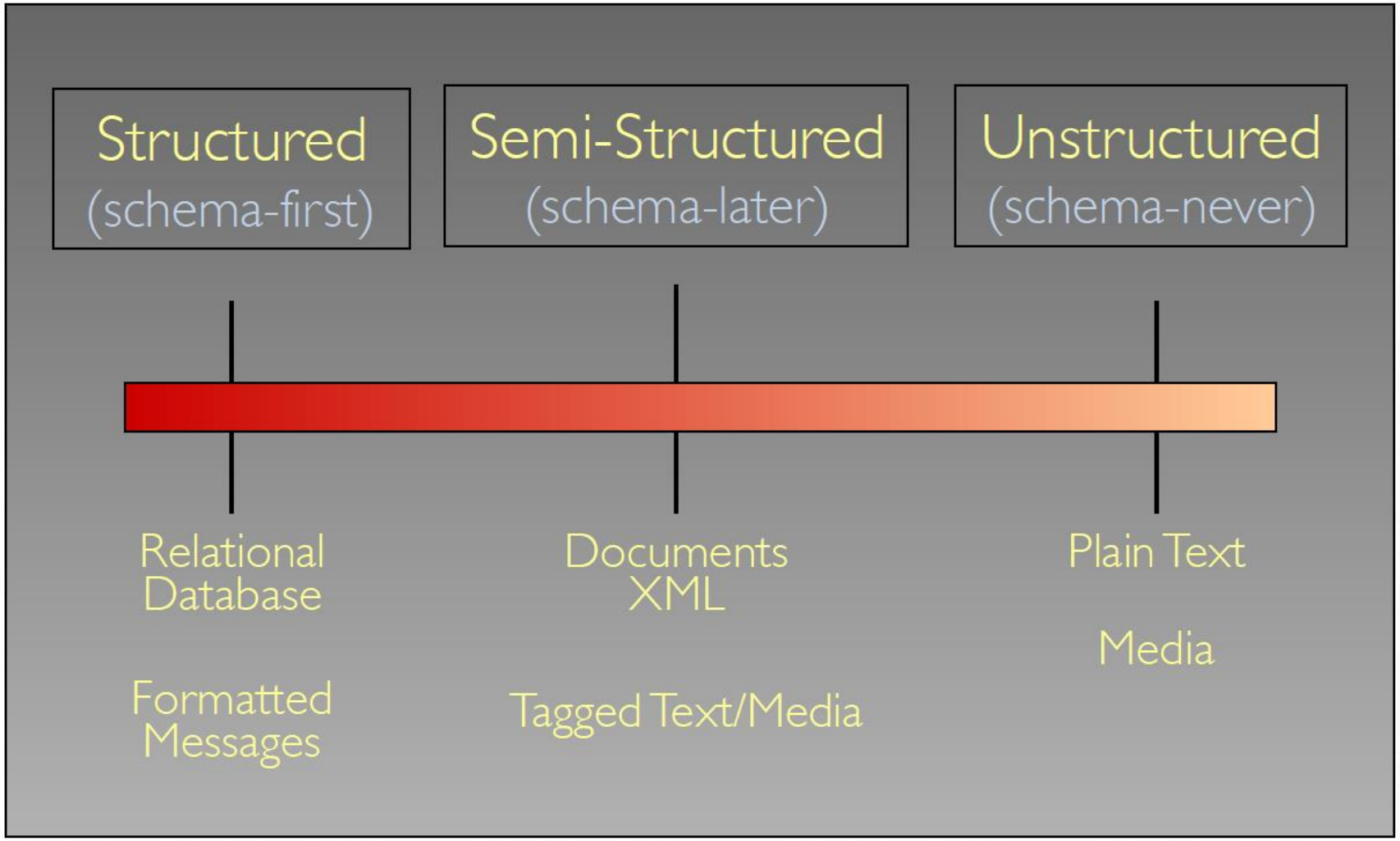
데이터 과학 외전

Day 2 – 웹크롤링 & Text Mining

Unstructured Data Analysis



Structured Data:

Information stored in databases is known as structured data because it is represented in a strict format.
The DBMS then checks to ensure that all data follows the structures and constraints specified in the schema.

Semi-Structured Data:

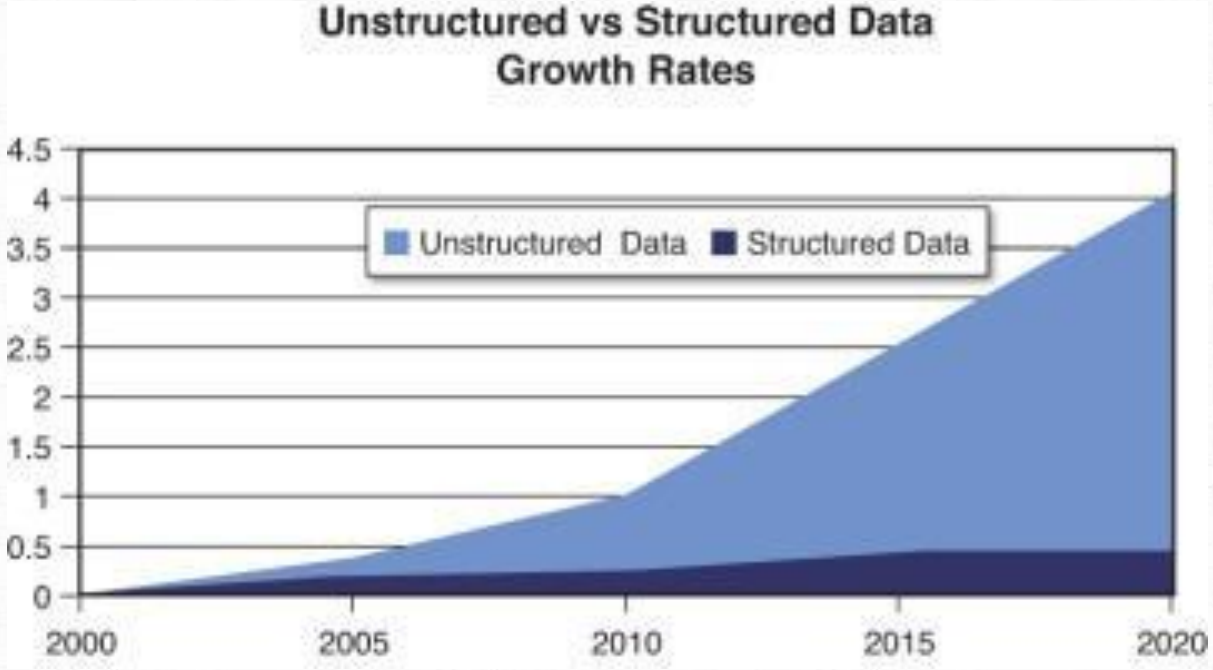
In some applications, data is collected in an ad-hoc manner before it is known how it will be stored and managed.
This data may have a certain structure, but not all the information collected will have identical structure.
This type of data is known as semi-structured data.
In semi-structured data, the schema information is mixed in with the data values, since each data object can have different attributes that are not known in advance. Hence, this type of data is sometimes referred to as self-describing data.

Unstructured Data:

A third category is known as unstructured data, because there is very limited indication of the type of data. A typical example would be a text document that contains information embedded within it
For Example, images and graphics, pdf files, word document, audio, video, emails, powerpoint presentations, webpages and web contents, wikis, streaming data, location coordinates etc

Most of data are unstructured, less are semi-structured and little are structured

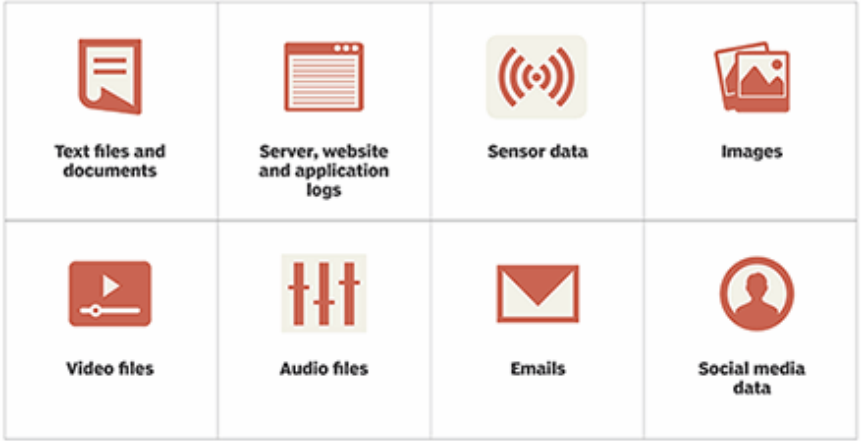
Unstructured Data Grows Faster



- It is said that Unstructured data takes about 80% of all data

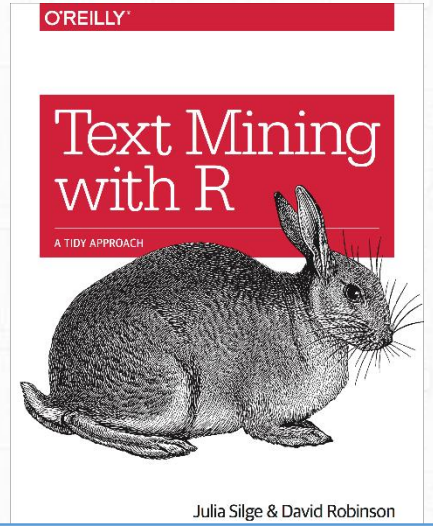
Unstructured Data

- Various ways to approach different types of unstructured data
  - Image
    - Image processing, image recognition, ... Many of AI application
  - Sound
    - Voice Recognition, ... Many of AI application
  - Text
    - Text Mining, Natural Language Processing (NLP)
  - Sensors
    - Time Series Data Analysis, Data Synchronization, ...
  - Hybrid
    - ...



The tidy text format

- As described by Hadley Wickham (Wickham 2014), tidy data has a specific structure:
  - Each variable is a column
  - Each observation is a row
  - Each type of observational unit is a table
- Here, a table with one-token-per-row. A token is a meaningful unit of text, such as a word
  - most often a single word, but can also be an n-gram, sentence, or paragraph.
- Tokenization is the process of splitting text into tokens.



www.tidytextmining.com

The unnest_tokens Function in tidytext

Emily Dickinson wrote some lovely text in her time.

```
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"
```

The unnest_tokens Function in tidytext

```
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
```

```
library(tidytext)
text_df <- data_frame(line = 1:4, text = text)
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
## # ... with 10 more rows
```

word: the output column name that will be created as the text is unnested into it
text: the input column that the text comes from

Tidying the works of Jane Austen

```r
library(janeaustenr)
library(dplyr)
library(stringr)

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

```r
original_books

## # A tibble: 73,422 x 4
##    text                  book                linenumber chapter
##    <chr>                 <fct>                    <int>   <int>
##  1 SENSE AND SENSIBILITY Sense & Sensibility          1       0
##  2 ""                    Sense & Sensibility          2       0
##  3 by Jane Austen        Sense & Sensibility          3       0
##  4 ""                    Sense & Sensibility          4       0
##  5 (1811)                Sense & Sensibility          5       0
##  6 ""                    Sense & Sensibility          6       0
##  7 ""                    Sense & Sensibility          7       0
##  8 ""                    Sense & Sensibility          8       0
##  9 ""                    Sense & Sensibility          9       0
## 10 CHAPTER 1             Sense & Sensibility         10       1
## # ... with 73,412 more rows
```

```r
library(tidytext)
tidy_books <- original_books %>%
  unnest_tokens(word, text)

tidy_books
```

```
## # A tibble: 725,055 x 4
##    book                linenumber chapter word
##    <fct>                    <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 725,045 more rows
```

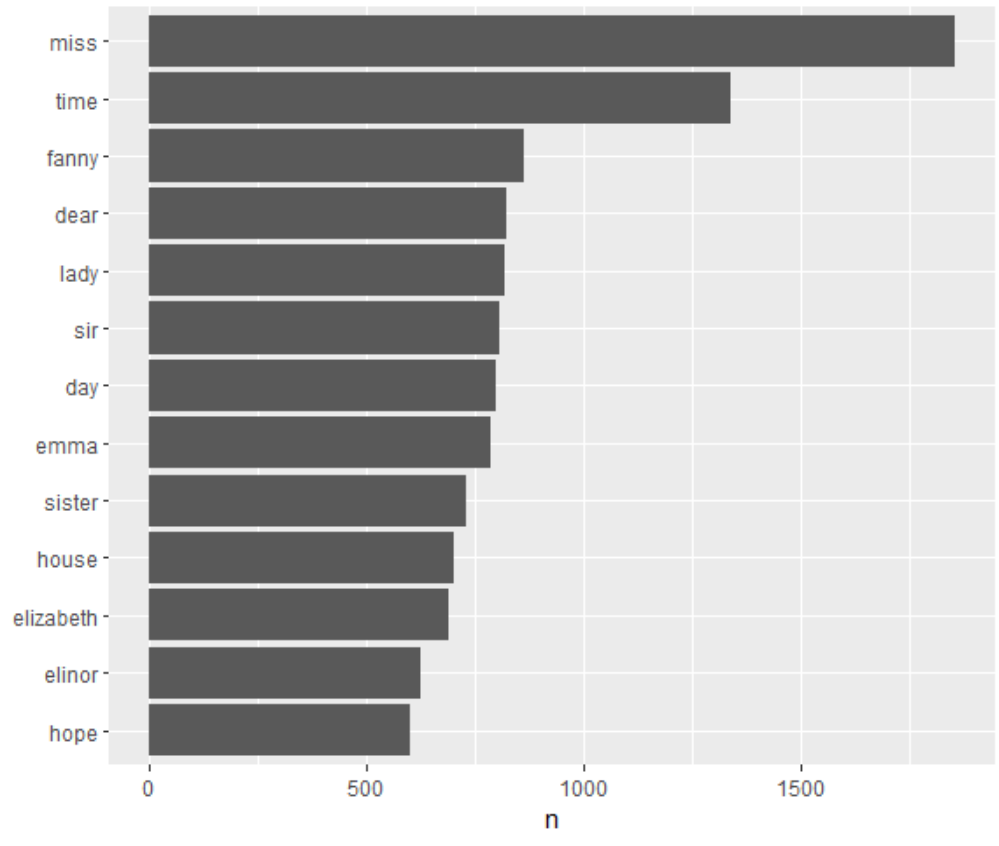
removing stop-words

```
data(stop_words)

tidy_books <- tidy_books %>%
  anti_join(stop_words)

tidy_books %>%
  count(word, sort = TRUE)

## # A tibble: 13,914 x 2
##    word        n
##    <chr>   <int>
##  1 miss     1855
##  2 time     1337
##  3 fanny     862
##  4 dear      822
##  5 lady      817
##  6 sir       806
##  7 day       797
##  8 emma      787
##  9 sister    727
## 10 house     699
## # ... with 13,904 more rows
```

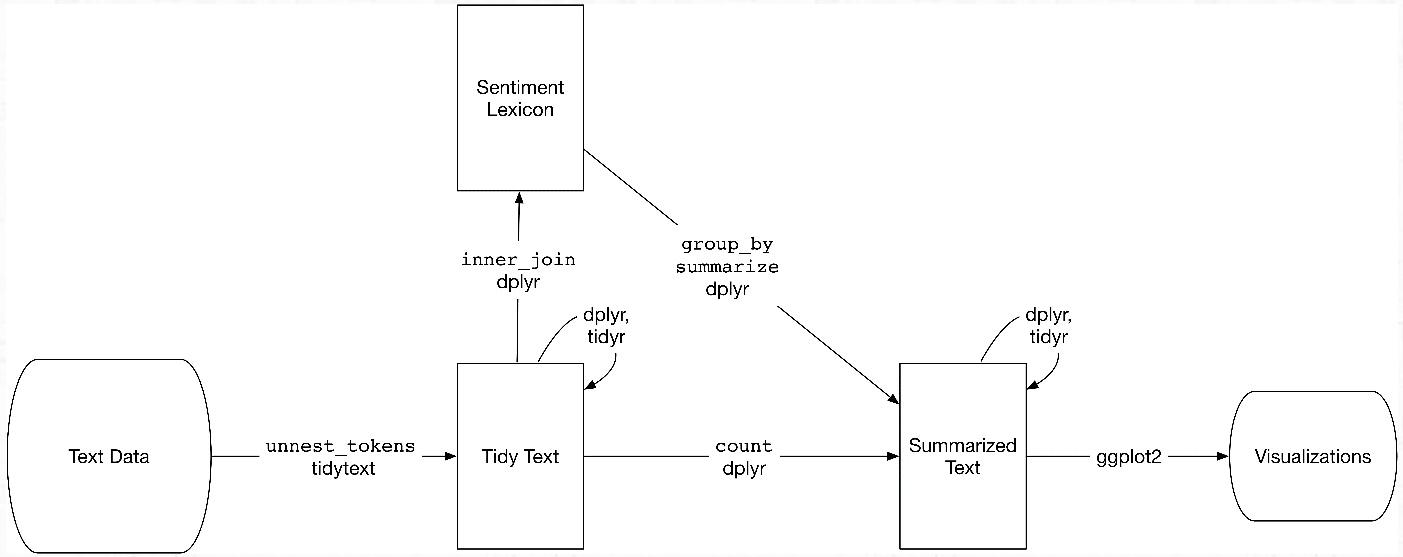
Visualization

```
library(ggplot2)

tidy_books %>%
  count(word, sort = TRUE) %>%
  filter(n > 600) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```

The most common words in Jane Austen's novels



Sentiment analysis with tidy data



A flowchart of a typical text analysis that uses tidytext for sentiment analysis.

The sentiments dataset

```r
library(tidytext)

sentiments
```

```
## # A tibble: 27,314 x 4
##    word        sentiment lexicon score
##    <chr>       <chr>     <chr>   <int>
##  1 abacus      trust     nrc        NA
##  2 abandon     fear      nrc        NA
##  3 abandon     negative  nrc        NA
##  4 abandon     sadness   nrc        NA
##  5 abandoned   anger     nrc        NA
##  6 abandoned   fear      nrc        NA
##  7 abandoned   negative  nrc        NA
##  8 abandoned   sadness   nrc        NA
##  9 abandonment anger     nrc        NA
## 10 abandonment fear      nrc        NA
## # ... with 27,304 more rows
```

Three different sentiment lexicon in sentiments data		
AFINN	Finn Årup Nielsen	score that runs between -5 and 5
bing	Bing Liu and collaborators	positive and negative
NRC	Saif Mohammad and Peter Turney	positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust

```r
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
```

```r
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
```

```r
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 analysis with inner join

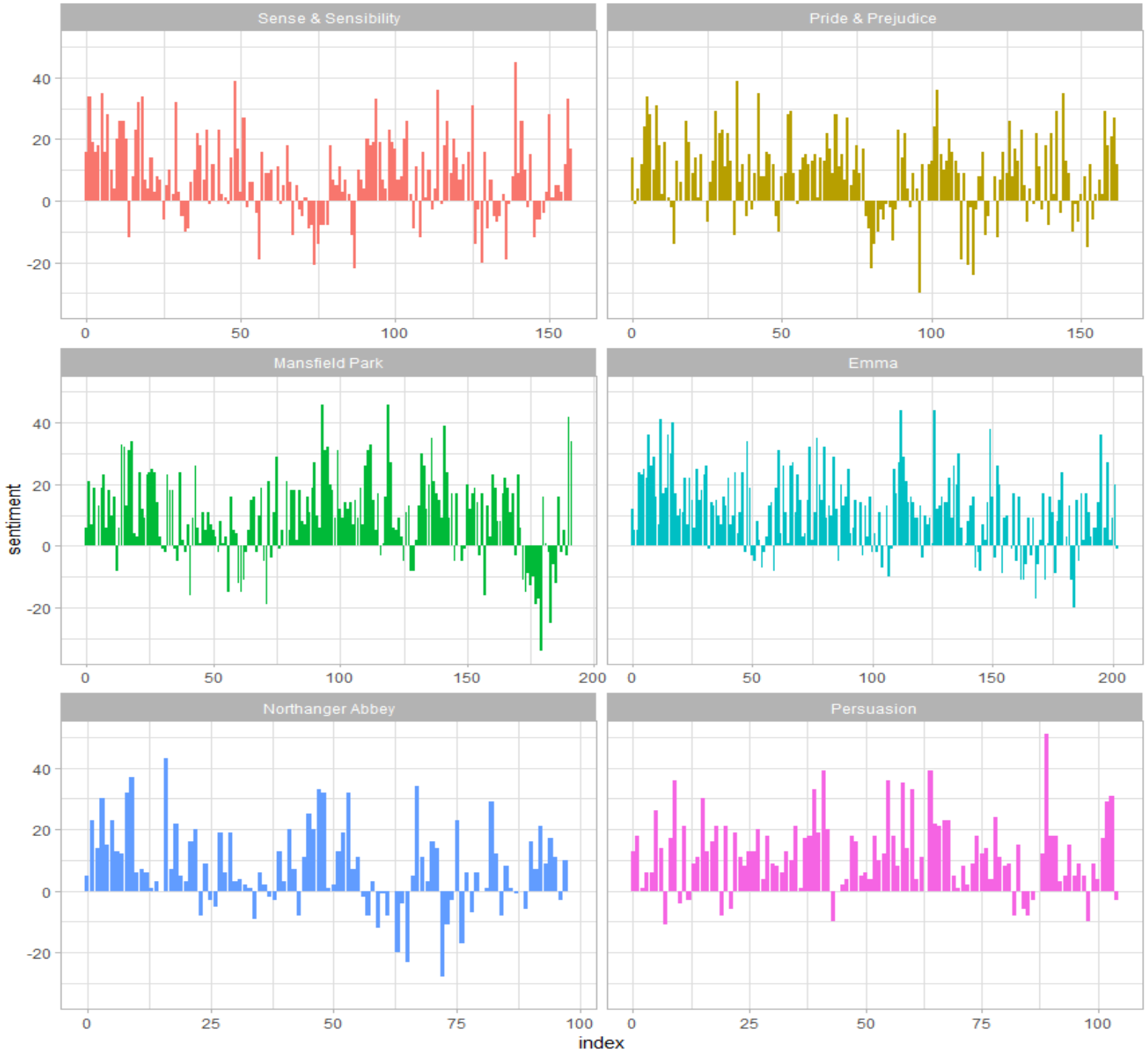
- What are the most common joy words in *Emma*?

```
library(janeaustenr)
library(dplyr)
library(stringr)

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)

nrc_joy <- get_sentiments("nrc") %>%
  filter(sentiment == "joy")
```

```
tidy_books %>%
  filter(book == "Emma") %>%
  inner_join(nrc_joy) %>%
  count(word, sort = TRUE)

## Joining, by = "word"

## # A tibble: 303 x 2
##    word        n
##    <chr>   <int>
##  1 good      359
##  2 young     192
##  3 friend    166
##  4 hope      143
##  5 happy     125
##  6 love      117
##  7 deal       92
##  8 found      92
##  9 present    89
## 10 kind       82
## # ... with 293 more rows
```

Show sentiment trend over section (80 lines)

```
library(tidyr)

jane_austen_sentiment <- tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(book, index = linenumber %/% 80, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)

library(ggplot2)

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


Sense & Sensibility
Pride & Prejudice
Mansfield Park
Emma
Northanger Abbey
Persuasion
sentiment
index
40
20
0
-20
0
50
100
150
200
25
75

Most common positive and negative words

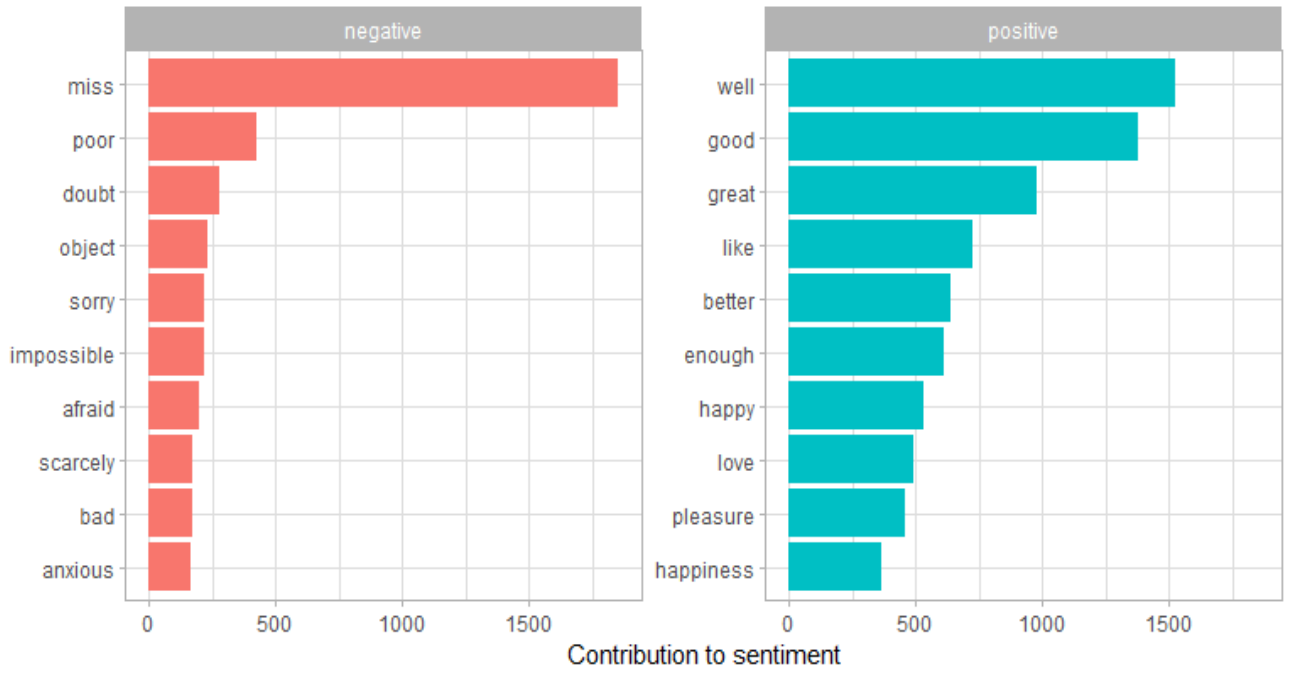
```
bing_word_counts <- tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

```
bing_word_counts
```

```
## # A tibble: 2,585 x 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
```

Most common positive and negative words

```
bing_word_counts %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Contribution to sentiment",
       x = NULL) +
  coord_flip()
```



Words that contribute to positive and negative sentiment in Jane Austen's novels

Add a new word to stopwords list

```
custom_stop_words <- bind_rows(data_frame(word = c("miss"),
                                          lexicon = c("custom")),
                               stop_words)

custom_stop_words
```

```
## # A tibble: 1,150 x 2
##    word        lexicon
##    <chr>       <chr>
##  1 miss        custom
##  2 a           SMART
##  3 a's         SMART
##  4 able        SMART
##  5 about       SMART
##  6 above       SMART
##  7 according   SMART
##  8 accordingly SMART
##  9 across      SMART
## 10 actually    SMART
## # ... with 1,140 more rows
```

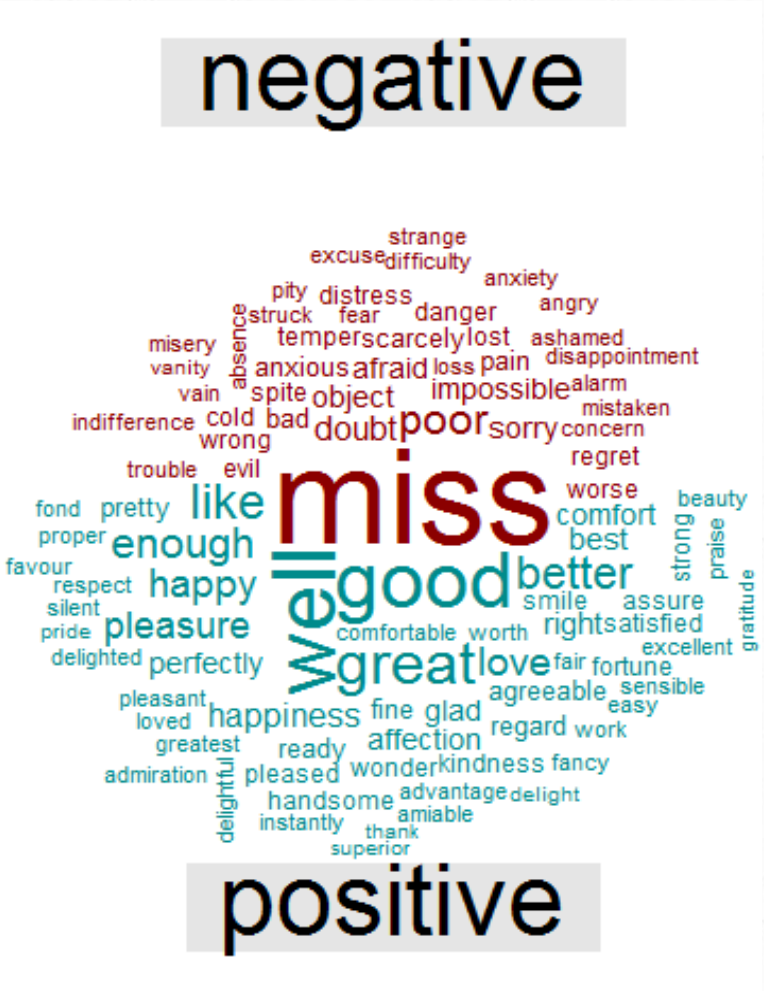
wordcloud

```
library(wordcloud)

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

The most common words in Jane Austen's novels

Comparison wordcloud

```
library(reshape2)

tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("darkred", "darkcyan"),
                   max.words = 100)
```



Most common positive and negative words in Jane Austen's novels

what are the most negative chapters in each of Jane Austen's novels?

```
bingnegative <- get_sentiments("bing") %>%
  filter(sentiment == "negative")

wordcounts <- tidy_books %>%
  group_by(book, chapter) %>%
  summarize(words = n())

tidy_books %>%
  semi_join(bingnegative) %>%
  group_by(book, chapter) %>%
  summarize(negativewords = n()) %>%
  left_join(wordcounts, by = c("book",
"chapter")) %>%
  mutate(ratio = negativewords/words) %>%
  filter(chapter != 0) %>%
  top_n(1) %>%
  ungroup()
```

```
## # A tibble: 6 x 5
##   book                chapter negativewords words  ratio
##   <fct>                 <int>         <int> <int>  <dbl>
## 1 Sense & Sensibility      43           161  3405 0.0473
## 2 Pride & Prejudice        34           111  2104 0.0528
## 3 Mansfield Park           46           173  3685 0.0469
## 4 Emma                     15           151  3340 0.0452
## 5 Northanger Abbey         21           149  2982 0.0500
## 6 Persuasion                4            62  1807 0.0343
```

TF/IDF
What are the most representative words in a document

- To quantify what a document is about.
- *term frequency* (tf):
  - how frequently a word occurs in a document
- *inverse document frequency* (idf)
  - How rare a word occurs across entire documents
  - The word uniqueness in a document
- tf-idf = tf * idf

$$tf(term) = \frac{n_{term\ occurrence\ in\ a\ document}}{n_{all\ words\ occurrence\ in\ a\ document}}$$

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

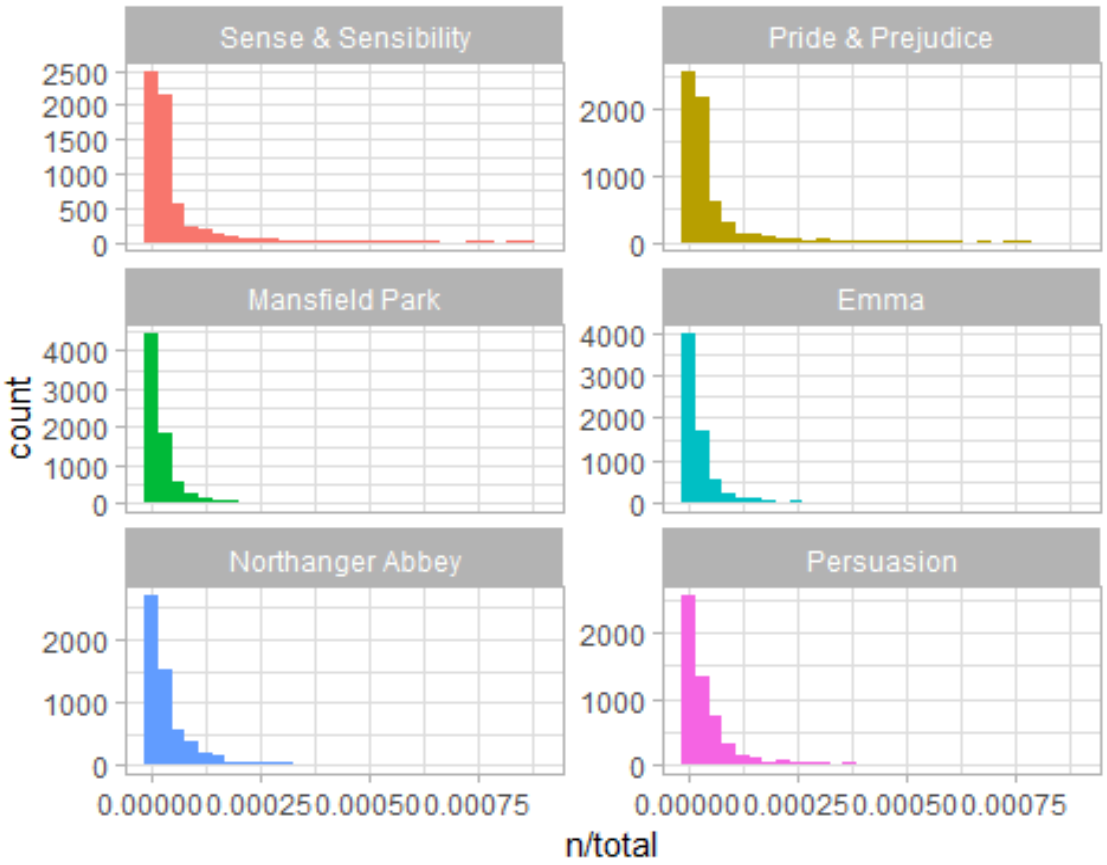
Counting word frequency of Jane Austen's works

```r
library(dplyr)
library(janeaustenr)
library(tidytext)

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)
```

```r
book_words

## # A tibble: 40,379 x 4
##    book                word      n  total
##    <fct>               <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
```

Counting word frequency of Jane Austen's works

```
ggplot(book_words, aes(n/total, fill = book)) +
  geom_histogram(show.legend = FALSE) +
  xlim(NA, 0.0009) +
  facet_wrap(~book, ncol = 2, scales = "free_y")
```



term_frequency

```r
freq_by_rank <- book_words %>%
  group_by(book) %>%
  mutate(rank = row_number(),
         `term frequency` = n/total)
```

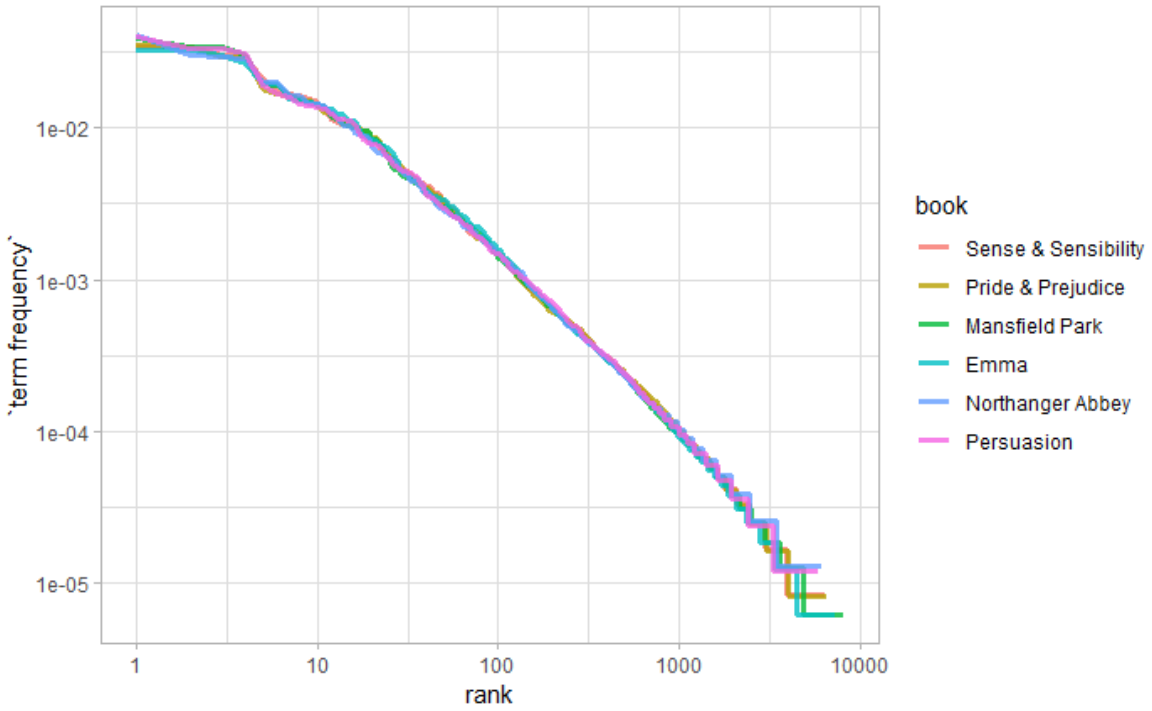
```r
freq_by_rank
```

```
## # A tibble: 40,379 x 6
## # Groups:   book [6]
##    book              word      n  total  rank `term frequency`
##    <fct>             <chr> <int>  <int> <int>            <dbl>
##  1 Mansfield Park    the    6206 160460     1           0.0387
##  2 Mansfield Park    to     5475 160460     2           0.0341
##  3 Mansfield Park    and    5438 160460     3           0.0339
##  4 Emma              to     5239 160996     1           0.0325
##  5 Emma              the    5201 160996     2           0.0323
##  6 Emma              and    4896 160996     3           0.0304
##  7 Mansfield Park    of     4778 160460     4           0.0298
##  8 Pride & Prejudice the    4331 122204     1           0.0354
##  9 Emma              of     4291 160996     4           0.0267
## 10 Pride & Prejudice to     4162 122204     2           0.0341
## # ... with 40,369 more rows
```

Zipf's law

Zipf's law states that the frequency that a word appears is inversely proportional to its rank.

```
freq_by_rank %>%
  ggplot(aes(rank, `term frequency`, color = book)) +
  geom_line(size = 1.1, alpha = 0.8) +
  scale_x_log10() +
  scale_y_log10()
```



bind_tf_idf

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

```
## # A tibble: 40,379 x 7
##    book              word      n  total     tf   idf tf_idf
##    <fct>             <chr> <int>  <int>  <dbl> <dbl>  <dbl>
##  1 Mansfield Park    the    6206 160460 0.0387     0      0
##  2 Mansfield Park    to     5475 160460 0.0341     0      0
##  3 Mansfield Park    and    5438 160460 0.0339     0      0
##  4 Emma              to     5239 160996 0.0325     0      0
##  5 Emma              the    5201 160996 0.0323     0      0
##  6 Emma              and    4896 160996 0.0304     0      0
##  7 Mansfield Park    of     4778 160460 0.0298     0      0
##  8 Pride & Prejudice the    4331 122204 0.0354     0      0
##  9 Emma              of     4291 160996 0.0267     0      0
## 10 Pride & Prejudice to     4162 122204 0.0341     0      0
## # ... with 40,369 more rows
```

bind_tf_idf

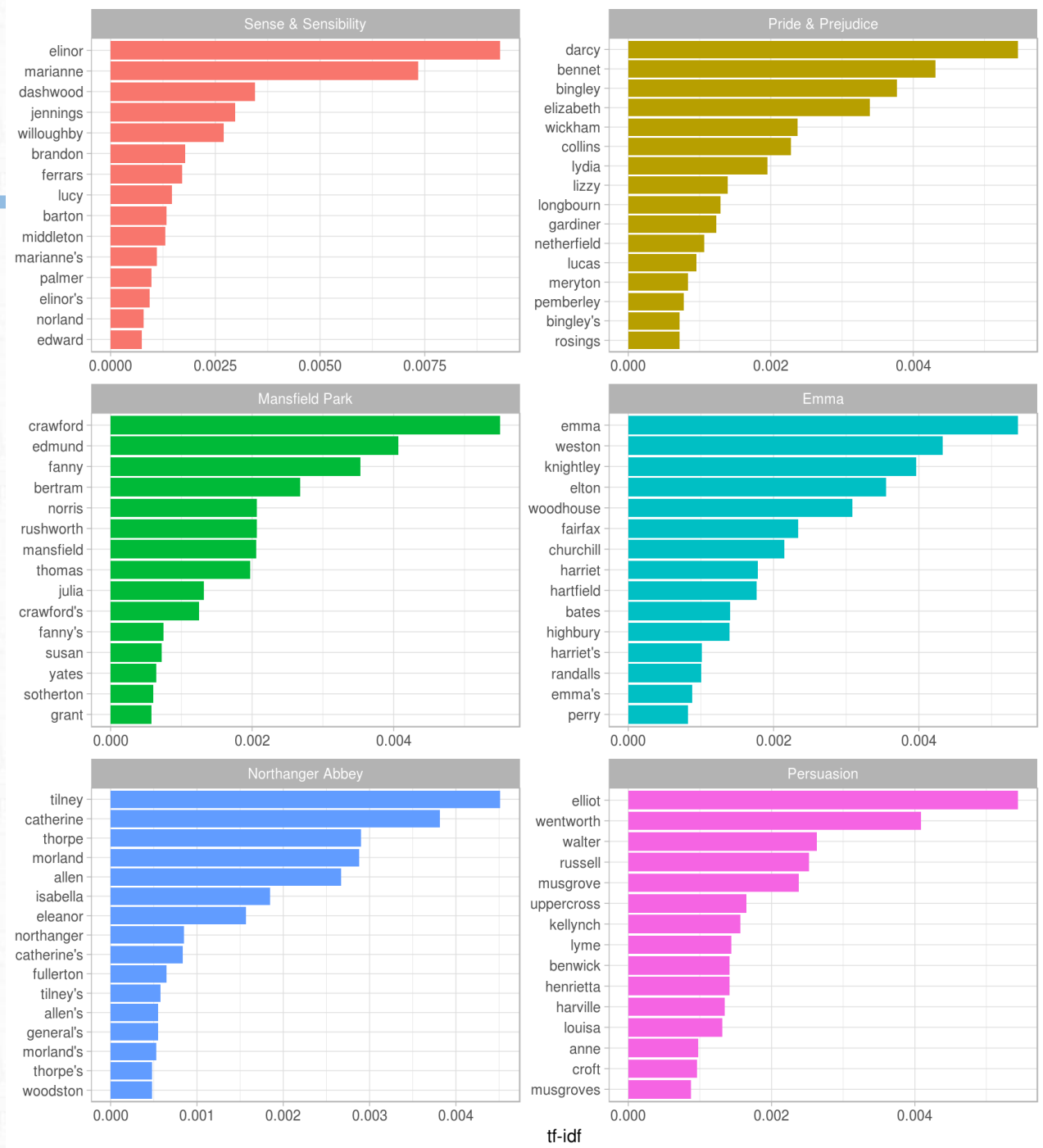
```
book_words %>%
  select(-total) %>%
  arrange(desc(tf_idf))
```

```
## # A tibble: 40,379 x 6
##    book                  word          n      tf   idf  tf_idf
##    <fct>                 <chr>     <int>   <dbl> <dbl>   <dbl>
##  1 Sense & Sensibility elinor       623 0.00519  1.79 0.00931
##  2 Sense & Sensibility marianne     492 0.00410  1.79 0.00735
##  3 Mansfield Park      crawford     493 0.00307  1.79 0.00551
##  4 Pride & Prejudice   darcy        373 0.00305  1.79 0.00547
##  5 Persuasion          elliot       254 0.00304  1.79 0.00544
##  6 Emma                emma         786 0.00488  1.10 0.00536
##  7 Northanger Abbey    tilney       196 0.00252  1.79 0.00452
##  8 Emma                weston       389 0.00242  1.79 0.00433
##  9 Pride & Prejudice   bennet       294 0.00241  1.79 0.00431
## 10 Persuasion          wentworth    191 0.00228  1.79 0.00409
## # ... with 40,369 more rows
```

bind_tf_idf

```
book_words %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  group_by(book) %>%
  top_n(15) %>%
  ungroup %>%
  ggplot(aes(word, tf_idf, fill = book)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~book, ncol = 2, scales = "free") +
  coord_flip()
```

bind_tf_idf


Sense & Sensibility
elinor
marianne
dashwood
jennings
willoughby
brandon
ferrars
lucy
barton
middleton
marianne's
palmer
elinor's
norland
edward
0.0000
0.0025
0.0050
0.0075
Pride & Prejudice
darcy
bennet
bingley
elizabeth
wickham
collins
lydia
lizzy
longbourn
gardiner
netherfield
lucas
meryton
pemberley
bingley's
rosings
0.000
0.002
0.004
Mansfield Park
crawford
edmund
fanny
bertram
norris
rushworth
mansfield
thomas
julia
crawford's
fanny's
susan
yates
sotherton
grant
0.000
0.002
0.004
Emma
emma
weston
knightley
elton
woodhouse
fairfax
churchill
harriet
hartfield
bates
highbury
harriet's
randalls
emma's
perry
0.000
0.002
0.004
Northanger Abbey
tilney
catherine
thorpe
morland
allen
isabella
eleanor
northanger
catherine's
fullerton
tilney's
allen's
general's
morland's
thorpe's
woodston
0.000
0.001
0.002
0.003
0.004
Persuasion
elliot
wentworth
walter
russell
musgrove
uppercross
kellynch
lyme
benwick
henrietta
harville
louisa
anne
croft
musgroves
0.000
0.002
0.004
tf-idf