# 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
```

```r
# Use *to_lower = FALSE* to converts 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

```
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 ~ Baum,~               42 en       Children's Lite~ Publi~
##  2           55 The ~ Baum,~               42 en       Children's Lite~ Publi~
##  3          419 The ~ Baum,~               42 en       Children's Lite~ Publi~
##  4          420 Doro~ Baum,~               42 en       Children's Lite~ Publi~
##  5          485 The ~ Baum,~               42 en       Children's Lite~ Publi~
##  6          486 Ozma~ Baum,~               42 en       Fantasy/Childre~ Publi~
##  7          517 The ~ Baum,~               42 en       Children's Lite~ Publi~
##  8          955 The ~ Baum,~               42 en       Children's Lite~ Publi~
##  9          956 Tik-~ Baum,~               42 en       Children's Lite~ Publi~
## 10          957 The ~ Baum,~               42 en       Children's Lite~ Publi~
## # ... with 39 more rows, and 1 more variable: has_text <lgl>
```

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

WWOz <- gutenberg_download(55)

WWOz
```

```
## # A tibble: 4,721 x 2
##    gutenberg_id text
##           <int> <chr>
##  1           55 "The Wonderful Wizard of Oz"
##  2           55 ""
##  3           55 ""
##  4           55 "by"
##  5           55 ""
##  6           55 "L. Frank Baum"
##  7           55 ""
##  8           55 ""
##  9           55 ""
## 10           55 " Contents"
## # ... with 4,711 more rows
```

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

tidy_books_Oz
```

```
## # A tibble: 39,704 x 2
##    gutenberg_id words
##           <int> <chr>
##  1           55 the
##  2           55 wonderful
##  3           55 wizard
##  4           55 of
##  5           55 oz
##  6           55 by
```

```
##   7           55 l
##   8           55 frank
##   9           55 baum
##  10           55 contents
## # ... with 39,694 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:2507        Min.   :  1.00
##  Class :character   1st Qu.:  1.00
##  Mode  :character   Median :  2.00
##                     Mean   :  4.91
##                     3rd Qu.:  4.00
##                     Max.   :347.00
```

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

```
## # A tibble: 2,507 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 2,497 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
```

```
tidy_books_Oz %>%
  count(word, sort = TRUE) %>%
  dplyr::filter(n > 50 ) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```

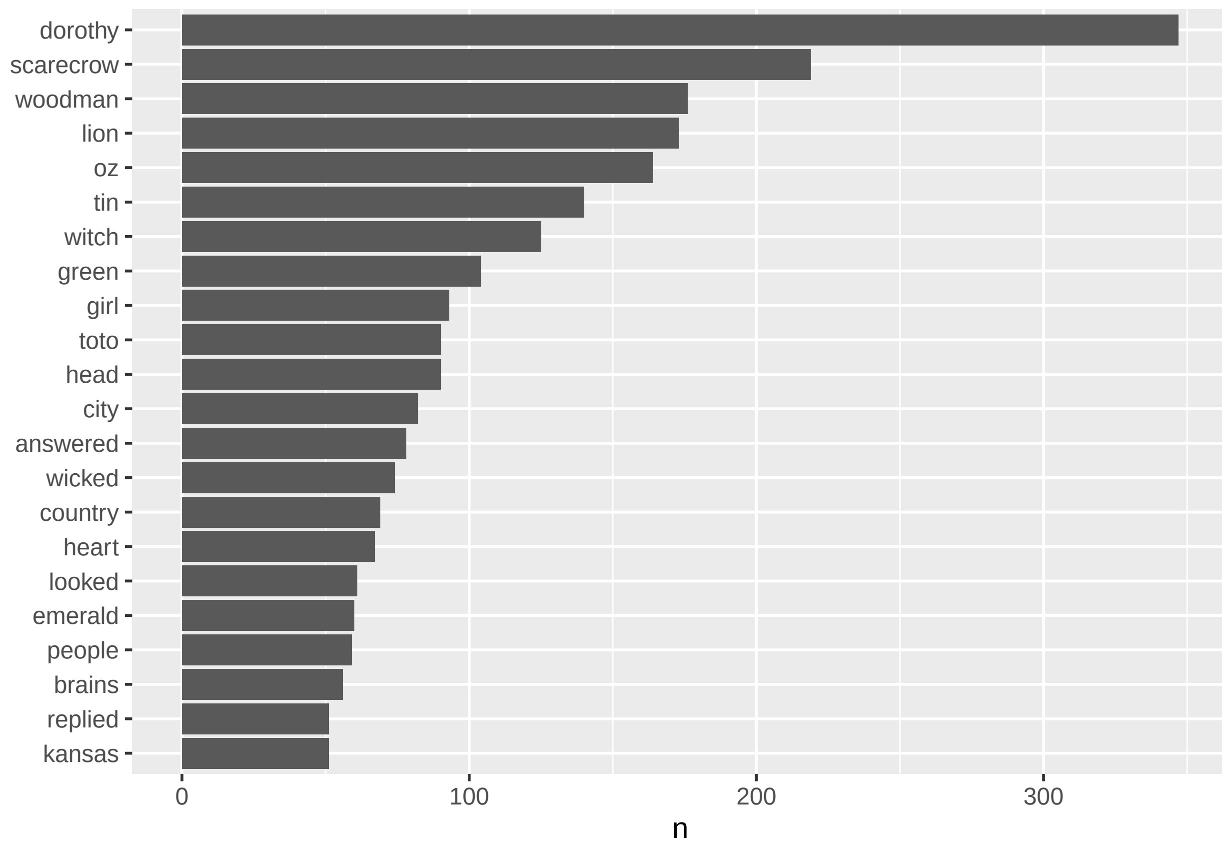



Figure 1: Words mentioned more than 50 times in The Wonderful Wizard of Oz

### 1.5 Word frequencies

```r
# H.G. Wells download

hgwells <- gutenberg_download(c(35,36,5230,159))

tidy_hgwells <- hgwells %>%
  unnest_tokens(words, text) %>%
  rename("word" = "words") %>%
  anti_join(stop_words, by = "word")

# most common words in these novels of H.G. Wells

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

```
## # A tibble: 11,769 x 2
##    word       n
##    <chr>  <int>
##  1 time     454
##  2 people   302
##  3 door     260
##  4 heard    249
##  5 black    232
##  6 stood    229
##  7 white    222
##  8 hand     218
##  9 kemp     213
## 10 eyes     210
## # ... with 11,759 more rows
```

```r
# Brontë sisters download - wrote in different style than  Jane Austen

bronte <- gutenberg_download(c(1260, 768, 969, 9182, 767))

tidy_bronte <- bronte %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words, by = "word")

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

```
## # A tibble: 23,050 x 2
##    word       n
##    <chr>  <int>
##  1 time    1065
##  2 miss     855
##  3 day      827
##  4 hand     768
##  5 eyes     713
##  6 night    647
##  7 heart    638
##  8 looked   601
##  9 door     592
## 10 half     586
```

```
## # ... with 23,040 more rows
```

```r
# The workbook says that 'Interesting that "time", "eyes", and "hand" are in the top 10
  # for both H.G. Wells and the Brontë sisters' but lets create a inner join of the top
  # 20 words of each word list.

# Bronte top 20 words used

bronte_T20 <- head(tidy_bronte %>%
  count(word, sort = TRUE), 20)

# H.G. Wells top 20 words used

hgwells_T20 <- head(tidy_hgwells %>%
  count(word, sort = TRUE), 20)

# Inner join. Key == "word." The words returned are those that appear in both

bronte_T20 %>%
  inner_join(hgwells_T20, by = "word") %>%
  rename("bronte" = "n.x", "hgwells" = "n.y")
```

```
## # A tibble: 8 x 3
##   word  bronte hgwells
##   <chr>  <int>   <int>
## 1 time    1065     454
## 2 day      827     193
## 3 hand     768     218
## 4 eyes     713     210
## 5 night    647     200
## 6 door     592     260
## 7 house    582     172
## 8 heard    510     249
```

```r
# if we were not using a tidy df, we would want to double check
  # how many rows we have since it would not be explict.

comparision1 <- bronte_T20 %>%
  inner_join(hgwells_T20, by = "word") %>%
  rename("bronte" = "n.x", "hgwells" = "n.y")

nrow(comparision1)
```

```
## [1] 8
```

Calculate the frequency for each word for the works of Jane Austen, the Brontë sisters, and H.G. Wells by binding the data frames together. We can use spread and gather from tidyr to reshape our dataframe so that it is just what we need for plotting and comparing the three sets of novels.

```r
frequency <-  bind_rows(mutate(tidy_bronte, author = "Brontë Sisters"),
                        mutate(tidy_hgwells, author = "H.G. Wells"),
                        mutate(tidy_books, author = "Jane Austen")) %>%
  mutate(word = str_extract(word, "[a-z']+")) %>%
  count(author, word) %>%
  group_by(author) %>%
  mutate(proportion = n/sum(n)) %>%
```

```
  select(-n) %>%
  spread(author, proportion) %>%
  gather(author, proportion, `Brontë Sisters`:`H.G. Wells`)
```

```r
ggplot(frequency, aes(x = proportion, y = `Jane Austen`, color = abs(`Jane Austen` - proportion))) +
  geom_abline(color = "gray40", lty = 2) +
  geom_jitter(alpha = 0.1, size = 2.5, width = 0.3, height = 0.3) +
  geom_text(aes(label = word), check_overlap = TRUE, vjust = 1.5) +
  scale_x_log10(labels = percent_format()) +
  scale_y_log10(labels = percent_format()) +
  scale_color_gradient(limits = c(0, 0.001), low = "darkslategray4", high = "gray75") +
  facet_wrap(~author, ncol = 2) +
  theme(legend.position="none") +
  labs(y = "Jane Austen", x = NULL)
```


Figure 2: Comparing the word frequencies of Jane Austen, the Brontë sisters, and H.G. Wells

How correlated are the word frequencies between Austen and the Brontë sisters, and between Austen and Wells? Consistent with the graphics, the word frequencies are more correlated between the Austen and Brontë novels than between Austen and H.G. Wells.

```
cor.test(data = frequency[frequency$author == "Brontë Sisters",],
         ~ proportion + `Jane Austen`)
```

```
##
##  Pearson's product-moment correlation
##
## data:  proportion and Jane Austen
## t = 119.65, df = 10404, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.7527869 0.7689642
## sample estimates:
##       cor
## 0.7609938
```

```
cor.test(data = frequency[frequency$author == "H.G. Wells",],
         ~ proportion + `Jane Austen`)
```

```
##
##  Pearson's product-moment correlation
##
## data:  proportion and Jane Austen
## t = 36.441, df = 6053, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4032800 0.4445987
## sample estimates:
##       cor
## 0.4241601
```

# 2 Sentiment analysis with tidy data

## 2.1 The sentiments dataset

```r
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
```

```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
```

```r
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

```r
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
```

```r
# examine how sentiment changes throughout each novel

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)
```

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

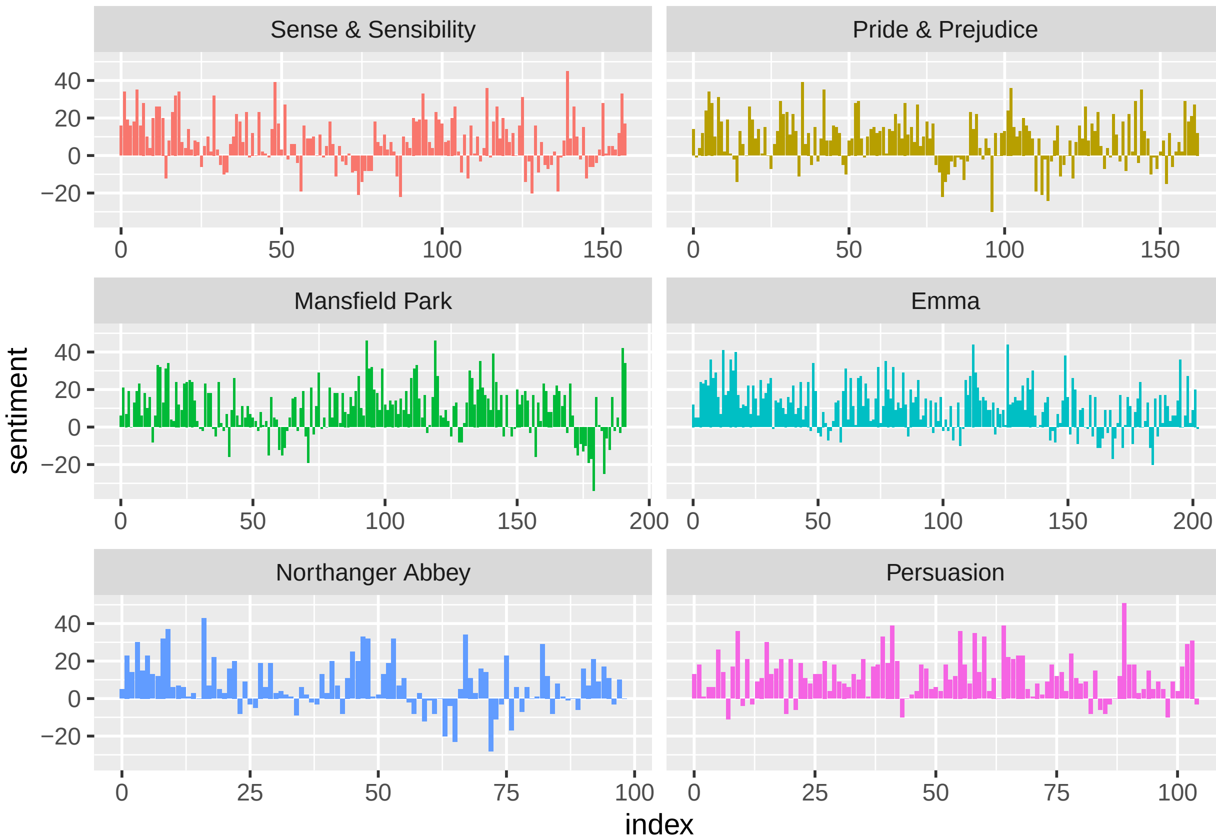



Figure 3: Sentiment through the narratives of Jane Austen's novels

## 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")) %>%
                            mutate(method = "Bing et al."),
                          pride_prejudice %>%
                            inner_join(get_sentiments("nrc") %>%
                                         filter(sentiment %in% c("positive",
                                                                 "negative"))) %>%
                            mutate(method = "NRC")) %>%
  count(method, index = linenumber %/% 80, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)
```

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

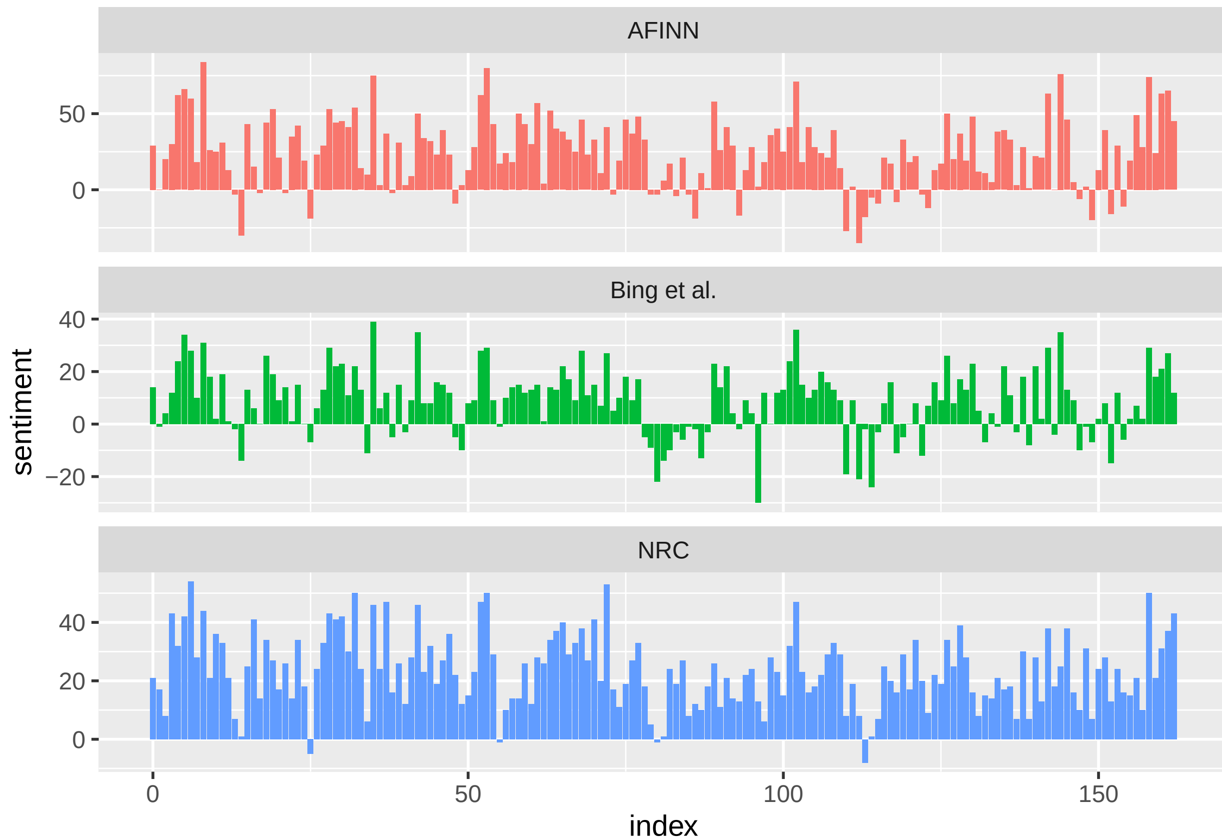



Figure 4: Comparing three sentiment lexicons using Pride and Prejudice

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
get_sentiments("nrc") %>%
  filter(sentiment %in% c("positive", "negative")) %>%
  count(sentiment)
```

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative   3324
## 2 positive   2312
```

```r
get_sentiments("bing") %>%
  count(sentiment)
```

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative   4781
## 2 positive   2005
```

## 2.5 Wordclouds

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

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

```r
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] "however little known the feelings or views of such a man may be on his first entering a neighbou
```

```
# 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?

```r
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 <- left_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")
```

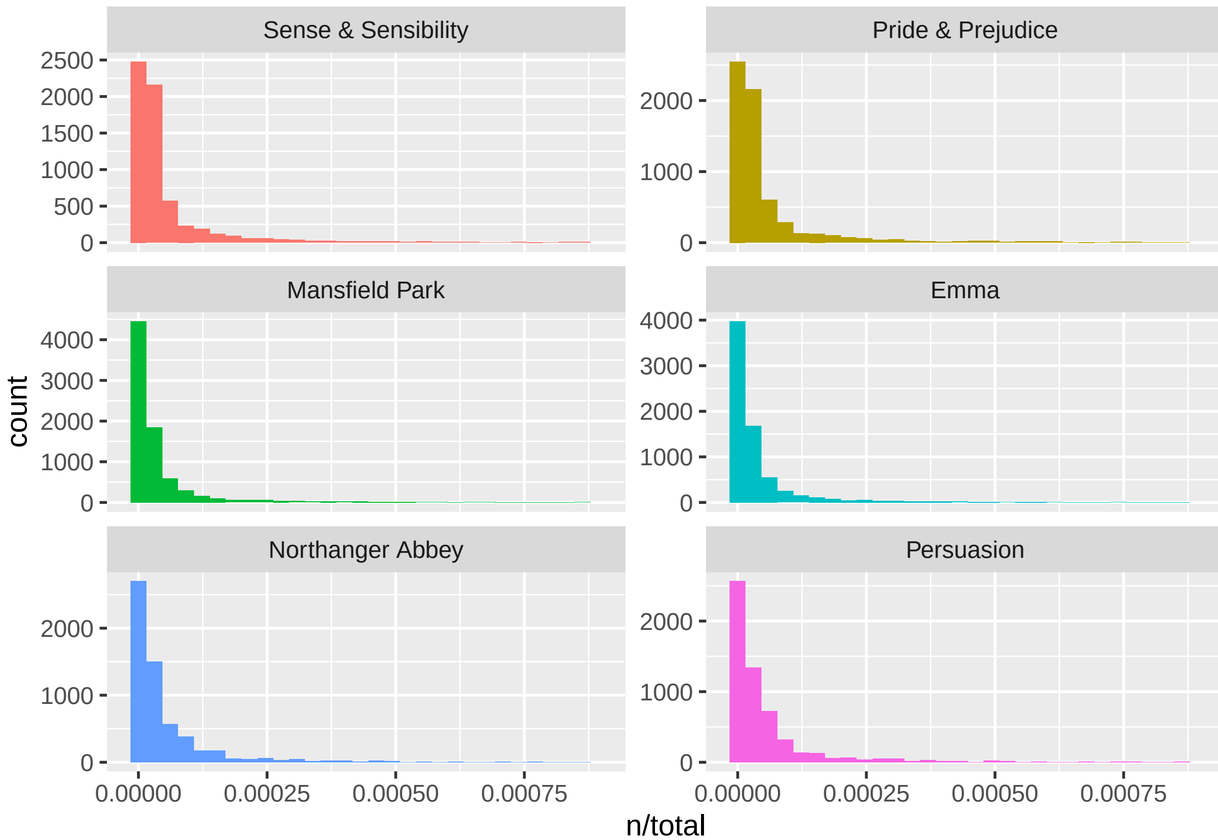



Figure 5: Term Frequency Distribution in Jane Austen's Novels

```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
```

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

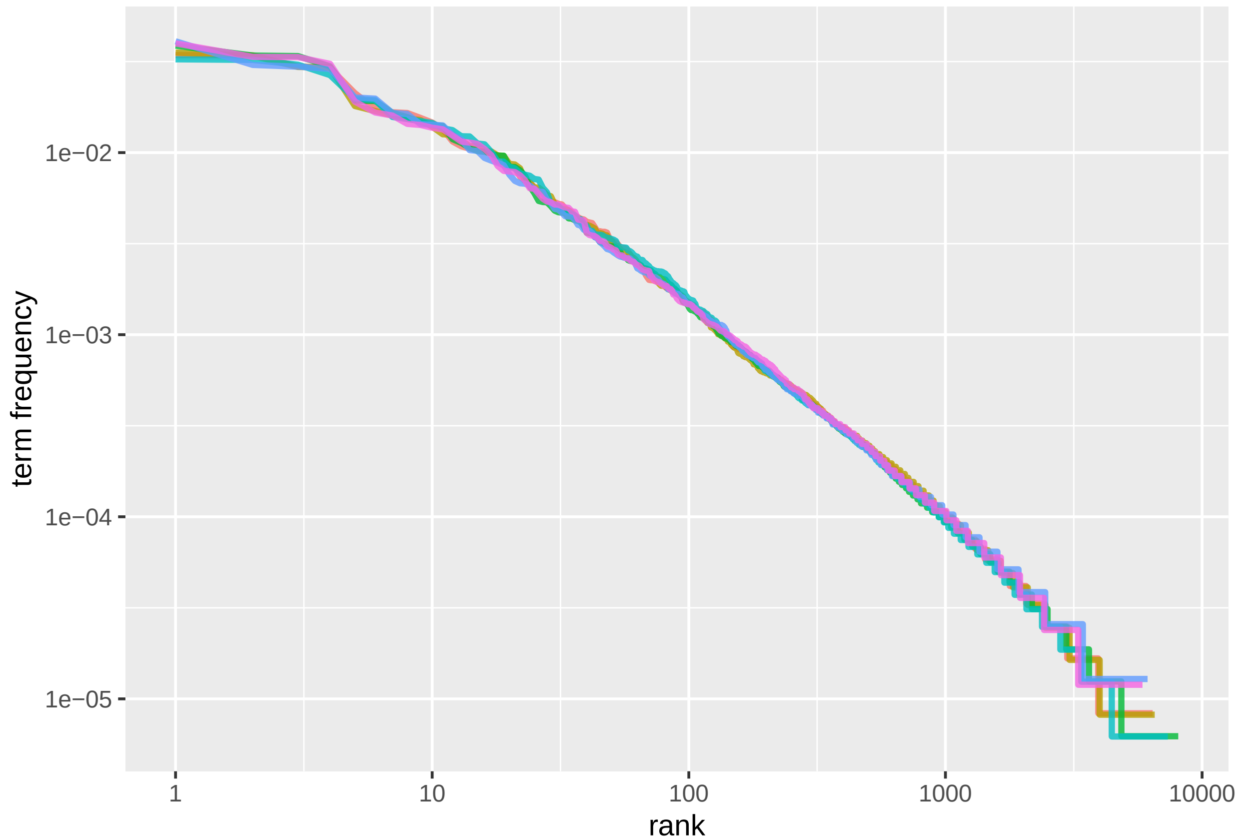



Figure 6: Zipf's law for Jane Austen's novels

```r
# 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
```

```r
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()
```

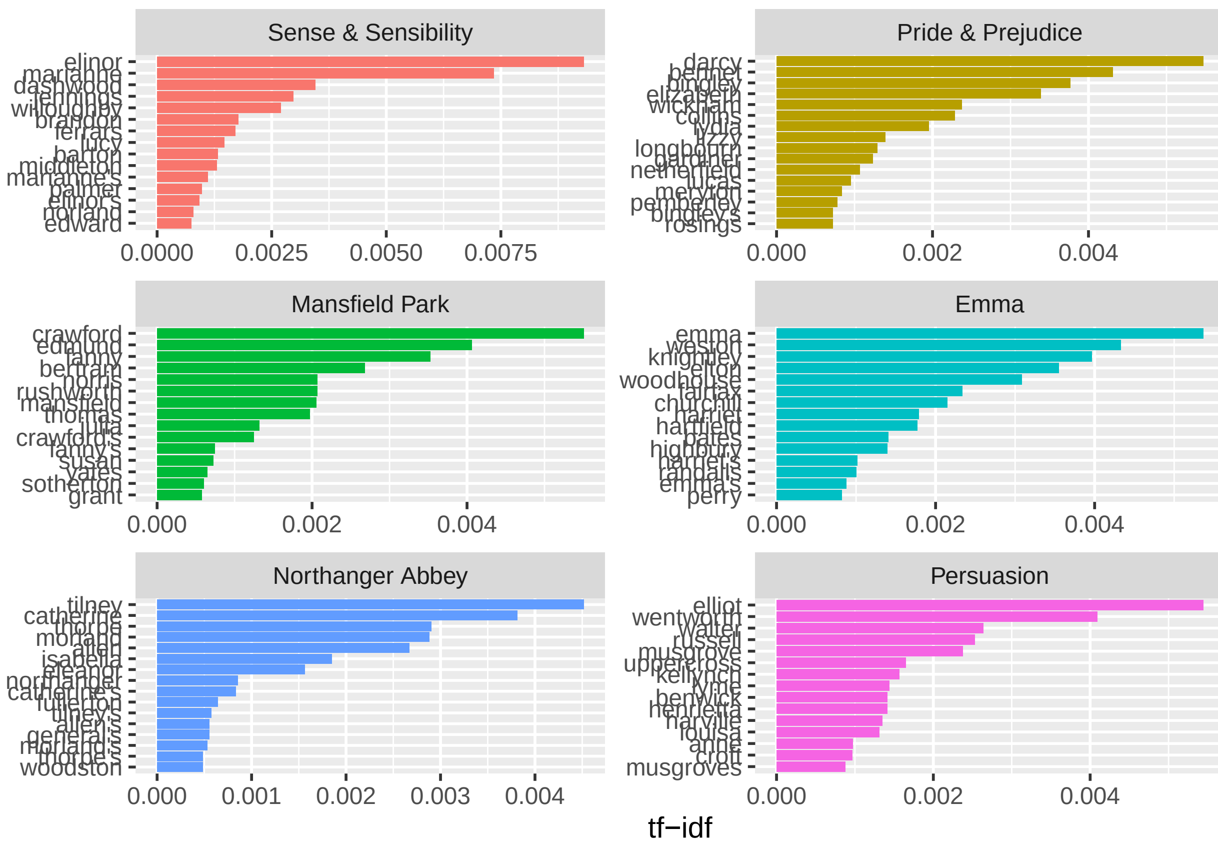



Figure 7: Highest tf-idf words in each of Jane Austen's Novels

## 3.4 A corpus of physics texts

```r
physics <- gutenberg_download(c(37729, 14725, 13476, 30155), meta_fields = "author")

physics_words <- physics %>%
  unnest_tokens(word, text) %>%
  count(author, word, sort = TRUE)

physics_words
```

```
## # A tibble: 12,671 x 3
##    author              word      n
##    <chr>               <chr> <int>
##  1 Galilei, Galileo    the    3760
##  2 Tesla, Nikola       the    3604
##  3 Huygens, Christiaan the    3553
##  4 Einstein, Albert    the    2993
##  5 Galilei, Galileo    of     2049
##  6 Einstein, Albert    of     2028
##  7 Tesla, Nikola       of     1737
##  8 Huygens, Christiaan of     1708
##  9 Huygens, Christiaan to     1207
## 10 Tesla, Nikola       a      1176
## # ... with 12,661 more rows
```

```r
# calculate tf-idf

plot_physics <- physics_words %>%
  bind_tf_idf(word, author, n) %>%
  mutate(word = fct_reorder(word, tf_idf)) %>%
  mutate(author = factor(author, levels = c("Galilei, Galileo",
                                            "Huygens, Christiaan",
                                            "Tesla, Nikola",
                                            "Einstein, Albert")))
```

```r
plot_physics %>%
  group_by(author) %>%
  top_n(15, tf_idf) %>%
  ungroup() %>%
  mutate(word = reorder(word, tf_idf)) %>%
  ggplot(aes(word, tf_idf, fill = author)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~author, ncol = 2, scales = "free") +
  coord_flip()
```

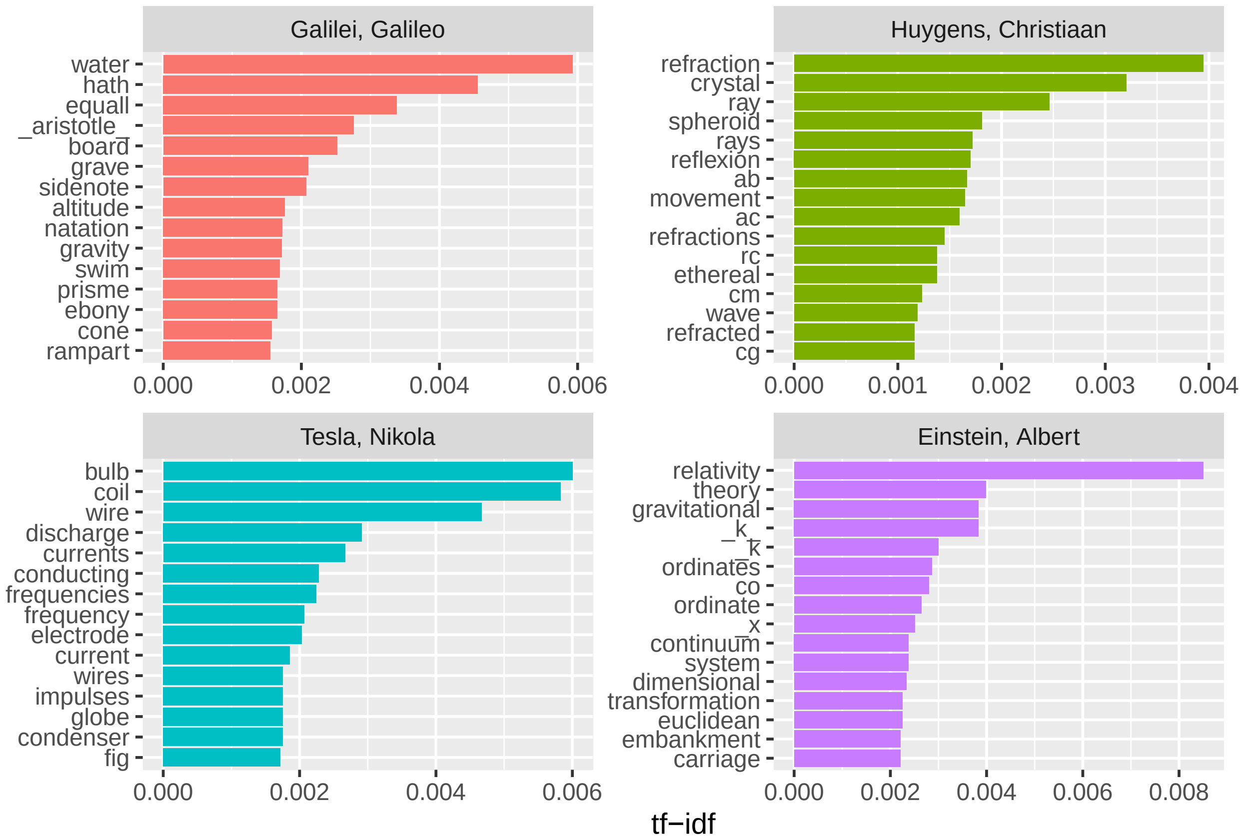



Figure 8: Highest tf-idf words in each physics texts

```r
# investigate the "k" in the Einstein text

physics %>%
  filter(str_detect(text, "_k_")) %>%
  select(text)
```

```
## # A tibble: 7 x 1
##   text
##   <chr>
## 1 surface AB at the points AK_k_B. Then instead of the hemispherical
## 2 would needs be that from all the other points K_k_B there should
## 3 necessarily be equal to CD, because C_k_ is equal to CK, and C_g_ to
## 4 the crystal at K_k_, all the points of the wave CO_oc_ will have
## 5 O_o_ has reached K_k_. Which is easy to comprehend, since, of these
## 6 CO_oc_ in the crystal, when O_o_ has arrived at K_k_, because it forms
## 7 <U+03C1> is the average density of the matter and _k_ is a constant connected
```

```r
# make a custom list of stop words and use anti_join() to remove them

mystopwords <- tibble(word = c("eq", "co", "rc", "ac", "ak", "bn",
                                   "fig", "file", "cg", "cb", "cm",
                               "ab", "_k", "_k_", "_x"))

physics_words <- anti_join(physics_words, mystopwords, by = "word")
```

```r
plot_physics <- physics_words %>%
  bind_tf_idf(word, author, n) %>%
  mutate(word = str_remove_all(word, "_")) %>%
  group_by(author) %>%
  top_n(15, tf_idf) %>%
  ungroup() %>%
  mutate(word = reorder_within(word, tf_idf, author)) %>%
  mutate(author = factor(author, levels = c("Galilei, Galileo",
                                            "Huygens, Christiaan",
                                            "Tesla, Nikola",
                                            "Einstein, Albert")))

ggplot(plot_physics, aes(word, tf_idf, fill = author)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~author, ncol = 2, scales = "free") +
  coord_flip() +
  scale_x_reordered()
```

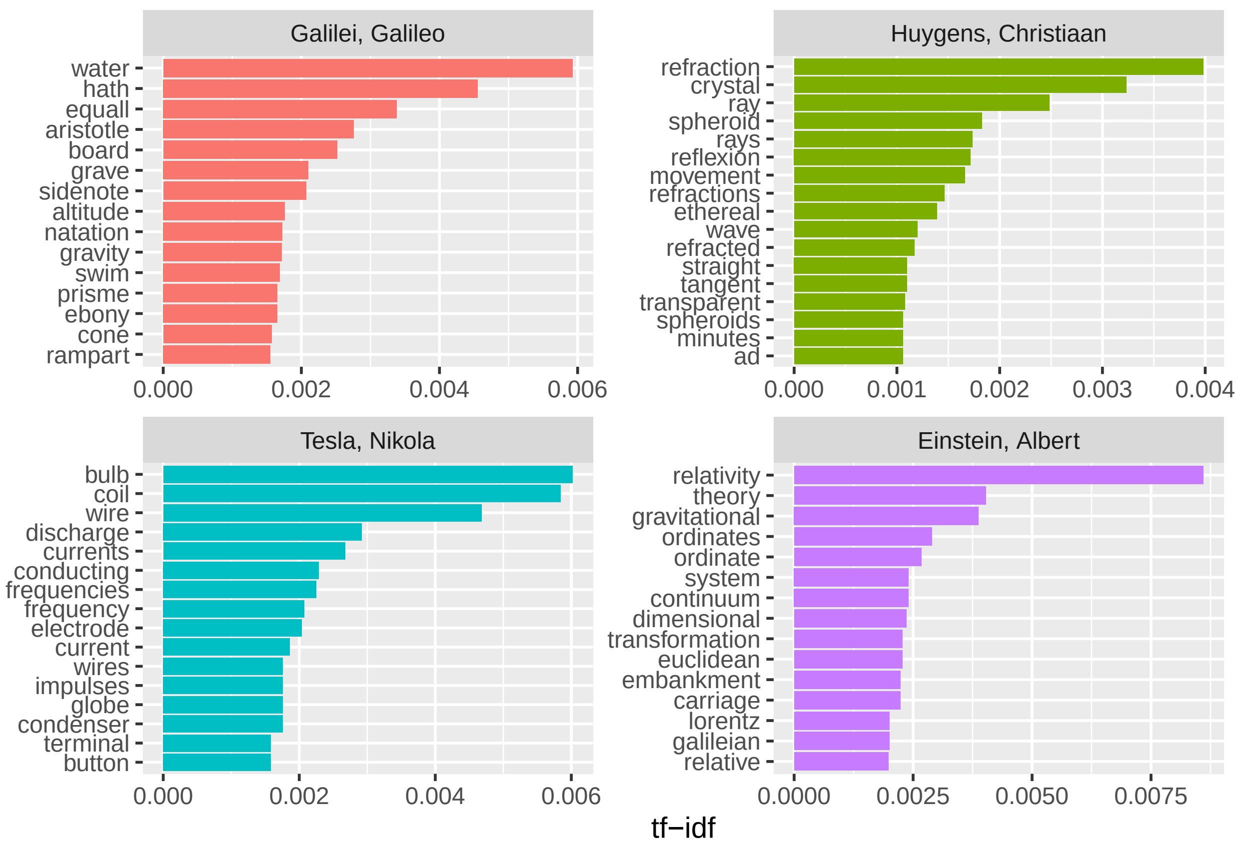



Figure 9: Highest tf-idf words in classic physics texts

# 4 Relationships between words: n-grams and correlations

## 4.1 Tokenizing by n-gram

```
# build a model of the relationships between words

 austen_bigrams <- austen_books() %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)
# n is the number of consecutive words that we are analyzing

austen_bigrams
```

```
## # A tibble: 725,049 x 2
##    book                bigram
##    <fct>               <chr>
##  1 Sense & Sensibility sense and
##  2 Sense & Sensibility and sensibility
##  3 Sense & Sensibility sensibility by
##  4 Sense & Sensibility by jane
##  5 Sense & Sensibility jane austen
##  6 Sense & Sensibility austen 1811
##  7 Sense & Sensibility 1811 chapter
##  8 Sense & Sensibility chapter 1
##  9 Sense & Sensibility 1 the
## 10 Sense & Sensibility the family
## # ... with 725,039 more rows
```

### 4.1.1 Counting and filtering n-grams

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

```
## # A tibble: 211,236 x 2
##    bigram       n
##    <chr>    <int>
##  1 of the    3017
##  2 to be     2787
##  3 in the    2368
##  4 it was    1781
##  5 i am      1545
##  6 she had   1472
##  7 of her    1445
##  8 to the    1387
##  9 she was   1377
## 10 had been  1299
## # ... with 211,226 more rows
```

```
# use separate() to split a column into multiple based on a delimiter -->
  # lets us separate it into two columns, "word1" and "word2"
  # at which point we can remove cases where either is a stop-word.

bigrams_separated <- austen_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")

bigrams_filtered <- bigrams_separated %>%
```

```r
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

# new bigram counts:
bigram_counts <- bigrams_filtered %>%
  count(word1, word2, sort = TRUE)


bigram_counts
```

```
## # A tibble: 33,421 x 3
##    word1   word2         n
##    <chr>   <chr>     <int>
##  1 sir     thomas      287
##  2 miss    crawford    215
##  3 captain wentworth   170
##  4 miss    woodhouse   162
##  5 frank   churchill   132
##  6 lady    russell     118
##  7 lady    bertram     114
##  8 sir     walter      113
##  9 miss    fairfax     109
## 10 colonel brandon     108
## # ... with 33,411 more rows
```

```r
# unite function is the inverse of separate(), and lets us recombine the columns into one.

bigrams_united <- bigrams_filtered %>%
  unite(bigram, word1, word2, sep = " ") %>%
   count(bigram, sort = TRUE)

bigrams_united
```

```
## # A tibble: 33,421 x 2
##    bigram                n
##    <chr>             <int>
##  1 sir thomas          287
##  2 miss crawford       215
##  3 captain wentworth   170
##  4 miss woodhouse      162
##  5 frank churchill     132
##  6 lady russell        118
##  7 lady bertram        114
##  8 sir walter          113
##  9 miss fairfax        109
## 10 colonel brandon     108
## # ... with 33,411 more rows
```

```r
# Trigrams -->  consecutive sequences of 3 words | n = 3

austen_books() %>%
  unnest_tokens(trigram, text, token = "ngrams", n = 3) %>%
  separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%
  filter(!word1 %in% stop_words$word,
         !word2 %in% stop_words$word,
```

```r
         !word3 %in% stop_words$word) %>%
  count(word1, word2, word3, sort = TRUE)
```

```
## # A tibble: 8,757 x 4
##    word1     word2     word3         n
##    <chr>     <chr>     <chr>     <int>
##  1 dear      miss      woodhouse    23
##  2 miss      de        bourgh       18
##  3 lady      catherine de           14
##  4 catherine de        bourgh       13
##  5 poor      miss      taylor       11
##  6 sir       walter    elliot       11
##  7 ten       thousand  pounds       11
##  8 dear      sir       thomas       10
##  9 twenty    thousand  pounds        8
## 10 replied   miss      crawford      7
## # ... with 8,747 more rows
```

### 4.1.3 Using bigrams to provide context in sentiment analysis

A word's context can matter nearly as much as its presence. For example, the words "happy" and "like" will be counted as positive, even in a sentence like "I'm not happy and I don't like it!." We examine how often sentiment-associated words are preceded by "not" or other negating words. We could use this to ignore or even reverse their contribution to the sentiment score. Use AFINN lexicon.

Examine the most frequent words that were preceded by "not" and were associated with a sentiment.

```
bigrams_separated %>%
  filter(word1 == "not") %>%
  count(word1, word2, sort = TRUE)
```

```
## # A tibble: 1,246 x 3
##    word1 word2     n
##    <chr> <chr> <int>
##  1 not   be      610
##  2 not   to      355
##  3 not   have    327
##  4 not   know    252
##  5 not   a       189
##  6 not   think   176
##  7 not   been    160
##  8 not   the     147
##  9 not   at      129
## 10 not   in      118
## # ... with 1,236 more rows
```

```
AFINN <- get_sentiments("afinn")

not_words <- bigrams_separated %>%
  filter(word1 == "not") %>%
  inner_join(AFINN, by = c(word2 = "word")) %>%
  count(word2, value, sort = TRUE)

not_words
```

```
## # A tibble: 245 x 3
##    word2   value     n
##    <chr>   <dbl> <int>
##  1 like        2    99
##  2 help        2    82
##  3 want        1    45
##  4 wish        1    39
##  5 allow       1    36
##  6 care        2    23
##  7 sorry      -1    21
##  8 leave      -1    18
##  9 pretend    -1    18
## 10 worth       2    17
## # ... with 235 more rows
```

Which words contributed the most in the "wrong" direction –> multiply their value by the number of times they appear

```
not_words %>%
  mutate(contribution = n * value) %>%
  arrange(desc(abs(contribution))) %>%
  head(20) %>%
  mutate(word2 = reorder(word2, contribution)) %>%
  ggplot(aes(word2, n * value, fill =n *value > 0)) +
  geom_col(show.legend = FALSE) + xlab("Words preceded by \"not\"") +
  ylab("sentiment value * number of occurences") +
  coord_flip()
```

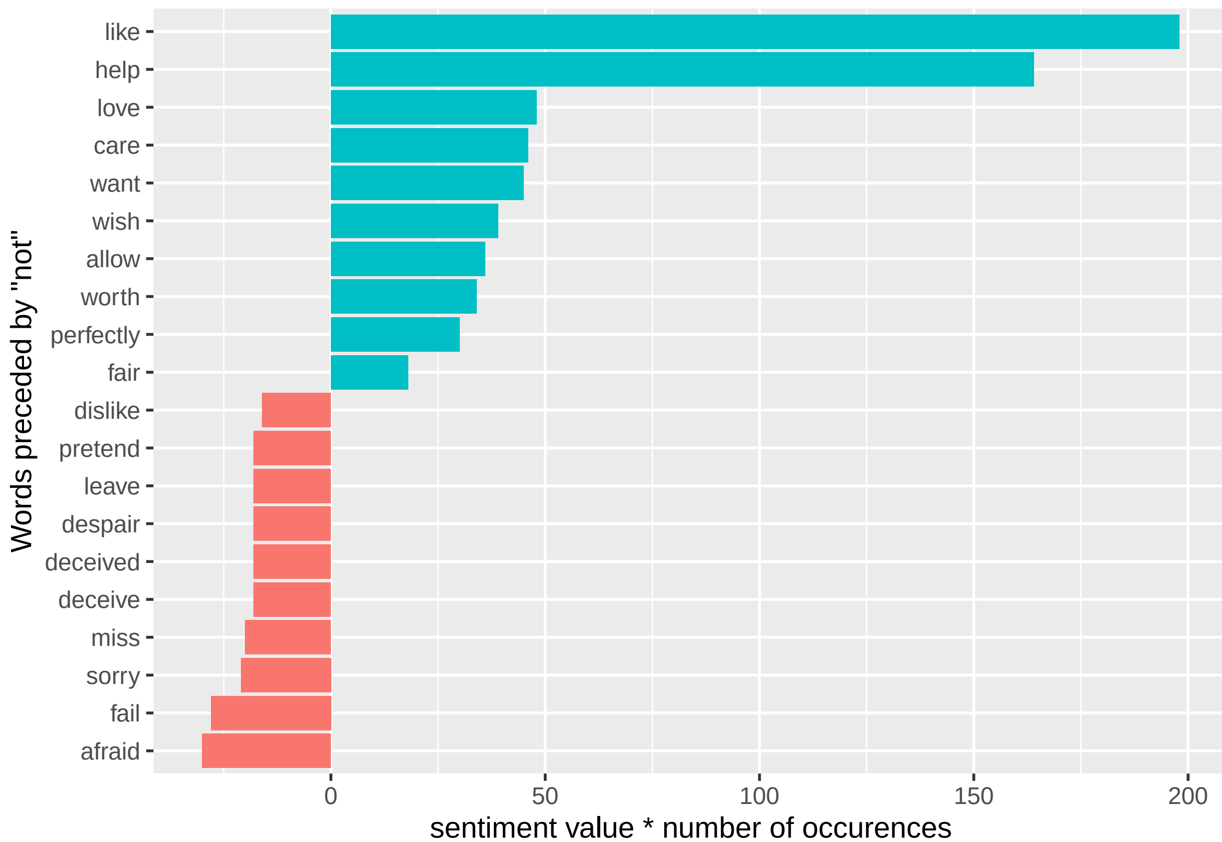



Figure 10: The 20 words preceded by 'not' that had the greatest contribution to sentiment values, in either a positive or negative direction

Exploring different negative words

Examine the most frequent words that were preceded by "never" and were associated with a sentiment.

```
bigrams_separated %>%
  filter(word1 == "never") %>%
  count(word1, word2, sort = TRUE)
```

```
## # A tibble: 394 x 3
##    word1 word2     n
##    <chr> <chr> <int>
##  1 never been     68
##  2 never be       63
##  3 never have     59
##  4 never to       49
##  5 never had      48
##  6 never seen     45
##  7 never saw      43
##  8 never was      34
##  9 never heard    33
## 10 never could    26
## # ... with 384 more rows
```

```
never_words <- bigrams_separated %>%
  filter(word1 == "never") %>%
  inner_join(AFINN, by = c(word2 = "word")) %>%
  count(word2, value, sort = TRUE)

never_words
```

```
## # A tibble: 65 x 3
##    word2   value     n
##    <chr>   <dbl> <int>
##  1 forget     -1    12
##  2 failed     -2     8
##  3 want        1     7
##  4 allow       1     5
##  5 agree       1     4
##  6 failing    -2     4
##  7 loved       3     4
##  8 liked       2     3
##  9 wish        1     3
## 10 consent     2     2
## # ... with 55 more rows
```

Examine the most frequent words that were preceded by "no" and were associated with a sentiment.

```
bigrams_separated %>%
  filter(word1 == "no") %>%
  count(word1, word2, sort = TRUE)
```

```
## # A tibble: 876 x 3
##    word1 word2      n
##    <chr> <chr>  <int>
##  1 no    more     190
##  2 no    longer   125
##  3 no    doubt    102
```

```
##   4 no    one       96
##   5 no    means     87
##   6 no    i         78
##   7 no    other     68
##   8 no    no        60
##   9 no    answer    38
##  10 no    reason    38
## # ... with 866 more rows
```

```
no_words <- bigrams_separated %>%
  filter(word1 == "no") %>%
  inner_join(AFINN, by = c(word2 = "word")) %>%
  count(word2, value, sort = TRUE)

no_words
```

```
## # A tibble: 153 x 3
##    word2    value     n
##    <chr>    <dbl> <int>
##  1 doubt       -1   102
##  2 no          -1    60
##  3 harm        -2    22
##  4 great        3    19
##  5 pleasure     3    16
##  6 danger      -2    15
##  7 want         1    15
##  8 good         3    13
##  9 matter       1    12
## 10 chance       2    11
## # ... with 143 more rows
```

Examine the most frequent words that were preceded by "without" and were associated with a sentiment.

```
bigrams_separated %>%
  filter(word1 == "without") %>%
  count(word1, word2, sort = TRUE)
```

```
## # A tibble: 370 x 3
##    word1   word2       n
##    <chr>   <chr>   <int>
##  1 without any       111
##  2 without a          64
##  3 without the        61
##  4 without being      45
##  5 without her        36
##  6 without having     22
##  7 without knowing    21
##  8 without saying     20
##  9 without feeling    13
## 10 without much       13
## # ... with 360 more rows
```

```
without_words <- bigrams_separated %>%
  filter(word1 == "without") %>%
  inner_join(AFINN, by = c(word2 = "word")) %>%
  count(word2, value, sort = TRUE)
```

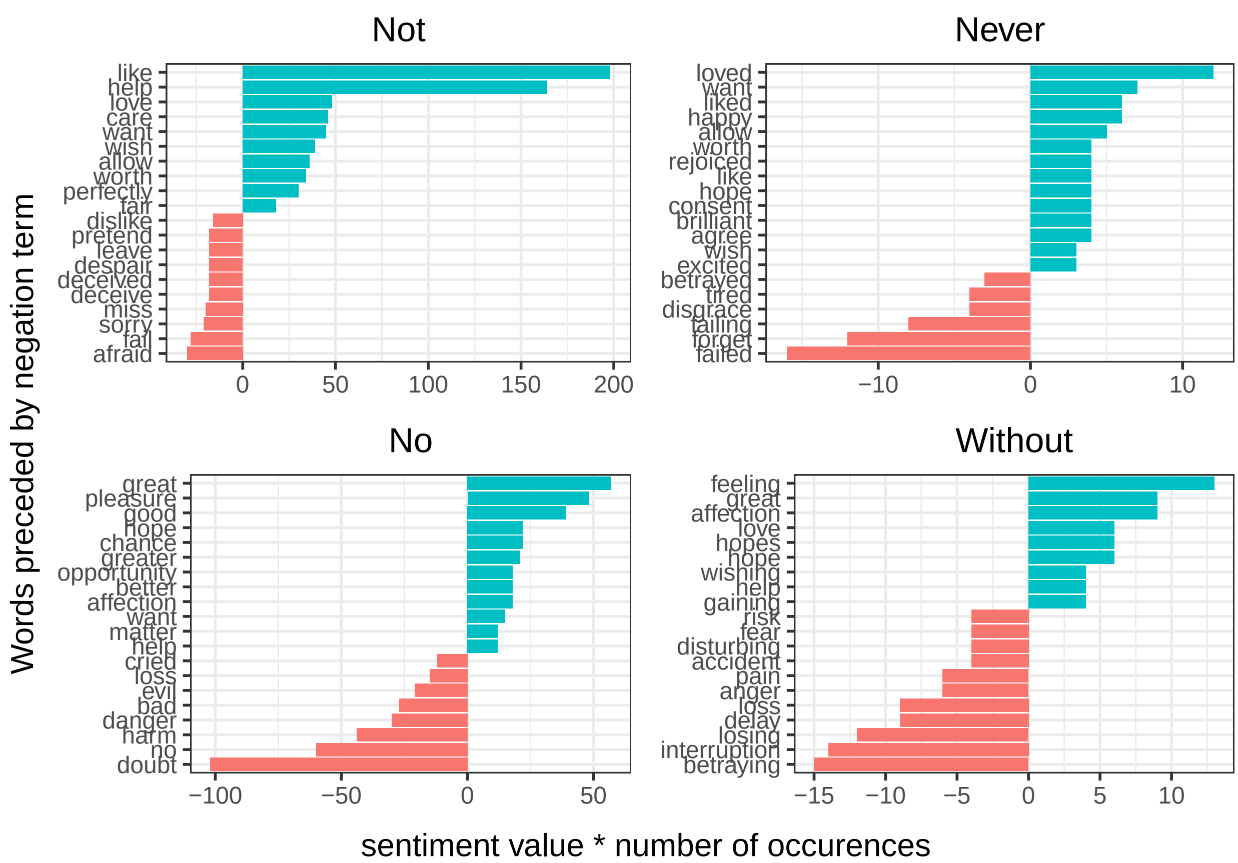
```r
without_words
```

```
## # A tibble: 68 x 3
##    word2        value     n
##    <chr>        <dbl> <int>
##  1 feeling          1    13
##  2 delay           -1     9
##  3 interruption    -2     7
##  4 betraying       -3     5
##  5 losing          -3     4
##  6 wishing          1     4
##  7 affection        3     3
##  8 great            3     3
##  9 hope             2     3
## 10 hopes            2     3
## # ... with 58 more rows
```

Which words contributed the most in the "wrong" direction –> multiply their value by the number of times they appear

```
# individual ggplot graphs --> goal: try to get a facet wrap!

w1 <- not_words %>%
  mutate(contribution = n * value) %>%
  arrange(desc(abs(contribution))) %>%
  head(20) %>%
  mutate(word2 = reorder(word2, contribution)) %>%
  ggplot(aes(word2, n * value, fill =n *value > 0)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  theme_bw() +
  labs(title = "Not") +
  theme(
    plot.title = element_text(hjust = 0.5),
    axis.title.x = element_blank(),
    axis.title.y = element_blank())

w2 <- never_words %>%
  mutate(contribution = n * value) %>%
  arrange(desc(abs(contribution))) %>%
  head(20) %>%
  mutate(word2 = reorder(word2, contribution)) %>%
  ggplot(aes(word2, n * value, fill =n *value > 0)) +
  geom_col(show.legend = FALSE)+
  coord_flip() +
  theme_bw() +
  labs(title = "Never") +
  theme(
    plot.title = element_text(hjust = 0.5),
    axis.title.x = element_blank(),
    axis.title.y = element_blank())

w3 <- no_words %>%
  mutate(contribution = n * value) %>%
  arrange(desc(abs(contribution))) %>%
  head(20) %>%
  mutate(word2 = reorder(word2, contribution)) %>%
  ggplot(aes(word2, n * value, fill =n *value > 0)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  theme_bw() +
  labs(title = "No") +
  theme(
    plot.title = element_text(hjust = 0.5),
    axis.title.x = element_blank(),
    axis.title.y = element_blank())

w4 <- without_words %>%
  mutate(contribution = n * value) %>%
  arrange(desc(abs(contribution))) %>%
  head(20) %>%
```

```
  mutate(word2 = reorder(word2, contribution)) %>%
  ggplot(aes(word2, n * value, fill =n *value > 0)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  theme_bw() +
  labs(title = "Without") +
  theme(
    plot.title = element_text(hjust = 0.5),
    axis.title.x = element_blank(),
    axis.title.y = element_blank())

figure <- ggarrange(w1, w2, w3, w4)

annotate_figure(figure,
                bottom = text_grob("sentiment value * number of occurences"),
                left = text_grob("Words preceded by negation term" , rot = 90))
```



### 4.1.4 Visualizing a network of bigrams with ggraph

```r
# original counts
bigram_counts
```

```
## # A tibble: 33,421 x 3
##    word1   word2         n
##    <chr>   <chr>     <int>
##  1 sir     thomas      287
##  2 miss    crawford    215
##  3 captain wentworth   170
##  4 miss    woodhouse   162
##  5 frank   churchill   132
##  6 lady    russell     118
##  7 lady    bertram     114
##  8 sir     walter      113
##  9 miss    fairfax     109
## 10 colonel brandon     108
## # ... with 33,411 more rows
```

```r
# filter for only relatively common combinations

bigram_graph <- bigram_counts %>%
  filter(n > 20) %>%
  graph_from_data_frame()

bigram_graph
```

```
## IGRAPH fee23f4 DN-- 91 77 --
## + attr: name (v/c), n (e/n)
## + edges from fee23f4 (vertex names):
##  [1] sir     ->thomas     miss    ->crawford   captain ->wentworth
##  [4] miss    ->woodhouse  frank   ->churchill  lady    ->russell
##  [7] lady    ->bertram    sir     ->walter     miss    ->fairfax
## [10] colonel ->brandon    miss    ->bates      lady    ->catherine
## [13] sir     ->john       jane    ->fairfax    miss    ->tilney
## [16] lady    ->middleton  miss    ->bingley    thousand->pounds
## [19] miss    ->dashwood   miss    ->bennet     john    ->knightley
## [22] miss    ->morland    captain ->benwick    dear    ->miss
## + ... omitted several edges
```

Salutations such as “miss”, “lady”, “sir”, “and”colonel" form common centers of nodes, which are often followed by names.

```r
set.seed(2017)

ggraph(bigram_graph, layout = "fr") +
  geom_edge_link() +
  geom_node_point() +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1)
```

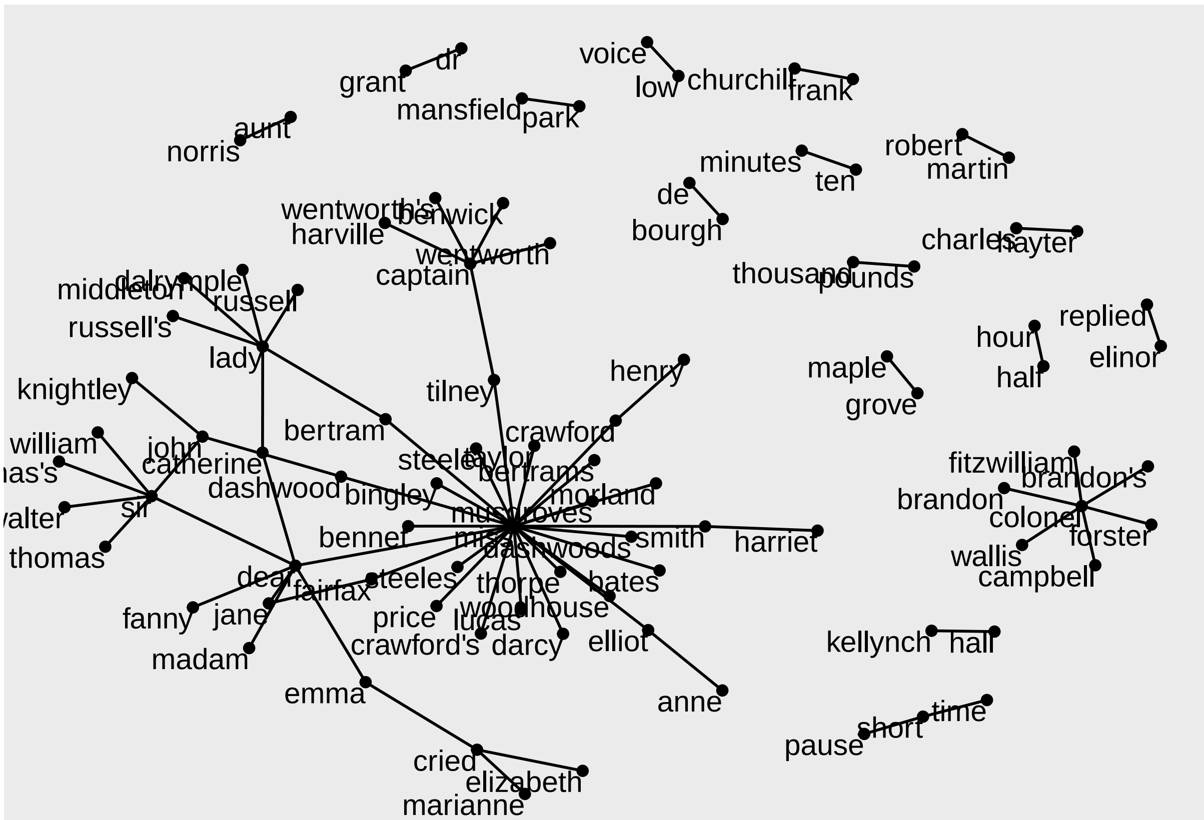



Figure 11: Common bigrams in Jane Austen’s novels, showing those that occurred more than 20 times and where neither word was a stop word

```r
set.seed(2016)

a <- grid::arrow(type = "closed", length = unit(.15, "inches"))

ggraph(bigram_graph, layout = "fr") +
  geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                 arrow = a, end_cap = circle(.07, 'inches')) +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
  theme_void()
```

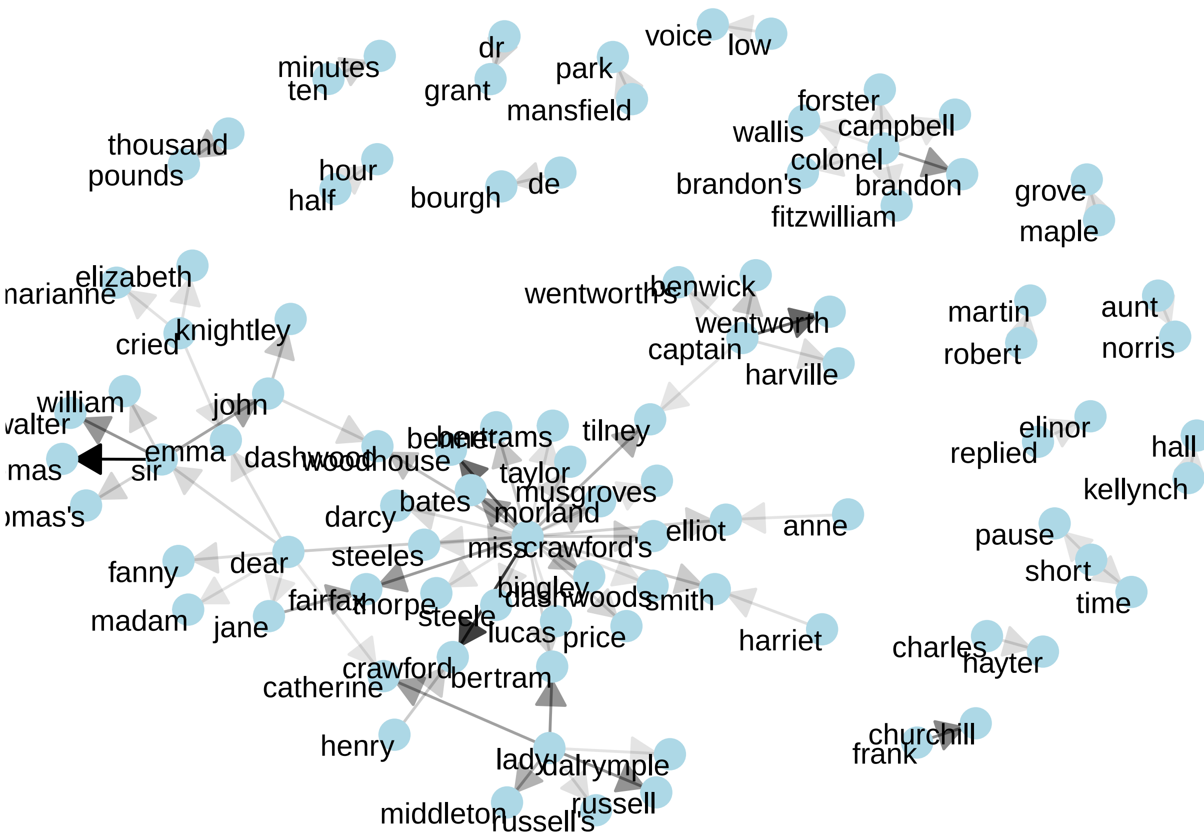


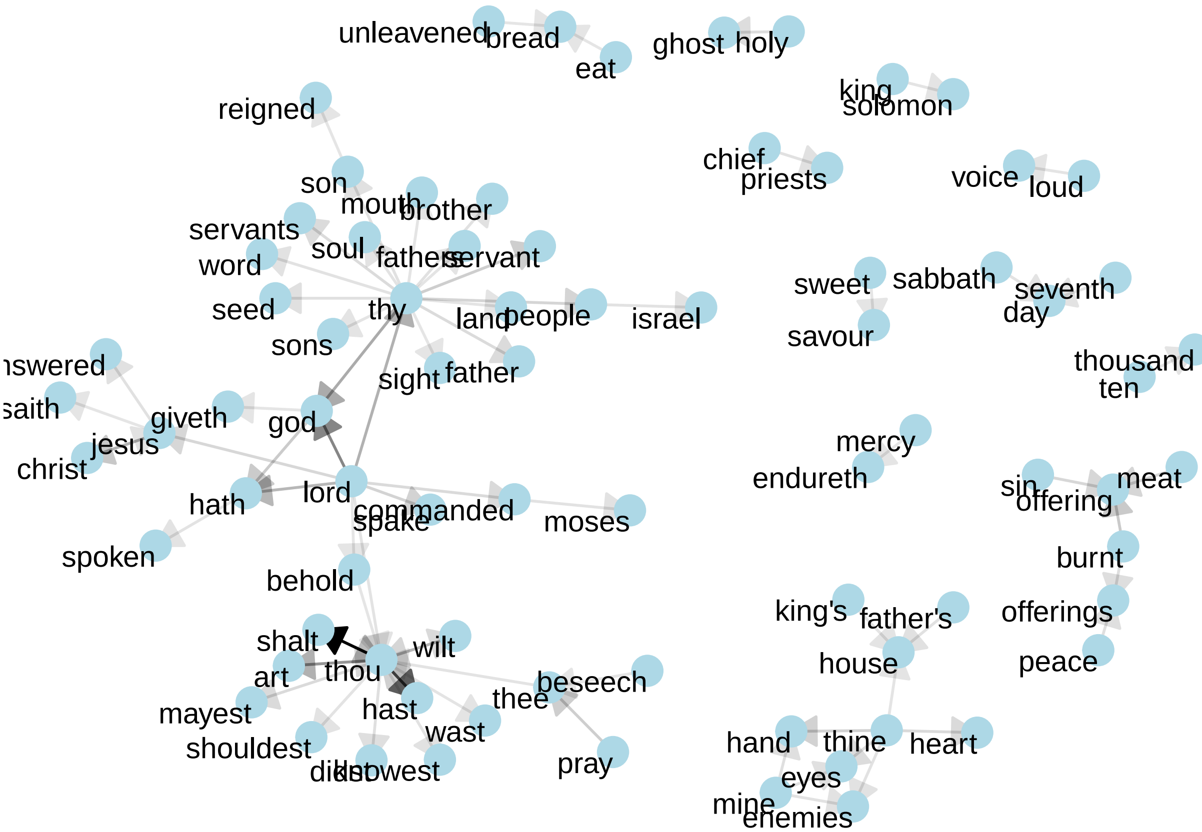

Figure 12: Common bigrams in Jane Austen's novels, with some polishing

```r
# edge_alpha aesthetic to the link layer to make links transparent
  # based on how common or rare the bigram is
# grid::arrow -> add directionality with an arrow, constructed using grid::arrow(),
  # including an end_cap option that tells the arrow to end before touching the node
```

### 4.1.5 Visualizing bigrams in other texts

```
# creating a function
count_bigrams <- function(dataset) {
  dataset %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
    separate(bigram, c("word1", "word2"), sep = " ") %>%
    filter(!word1 %in% stop_words$word,
           !word2 %in% stop_words$word) %>%
    count(word1, word2, sort = TRUE)
}

visualize_bigrams <- function(bigrams) {
  set.seed(2016)
  a <- grid::arrow(type = "closed", length = unit(.15, "inches"))

  bigrams %>%
    graph_from_data_frame() %>%
    ggraph(layout = "fr") +
    geom_edge_link(aes(edge_alpha = n), show.legend = FALSE, arrow = a) +
    geom_node_point(color = "lightblue", size = 5) +
    geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
    theme_void()
}

# Example using function using the King James Bible

kjv <- gutenberg_download(10)

kjv_bigrams <- kjv %>%
  count_bigrams()
```

```r
# filter out rare combinations, as well as digits
kjv_bigrams %>%
  filter(n > 40,
         !str_detect(word1, "\\d"),
         !str_detect(word2, "\\d")) %>%
  visualize_bigrams()
```



## 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-occurring 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)
```

```r
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 correlation

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

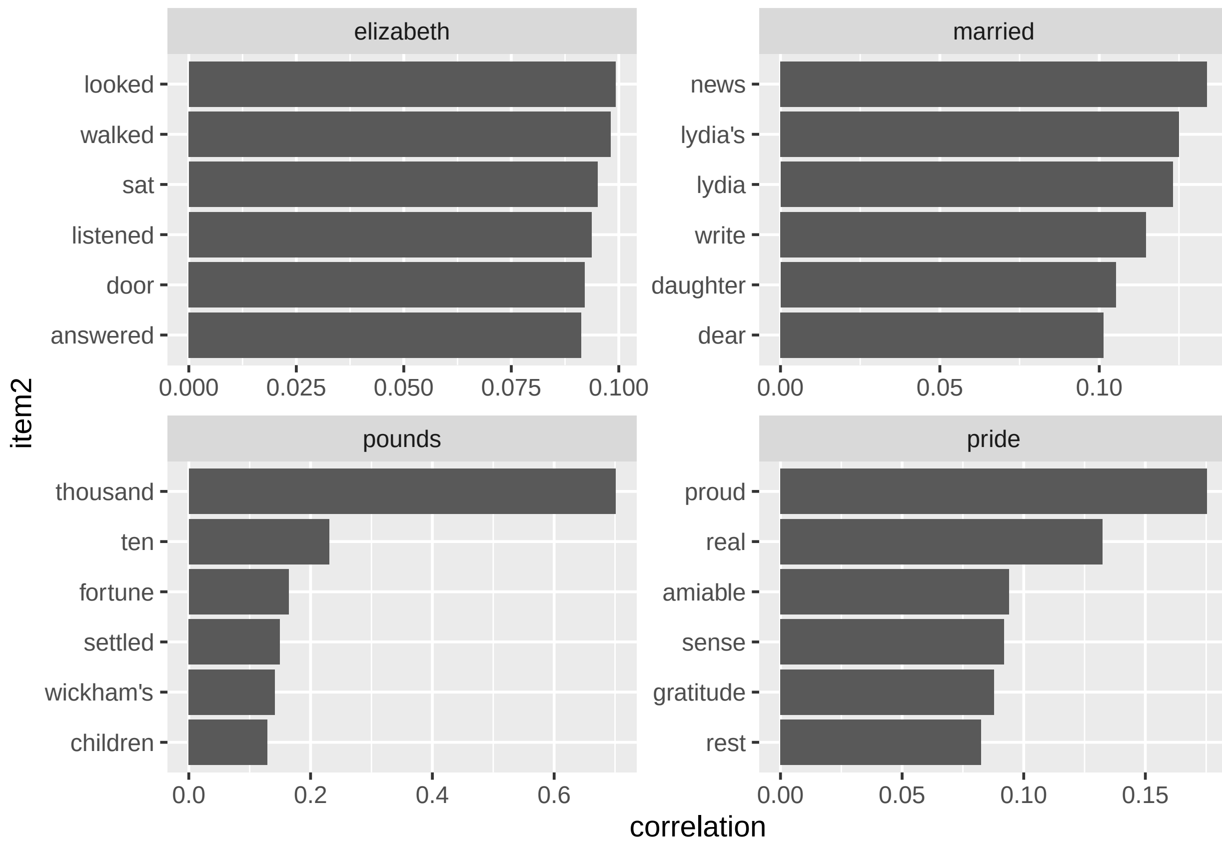
```
# filter for common words first

word_cors <-  austen_section_words %>%
  group_by(word) %>%
  filter(n() >= 20) %>%
  pairwise_cor(word, section, sort = TRUE)

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
```

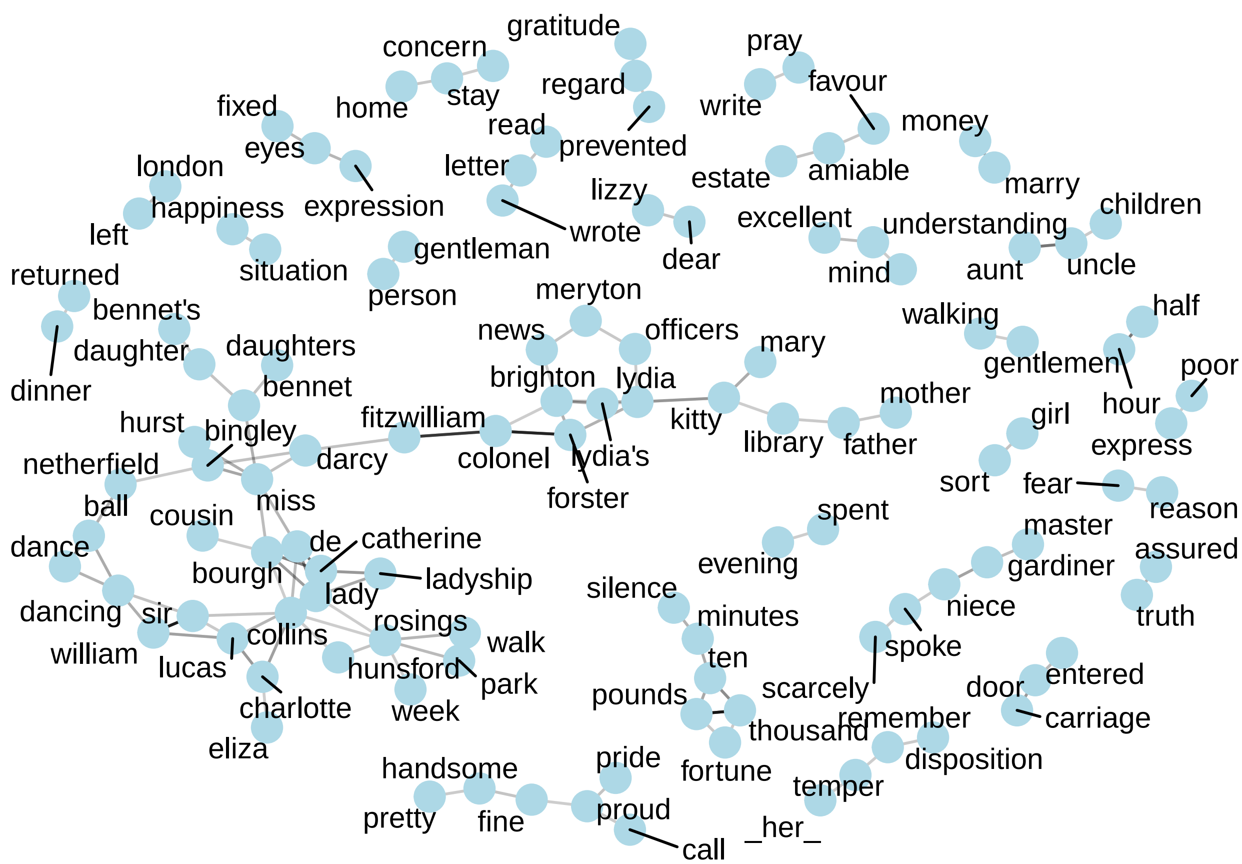
```
# find the words most correlated with a word like "pounds" using a filter operation.

word_cors %>%
  filter(item1 == "pounds")
```

```
## # 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
```

```r
# 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(item2, correlation)) +
  geom_bar(stat = "identity") +
  facet_wrap(~ item1, scales = "free") +
  coord_flip()
```



```r
# visualization

set.seed(2016)

word_cors %>%
  filter(correlation > .15) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
   geom_edge_link(aes(edge_alpha = correlation), show.legend = FALSE) +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void()
```




```r
# Note: relationships here are symmetrical and not directional as in the previous bigram example.
```

# 5 Converting to and from non-tidy formats

## 5.1 Tidying a document-term matrix

### 5.1.1 Tidying DocumentTermMatrix objects

```
data("AssociatedPress", package = "topicmodels")

AssociatedPress
```

```
## <<DocumentTermMatrix (documents: 2246, terms: 10473)>>
## Non-/sparse entries: 302031/23220327
## Sparsity           : 99%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

```
# documents = number of documents i.e. articles
# terms = distinct words

# access the terms in the document with the Terms() function

terms <-  Terms(AssociatedPress)
head(terms)
```

```
## [1] "aaron"      "abandon"    "abandoned"  "abandoning" "abbott"
## [6] "abboud"
```

```
#  using the tidy() verb, which takes a non-tidy object and turns it into a
  # tidy data frame, we can now analyze the data

ap_td <-  tidy(AssociatedPress)
ap_td
```

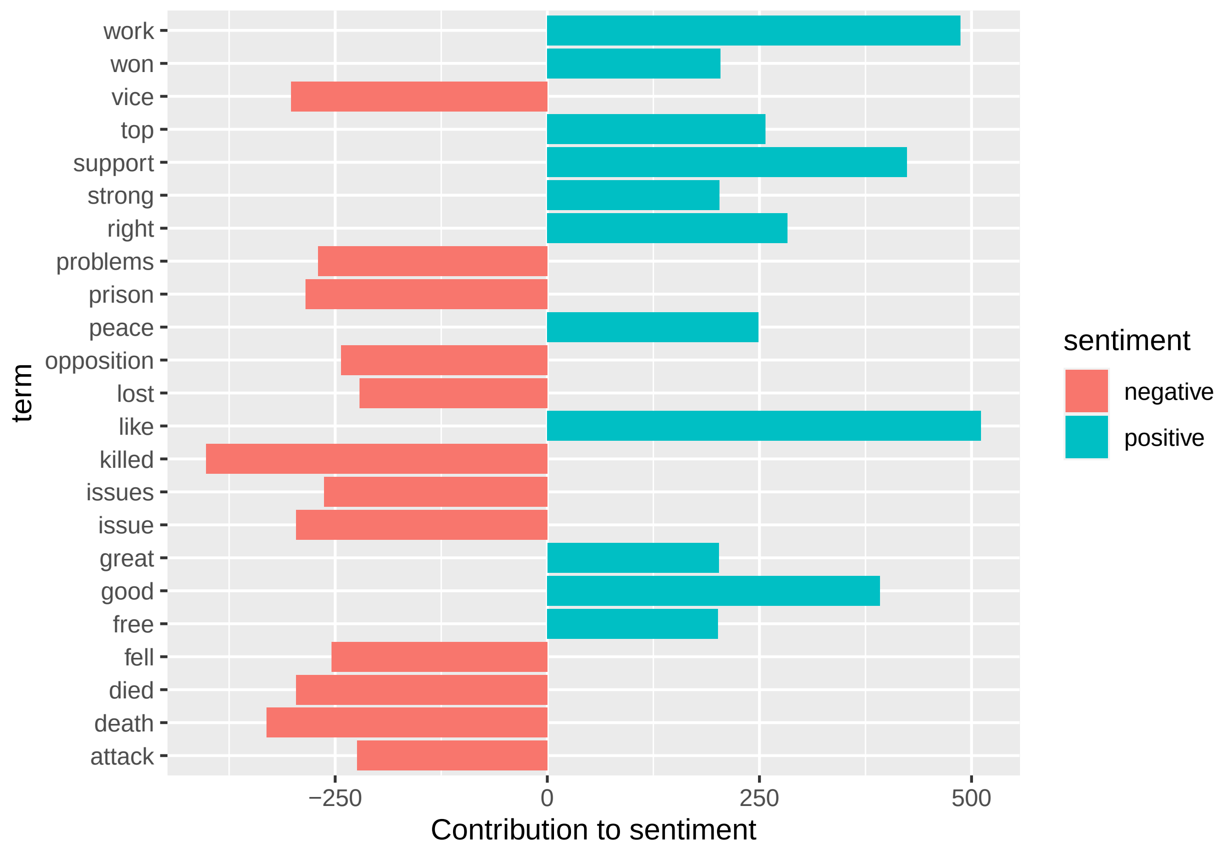
```
## # A tibble: 302,031 x 3
##    document term       count
##       <int> <chr>      <dbl>
##  1        1 adding         1
##  2        1 adult          2
##  3        1 ago            1
##  4        1 alcohol        1
##  5        1 allegedly      1
##  6        1 allen          1
##  7        1 apparently     2
##  8        1 appeared       1
##  9        1 arrested       1
## 10        1 assault        1
## # ... with 302,021 more rows
```

```
# sentiment analysis

ap_sentiments <-  ap_td %>%
  inner_join(get_sentiments("bing"), by = c(term = "word"))

ap_sentiments
```

```
## # A tibble: 30,094 x 4
##    document term    count sentiment
```

```
##       <int> <chr>   <dbl> <chr>
##  1        1 assault     1 negative
##  2        1 complex     1 negative
##  3        1 death       1 negative
##  4        1 died        1 negative
##  5        1 good        2 positive
##  6        1 illness     1 negative
##  7        1 killed      2 negative
##  8        1 like        2 positive
##  9        1 liked       1 positive
## 10        1 miracle     1 positive
## # ... with 30,084 more rows
```

```r
# visualization

ap_sentiments %>%
  count(sentiment, term, wt = count) %>%
  ungroup() %>%
  filter(n > 200) %>%
  mutate(n = ifelse(sentiment == "negative", -n, n)) %>%
  mutate(term = reorder(term, n)) %>%
  ggplot(aes(term, n, fill = sentiment)) +
  geom_bar(stat = "identity") +
  ylab("Contribution to sentiment") +
  coord_flip()
```



### 5.1.2 Tidying dfm objects

```r
# corpus of presidential inauguration speeches

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

inaug_dfm <- quanteda::dfm(data_corpus_inaugural, verbose = FALSE)
# dfm = document-feature-matrix

view(inaug_dfm)

# integrate into tidy

inaug_td <- tidy(inaug_dfm)
inaug_td
```

```
## # A tibble: 44,710 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 44,700 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: 44,710 x 6
##    document        term        count      tf   idf tf_idf
##    <chr>           <chr>       <dbl>   <dbl> <dbl>  <dbl>
##  1 1793-Washington arrive          1 0.00680  4.06 0.0276
##  2 1793-Washington upbraidings     1 0.00680  4.06 0.0276
##  3 1793-Washington violated        1 0.00680  3.37 0.0229
##  4 1793-Washington willingly       1 0.00680  3.37 0.0229
##  5 1793-Washington incurring       1 0.00680  3.37 0.0229
##  6 1793-Washington previous        1 0.00680  2.96 0.0201
##  7 1793-Washington knowingly       1 0.00680  2.96 0.0201
##  8 1793-Washington injunctions     1 0.00680  2.96 0.0201
##  9 1793-Washington witnesses       1 0.00680  2.96 0.0201
## 10 1793-Washington besides         1 0.00680  2.67 0.0182
## # ... with 44,700 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 = "",
       y = "tf-idf")
```

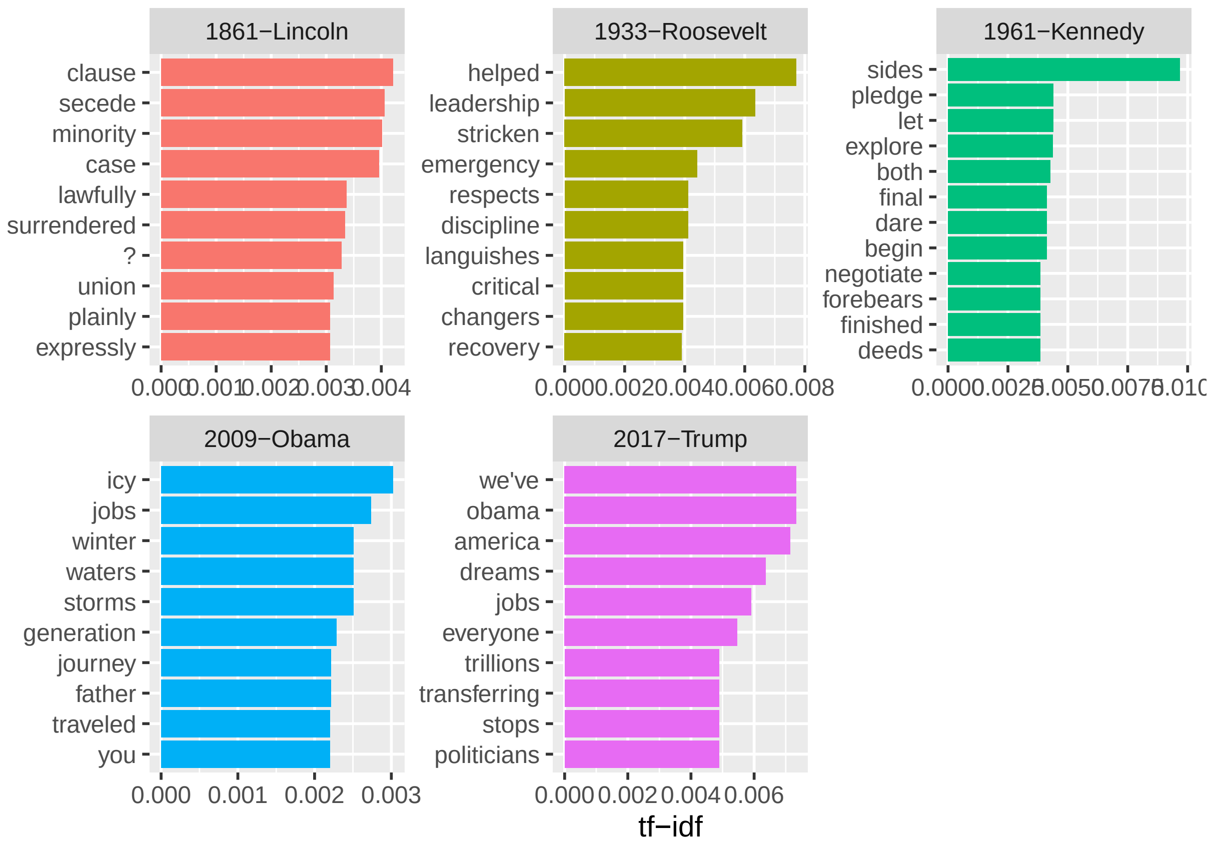



Figure 13: The terms with the highest tf-idf from each of four selected inaugural addresses. Note that quanteda's tokenizer includes the '?' punctuation mark as a term, though the texts we've tokenized ourselves with unnest_tokens do not.

```r
#  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")
```

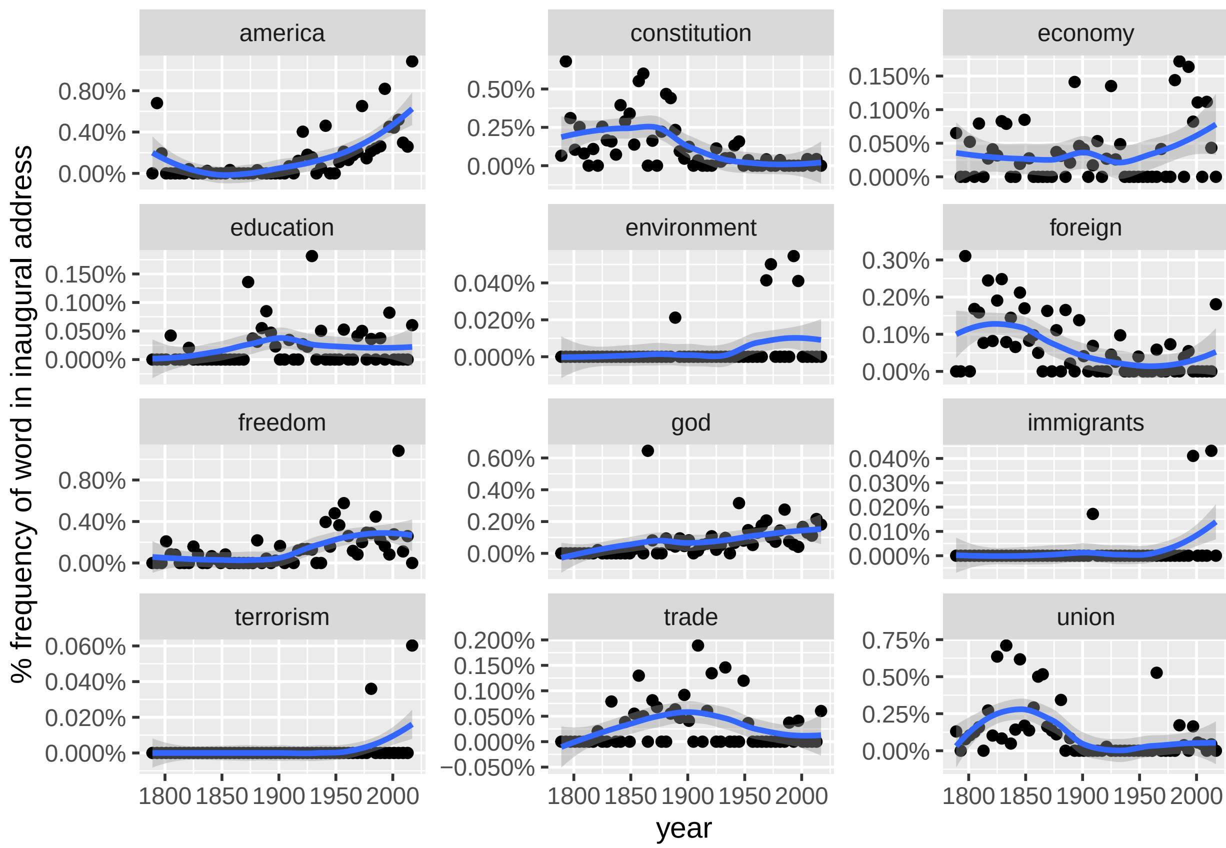



Figure 14: Changes in word frequency over time within Presidential inaugural addresses, for twelve selected terms

## 5.2 Casting tidy text data into a matrix

From tidy –> document-term matrix

```
ap_td %>%
  cast_dtm(document, term, count)
```

```
## <<DocumentTermMatrix (documents: 2246, terms: 10473)>>
## Non-/sparse entries: 302031/23220327
## Sparsity           : 99%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

```
# cast into a Matrix object

m <- ap_td %>%
  cast_sparse(document, term, count)

class(m)
```

```
## [1] "dgCMatrix"
## attr(,"package")
## [1] "Matrix"
```

```
dim(m)
```

```
## [1]  2246 10473
```

## 5.3 Tidying corpus objects with metadata

```
data("acq")
acq
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 50
```

```
# first document

acq[[1]]
```

```
## <<PlainTextDocument>>
## Metadata:  15
## Content:  chars: 1287
```

```
#  use the tidy() method to construct a table with one row per document

acq_td <- tidy(acq)
acq_td
```

```
## # A tibble: 50 x 16
##    author datetimestamp       description heading id    language origin topics
##    <chr>  <dttm>              <chr>       <chr>   <chr> <chr>    <chr>  <chr>
##  1 <NA>   1987-02-26 08:18:06 ""          COMPUT~ 10    en       Reute~ YES
##  2 <NA>   1987-02-26 08:19:15 ""          OHIO M~ 12    en       Reute~ YES
##  3 <NA>   1987-02-26 08:49:56 ""          MCLEAN~ 44    en       Reute~ YES
##  4 By Ca~ 1987-02-26 08:51:17 ""          CHEMLA~ 45    en       Reute~ YES
##  5 <NA>   1987-02-26 09:08:33 ""          <COFAB~ 68    en       Reute~ YES
##  6 <NA>   1987-02-26 09:32:37 ""          INVEST~ 96    en       Reute~ YES
##  7 By Pa~ 1987-02-26 09:43:13 ""          AMERIC~ 110   en       Reute~ YES
##  8 <NA>   1987-02-26 09:59:25 ""          HONG K~ 125   en       Reute~ YES
##  9 <NA>   1987-02-26 10:01:28 ""          LIEBER~ 128   en       Reute~ YES
## 10 <NA>   1987-02-26 10:08:27 ""          GULF A~ 134   en       Reute~ YES
## # ... with 40 more rows, and 8 more variables: lewissplit <chr>,
## #   cgisplit <chr>, oldid <chr>, places <named list>, people <lgl>, orgs <lgl>,
## #   exchanges <lgl>, text <chr>
```

This can then be used with unnest_tokens() to, for example, find the most common words across the 50 Reuters articles, or the ones most specific to each article.

```
acq_tokens <- acq_td %>%
  select(-places) %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words, by = "word")

# most commmon words

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

```
## # A tibble: 1,566 x 2
##    word        n
##    <chr>   <int>
##  1 dlrs      100
##  2 pct        70
```

```
##  3 mln         65
##  4 company     63
##  5 shares      52
##  6 reuter      50
##  7 stock       46
##  8 offer       34
##  9 share       34
## 10 american    28
## # ... with 1,556 more rows
```

```
# tf-idf

acq_tokens %>%
  count(id, word) %>%
  bind_tf_idf(word, id, n) %>% arrange(desc(tf_idf))
```

```
## # A tibble: 2,853 x 6
##    id    word         n     tf   idf tf_idf
##    <chr> <chr>    <int>  <dbl> <dbl>  <dbl>
##  1 186   groupe       2 0.133   3.91  0.522
##  2 128   liebert      3 0.130   3.91  0.510
##  3 474   esselte      5 0.109   3.91  0.425
##  4 371   burdett      6 0.103   3.91  0.405
##  5 442   hazleton     4 0.103   3.91  0.401
##  6 199   circuit      5 0.102   3.91  0.399
##  7 162   suffield     2 0.1     3.91  0.391
##  8 498   west         3 0.1     3.91  0.391
##  9 441   rmj          8 0.121   3.22  0.390
## 10 467   nursery      3 0.0968  3.91  0.379
## # ... with 2,843 more rows
```