02 Sentiment Analysis with Tidy Data

In this project, I want to address the topic of opinion mining or sentiment analysis. When human readers approach a text, we use our understanding of the emotional intent of words to infer whether a section of text is positive or negative, or perhaps characterized by some other more nuanced emotion like surprise or disgust. I want to use the tools of text mining to approach the emotional content of text programmatically.

One way to analyze the sentiment of a text is to consider the text as a combination of its individual words and the sentiment content of the whole text as the sum of the sentiment content of the individual words. This isn't the only way to approach sentiment analysis, but it is an often-used approach, and an approach that naturally takes advantage of the tidy tool ecosystem.

02_01 The Sentiments Dataset

```
sentiments
```

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

There are three general-purpose lexicons. All three of these lexicons are based on unigrams, i.e., single words. These lexicons contain many English words and the words are assigned scores for positive/negative sentiment, and also possibly emotions like joy, anger, sadness, and so forth.

This three lexicons are:

- AFINN lexicon from Finn Årup Nielsen, which assigns words with a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment.
- bing lexicon from Bing Liu and collaborators, which categorizes words in a binary fashion into positive and negative categories.
- nrc lexicon from Saif Mohammad and Peter Turney, which categorizes words in a binary fashion ("yes"/"no") into categories of positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust.

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

```
## # A tibble: 2,476 × 2
##          word score
##         <chr> <int>
## 1     abandon    -2
## 2   abandoned    -2
## 3    abandons    -2
## 4    abducted    -2
## 5   abduction    -2
## 6  abductions    -2
## 7       abhor    -3
## 8    abhorred    -3
## 9   abhorrent    -3
## 10     abhors    -3
## # ... with 2,466 more rows
```

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

```
## # A tibble: 6,788 × 2
##            word sentiment
##           <chr>     <chr>
## 1       2-faced  negative
## 2       2-faces  negative
## 3            a+  positive
## 4      abnormal  negative
## 5       abolish  negative
## 6    abominable  negative
## 7    abominably  negative
## 8     abominate  negative
## 9   abomination  negative
## 10        abort  negative
## # ... with 6,778 more rows
```

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

```
## # A tibble: 13,901 × 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
```

02_02 Sentiment Analysis with Inner Join

With data in a tidy format, sentiment analysis can be done as an inner join. This is another of the great successes of viewing text mining as a tidy data analysis task. So I want to look at the words with a joy score from the NRC lexicon. I ask myself, what the most common joy words in Emma are.

First, I take the text of the novels and convert the text to the tidy format. Then I set up some other columns to keep track of which line and chapter of the book each word comes from.

```
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)
```

Now that the text is in a tidy format with one word per row, I will do the sentiment analysis.

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

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

```
## Joining, by = "word"
```

```
## # A tibble: 303 × 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
```

Next, I count up how many positive and negative words there are in defined sections of each book. I define an index here to keep track of where I am in the narrative, which (using integer division) counts up sections of 100 lines of text.

Small sections of text may not have enough words in them to get a good estimate of sentiment while really large sections can wash out narrative structure. For these books, using 80 lines works well, but this can vary depending on individual texts, how long the lines were to start with, etc.

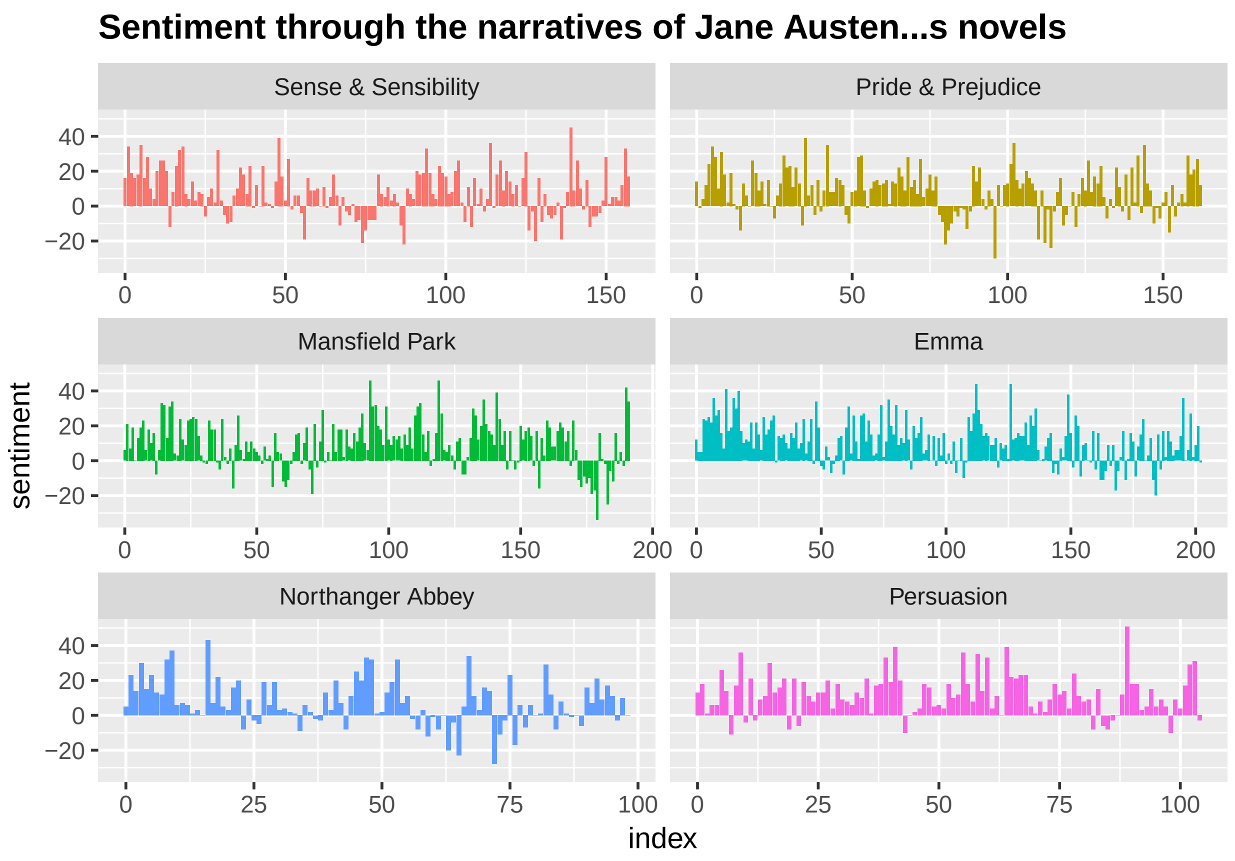
```
janeaustensentiment <- tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(book, index = linenumber %/% 80, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)
```

```
## Joining, by = "word"
```

Now I can plot these sentiment scores across the plot trajectory of each novel.

```
ggplot(janeaustensentiment, aes(index, sentiment, fill = book)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~book, ncol = 2, scales = "free_x") +
  ggtitle("Sentiment through the narratives of Jane Austen's novels") +
     theme(plot.title = element_text(lineheight=.8, face="bold"))
```

```
## Warning in grid.Call(L_stringMetric, as.graphicsAnnot(x$label)): Fontmetrik
## für das Zeichen 0x1e unbekannt
## Warning in grid.Call(L_stringMetric, as.graphicsAnnot(x$label)): Fontmetrik
## für das Zeichen 0x80 unbekannt

## Warning in grid.Call(L_stringMetric, as.graphicsAnnot(x$label)):
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <e2>

## Warning in grid.Call(L_stringMetric, as.graphicsAnnot(x$label)):
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <80>

## Warning in grid.Call(L_stringMetric, as.graphicsAnnot(x$label)):
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <99>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <e2>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <80>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <99>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <e2>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <80>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <99>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <e2>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <80>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <99>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <e2>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <80>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen's
## novels' in 'mbcsToSbcs': Punkt ersetzt <99>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen’s
## novels' in 'mbcsToSbcs': Punkt ersetzt <e2>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen’s
## novels' in 'mbcsToSbcs': Punkt ersetzt <80>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen’s
## novels' in 'mbcsToSbcs': Punkt ersetzt <99>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen’s
## novels' in 'mbcsToSbcs': Punkt ersetzt <e2>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen’s
## novels' in 'mbcsToSbcs': Punkt ersetzt <80>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen’s
## novels' in 'mbcsToSbcs': Punkt ersetzt <99>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen’s
## novels' in 'mbcsToSbcs': Punkt ersetzt <e2>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen’s
## novels' in 'mbcsToSbcs': Punkt ersetzt <80>

## Warning in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## Konvertierungsfehler für 'Sentiment through the narratives of Jane Austen’s
## novels' in 'mbcsToSbcs': Punkt ersetzt <99>

## Warning in grid.Call.graphics(L_text, as.graphicsAnnot(x$label), x$x, x
## $y, : Konvertierungsfehler für 'Sentiment through the narratives of Jane
## Austen’s novels' in 'mbcsToSbcs': Punkt ersetzt <e2>

## Warning in grid.Call.graphics(L_text, as.graphicsAnnot(x$label), x$x, x
## $y, : Konvertierungsfehler für 'Sentiment through the narratives of Jane
## Austen’s novels' in 'mbcsToSbcs': Punkt ersetzt <80>

## Warning in grid.Call.graphics(L_text, as.graphicsAnnot(x$label), x$x, x
## $y, : Konvertierungsfehler für 'Sentiment through the narratives of Jane
## Austen’s novels' in 'mbcsToSbcs': Punkt ersetzt <99>
```



I can see, how the plot of each novel changes toward more positive or negative sentiment over the trajectory of the story.

02_03 Comparing the three Sentiment Dictionaries

Next, I want to use all three sentiment lexicons and examine how the sentiment changes across the narrative arc of Pride and Prejudice. First, I choose only the words from the one novel I am interested in.

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

pride_prejudice
```

```
## # A tibble: 122,204 × 4
##                  book linenumber chapter      word
##                <fctr>      <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
```

In the next step, I define larger sections of text that span multiple lines. My aim is to find the net sentiment in each of these sections of text.

```
afinn <- pride_prejudice %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(index = linenumber %/% 80) %>%
  summarise(sentiment = sum(score)) %>%
  mutate(method = "AFINN")
```

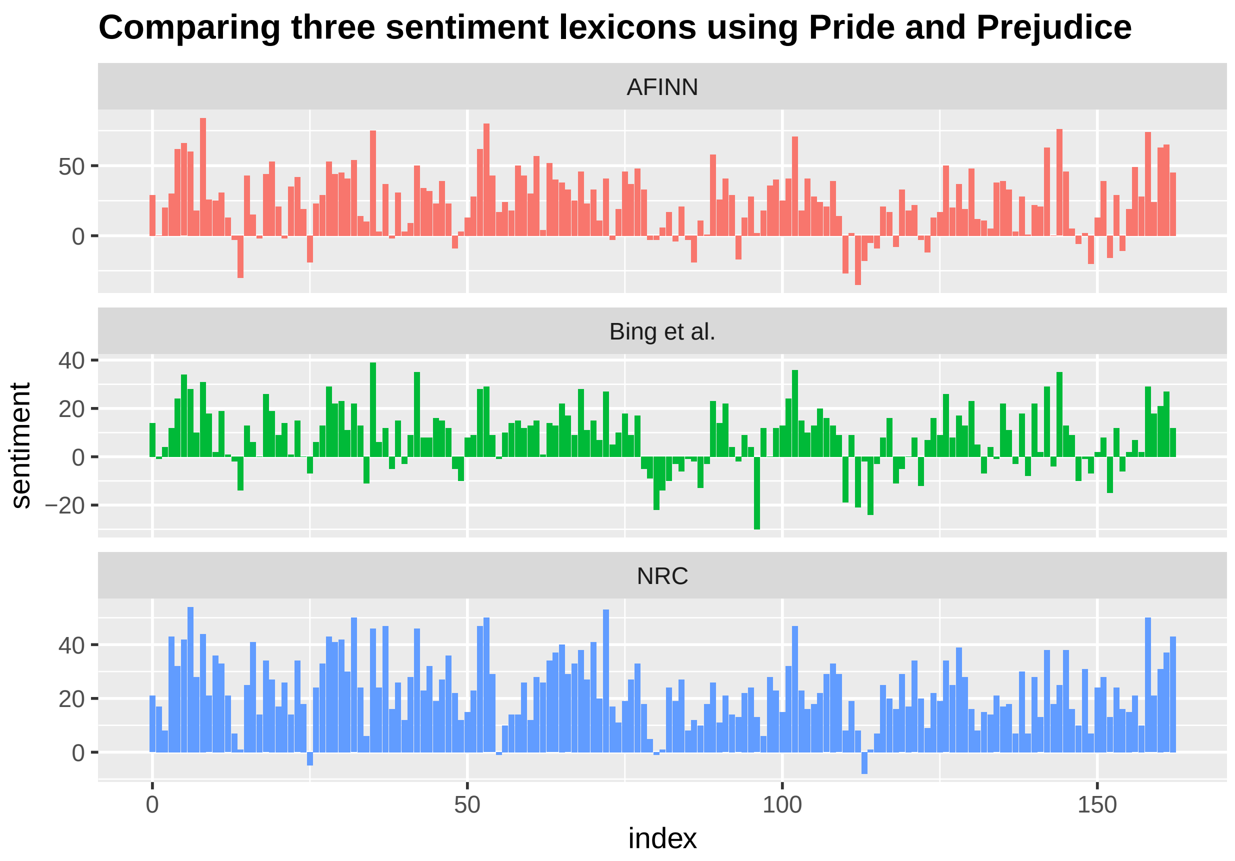
```
## Joining, by = "word"
```

```
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)
```

```
## Joining, by = "word"
## Joining, by = "word"
```

I now have an estimate of the net sentiment (positive - negative) in each chunk of the novel text for each sentiment lexicon. Next, I bind them together and visualize them.

```
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") +
  ggtitle("Comparing three sentiment lexicons using Pride and Prejudice") +
     theme(plot.title = element_text(lineheight=.8, face="bold"))
```



As result I see that the different lexicons for calculating sentiment give results that are different in an absolute sense but have similar relative trajectories through the novel. I am recognizing similar dips and peaks in sentiment at about the same places in the novel, but the absolute values are significantly different.

The AFINN lexicon gives the largest absolute values, with high positive values. The lexicon from Bing et al. has lower absolute values and seems to label larger blocks of contiguous positive or negative text. The NRC results are shifted higher relative to the other two, labeling the text more positively, but detects similar relative changes in the text. The NRC results are shifted higher relative to the other two, labeling the text more positively, but detects similar relative changes in the text. I find similar differences between the methods when looking at other novels. The NRC sentiment is high, the AFINN sentiment has more variance, the Bing et al. sentiment appears to find longer stretches of similar text, but all three agree roughly on the overall trends in the sentiment through a narrative arc.

I want to look briefly at how many positive and negative words are in these lexicons.

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

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

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

```
## # A tibble: 2 × 2
##   sentiment     n
##       <chr> <int>
## 1  negative  4782
## 2  positive  2006
```

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. This will contribute to the effect I see in the plot above, as will any systematic difference in word matches, e.g. if the negative words in the NRC lexicon do not match the words that Jane Austen uses very well.

So I noticed differences in absolute sentiment from lexicon to lexicon. I should keep this in mind when I am choosing a sentiment lexicon for analysis.

02_04 Most common positive and negative Words

One advantage of having the data frame with both sentiment and word is that I can analyze word counts that contribute to each sentiment.

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

```
## Joining, by = "word"
```

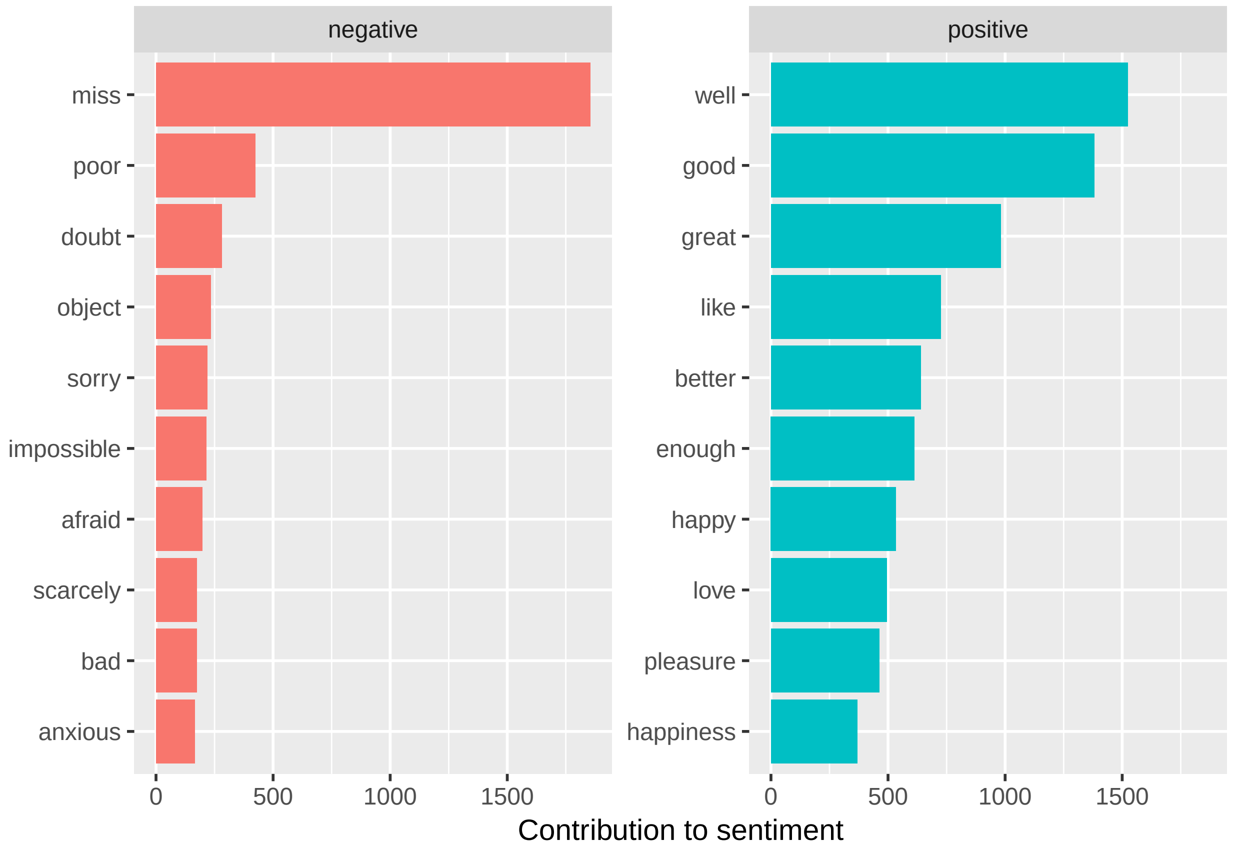
```
bing_word_counts
```

```
## # A tibble: 2,585 × 3
##        word sentiment     n
##       <chr>     <chr> <int>
## 1      miss  negative  1855
## 2      well  positive  1523
## 3      good  positive  1380
## 4     great  positive   981
## 5      like  positive   725
## 6    better  positive   639
## 7    enough  positive   613
## 8     happy  positive   534
## 9      love  positive   495
## 10 pleasure  positive   462
## # ... with 2,575 more rows
```

Now, this can be shown visually.

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

```
## Selecting by n
```



An anomaly in the sentiment analysis is, that the word "miss" is coded as negative but it is used as a title for young, unmarried women in Jane Austen's works. If it were appropriate for my purposes, I could easily add "miss" to a custom stop-words list using.

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

custom_stop_words
```

```
## # A tibble: 1,150 × 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
```

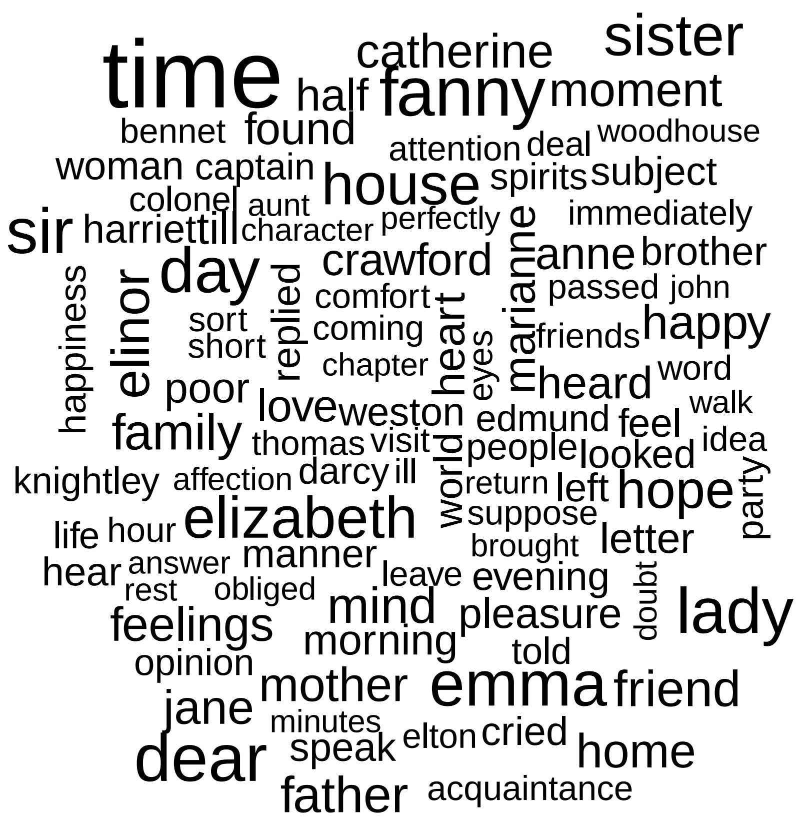
02_05 Wordclouds

I want to look at the most common words in Jane Austen's works as a whole again, but this time as a wordcloud.

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

```
## Joining, by = "word"
```

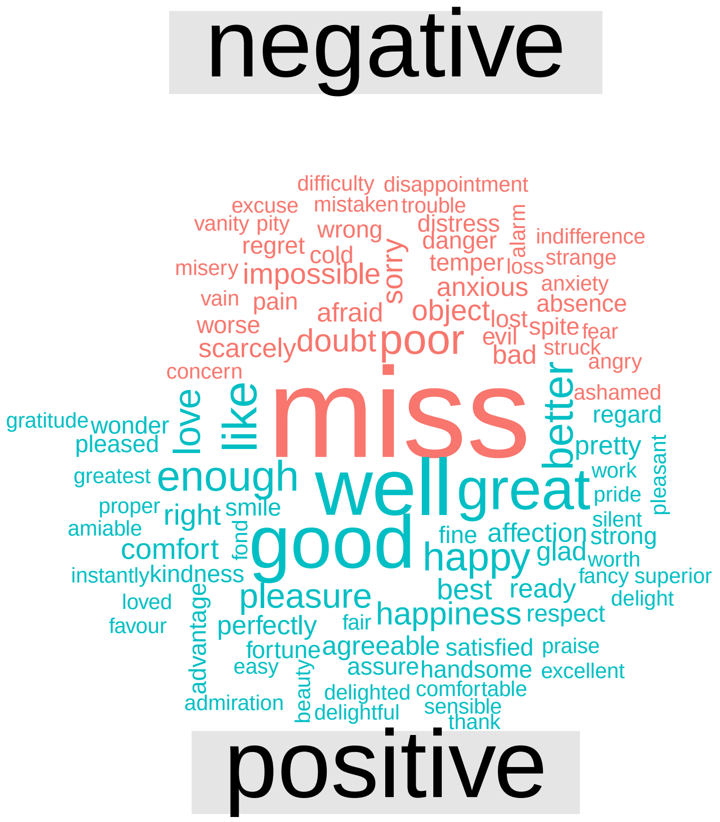
```
## Warning in wordcloud(word, n, max.words = 100): miss could not be fit on
## page. It will not be plotted.
```



Let's do the sentiment analysis to tag positive and negative words using an inner join, then find the most common positive and negative words.

```
tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("#F8766D", "#00BFC4"),
                   max.words = 100)
```

```
## Joining, by = "word"
```



So I see the most important positive and negative words, but the sizes of the words are not comparable across sentiments.

02_06 Looking at Units beyond just Words

Lots of useful work can be done by tokenizing at the word level, but sometimes it is useful or necessary to look at different units of text. For example, some sentiment analysis algorithms look beyond only unigrams (i.e. single words) to try to understand the sentiment of a sentence as a whole. These algorithms try to understand that

I am not having a good day.

is a sad sentence, not a happy one, because of negation. R packages included coreNLP (T. Arnold and Tilton 2016), cleanNLP (T. B. Arnold 2016), and sentimentr (Rinker 2017) are examples of such sentiment analysis algorithms. For these, I may want to tokenize text into sentences, and it makes sense to use a new name for the output column in such a case.

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

```r
PandP_sentences$sentence[2]
```

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

The sentence tokenizing does seem to have a bit of trouble with UTF-8 encoded text, especially with sections of dialogue; it does much better with punctuation in ASCII.

I split into tokens using a regex pattern. I can use this, for example, to split the text of Jane Austen's novels into a data frame by chapter.

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

At the beginning of this project, I used a similar regex to find where all the chapters were in Austen's novels for a tidy data frame organized by one-word-per-row. I can use tidy text analysis to ask questions such as what are the most negative chapters in each of Jane Austen's novels?

First, I get the list of negative words from the Bing lexicon. Second, I make a data frame of how many words are in each chapter so I can normalize for the length of chapters. Last, I find the number of negative words in each chapter and divide by the total words in each chapter. So I can analyze for each book, which chapter has the highest proportion of negative words.

```
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()
```

```
## Joining, by = "word"
```

```
## Selecting by ratio
```

```
## # A tibble: 6 × 5
##                  book chapter negativewords words      ratio
##                <fctr>   <int>         <int> <int>      <dbl>
## 1 Sense & Sensibility      43           161  3405 0.04728341
## 2   Pride & Prejudice      34           111  2104 0.05275665
## 3      Mansfield Park      46           173  3685 0.04694708
## 4               Emma      15            151  3340 0.04520958
## 5    Northanger Abbey      21            149  2982 0.04996647
## 6          Persuasion       4             62  1807 0.03431101
```

These are the chapters with the most sad words in each book, normalized for number of words in the chapter. When I will have a short review, what is happening in these chapters. So for example, in chapter 43 of "Sense and Sensibility" Marianne is seriously ill, near death, or chapter 46 of "Mansfield Park" is almost the end, when everyone learns of Henry's scandalous adultery.

Summary

Sentiment analysis provides a way to understand the attitudes and opinions expressed in texts. In this project, I explored how to approach sentiment analysis using tidy data principles. When text data is in a tidy data structure, sentiment analysis can be implemented as an inner join. I use sentiment analysis to understand how a narrative arc changes throughout its course or what words with emotional and opinion content are important for a particular text.