Introduction to tidytext

Data Wrangling and Husbandry

2/25/2019

Textual Analysis

There are many analyses of text that one might want to do, including

- Frequencies of words
- Sentiment of words
- Discovery of topics

Until recently, the main formats have been

- Raw strings
- Corpus: raw string content with additional metadata and details
- Document-term matrix

Julia Silge and David Robinson have introduced the `tidytext` package and new book which offers the tidytext format. That format is simply a table with one-term-per-row.

(Installing the `tidytext` package may take a while if you install from source, since it requires installing many additional packages.)

This is not a replacement for full-fledged natural language processing, but is an easy entry point for many analyses.

An extended example (from Tidy Text Mining with R)

We will make a frequency table of words in Pride and Prejudice and follow that with sentiment analysis of all of Jane Austen's novels.

```
library(janeaustenr)
prideprejudice[1:11]
```

```
##  [1] "PRIDE AND PREJUDICE"
##  [2] ""
##  [3] "By Jane Austen"
##  [4] ""
##  [5] ""
##  [6] ""
##  [7] "Chapter 1"
##  [8] ""
##  [9] ""
## [10] "It is a truth universally acknowledged, tha
## [11] "of a good fortune, must be in want of a wif
```

7/31

```
library(stringr)
prideprejudice.tbl <- tibble(text = prideprejudice)
mutate(linenumber = row_number(),
         chapter = cumsum(str_detect(text, "(?i)^chap
prideprejudice.tbl[10:15,]
```

```
## # A tibble: 6 x 3
##   text
##   <chr>
## 1 It is a truth universally acknowledged, that a
## 2 of a good fortune, must be in want of a wife.
## 3 ""
## 4 However little known the feelings or views of su
## 5 first entering a neighbourhood, this truth is so
## 6 of the surrounding families, that he is conside
```

`unnest_tokens()`

We can move from strings to words with the `unnest_tokens()` function (which by default changes everything to lower case).

```
library(tidytext)
tidy_prideprejudice <- prideprejudice.tbl %>%
  unnest_tokens(word, text)

tidy_prideprejudice[9:14,]
```

```
## # A tibble: 6 x 3
##   linenumber chapter word
##        <int>   <int> <chr>
## 1         10       1 it
## 2         10       1 is
## 3         10       1 a
## 4         10       1 truth
## 5         10       1 universally
## 6         10       1 acknowledged
```

Now that the data is tidy, we can use regular tools from the tidyverse. For example, what are the most common words?

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

```
## # A tibble: 6,538 x 2
##    word      n
##    <chr> <int>
##  1 the    4331
##  2 to     4162
##  3 of     3610
##  4 and    3585
##  5 her    2203
##  6 i      2065
##  7 a      1954
##  8 in     1880
##  9 was    1843
## 10 she    1695
## # … with 6,528 more rows
```

Hmm, that's not quite what we want. We can eliminate what are known as *stop words*.

```
data(stop_words)
sample_n(stop_words, 10)
```

```
## # A tibble: 10 x 2
##    word       lexicon
##    <chr>      <chr>
##  1 needs      SMART
##  2 ever       SMART
##  3 seems      SMART
##  4 ourselves  snowball
##  5 can        onix
##  6 serious    SMART
##  7 regardless SMART
##  8 further    SMART
##  9 immediate  SMART
## 10 exactly    SMART
```

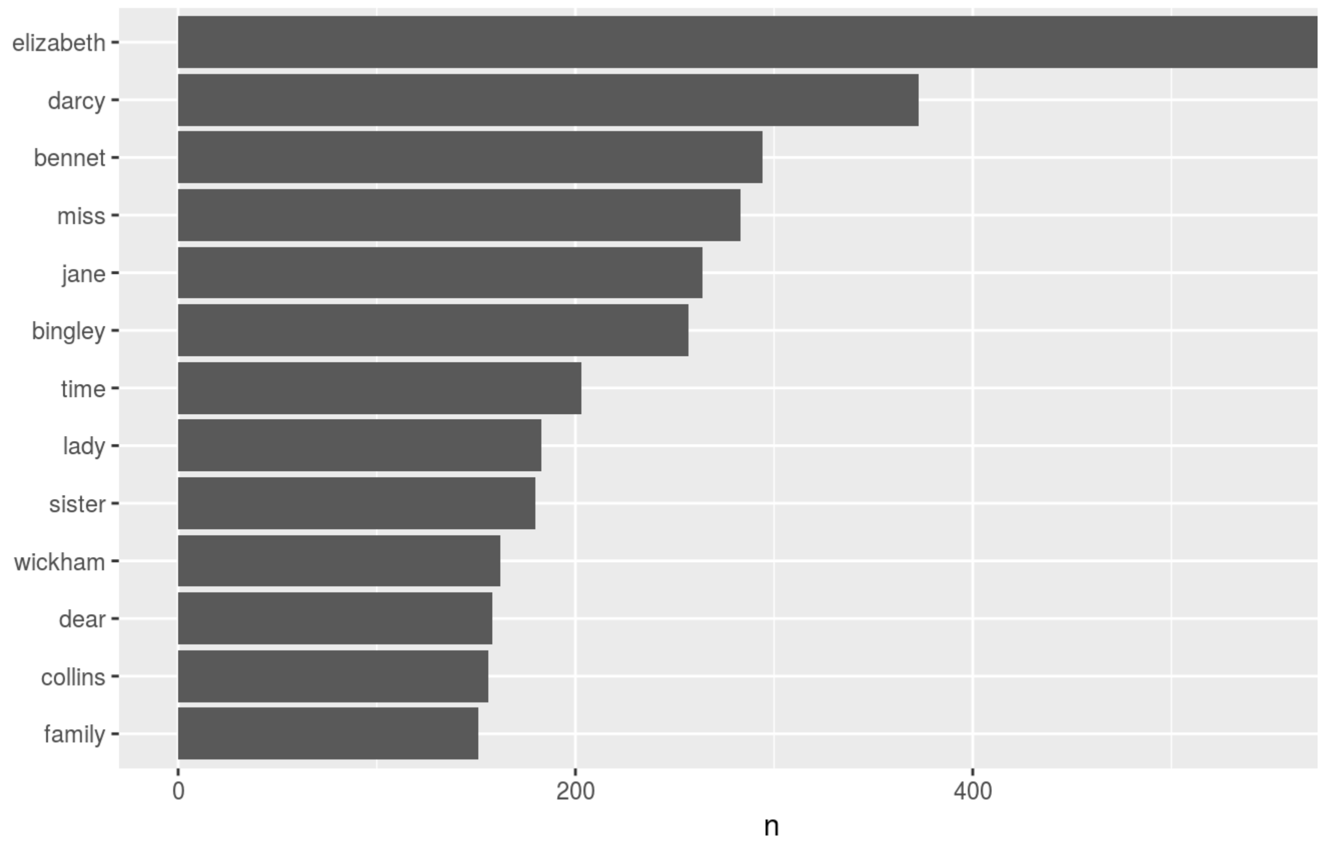
```
tidy_prideprejudice %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE)
```

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

```
## # A tibble: 6,009 x 2
##    word          n
##    <chr>     <int>
##  1 elizabeth   597
##  2 darcy       373
##  3 bennet      294
##  4 miss        283
##  5 jane        264
##  6 bingley     257
##  7 time        203
##  8 lady        183
##  9 sister      180
## 10 wickham     162
## # … with 5,999 more rows
```

```
library(ggplot2)

tidy_prideprejudice %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  filter(n > 150) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_bar(stat = "identity") +
  xlab(NULL) + coord_flip()
```


elizabeth
darcy
bennet
miss
jane
bingley
time
lady
sister
wickham
dear
collins
family
0
200
400
n

N-grams

What about pairs or triples of words? These are known as “n-grams”, and can be obtained by an optional argument to `unnest_tokens()`.

```
tidy_prideprejudice_bigram <- prideprejudice.tbl %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 
tidy_prideprejudice_bigram
```

```
## # A tibble: 114,045 x 3
##    linenumber chapter bigram
##         <int>   <int> <chr>
##  1          1       0 pride and
##  2          1       0 and prejudice
##  3          2       0 <NA>
##  4          3       0 by jane
##  5          3       0 jane austen
##  6          4       0 <NA>
##  7          5       0 <NA>
##  8          6       0 <NA>
##  9          7       1 chapter 1
## 10          8       1 <NA>
## # … with 114,035 more rows
```

We can’t just drop stop words, since that will mess up the pairs.

Instead, we can split out the words in a pair and delete the pairs with a stop word in them

```
prideprejudice_bigrams_separated <- tidy_prideprejudi
  separate(bigram, c("word1", "word2"), sep = " ")

prideprejudice_bigrams_filtered <- prideprejudice_big
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)
```

```
prideprejudice_bigrams_filtered
```

```
## # A tibble: 8,835 x 4
##    linenumber chapter word1       word2
##         <int>   <int> <chr>       <chr>
##  1          2       0 <NA>        <NA>
##  2          3       0 jane        austen
##  3          4       0 <NA>        <NA>
##  4          5       0 <NA>        <NA>
##  5          6       0 <NA>        <NA>
##  6          7       1 chapter     1
##  7          8       1 <NA>        <NA>
##  8          9       1 <NA>        <NA>
##  9         10       1 truth       universally
## 10         10       1 universally acknowledged
## # … with 8,825 more rows
```

```
prideprejudice_bigrams_filtered %>%
  count(word1, word2, sort = TRUE)
```

```
## # A tibble: 5,109 x 3
##    word1   word2         n
##    <chr>   <chr>     <int>
##  1 <NA>    <NA>       2556
##  2 lady    catherine    87
##  3 miss    bingley      67
##  4 miss    bennet       52
##  5 sir     william      35
##  6 de      bourgh       32
##  7 miss    darcy        32
##  8 cried   elizabeth    24
##  9 colonel forster      23
## 10 miss    lucas        23
## # … with 5,099 more rows
```

Sentiment Analysis

Sentiment analysis is an attempt to extract the sentiment, or feeling or attitude, from text. The most popular approach is just to match individual words against a lexicon. For later, note that `get_sentiments()` extracts a particular lexicon.

```
sentiments
```

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

(All code from Text Mining With R)

```
tidy_books <- austen_books() %>%
  group_by(book) %>%
  mutate(linenumber = row_number(),
         chapter = cumsum(str_detect(text, regex("^cl
                                                  ign
  ungroup()  %>%
  unnest_tokens(word, text)
```

It helps to know your subject

```
sentiments %>% filter(word == "miss")
```

```
## # A tibble: 3 x 4
##   word  sentiment lexicon  score
##   <chr> <chr>     <chr>    <int>
## 1 miss  negative  bing        NA
## 2 miss  <NA>      AFINN       -2
## 3 miss  negative  loughran    NA
```

We could carefully drop “miss” when it’s followed by a character name, but we’ll take the sloppier approach and just filter it out.

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

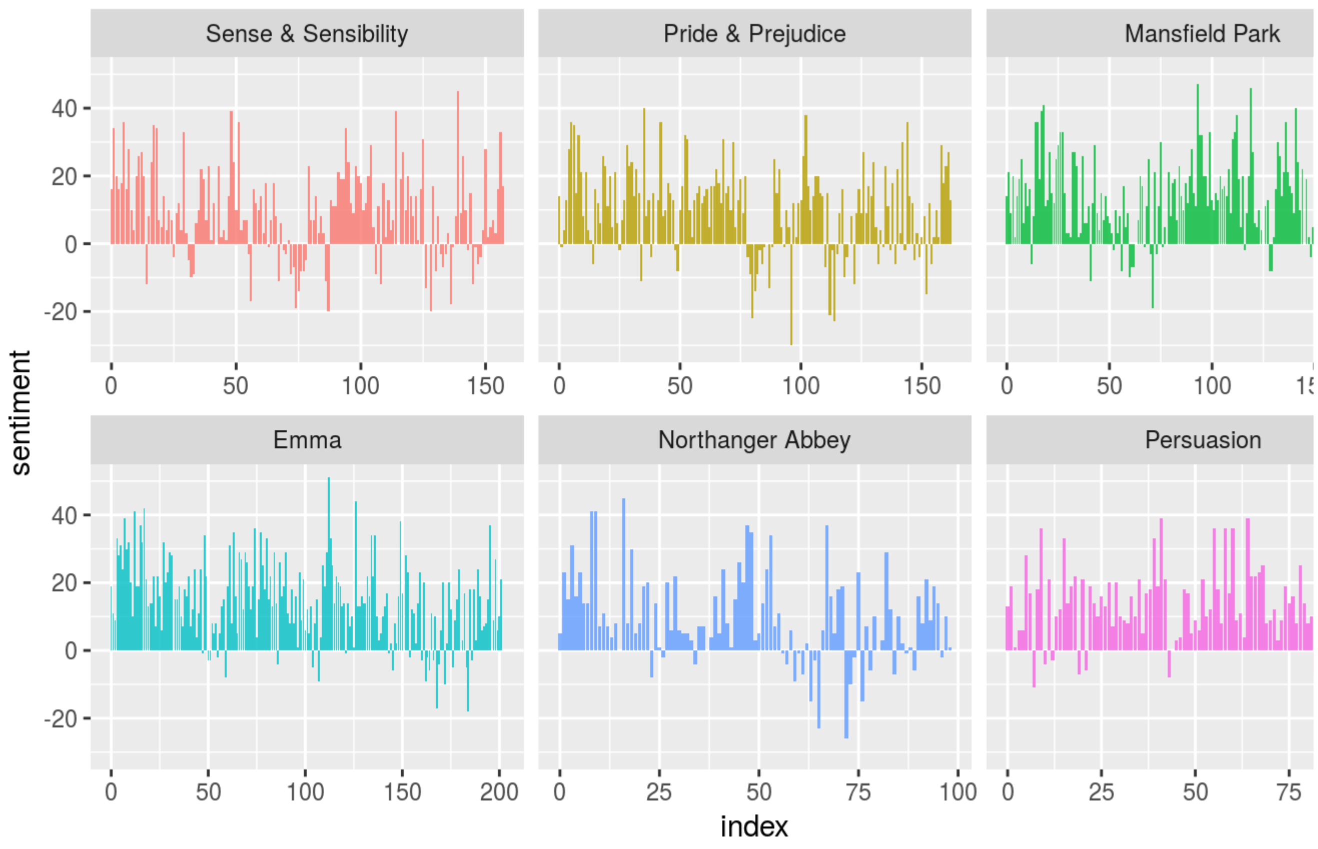
```
## Joining, by = "word"
```

```
janeaustensentiment
```

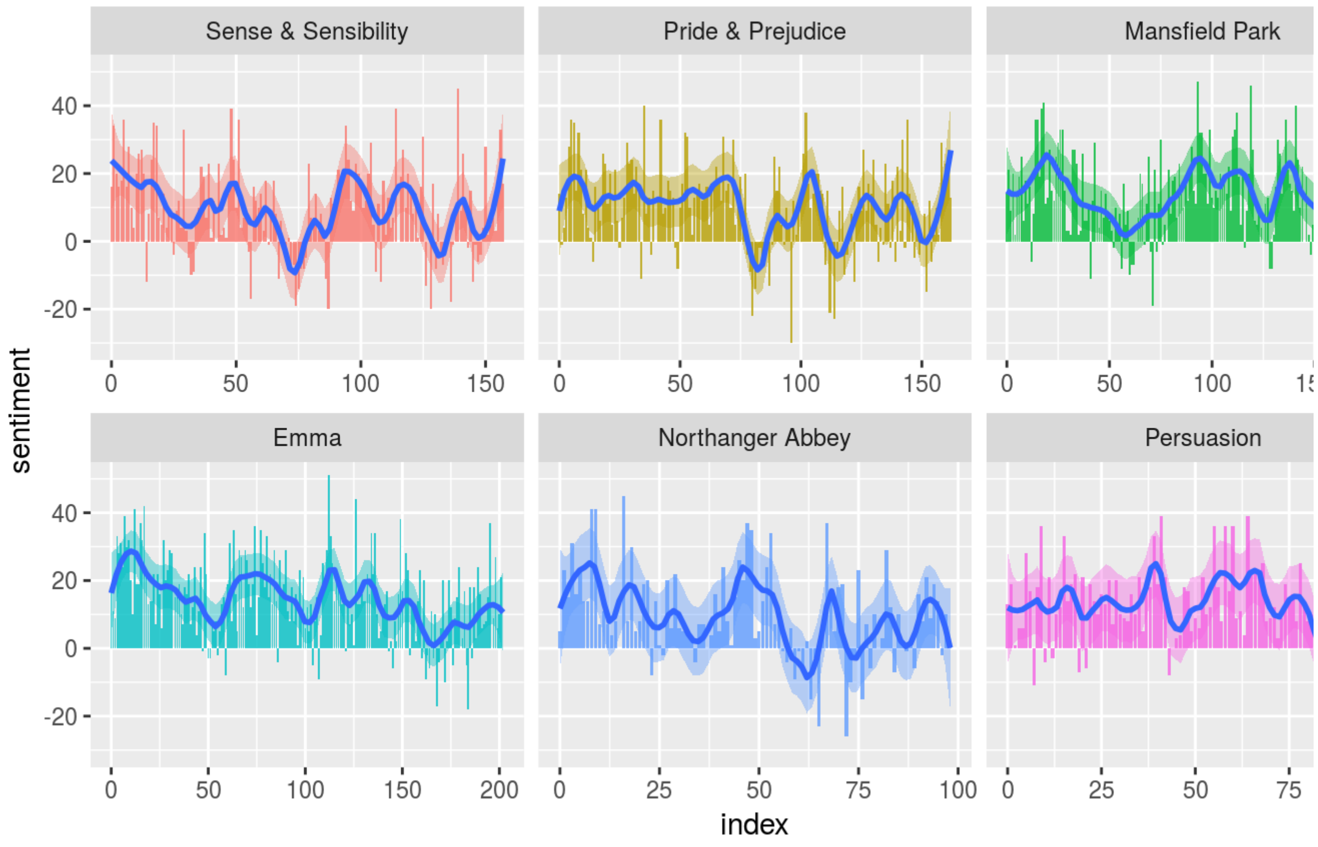
```
## # A tibble: 920 x 5
##    book                index negative positive se
##    <fct>               <dbl>    <dbl>    <dbl>
##  1 Sense & Sensibility     0       16       32
##  2 Sense & Sensibility     1       19       53
##  3 Sense & Sensibility     2       11       31
##  4 Sense & Sensibility     3       15       31
##  5 Sense & Sensibility     4       16       34
##  6 Sense & Sensibility     5       15       51
##  7 Sense & Sensibility     6       24       40
##  8 Sense & Sensibility     7       23       51
##  9 Sense & Sensibility     8       30       40
## 10 Sense & Sensibility     9       15       19
## # … with 910 more rows
```

```
ggplot(janeaustensentiment, aes(index, sentiment, fi
  geom_bar(alpha = 0.8, stat = "identity", show.legen
  facet_wrap(~book, ncol = 3, scales = "free_x")
```

```
ggplot(janeaustensentiment, aes(index, sentiment, fi
  geom_bar(alpha = 0.8, stat = "identity") +
  facet_wrap(~book, ncol = 3, scales = "free_x") +
  theme(legend.position = "none")
```



```
ggplot(janeaustensentiment, aes(index, sentiment, fi
  geom_bar(alpha = 0.8, stat = "identity", show.lege
  facet_wrap(~book, ncol = 3, scales = "free_x") +
  theme(legend.position = "none") +
  geom_smooth(span = .15)
```



In class exercises

1. Use the `gutenbergr` package to download the texts of Wuthering Heights and Jane Eyre.

```
library(gutenbergr)
bronte_books <- gutenberg_download(gutenberg_id = c(
    meta_fields = "title")
```

1. Tidy the downloaded data so that each row is one word.
2. Find the most popular non-stop-words in Jane Eyre
3. Plot the sentiment for the two books (just re-use the code above)