Handout 10: Tidytext

Taylor Arnold

This handout describes how to start extracting words from raw text using the package **tidytext**. To load the package call

```
library(tidytext)
```

I will also make use of the package `janeaustenr` which contains the complete text of every Jane Austen novel as a useful source of data for this handout:

```
library(janeaustenr)
```

You do not need to install or load the package unless you want to replicate the code here. I will save the data to a new data frame named `ja`, removing empty lines:

```
ja <- janeaustenr::austen_books()
ja <- mutate(ja, book = as.character(book))
ja
```

```
## # A tibble: 73,422 × 2
##                      text                book
##                     <chr>               <chr>
## 1  SENSE AND SENSIBILITY Sense & Sensibility
## 2                        Sense & Sensibility
## 3         by Jane Austen Sense & Sensibility
## 4                        Sense & Sensibility
## 5                 (1811) Sense & Sensibility
## 6                        Sense & Sensibility
## 7                        Sense & Sensibility
## 8                        Sense & Sensibility
## 9                        Sense & Sensibility
## 10             CHAPTER 1 Sense & Sensibility
## # ... with 73,412 more rows
```

```
ja <- filter(ja, text != "")
```

Tokens

Tokenization is the process of splitting text into individual words. To do this in the **tidytext** package, we use the function `unnest_tokens`. It takes a dataset, the name of the column that will hold the words, and the name of the variable that currently holds the raw text. To use it on the dataset `mansfieldpark`, for example, we can do this:

```
ja_word <- unnest_tokens(ja, word, text)
ja_word
```

```
## # A tibble: 725,054 × 2
##                   book        word
##                  <chr>       <chr>
## 1  Sense & Sensibility       sense
## 2  Sense & Sensibility         and
## 3  Sense & Sensibility sensibility
## 4  Sense & Sensibility          by
## 5  Sense & Sensibility        jane
## 6  Sense & Sensibility      austen
## 7  Sense & Sensibility        1811
## 8  Sense & Sensibility     chapter
## 9  Sense & Sensibility           1
## 10 Sense & Sensibility         the
## # ... with 725,044 more rows
```

The output is a new data frame with exactly one word per line. The first column keeps track of which text the word is a part of.

What might be want to do with this dataset? As a start, perhaps would could figure out what the primary themes are based on the words that are used. The **dplyr** verb `count`, along with arrange is a useful place to start:

```
count(ja_word, book, word, sort = TRUE)
```

```
## Source: local data frame [40,379 x 3]
## Groups: book [6]
##
##                 book  word     n
##                <chr> <chr> <int>
## 1     Mansfield Park   the  6206
## 2     Mansfield Park    to  5475
## 3     Mansfield Park   and  5438
## 4               Emma    to  5239
## 5               Emma   the  5201
## 6               Emma   and  4896
## 7     Mansfield Park    of  4778
## 8  Pride & Prejudice   the  4331
## 9               Emma    of  4291
## 10 Pride & Prejudice    to  4162
## # ... with 40,369 more rows
```

Unfortunately, the top words are all very common functional words that don't tell us much about what is actually going on. What to do

now? Lucky for us, **tidytext** contains a list of common words such as these called `stop_words`.

```
stop_words
```

```
## # A tibble: 1,149 × 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
```

We can remove these then with a call to the function `anti_join`:

```
ja_word <- anti_join(ja_word, stop_words)
count(ja_word, book, word, sort = TRUE)
```

```
## Source: local data frame [37,224 x 3]
## Groups: book [6]
##
##                   book      word     n
##                  <chr>     <chr> <int>
## 1       Mansfield Park     fanny   816
## 2                 Emma      emma   786
## 3  Sense & Sensibility    elinor   623
## 4                 Emma      miss   599
## 5    Pride & Prejudice elizabeth   597
## 6       Mansfield Park  crawford   493
## 7  Sense & Sensibility  marianne   492
## 8           Persuasion      anne   447
## 9       Mansfield Park      miss   432
## 10    Northanger Abbey catherine   428
## # ... with 37,214 more rows
```

Ah okay, at least we are actually getting somewhere now. All of these are at least characters in the various texts! We need some more work for filtering out the words that are of interest to us.

Part of speech

The **tidytext** package also contains a dataset called `parts_of_speech`:

```
parts_of_speech
```

```
## # A tibble: 208,259 × 2
##       word       pos
##      <chr>     <chr>
## 1      3-d Adjective
## 2      3-d      Noun
## 3      4-f      Noun
## 4   4-h'er      Noun
## 5      4-h Adjective
## 6       a' Adjective
## 7      a-1      Noun
## 8   a-axis      Noun
## 9   a-bomb      Noun
## 10 a-frame      Noun
## # ... with 208,249 more rows
```

Using this we can do a very basic form of part of speech tagging by left joining the `ja_word` data with this. In fact, we will actually use an inner join here to ignore words that have no match:

```
ja_pos <- inner_join(ja_word, parts_of_speech)
```

Notice that here there is a copy of words that occur with multiple parts of speech. Now, what happens if we find the top words of specific parts of speech:

```
count(filter(ja_pos, pos == "Adjective"),
      book, word, sort = TRUE)
```

```
## Source: local data frame [7,493 x 3]
## Groups: book [6]
##
##                 book  word     n
##                <chr> <chr> <int>
## 1               Emma  dear   241
## 2               Emma frank   200
## 3     Mansfield Park  dear   161
## 4  Pride & Prejudice  dear   158
## 5     Mansfield Park  home   144
## 6               Emma  poor   136
## 7     Mansfield Park  half   136
## 8               Emma  home   130
```

```
## 9               Emma happy   125
## 10              Emma  half   118
## # ... with 7,483 more rows
```

```
count(filter(ja_pos, pos == "Noun"),
      book, word, sort = TRUE)
```

```
## Source: local data frame [15,496 x 3]
## Groups: book [6]
##
##                   book      word     n
##                  <chr>     <chr> <int>
## 1       Mansfield Park     fanny   816
## 2                 Emma      emma   786
## 3  Sense & Sensibility    elinor   623
## 4                 Emma      miss   599
## 5    Pride & Prejudice elizabeth   597
## 6       Mansfield Park  crawford   493
## 7  Sense & Sensibility  marianne   492
## 8           Persuasion      anne   447
## 9       Mansfield Park      miss   432
## 10    Northanger Abbey catherine   428
## # ... with 15,486 more rows
```

```
count(filter(ja_pos, str_detect(pos, "Verb")),
      book, word, sort = TRUE)
```

```
## Source: local data frame [10,931 x 3]
## Groups: book [6]
##
##                   book  word     n
##                  <chr> <chr> <int>
## 1                 Emma  miss  1198
## 2       Mansfield Park  miss   864
## 3    Pride & Prejudice  miss   566
## 4  Sense & Sensibility  miss   420
## 5     Northanger Abbey  miss   412
## 6                 Emma frank   400
## 7       Mansfield Park  feel   384
## 8       Mansfield Park house   356
## 9                 Emma  body   328
## 10 Sense & Sensibility house   322
## # ... with 10,921 more rows
```

Still somewhat rough, primarily because the part of speech names do not distinguish proper and non-proper nouns.

Sentiment

Along the same idea, the **tidytext** dataset also contains a set of sentiment scores for various words. There are three sets contained in the same data `sentiments`. The first is the NRC Emotion Lexicon:

```
filter(sentiments, lexicon == "nrc")
```

```
## # A tibble: 13,901 × 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 13,891 more rows
```

The sentiment lexicon from Bing Liu

```
filter(sentiments, lexicon == "bing")
```

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

And the lexicon of Finn Arup Nielsen:

```
filter(sentiments, lexicon == "AFINN")
```

```
## # A tibble: 2,476 × 4
```

```
##          word sentiment lexicon score
##         <chr>     <chr>   <chr> <int>
## 1     abandon      <NA>   AFINN    -2
## 2   abandoned      <NA>   AFINN    -2
## 3    abandons      <NA>   AFINN    -2
## 4    abducted      <NA>   AFINN    -2
## 5   abduction      <NA>   AFINN    -2
## 6  abductions      <NA>   AFINN    -2
## 7       abhor      <NA>   AFINN    -3
## 8    abhorred      <NA>   AFINN    -3
## 9   abhorrent      <NA>   AFINN    -3
## 10     abhors      <NA>   AFINN    -3
## # ... with 2,466 more rows
```

We can merge this into the novels as well to try to see the different moods of the various books:

```
bing <- filter(sentiments, lexicon == "bing")
ja_bing <- inner_join(ja_word, bing)
count(ja_bing, book, sentiment)
```

```
## Source: local data frame [12 x 3]
## Groups: book [?]
##
##                   book sentiment     n
##                  <chr>     <chr> <int>
## 1                 Emma  negative  4738
## 2                 Emma  positive  5150
## 3       Mansfield Park  negative  4784
## 4       Mansfield Park  positive  4886
## 5     Northanger Abbey  negative  2485
## 6     Northanger Abbey  positive  2441
## 7           Persuasion  negative  2178
## 8           Persuasion  positive  2511
## 9    Pride & Prejudice  negative  3611
## 10   Pride & Prejudice  positive  3910
## 11 Sense & Sensibility  negative  3625
## 12 Sense & Sensibility  positive  3852
```

So *Northanger Abbey* is the most positive novel and *Persuasion* is the most negative.