

DATA607 - Assignment Week 10

Sentiment Analysis

Donald Butler

10/31/2021

Introduction

We were asked to recreate the R code from chapter 2 of the book “Text Mining with R: A Tidy Approach” then perform similar analysis on another authors set of works.

Load Packages

Load the packages required to complete the sentiment analysis.

```
library(tidyverse)
library(tidytext)
library(janeaustenr)
library(wordcloud)
library(reshape2)
```

Recreate Sentiment Analysis

Recreate the sentiment analysis from Chapter 2 in “Text Mining with R: A Tidy Approach.” (Silge and Robinson 2017)

Sentiment Lexicons

A sentiment lexicon is a set of words and a sentiment score that is attributed to each word.

AFINN-111

The AFINN sentiment lexicon scores words on a scale of -5 (negative sentiment) to 5 (positive sentiment). (Nielsen 2011)

```
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
```

Bing

The Bing sentiment lexicon assigns words as either positive or negative. (Hu and Liu 2004)

```
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
```

NRC Word-Emotion Association Lexicon

The nrc sentiment lexicon classifies words as positive or negative, but also by emotions such as fear or anger. (Mohammad and Turney 2013)

```
get_sentiments("nrc")
```

```
## # A tibble: 13,875 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 13,865 more rows
```

Sentiment in Jane Austen's Books

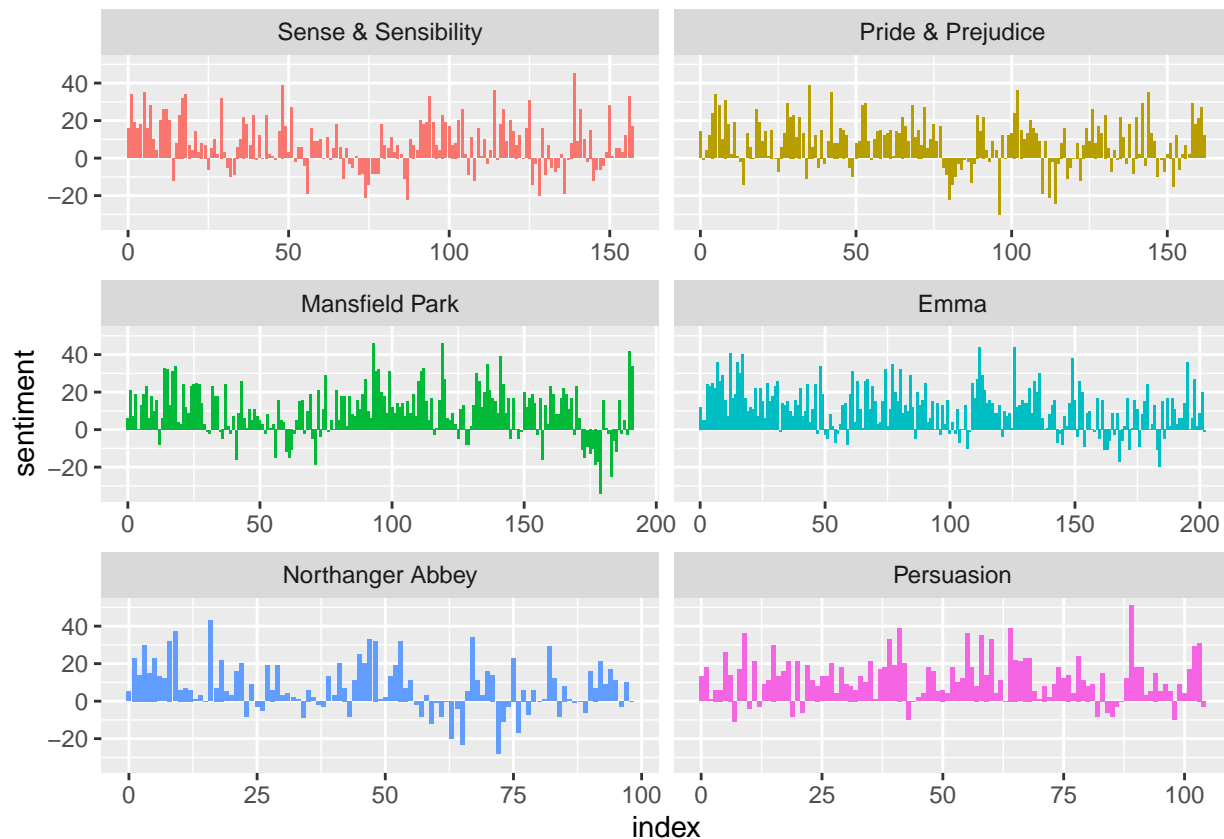
Using the Bing sentiment lexicon to compare the sentiment of all of Jane Austen's books.

```
lines_per_index <- 80

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)

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

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



Comparing Sentiment Lexicons

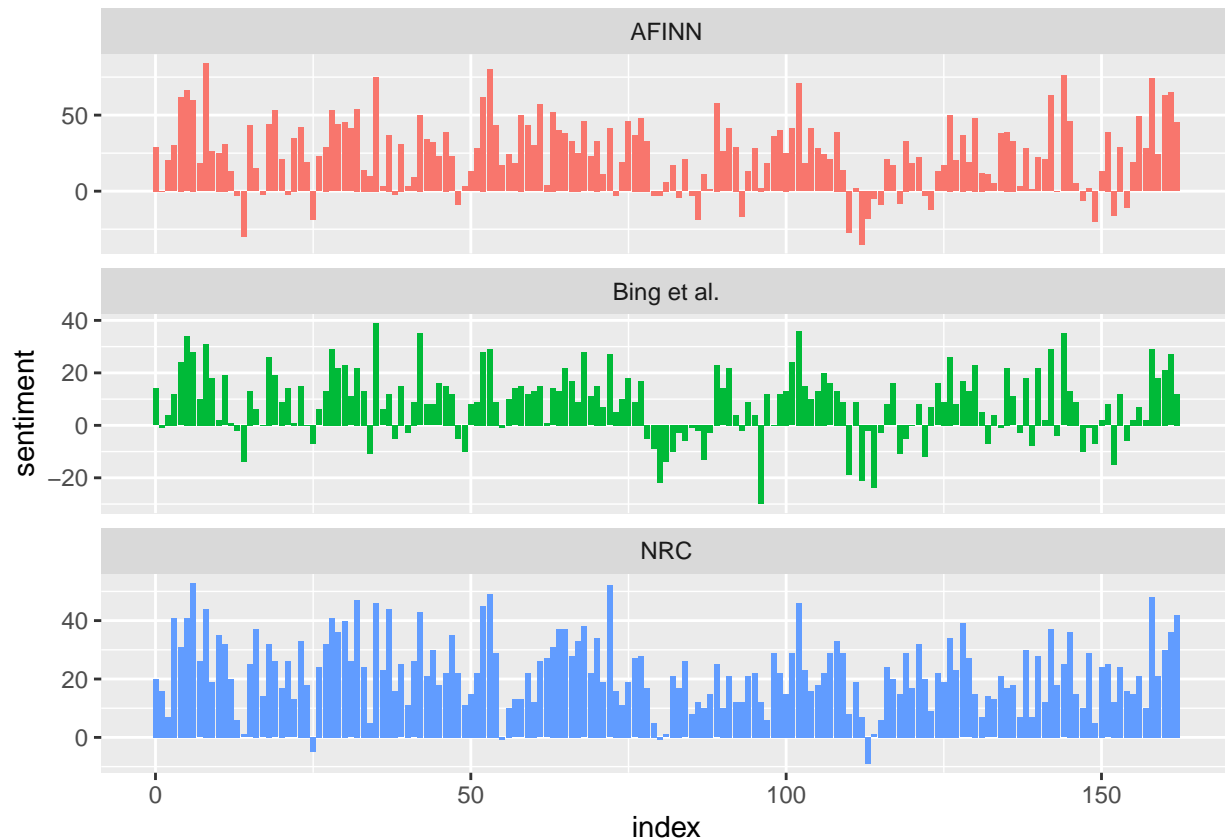
In Jane Austen's book "Pride and Prejudice," we will compare the three sentiment lexicons.

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

afinn <- pride_prejudice %>%
  inner_join(get_sentiments('afinn'), by = 'word') %>%
  group_by(index = linenummer %/% lines_per_index) %>%
  summarize(sentiment = sum(value)) %>%
  mutate(method = "AFINN")

bing_and_nrc <- bind_rows(
  pride_prejudice %>%
    inner_join(get_sentiments('bing'), by = 'word') %>%
    mutate(method = "Bing et al."),
  pride_prejudice %>%
    inner_join(get_sentiments('nrc') %>%
      filter(sentiment %in% c('positive', 'negative')), by = 'word') %>%
    mutate(method = 'NRC')
) %>%
  count(method, index = linenummer %/% lines_per_index, sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_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')
```

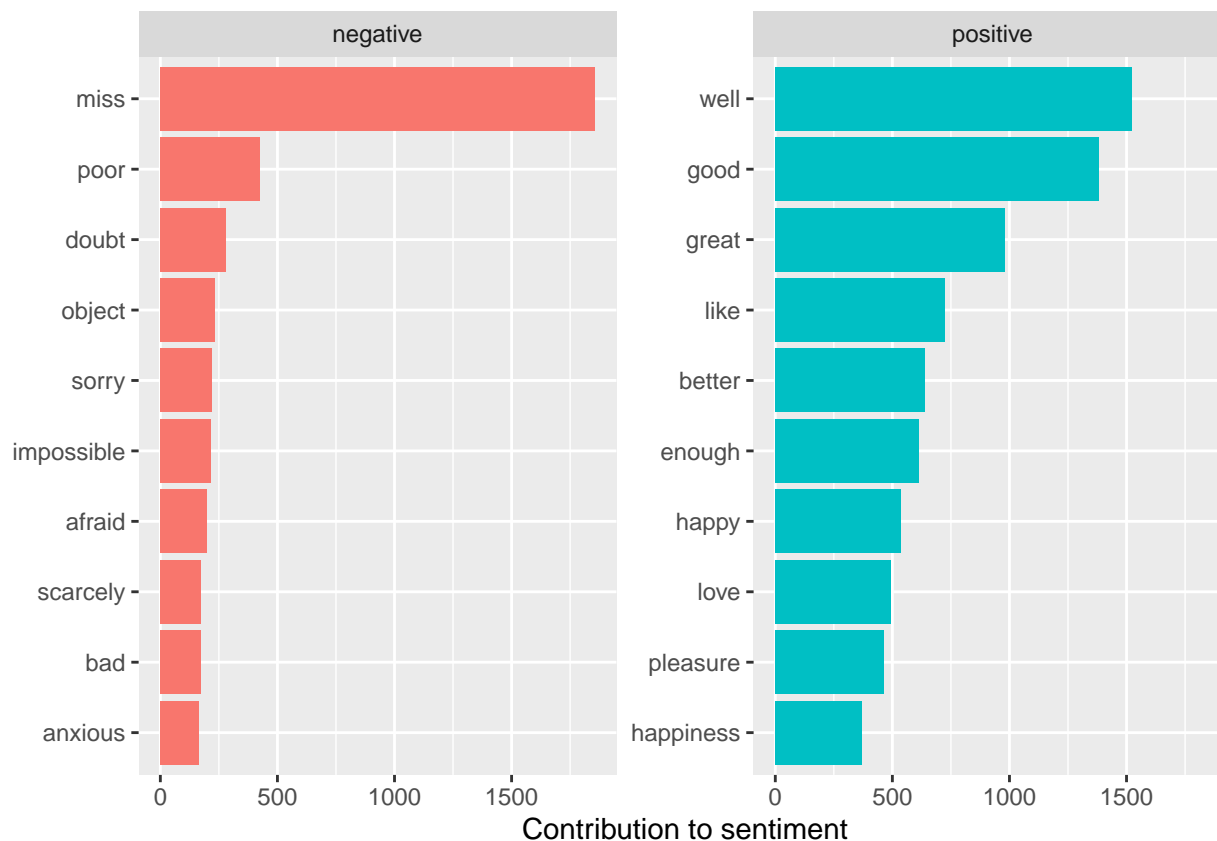


Common Positive and Negative Words

Determine the most common positive and negative words in Jane Austen's books.

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

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



Custom Stop Words

The word 'miss' is contributing significantly towards negative sentiment, but its use within Jane Austen's books isn't used in that context. Create a custom stop word list to adjust for anomalies in the sentiment lexicon used.

```
custom_stop_words <- bind_rows(tibble(word = c('miss'), lexicon = c('custom')), stop_words)
```

```
custom_stop_words
```

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

Construct a wordcloud of the most common words in Jane Austen's books.

```
## Warning in wordcloud(word, n, max.words = 100): happiness could not be fit on
## page. It will not be plotted.
```



Construct a wordcloud of the positive and negative words in Jane Austen's books.

```
tidy_books %>%
  inner_join(get_sentiments('bing'), by = 'word') %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = 'n', fill = 0) %>%
  comparison.cloud(colors = c('gray20', 'gray80'), max.words = 100)
```

negative



Sentiment based on other units

Instead of considering sentiment on a word for word basis, we can look at it based on other units of text such as chapters. Determine the most negative chapter in each book.

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

austen_chapters %>%
  group_by(book) %>%
  summarize(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
```



```
bing_negative <- get_sentiments('bing') %>%
  filter(sentiment == 'negative')

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

'summarise()' has grouped output by 'book'. You can override using the '.groups' argument.

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

'summarise()' has grouped output by 'book'. You can override using the '.groups' argument.

```
## # 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
```

Sir Arthur Conan Doyle

Perform a similar analysis based on the works of author Sir Arthur Conan Doyle.

Project Gutenberg

Project Gutenberg is a library of public domain works. The `gutenbergr` package allows access to these works. (Robinson 2021) Retrieve a list of the Sherlock Holmes books that are available.

```
library(gutenbergr)

doyle_books <- gutenberg_works(author == 'Doyle, Arthur Conan') %>%
  filter(gutenberg_bookshelf == 'Detective Fiction')

doyle_books
```

```
## # A tibble: 13 x 8
##   gutenberg_id title      author gutenberg_autho~ language gutenberg_books~ rights
```

```
##           <int> <chr>   <chr>                <int> <chr>   <chr>                <chr>
## 1           108 The Re~ Doyle~                69 en      Detective Ficti~ Publi~
## 2           244 A Stud~ Doyle~                69 en      Detective Ficti~ Publi~
## 3           834 The Me~ Doyle~                69 en      Detective Ficti~ Publi~
## 4          2097 The Si~ Doyle~                69 en      Detective Ficti~ Publi~
## 5          2343 The Ad~ Doyle~                69 en      Detective Ficti~ Publi~
## 6          2344 The Ad~ Doyle~                69 en      Detective Ficti~ Publi~
## 7          2345 The Ad~ Doyle~                69 en      Detective Ficti~ Publi~
## 8          2346 The Ad~ Doyle~                69 en      Detective Ficti~ Publi~
## 9          2347 The Ad~ Doyle~                69 en      Detective Ficti~ Publi~
## 10         2348 The Di~ Doyle~                69 en      Detective Ficti~ Publi~
## 11         2349 The Ad~ Doyle~                69 en      Detective Ficti~ Publi~
## 12         2350 His La~ Doyle~                69 en      Detective Ficti~ Publi~
## 13         3289 The Va~ Doyle~                69 en      Detective Ficti~ Publi~
## # ... with 1 more variable: has_text <lgl>
```

With the list of books, download the works and tidy.

```
tidy_doyle <- doyle_books %>%
  gutenberg_download(meta_fields = 'title') %>%
  group_by(gutenberg_id) %>%
  mutate(linenumber = row_number()) %>%
  ungroup() %>%
  unnest_tokens(word, text)
```

```
## Determining mirror for Project Gutenberg from http://www.gutenberg.org/robot/harvest
```

```
## Using mirror http://aleph.gutenberg.org
```

Unfortunately, in this collection of works, chapter pages are not present, so we are unable to consider a units by chapter.

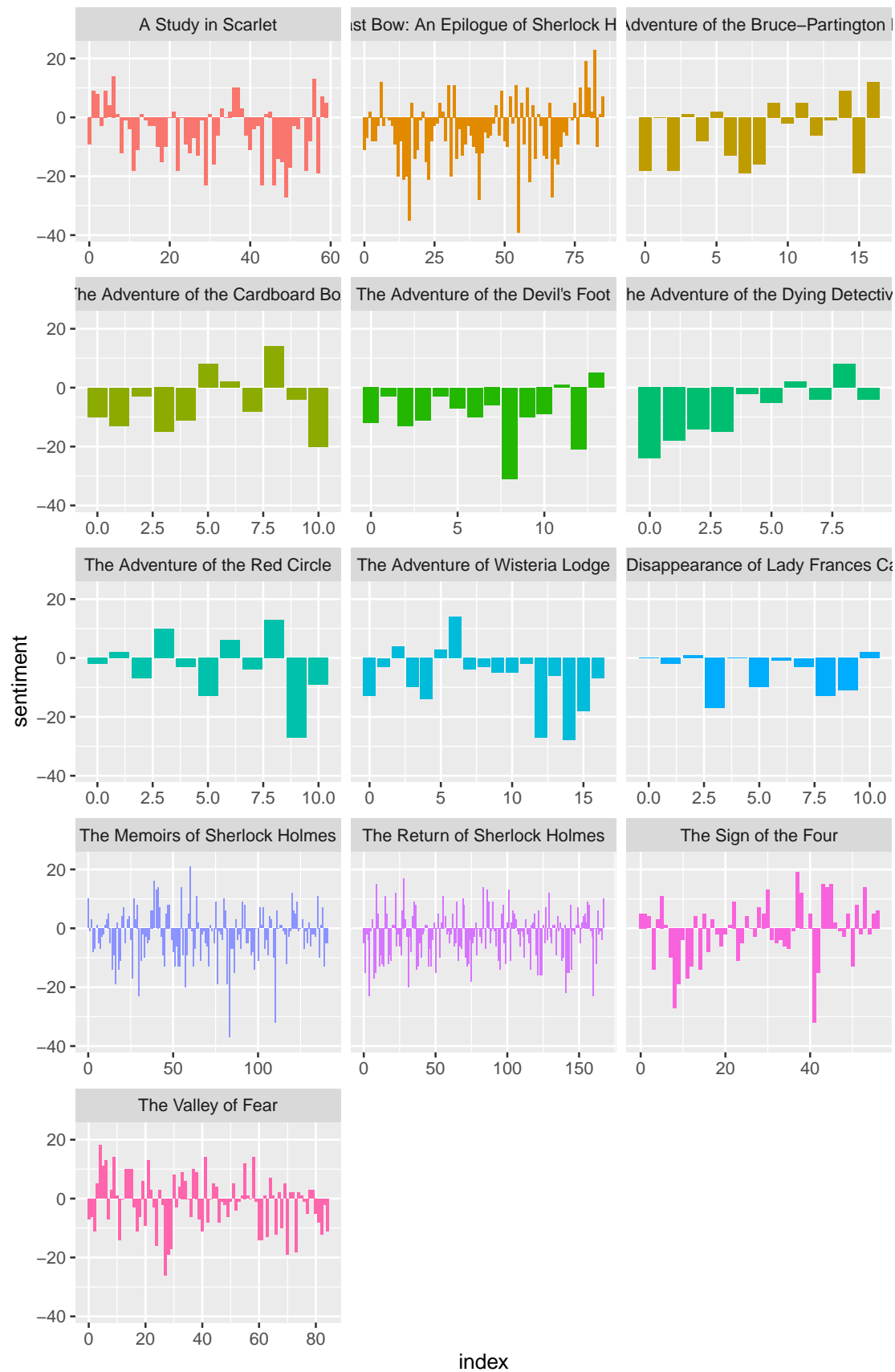
Sentiment in Sir Arthur Conan Doyal's Books

Using the Bing sentiment lexicon to compare the sentiment of all of the Sherlock Holmes books.

```
lines_per_index <- 80

doyle_sentiment <- tidy_doyle %>%
  inner_join(get_sentiments('bing'), by = 'word') %>%
  count(title, index = linenumber %% lines_per_index, sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(sentiment = positive - negative)

doyle_sentiment %>%
  ggplot(aes(index, sentiment, fill = title)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~title, ncol = 3, scales = "free_x")
```



Comparing Sentiment Lexicons

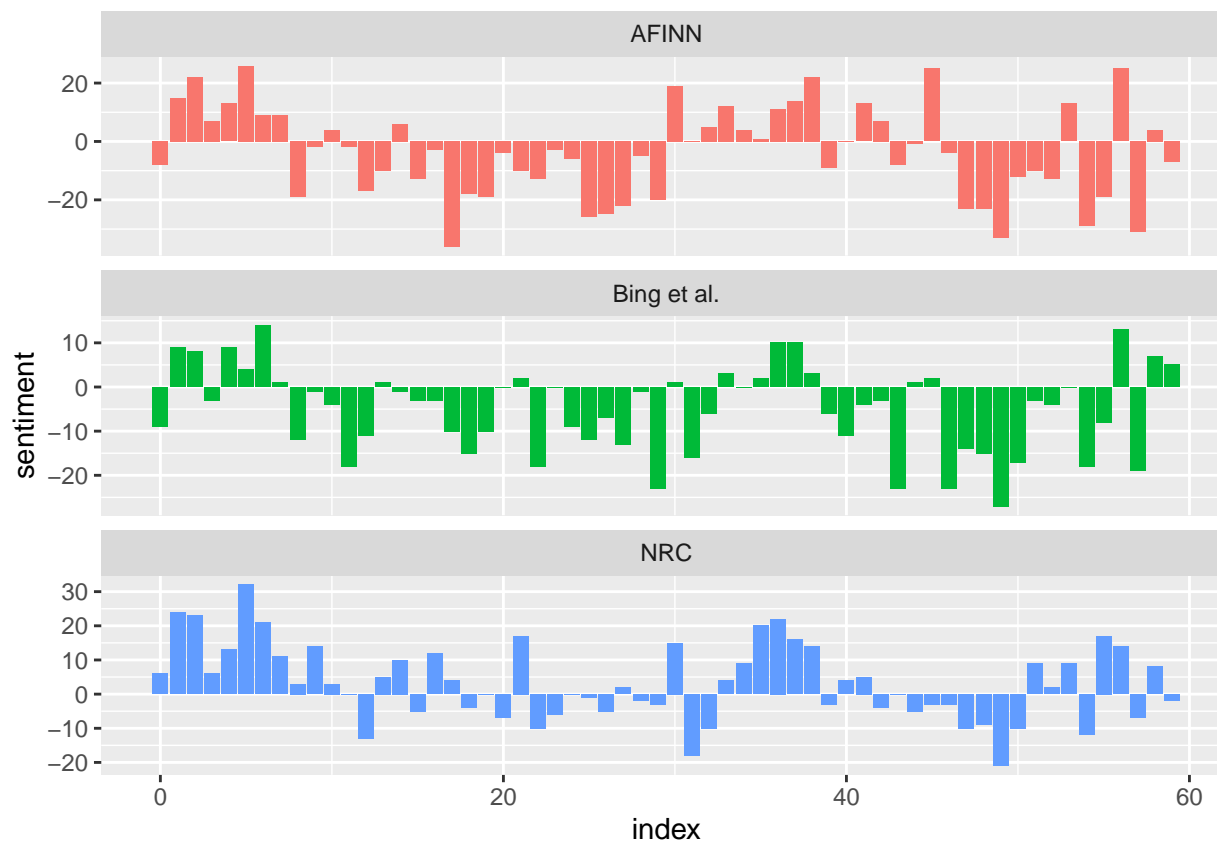
In “A Study in Scarlet,” we will compare the three sentiment lexicons.

```
study_scarlet <- tidy_doyle %>%
  filter(title == "A Study in Scarlet")

scarlet_afinn <- study_scarlet %>%
  inner_join(get_sentiments('afinn'), by = 'word') %>%
  group_by(index = linenumber %% lines_per_index) %>%
  summarize(sentiment = sum(value)) %>%
  mutate(method = "AFINN")

scarlet_bing_and_nrc <- bind_rows(
  study_scarlet %>%
    inner_join(get_sentiments('bing'), by = 'word') %>%
    mutate(method = "Bing et al."),
  study_scarlet %>%
    inner_join(get_sentiments('nrc') %>%
      filter(sentiment %in% c('positive', 'negative')), by = 'word') %>%
    mutate(method = 'NRC')
) %>%
  count(method, index = linenumber %% lines_per_index, sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(sentiment = positive - negative)

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



Syuzhet

I tried playing with the Syuzhet package to use another sentiment lexicon. (**syuzhet?**) The package computes a sentiment score for each line of the book text.

```
library(syuzhet)

study_in_scarlet <- doyle_books %>%
  filter(title == 'A Study in Scarlet') %>%
  gutenbergl_download(meta_fields = 'title') %>%
  filter(grepl('[A-Za-z]',text))

syuzhet_sentiment <- get_sentiment(study_in_scarlet$text, method = 'syuzhet')
summary(syuzhet_sentiment)
```

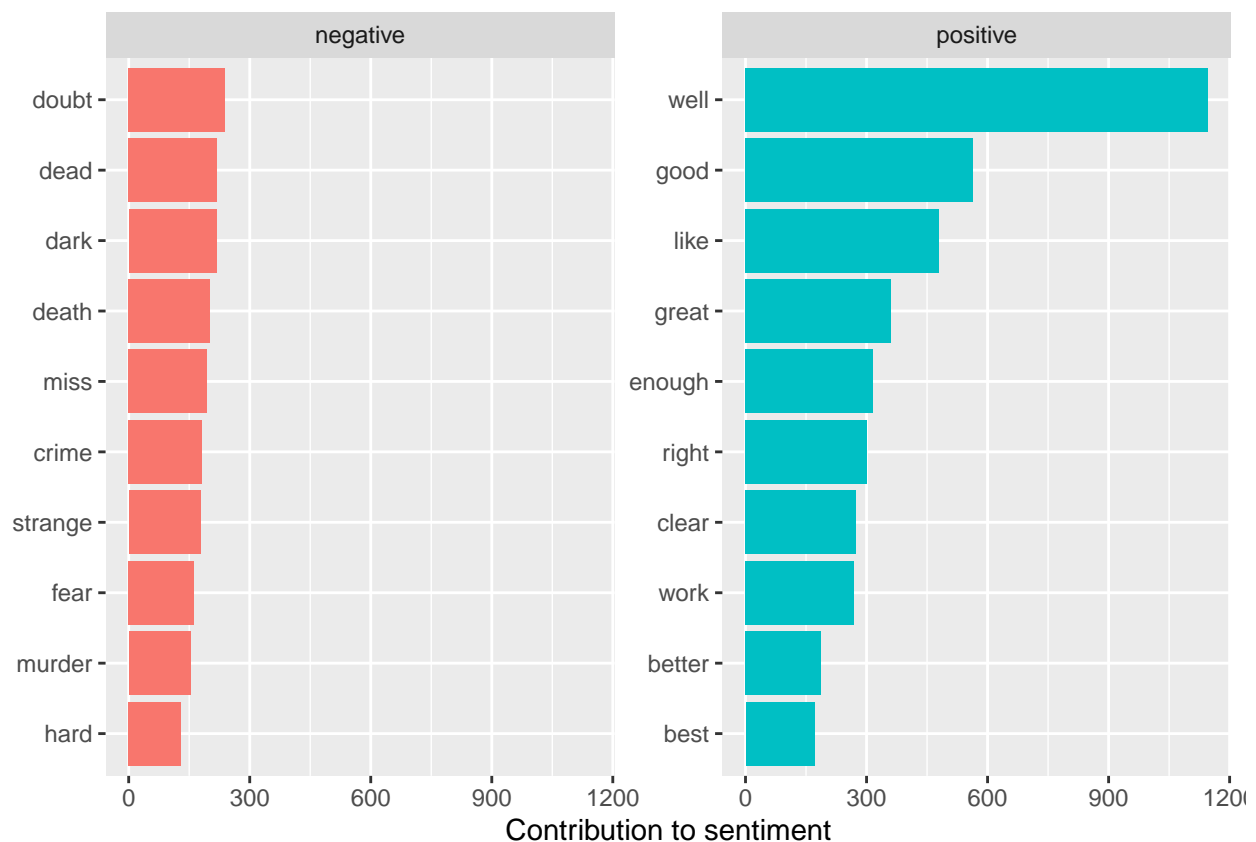
```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.60000 -0.40000  0.00000  0.01536  0.50000  2.60000
```

Common Positive and Negative Words

Determine the most common positive and negative words in the Sherlock Holmes books.

```
doyle_bing_word_counts <- tidy_doyle %>%
  inner_join(get_sentiments('bing'), by = 'word') %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()

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



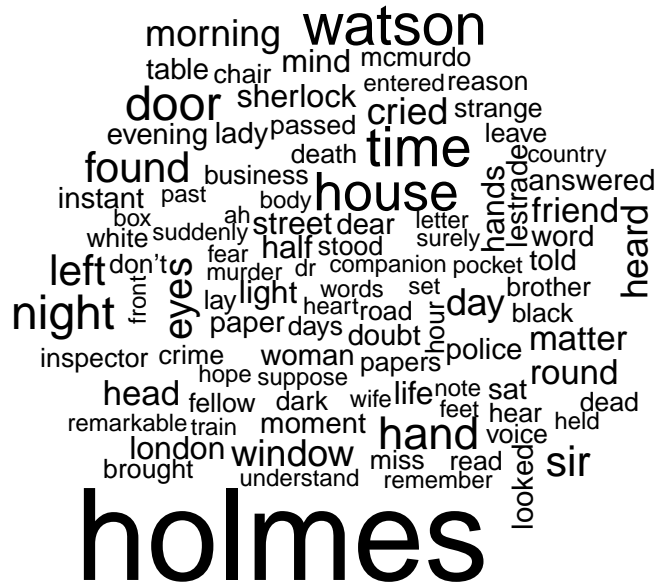
Custom Stop Words

Again we see the word 'miss' on the common negative list, although not as impacting with Jane Austen's work, but based on the time period I would expect a fair number of instances are referring to a women and not the negative context.

Wordclouds

Construct a wordcloud of the most common words in the Sherlock Holmes books.

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

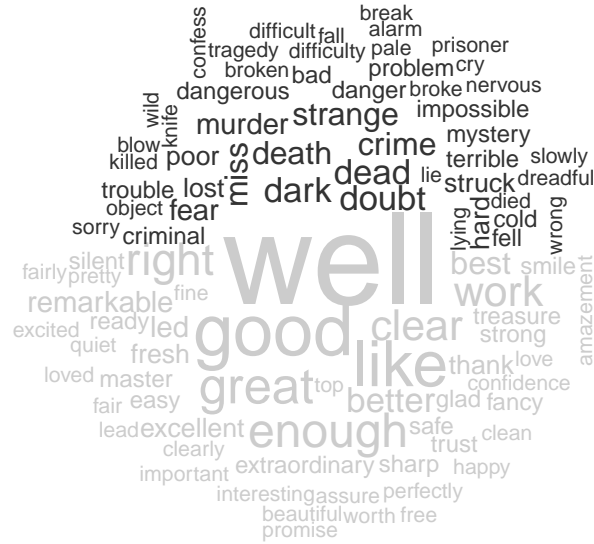


Wordclouds by Sentiment

Construct a wordcloud of the positive and negative words in Jane Austen's books.

```
tidy_doyle %>%
  inner_join(get_sentiments('bing'), by = 'word') %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = 'n', fill = 0) %>%
  comparison.cloud(colors = c('gray20', 'gray80'), max.words = 100)
```

negative



positive

Sentiment based on other units

Unfortunately the chapter pages that were in Jane Austen's books, do not appear in the files for Sir Arthur Conan Doyle, so a similar analysis cannot be performed.

References

- Hu, Mingming, and Bing Liu. 2004. “Mining and Summarizing Customer Reviews,” KDD ’04, 168–77. <https://doi.org/10.1145/1014052.1014073>.
- Mohammad, Saif M., and Peter D. Turney. 2013. “Crowdsourcing a Word-Emotion Association Lexicon.” *Computational Intelligence* 29 (3): 436–65. <https://doi.org/10.1111/j.1467-8640.2012.00460.x>.
- Nielsen, F. Å. 2011. “AFINN.” Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby: Informatics; Mathematical Modelling, Technical University of Denmark. <http://www2.compute.dtu.dk/pubdb/pubs/6010-full.html>.
- Robinson, David. 2021. *Gutenbergr: Download and Process Public Domain Works from Project Gutenberg*. <https://CRAN.R-project.org/package=gutenbergr>.
- Silge, Julia, and David Robinson. 2017. *Text Mining with r: A Tidy Approach*. 1st ed. O’Reilly Media, Inc. <https://www.tidytextmining.com>.