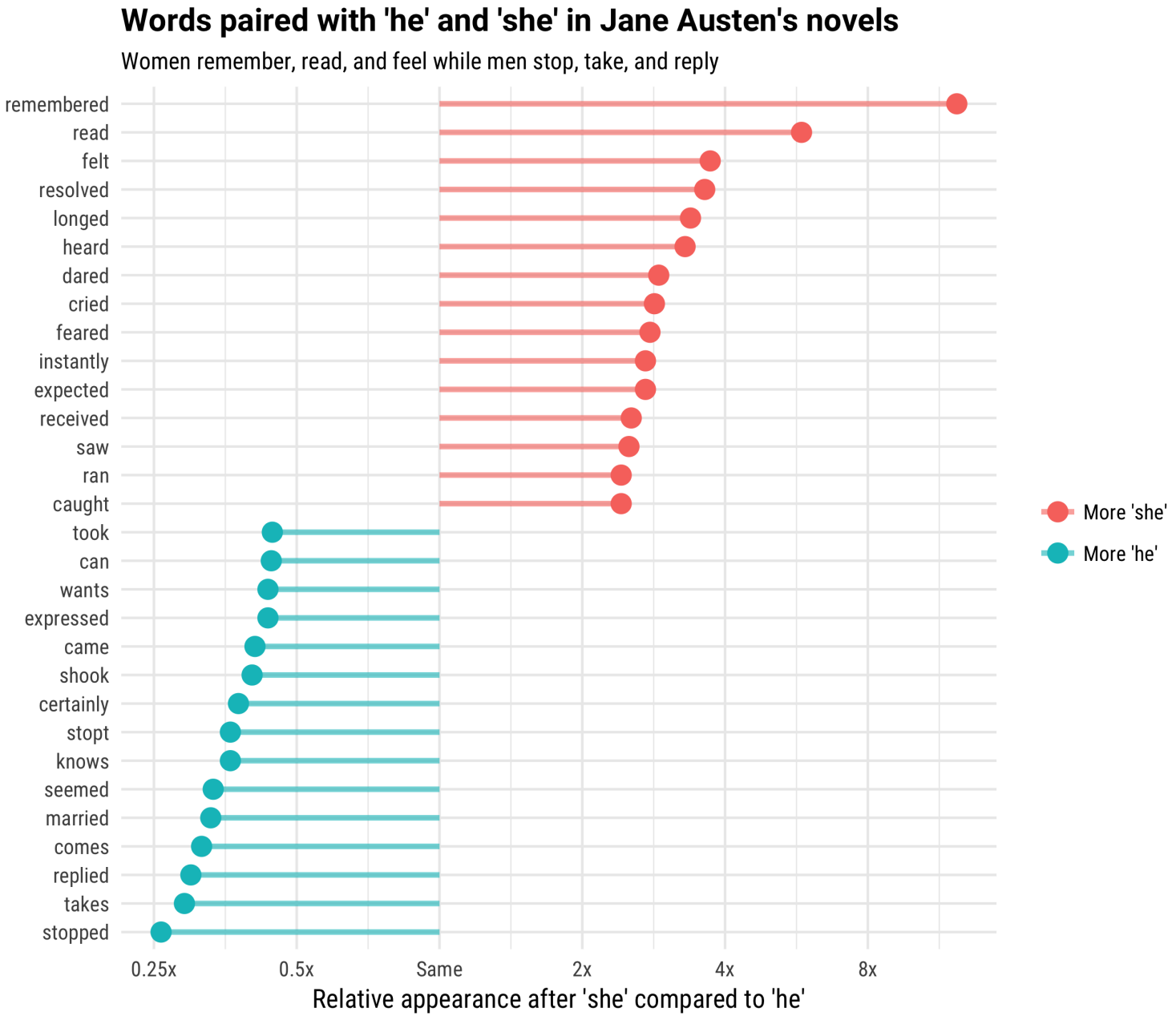
N-grams... and beyond! 🚀

```
tidy_ngram %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  filter(!word1 %in% stop_words$word,
         !word2 %in% stop_words$word) %>%
  count(word1, word2, sort = TRUE)
```

```
## # A tibble: 5,922 x 3
##    word1   word2           n
##    <chr>   <chr>       <int>
##  1 lady    catherine     100
##  2 miss    bingley        72
##  3 miss    bennet         60
##  4 sir     william        38
##  5 de      bourgh         35
##  6 miss    darcy          34
##  7 colonel forster        26
##  8 colonel fitzwilliam    25
##  9 cried   elizabeth      24
## 10 miss    lucas          23
## # … with 5,912 more rows
```

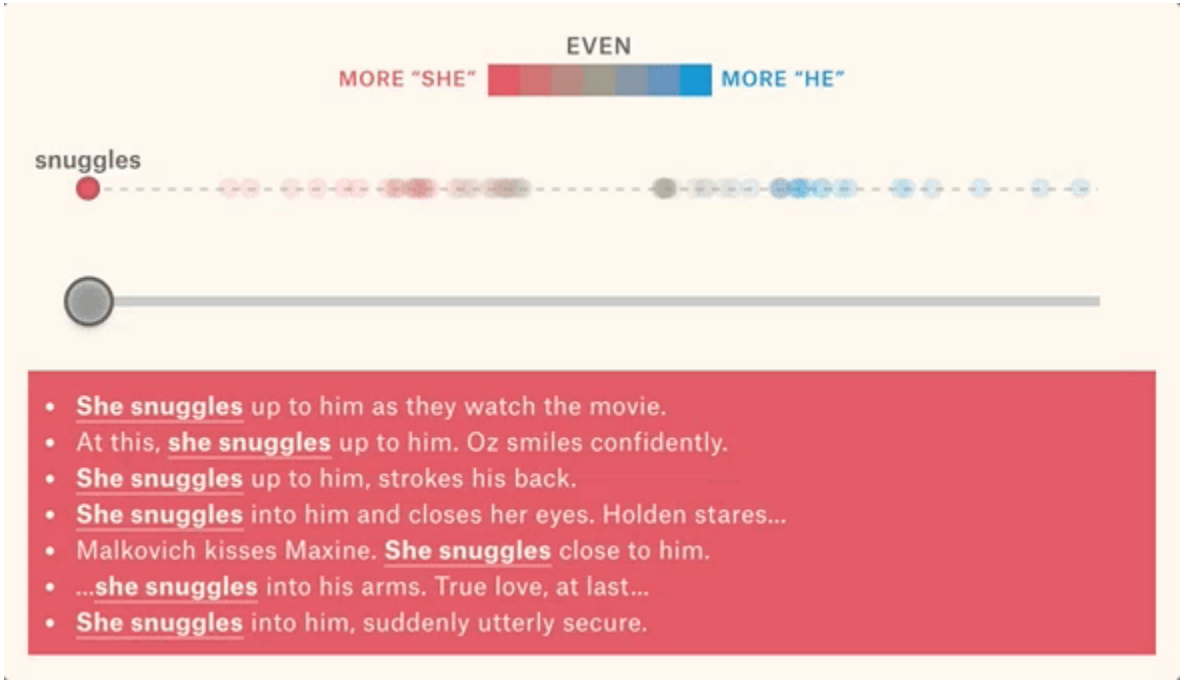
What can you do with n-grams?

- tf-idf of n-grams
- network analysis
- negation


Words paired with 'he' and 'she' in Jane Austen's novels
Women remember, read, and feel while men stop, take, and reply
remembered
read
felt
resolved
longed
heard
dared
cried
feared
instantly
expected
received
saw
ran
caught
took
can
wants
expressed
came
shook
certainly
stopt
knows
seemed
married
comes
replied
takes
stopped
0.25x
0.5x
Same
2x
4x
8x
Relative appearance after 'she' compared to 'he'
More 'she'
More 'he'

What can you do with n-grams?

She Giggles, He Gallops



Thanks!

- tidytextmining.com
- @juliasilge
- @juliasilge
- juliasilge.com
- @dataandme
- @batpigandme
- maraaverick.rbind.io

Slides created with **remark.js** and the R package **xaringan**