Text Mining

Monica Buczynski

October 09, 2020

Note: The purpose of this document is to showcase a sample of skills that I learned in *Text Mining with R: A Tidy Approach* by Julia Silge and David Robinson. Some scripts were taken from https://www.tidytextmining.com/s.html. The code for each exercise was studied carefully for understanding and then was retyped manually into R to maximize the learning experience; however, many of the scripts were altered for further analysis and presentation aesthetics. Additionally, I added my own code for further analysis and my own curiosity.

Skills that I focused on included:

- The tidy text format
- Sentiment analysis with tidy data
- Analyzing word and document frequency: tf-idf
- Relationships between words: n-grams and correlations
- Converting to and from non-tidy formats

1 The tidy text format

1.2 The unnest_tokens function

```
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"
```

```
# save as df

library(dplyr)
text_df <-  tibble(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
```

```
# tokenization

library(tidytext)

text_df %>%
  unnest_tokens(word, text, to_lower = FALSE)
```

```
## # 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
```

```
# do not use *to_lower = FALSE* to convert the tokens to lowercase

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
```

1.3 Tidying the works of Jane Austen

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

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
# restructure df in the one-token-per-row format with the unnest_tokens()


tidy_books <- original_books %>%
  unnest_tokens(words, text)

tidy_books
```

```
## # A tibble: 725,055 x 4
##    book                linenumber chapter words
##    <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
```

```r
# add stop words - words that are not usefull to us for analysis

stop_words
```

```
## # A tibble: 1,149 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 1,139 more rows
```

```
# Practice adding a new row
newRow <- data.frame(word="AAAAA",lexicon = "SMART" )
stop_words <- rbind(stop_words, newRow)

tidy_books <- tidy_books %>%
  rename("word" = "words") %>%  # rename column name "words" to "word" in tidy_books
                                # so that there is a key between tidy_books and
                                # stop_words for anti_join()
  anti_join(stop_words, by = "word") # drops all observations in x that have a match in y

tidy_books
```

```
## # A tibble: 217,609 x 4
##    book                linenumber chapter word
##    <fct>                    <int>   <int> <chr>
##  1 Sense & Sensibility          1       0 sense
##  2 Sense & Sensibility          1       0 sensibility
##  3 Sense & Sensibility          3       0 jane
##  4 Sense & Sensibility          3       0 austen
##  5 Sense & Sensibility          5       0 1811
##  6 Sense & Sensibility         10       1 chapter
##  7 Sense & Sensibility         10       1 1
##  8 Sense & Sensibility         13       1 family
##  9 Sense & Sensibility         13       1 dashwood
## 10 Sense & Sensibility         13       1 settled
## # ... with 217,599 more rows
```

```
# use count() to find the most common 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
```

1.4 The gutenbergr package

```r
BooksOz <- gutenberg_metadata[grep("Oz", gutenberg_metadata$title), ]
BooksOz
```

```
## # A tibble: 49 x 8
##    gutenberg_id title  author  gutenberg_autho~ language gutenberg_books~ rights
##           <int> <chr>  <chr>              <int> <chr>    <chr>            <chr>
##  1           54 The M~ Baum, ~               42 en       Children's Lite~ Publi~
##  2           55 The W~ Baum, ~               42 en       Children's Lite~ Publi~
##  3          419 The M~ Baum, ~               42 en       Children's Lite~ Publi~
##  4          420 Dorot~ Baum, ~               42 en       Children's Lite~ Publi~
##  5          485 The R~ Baum, ~               42 en       Children's Lite~ Publi~
##  6          486 Ozma ~ Baum, ~               42 en       Fantasy/Childre~ Publi~
##  7          517 The E~ Baum, ~               42 en       Children's Lite~ Publi~
##  8          955 The P~ Baum, ~               42 en       Children's Lite~ Publi~
##  9          956 Tik-T~ Baum, ~               42 en       Children's Lite~ Publi~
## 10          957 The S~ Baum, ~               42 en       Children's Lite~ Publi~
## # ... with 39 more rows, and 1 more variable: has_text <lgl>
```

```r
#gutenberg_metadata %>%
 #filter(title == "Oz")

WWOz <- gutenberg_download(55)

WWOz
```

```
## # A tibble: 4,750 x 2
##    gutenberg_id text
##           <int> <chr>
##  1           55 "[Illustration]"
##  2           55 ""
##  3           55 ""
##  4           55 ""
##  5           55 ""
##  6           55 "The Wonderful Wizard of Oz"
##  7           55 ""
##  8           55 "by L. Frank Baum"
##  9           55 ""
## 10           55 ""
## # ... with 4,740 more rows
```

```r
tidy_books_Oz <- WWOz %>%
  unnest_tokens(words, text)

tidy_books_Oz
```

```
## # A tibble: 39,765 x 2
##    gutenberg_id words
##           <int> <chr>
##  1           55 illustration
##  2           55 the
##  3           55 wonderful
##  4           55 wizard
##  5           55 of
##  6           55 oz
##  7           55 by
##  8           55 l
##  9           55 frank
## 10           55 baum
## # ... with 39,755 more rows
```

```
tidy_books_Oz <- tidy_books_Oz %>%
  rename("word" = "words") %>%
  # rename column name "words" to "word" in tidy_books so that there is a key
   # between tidy_books and stop_words for anti_join()
anti_join(stop_words, by = "word")
  # drops all observations in x that have a match in y

tidy_books_Oz %>%
  count(word, sort = TRUE) %>%
  summary(tidy_books_Oz$n)
```

```
##      word                 n
##  Length:2533        Min.   :  1.000
##  Class :character   1st Qu.:  1.000
##  Mode  :character   Median :  2.000
##                     Mean   :  4.941
##                     3rd Qu.:  4.000
##                     Max.   :347.000
```

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

```
## # A tibble: 2,533 x 2
##    word          n
##    <chr>     <int>
##  1 dorothy     347
##  2 scarecrow   219
##  3 woodman     176
##  4 lion        173
##  5 oz          164
##  6 tin         140
##  7 witch       125
##  8 green       104
##  9 girl         93
## 10 head         90
## # ... with 2,523 more rows
```

```
tidy_books_Oz %>%
  count(word, sort = TRUE) %>%
  dplyr::filter(word == "munchkins")
```

```
## # A tibble: 1 x 2
##   word          n
##   <chr>     <int>
## 1 munchkins    21
```

```
tidy_books_Oz %>%
  count(word, sort = TRUE) %>%
  dplyr::filter(word == "monkeys")
```

```
## # A tibble: 1 x 2
##   word        n
##   <chr>   <int>
## 1 monkeys    44
```

```r
library("RColorBrewer")
display.brewer.pal(n = 8, name = 'Dark2')
```

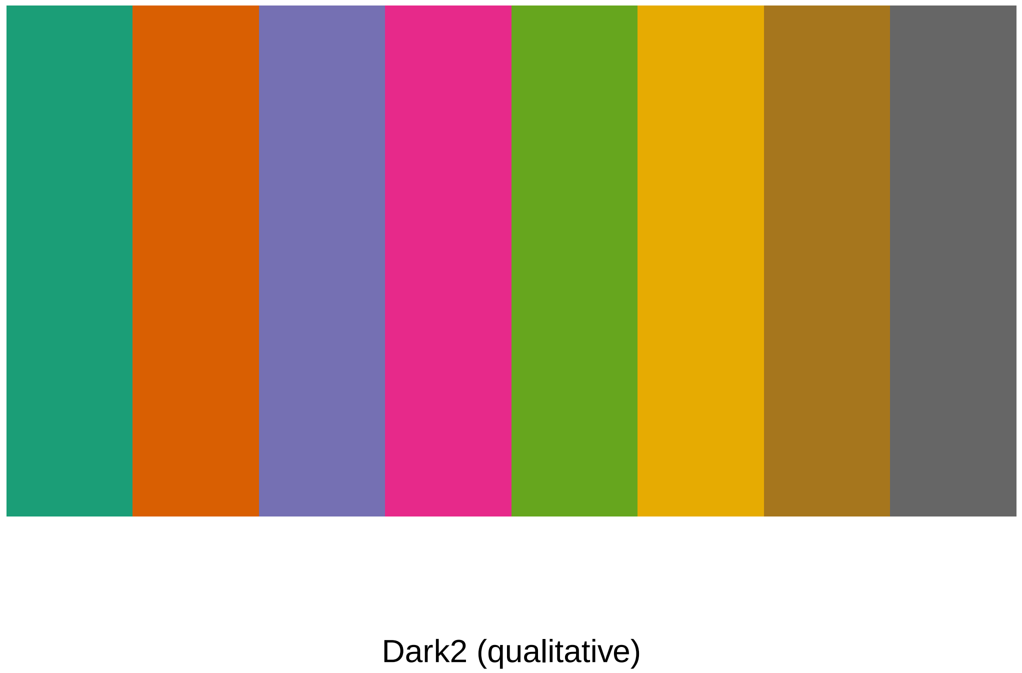


Figure 1: Words mentioned more than 50 times in The Wonderful Wizard of Oz

```r
brewer.pal(n = 8, name = 'Dark2')
```

```
## [1] "#1B9E77" "#D95F02" "#7570B3" "#E7298A" "#66A61E" "#E6AB02" "#A6761D"
## [8] "#666666"
```

```r
tidy_books_Oz %>%
  count(word, sort= TRUE) %>%
  dplyr::filter(n >50) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col(fill= "#1B9E77"
 ) +
  labs(y = "frequency") +
  coord_flip()
```

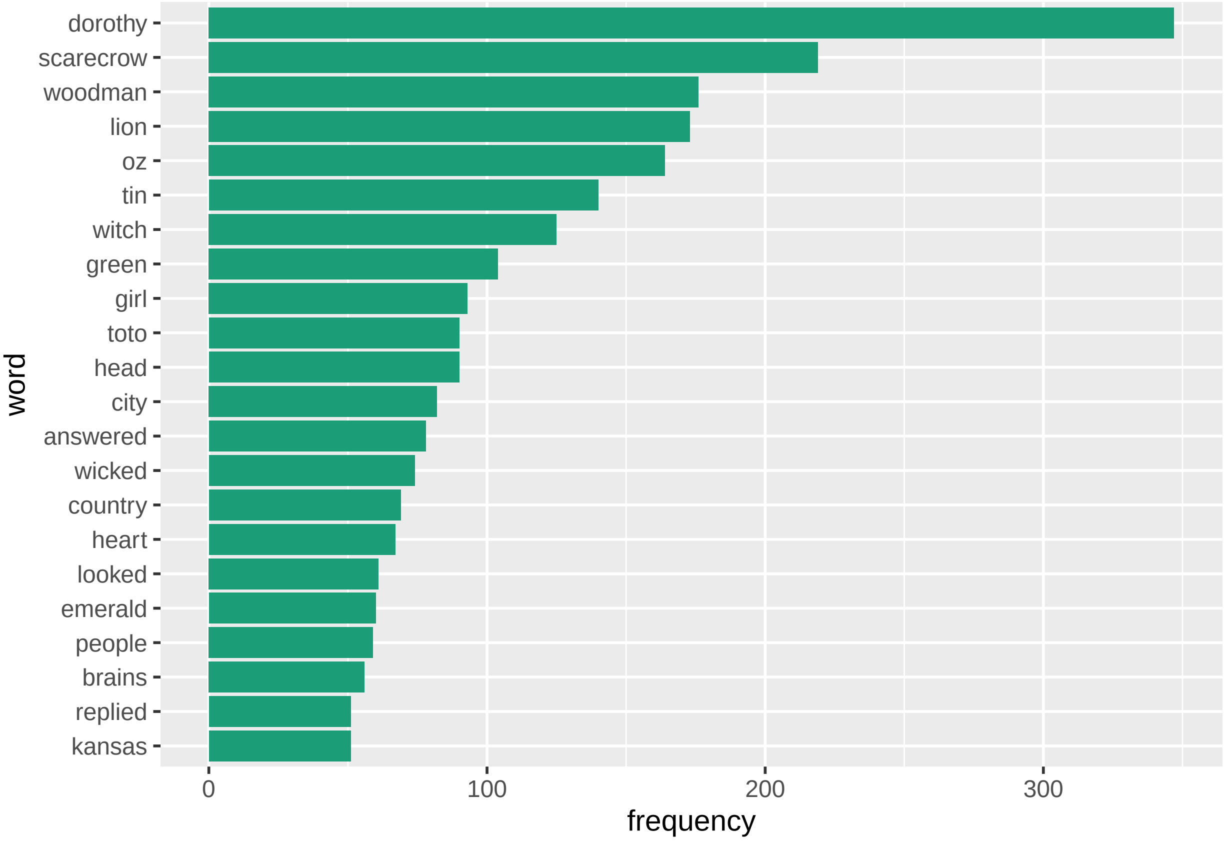


Figure 2: Words mentioned more than 50 times in The Wonderful Wizard of Oz

2 Sentiment analysis with tidy data

2.1 The sentiments dataset

```
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("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("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
```

2.2 Sentiment analysis with inner join

```
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)

# What are the most common joy words in Emma?

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

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

```
## # 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
```

```
# examine how sentiment changes throughout each novel

jane_austen_sentiment <- tidy_books %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  count(book, index = linenumber %/% 80, sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(sentiment = positive - negative)
```

```
ggplot(jane_austen_sentiment, aes(index, sentiment, fill = book)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~book, ncol = 2, scales = "free_x") +
  scale_fill_brewer(palette = "Dark2")
```

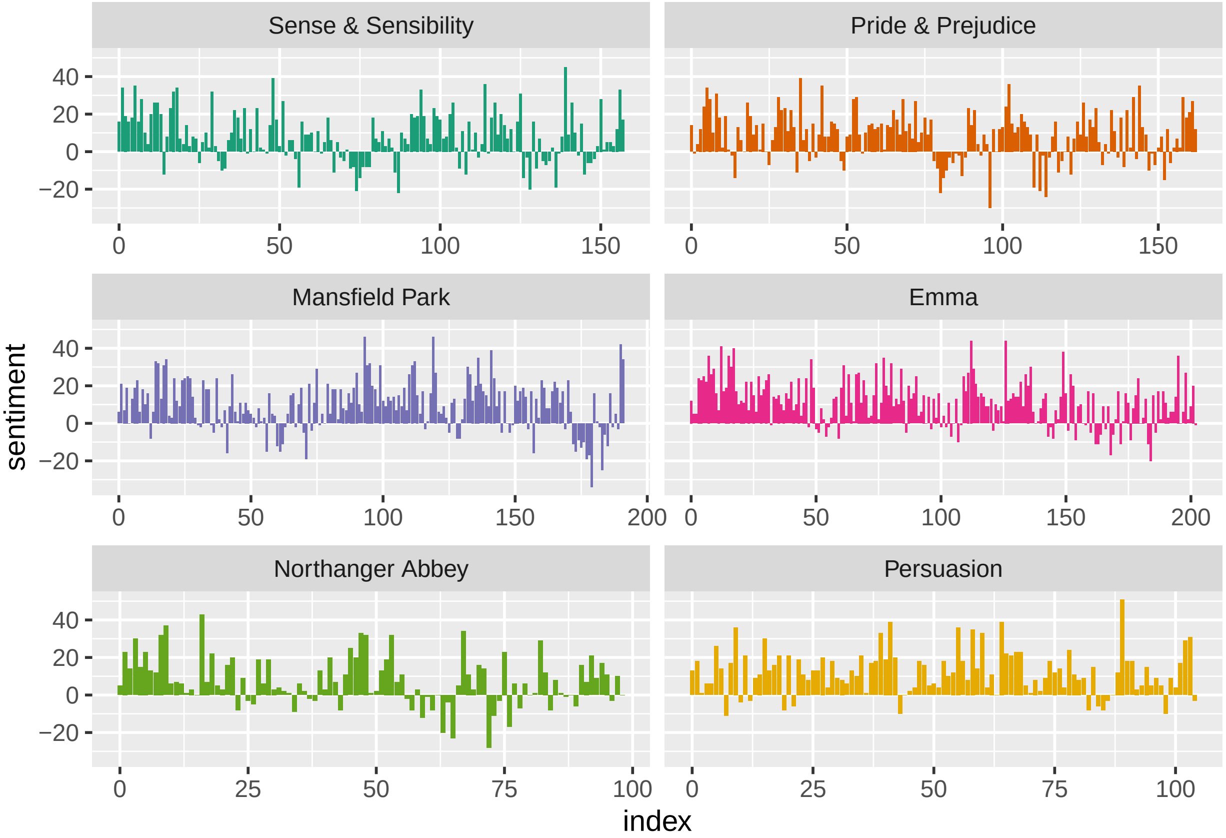


Figure 3: Sentiment through the narratives of Jane Austen's novels

2.3 Comparing the three sentiment dictionaries

```r
# filtering to one novel that I am interested in

pride_prejudice <- tidy_books %>%
  filter(book == "Pride & Prejudice")

pride_prejudice
```

```
## # A tibble: 122,204 x 4
##    book              linenumber chapter word
##    <fct>                  <int>   <int> <chr>
##  1 Pride & Prejudice          1       0 pride
##  2 Pride & Prejudice          1       0 and
##  3 Pride & Prejudice          1       0 prejudice
##  4 Pride & Prejudice          3       0 by
##  5 Pride & Prejudice          3       0 jane
##  6 Pride & Prejudice          3       0 austen
##  7 Pride & Prejudice          7       1 chapter
##  8 Pride & Prejudice          7       1 1
##  9 Pride & Prejudice         10       1 it
## 10 Pride & Prejudice         10       1 is
## # ... with 122,194 more rows
```

```r
# need two different patterns because AFINN has a numeric measure while bing and nrc are binary.

afinn <-  pride_prejudice %>%
  inner_join(get_sentiments("afinn"), by = "word") %>%
  group_by(index = linenumber %/% 80) %>%
  summarise(sentiment = sum(value)) %>%
  mutate(method = "AFINN")

bing_and_nrc <- bind_rows(pride_prejudice %>%
                            inner_join(get_sentiments("bing"), by = "word") %>%
                            mutate(method = "Bind et al."),
                          pride_prejudice %>%
                            inner_join(get_sentiments("nrc"), by = "word") %>%
                                         filter(sentiment %in% c("positive", "negative")) %>%
                            mutate(method = "NRC")) %>%
  count(method, index = linenumber%/% 80, sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(sentiment = positive - negative)
```

```
bind_rows(afinn, bing_and_nrc) %>%
  ggplot(aes(x = index, y = sentiment, fill = method)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~method, ncol = 1, scales = "free_y")  +
  scale_fill_brewer(palette = "Dark2")
```

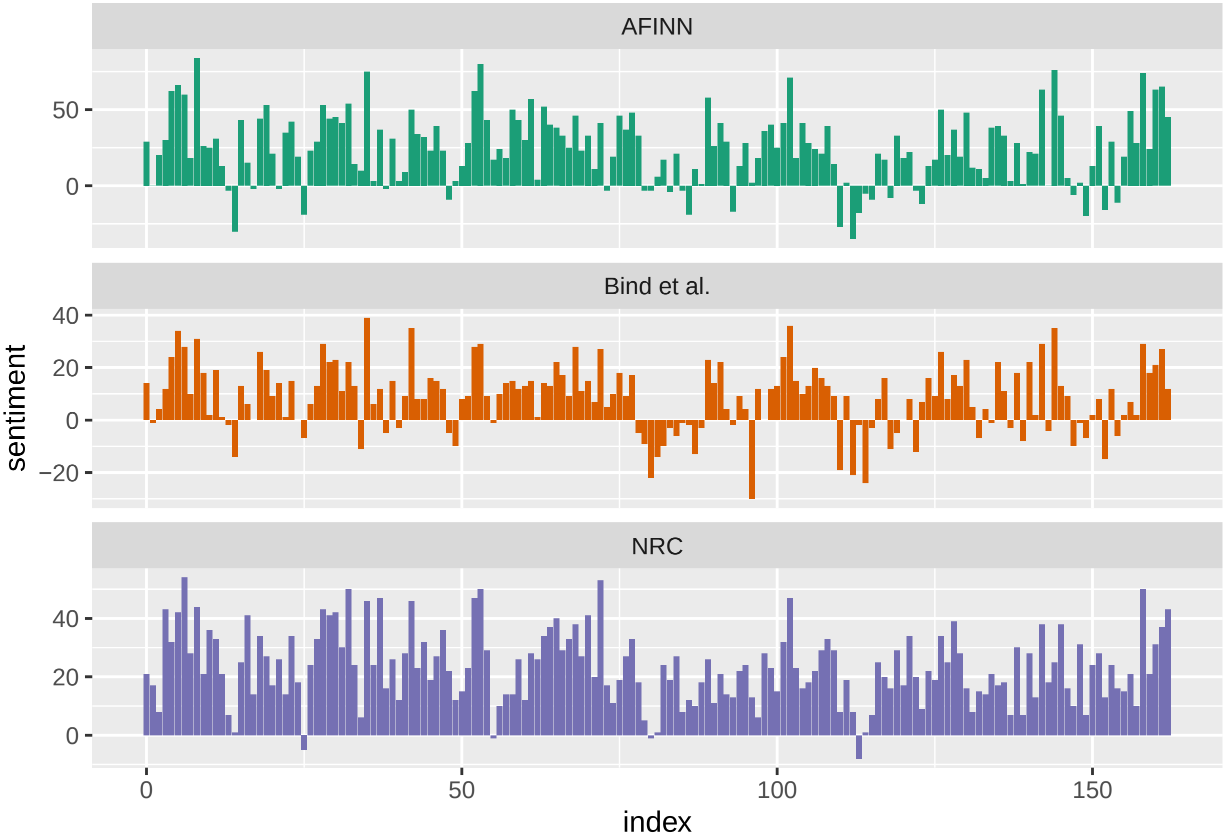


Figure 4: Comparing three sentiment lexicons using Pride and Prejudice

Why is, for example, the result for the NRC lexicon biased so high in sentiment compared to the Bing et al. result? Both lexicons have more negative than positive words, but the ratio of negative to positive words is higher in the Bing lexicon than the NRC lexicon.

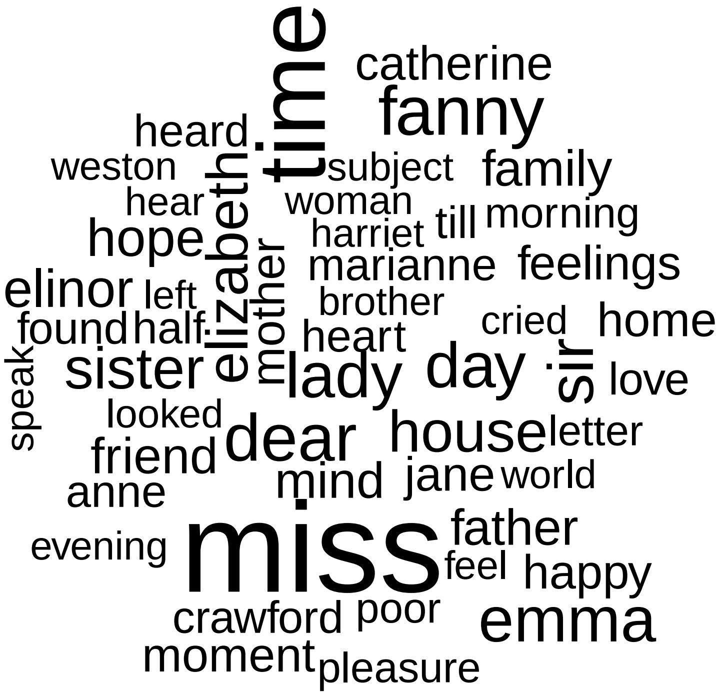
```r
nrc <- get_sentiments("nrc") %>%
  filter(sentiment %in% c("positive", "negative")) %>%
  count(sentiment) %>%
  mutate(proportion = n/sum(n),
         lexicon = "NRC")

bing <- get_sentiments("bing") %>%
  count(sentiment) %>%
  mutate(proportion = n/sum(n),
         lexicon = "bing")

full_join(bing, nrc)
```

```
## # A tibble: 4 x 4
##   sentiment     n proportion lexicon
##   <chr>     <int>      <dbl> <chr>
## 1 negative   4781      0.705 bing
## 2 positive   2005      0.295 bing
## 3 negative   3324      0.590 NRC
## 4 positive   2312      0.410 NRC
```

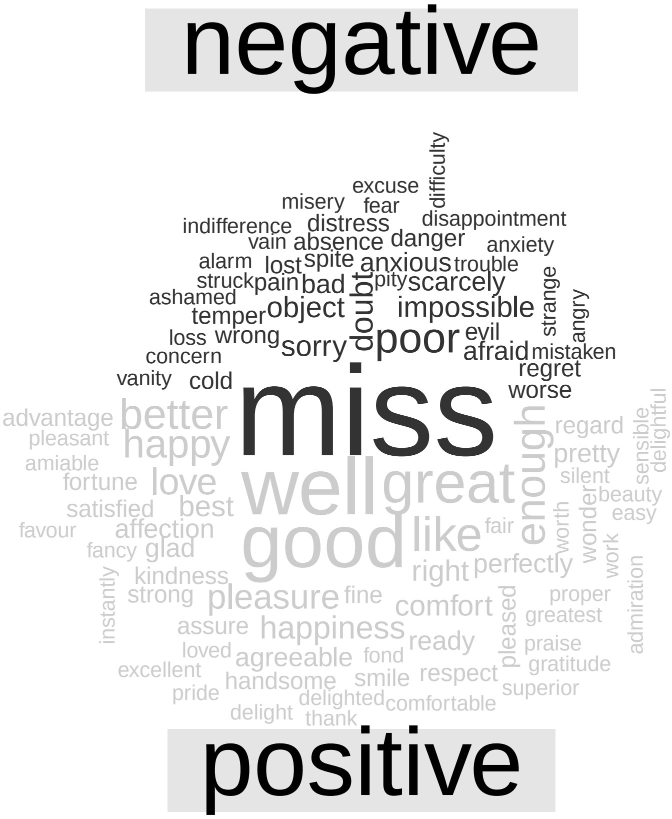
2.5 Wordclouds

Illustrate the most common words in Jane Austen's works in a world cloud

```
tidy_books %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 50))
```



```
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("gray20", "gray80"),
                   max.words = 100)
```



2.6 Looking at units beyond just words

```
# tokenizing at the sentence level

PandP_sentences <- tibble(text = prideprejudice) %>%
  unnest_tokens(sentence, text, token = "sentences")

# look at sentence #2

PandP_sentences$sentence[2]
```

```
## [1] "by jane austen"
```

```
# tokenizing at the chapter level

austen_chapters <- austen_books() %>%
  group_by(book) %>%
  unnest_tokens(chapter, text, token = "regex",
              pattern = "Chapter|CHAPTER [\\dIVXLC]") %>% ungroup()

austen_chapters %>%
  group_by(book) %>%
  summarise(chapters = n())
```

```
## # A tibble: 6 x 2
##   book                chapters
##   <fct>                  <int>
## 1 Sense & Sensibility       51
## 2 Pride & Prejudice         62
## 3 Mansfield Park            49
## 4 Emma                      56
## 5 Northanger Abbey          32
## 6 Persuasion                25
```

```
## 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
```

3 Analyzing word and document frequency: tf-idf

The statistic tf-idf is intended to measure how important a word is to a document in a collection (or corpus) of documents, for example, to one novel in a collection of novels or to one website in a collection of websites.

For a term t in a document d, the weight Wt,d of term t in document d is given by:

$$tf - idfWt, d = TFt, dlog(N/DFt)$$

Where:

TFt,d is the number of occurrences of t in document d. DFt is the number of documents containing the term t. N is the total number of documents in the corpus.

TF –> term frequency IDF –> inverse document frequency - decreases the weight for commonly used words and increases the weight for words that are not used very much in a collection of documents

The higher the TF*IDF score (weight), the rarer the term and vice versa

3.1 Term frequency in Jane Austen's novels

What are the most commonly used words in Jane Austen's novels?

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

total_words <- book_words %>%
  group_by(book) %>%
  summarize(total = sum(n))

book_words <- full_join(book_words, total_words)

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
## # ... with 40,369 more rows
```

```r
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") +
  scale_fill_brewer(palette = "Dark2")
```

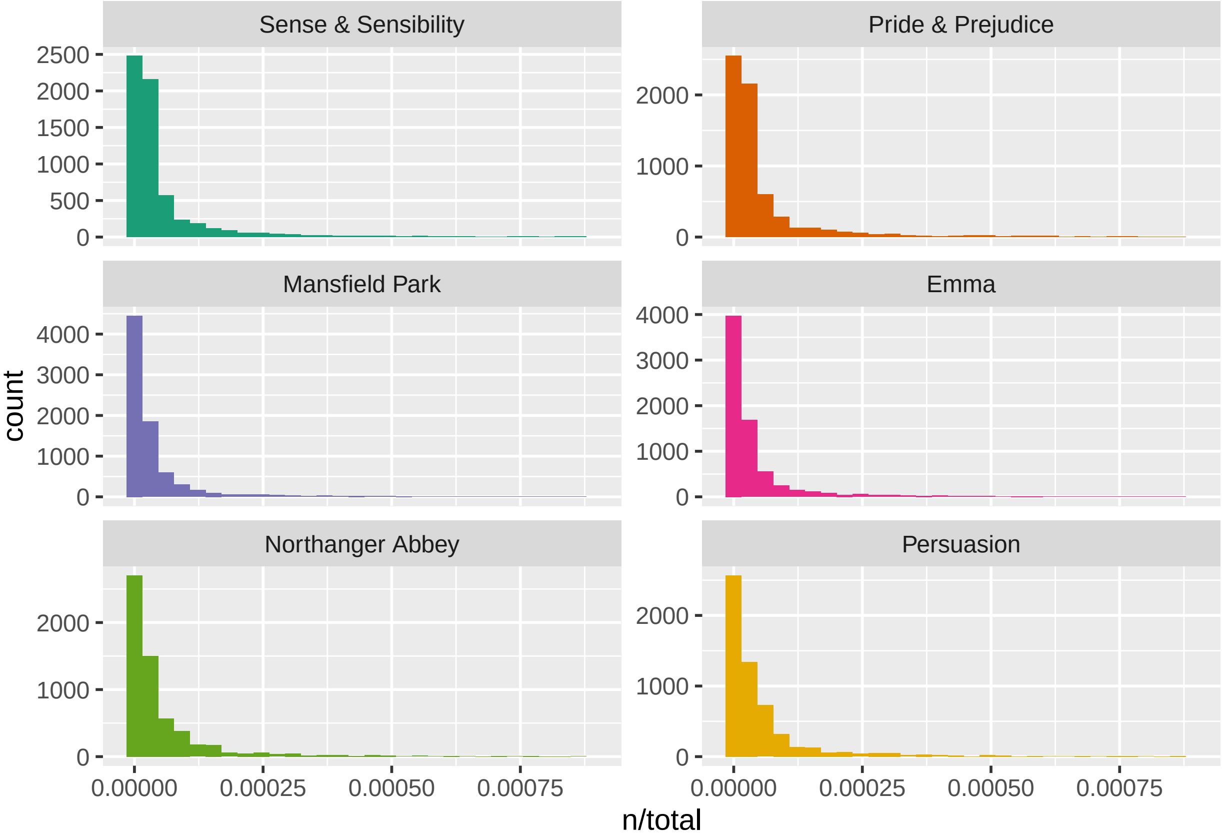


Figure 5: Term Frequency Distribution in Jane Austen's Novels

```r
# Observation: many words that occur rarely and fewer words that occur frequently
```

3.2 Zipf's law

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

```
freq_by_rank <- book_words %>%
  group_by(book) %>%
  mutate(rank = row_number(),
         "term_frequency" = n/total)

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
```

```
freq_by_rank %>%
  ggplot(aes(x = rank, y = term_frequency, color = book)) + geom_line(size =1.1, alpha = 0.8, show.legen
  scale_y_log10() +
  labs(y = "term frequency")
```

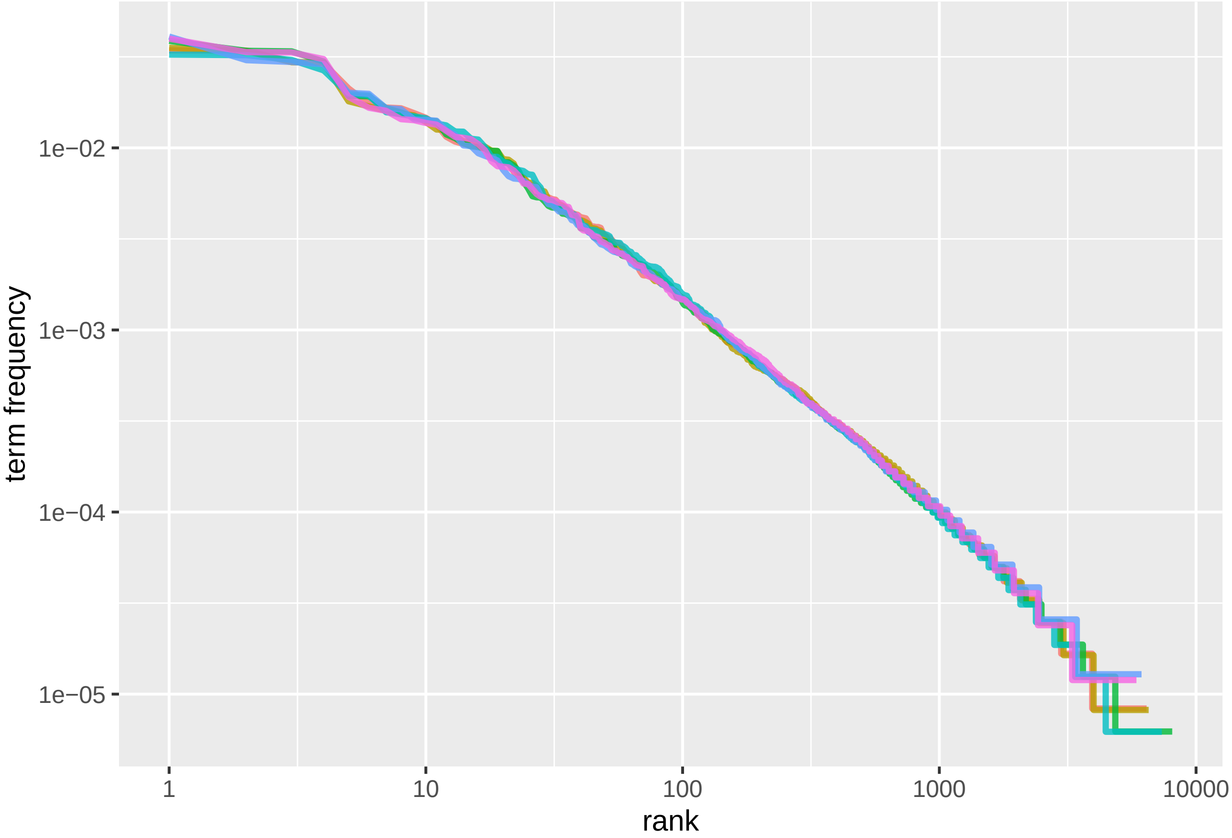


Figure 6: Zipf's law for Jane Austen's novels

```
# rank column tells the rank of each word within the frequency table
```

3.3 The bind_tf_idf function

```
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
```

```
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
```

```
book_words %>%
arrange(desc(tf_idf)) %>%
mutate(word = factor(word, levels = rev(unique(word)))) %>%
group_by(book) %>%
top_n(15) %>%
ungroup() %>%
ggplot(aes(x= word, y= tf_idf, fill = book)) +
geom_col(show.legend = FALSE) +
labs(x = NULL, y = "tf-idf") +
facet_wrap(~book, ncol = 2, scales = "free") +
coord_flip() +
  scale_fill_brewer(palette = "Dark2")
```

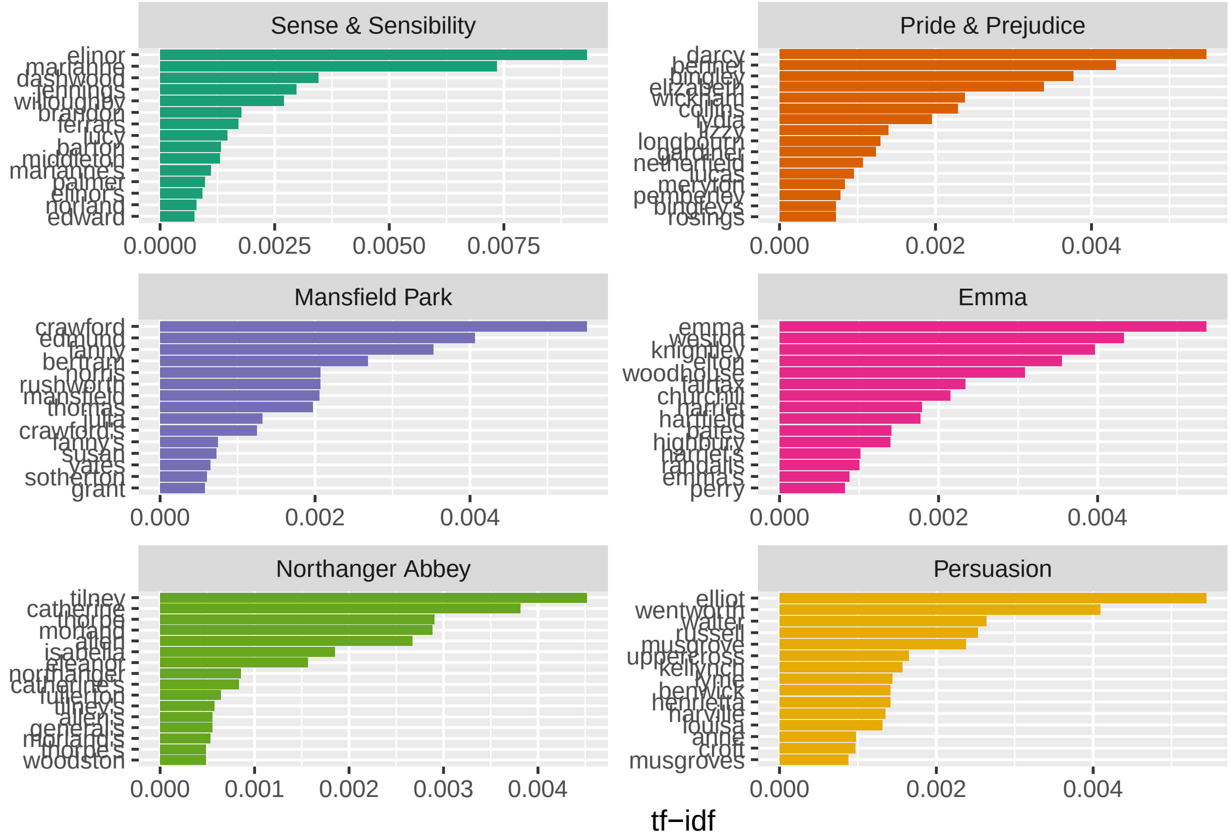


Figure 7: Highest tf-idf words in each of Jane Austen's Novels

4.2 Counting and correlating pairs of words with the widyr package

4.2.1 Counting and correlating among sections

The widyr package makes operations such as computing counts and correlations easy, by simplifying the pattern of "widen data, perform an operation, then re-tidy data". The book "Pride and Prejudice" divided into 10-line sections, as we did (with larger sections) for sentiment analysis in Chapter 2. We may be interested in what words tend to appear within the same section.

```
austen_section_words <- austen_books() %>%
filter(book ==  "Pride & Prejudice" ) %>%
mutate(section = row_number() %/% 10) %>%
filter(section > 0) %>%
unnest_tokens(word, text) %>%
filter(!word %in% stop_words$word)

austen_section_words
```

```
## # A tibble: 37,240 x 3
##    book              section word
##    <fct>               <dbl> <chr>
##  1 Pride & Prejudice       1 truth
##  2 Pride & Prejudice       1 universally
##  3 Pride & Prejudice       1 acknowledged
##  4 Pride & Prejudice       1 single
##  5 Pride & Prejudice       1 possession
##  6 Pride & Prejudice       1 fortune
##  7 Pride & Prejudice       1 wife
##  8 Pride & Prejudice       1 feelings
##  9 Pride & Prejudice       1 views
## 10 Pride & Prejudice       1 entering
## # ... with 37,230 more rows
```

```
# count words co-occuring within sections

word_pairs <- austen_section_words %>% pairwise_count(word, section, sort = TRUE)

word_pairs
```

```
## # A tibble: 796,008 x 3
##    item1     item2         n
##    <chr>     <chr>     <dbl>
##  1 darcy     elizabeth   144
##  2 elizabeth darcy       144
##  3 miss      elizabeth   110
##  4 elizabeth miss        110
##  5 elizabeth jane        106
##  6 jane      elizabeth   106
##  7 miss      darcy        92
##  8 darcy     miss         92
##  9 elizabeth bingley      91
## 10 bingley   elizabeth    91
## # ... with 795,998 more rows
```

```
# the most common pair of words in a section is "Elizabeth" and "Darcy"
# (the two main characters)
```

```
word_pairs %>%
  filter(item1 == "darcy")
```

```
## # A tibble: 2,930 x 3
##    item1 item2         n
##    <chr> <chr>     <dbl>
##  1 darcy elizabeth   144
##  2 darcy miss         92
##  3 darcy bingley      86
##  4 darcy jane         46
##  5 darcy bennet       45
##  6 darcy sister       45
##  7 darcy time         41
##  8 darcy lady         38
##  9 darcy friend       37
## 10 darcy wickham      37
## # ... with 2,920 more rows
```

4.2.2 Pairwise coreelations

Examine among words, which indicates how often they appear together relative to how often they appear separately.

```
#filter for common words first
word_cors <- austen_section_words %>%
  group_by(word) %>%
  filter(n() >= 20) %>%
  pairwise_cor(word, section) %>%
  arrange(correlation)

word_cors
```

```
## # A tibble: 154,842 x 3
##    item1     item2     correlation
##    <chr>     <chr>           <dbl>
##  1 darcy     lydia          -0.122
##  2 lydia     darcy          -0.122
##  3 collins   bingley        -0.122
##  4 bingley   collins        -0.122
##  5 jane      lady           -0.111
##  6 lady      jane           -0.111
##  7 collins   darcy          -0.100
##  8 darcy     collins        -0.100
##  9 longbourn darcy          -0.0946
## 10 darcy     longbourn      -0.0946
## # ... with 154,832 more rows
```

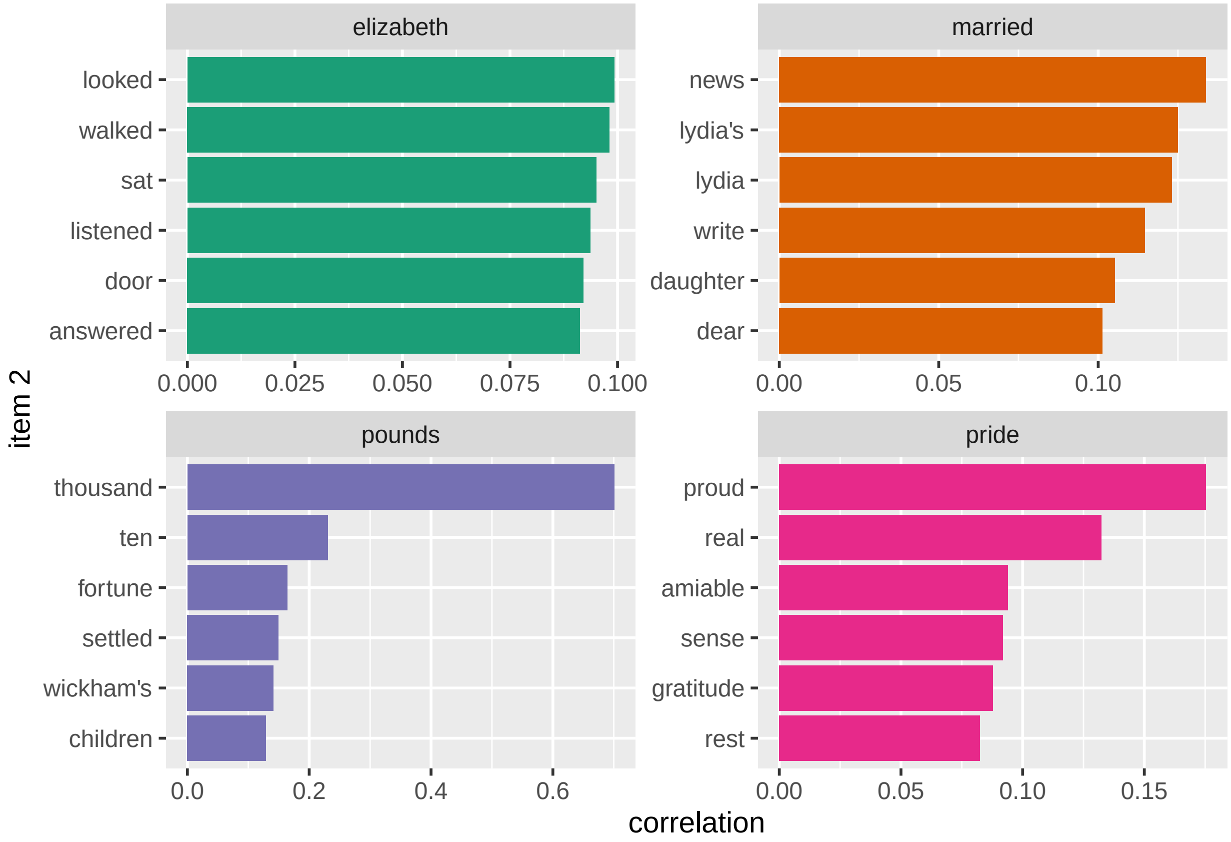
```
word_cors <- austen_section_words %>%
  group_by(word) %>%
  filter(n() >= 20) %>%
  pairwise_cor(word, section) %>%
  arrange(desc(correlation))

word_cors
```

```
## # A tibble: 154,842 x 3
##    item1     item2     correlation
##    <chr>     <chr>           <dbl>
##  1 bourgh    de              0.951
##  2 de        bourgh          0.951
##  3 pounds    thousand        0.701
##  4 thousand  pounds          0.701
##  5 william   sir             0.664
##  6 sir       william         0.664
##  7 catherine lady            0.663
##  8 lady      catherine       0.663
##  9 forster   colonel         0.622
## 10 colonel   forster         0.622
## # ... with 154,832 more rows
```

```
# find the words most correlated with a word like "pounds" using a filter operation.

word_cors %>%
  filter(item1 == "pounds") %>%
  arrange(desc(correlation))
```

```
## # A tibble: 393 x 3
##    item1  item2     correlation
##    <chr>  <chr>           <dbl>
##  1 pounds thousand       0.701
##  2 pounds ten            0.231
##  3 pounds fortune        0.164
##  4 pounds settled        0.149
##  5 pounds wickham's      0.142
##  6 pounds children       0.129
##  7 pounds mother's       0.119
##  8 pounds believed       0.0932
##  9 pounds estate         0.0890
## 10 pounds ready          0.0860
## # ... with 383 more rows
```

```
# pick particular interesting words and find the other words most associated with them

word_cors %>%
  filter(item1 %in% c("elizabeth", "pounds", "married", "pride")) %>%
  group_by(item1) %>%
  top_n(6) %>%
  ungroup() %>%
  mutate(item2 = reorder(item2, correlation)) %>%
  ggplot(aes(x = item2, y = correlation, fill = item1)) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  facet_wrap(~ item1, scales = "free") +
   labs(x = "item 2") +
  coord_flip() +
  scale_fill_brewer(palette = "Dark2")
```


elizabeth
looked
walked
sat
listened
door
answered
0.000
0.025
0.050
0.075
0.100
married
news
lydia's
lydia
write
daughter
dear
0.00
0.05
0.10
pounds
thousand
ten
fortune
settled
wickham's
children
0.0
0.2
0.4
0.6
pride
proud
real
amiable
sense
gratitude
rest
0.00
0.05
0.10
0.15
item 2
correlation

Converting to and from non-tidy formats

Tidying dfm objects

```r
data("data_corpus_inaugural", package = "quanteda")

inaug_dfm <- quanteda::dfm(data_corpus_inaugural, verbose = FALSE)

# dfm = document-feature-matrix

# integrate into tidy

inaug_td <- tidy(inaug_dfm)
inaug_td
```

```
## # A tibble: 45,453 x 3
##    document        term            count
##    <chr>           <chr>           <dbl>
##  1 1789-Washington fellow-citizens     1
##  2 1797-Adams      fellow-citizens     3
##  3 1801-Jefferson  fellow-citizens     2
##  4 1809-Madison    fellow-citizens     1
##  5 1813-Madison    fellow-citizens     1
##  6 1817-Monroe     fellow-citizens     5
##  7 1821-Monroe     fellow-citizens     1
##  8 1841-Harrison   fellow-citizens    11
##  9 1845-Polk       fellow-citizens     1
## 10 1849-Taylor     fellow-citizens     1
## # ... with 45,443 more rows
```

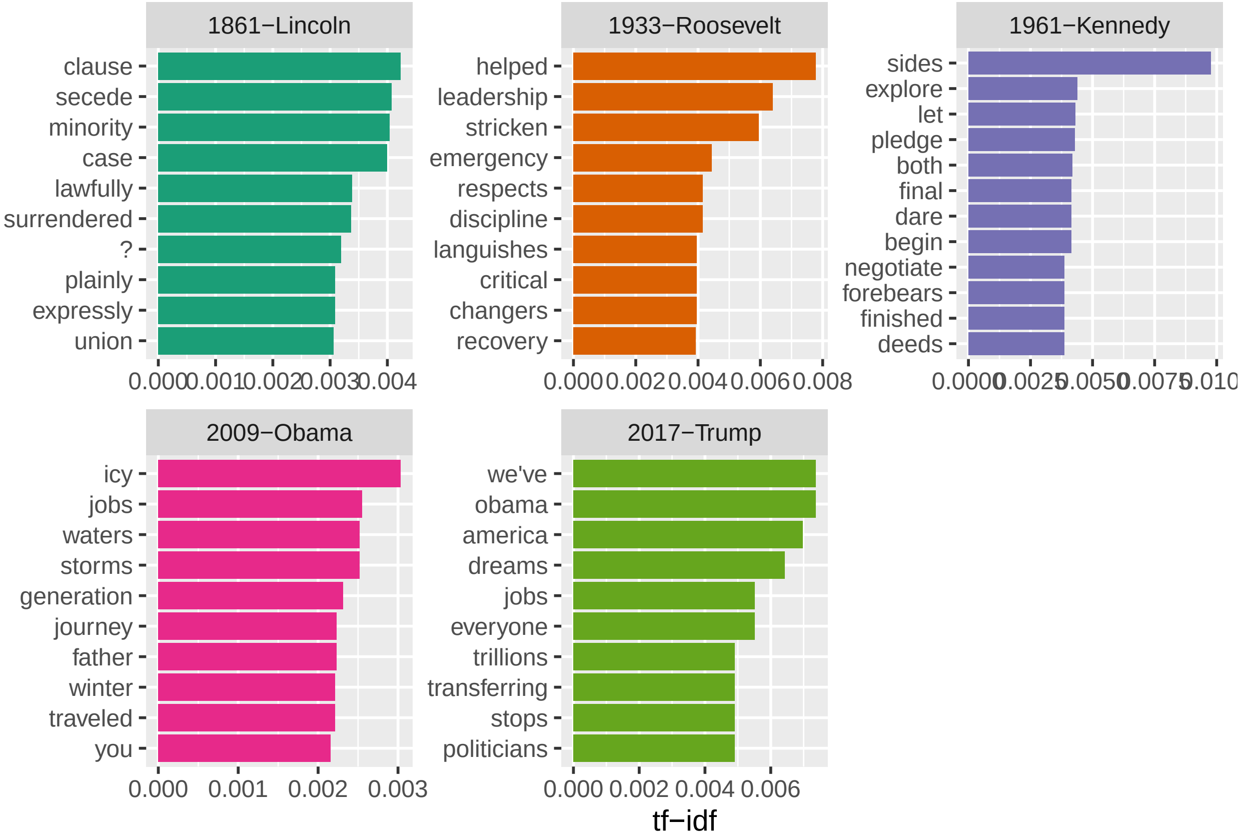
Find the words most specific to each of the inaugural speeches by calculating the tf-idf of each term-speech using the bind_tf_idf() function:

```r
inaug_tf_idf <- inaug_td %>%
  bind_tf_idf(term, document, count) %>%
  arrange(desc(tf_idf))

inaug_tf_idf
```

```
## # A tibble: 45,453 x 6
##    document        term        count      tf   idf tf_idf
##    <chr>           <chr>       <dbl>   <dbl> <dbl>  <dbl>
##  1 1793-Washington arrive          1 0.00680  4.08 0.0277
##  2 1793-Washington upbraidings     1 0.00680  4.08 0.0277
##  3 1793-Washington violated        1 0.00680  3.38 0.0230
##  4 1793-Washington willingly       1 0.00680  3.38 0.0230
##  5 1793-Washington incurring       1 0.00680  3.38 0.0230
##  6 1793-Washington previous        1 0.00680  2.98 0.0203
##  7 1793-Washington knowingly       1 0.00680  2.98 0.0203
##  8 1793-Washington injunctions     1 0.00680  2.98 0.0203
##  9 1793-Washington witnesses       1 0.00680  2.98 0.0203
## 10 1793-Washington besides         1 0.00680  2.69 0.0183
## # ... with 45,443 more rows
```

```r
speeches <- c("1933-Roosevelt", "1861-Lincoln", "1961-Kennedy", "2009-Obama", "2017-Trump")
```

```
inaug_tf_idf %>%
  filter(document %in% speeches) %>%
  group_by(document) %>%
  top_n(10,tf_idf) %>%
  ungroup %>%
  mutate(term=reorder_within(term, tf_idf, document)) %>%
  ggplot(aes(term, tf_idf, fill = document)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~document, scales = "free") +
  coord_flip() +
  scale_x_reordered() +
  labs(x = NULL,
       y = "tf-idf") +
  scale_fill_brewer(palette = "Dark2")
```



```
#  visualize how words changed in frequency over time

year_term_counts <- inaug_td %>%
  extract(document, "year", "(\\d+)", convert = TRUE) %>%
  complete(year, term, fill = list(count = 0)) %>%
  group_by(year) %>%
  mutate(year_total = sum(count))

year_term_counts %>%
  filter(term %in% c("god", "america", "foreign", "union",
                     "trade", "constitution", "freedom", "immigrants",
                     "economy", "education", "environment", "terrorism")) %>%
  ggplot(aes(year, count /year_total)) +
  geom_point() +
  geom_smooth() +
  facet_wrap(~ term, scales = "free_y", ncol = 3) +
  scale_y_continuous(labels = scales:: percent_format()) +
  ylab("% frequency of word in inaugural address")
```

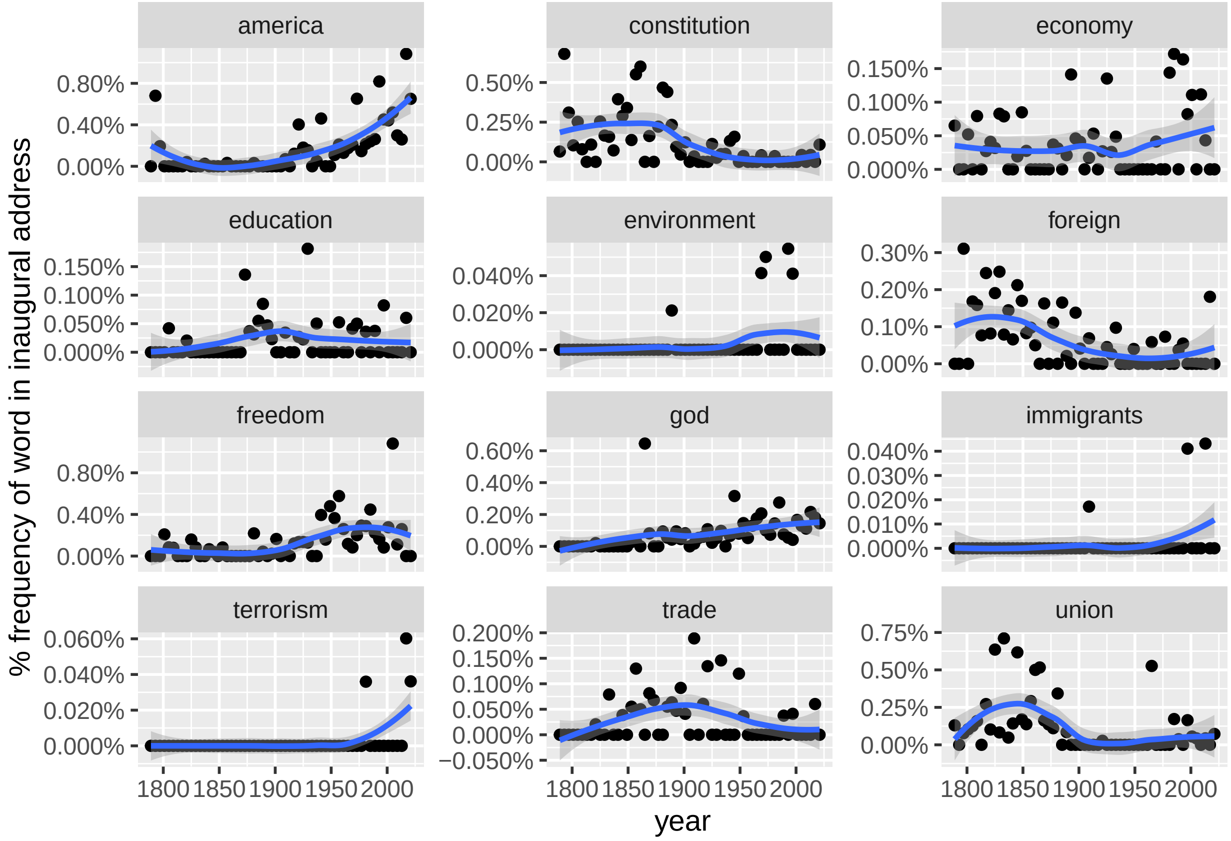


Figure 8: Changes in word frequency over time within Presidential inaugural addresses, for twelve selected terms