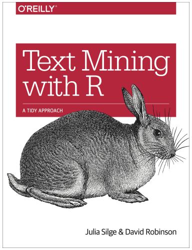
Introduction to Text Mining with R

RLadies Philadelphia

Today's "bookclub"

- Chapters 1 & 2 of *"Text Mining with R: A Tidy Approach"* by Julia Silge and David Robinson `https://www.tidytextmining.com/`
- Content overview
- Work together in small groups



The tidy text format

- Some keywords
  - **token** = meaningful unit of text
  - **tokenisation** = splitting text into tokens
  - **n-gram** = adjacent sequence of n items from sample of text
    - unigram, bigram, trigrams...
  - **regex** = “regular expression” = sequence of characters defining a search pattern
- What is “the tidy text format”?
  - one-token-per-row
  - dplyr, ggplot2

A minimal example

```r
text <- c("Because I could not stop for Death -",
          "He kindly stopped for me -",
          "The Carriage held but just Ourselves -",
          "and Immortality")
text_df <- data_frame(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
```

Tokenisation

- unnest_tokens() from tidytext to tokenize
  - words (default)
  - characters
  - n-grams
  - sentences
  - lines
  - paragraphs
  - regex pattern separation

```r
head(text_df %>%
  unnest_tokens(word, text))
```

```
## # A tibble: 6 x 2
##    line word
##   <int> <chr>
## 1     1 because
## 2     1 i
## 3     1 could
## 4     1 not
## 5     1 stop
## 6     1 for
```

Processing austen_books()

- austen_books() from janeaustenr
- mutate to add
  - line numbers for each row
  - chapter (w/ cumulative sum of regex string finds)

```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[1,]
```

```
## # A tibble: 1 x 4
##   text                  book                linenumber chapter
##   <chr>                 <fct>                    <int>   <int>
## 1 SENSE AND SENSIBILITY Sense & Sensibility          1       0
```

Processing austen_books()

- tokenise words

```
tidy_books <- original_books %>%
  unnest_tokens(word, text)
head(tidy_books)
```

```
## # A tibble: 6 x 4
##   book                linenumber chapter word
##   <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
```

stop_words

- stop_words from tidytext
- purpose?
- remove stop_words with an antijoin

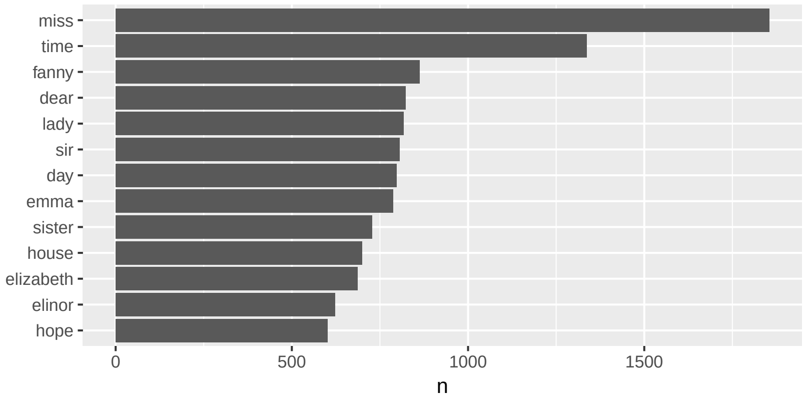
```
tidy_books <- tidy_books %>%
  anti_join(stop_words)
```

The most frequent words

```r
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
```

The most frequent words



Project Gutenberg

- 1971
- `https://www.gutenberg.org/`
- 56,000 ebooks
- public domain / expired copyright

gutenbergr

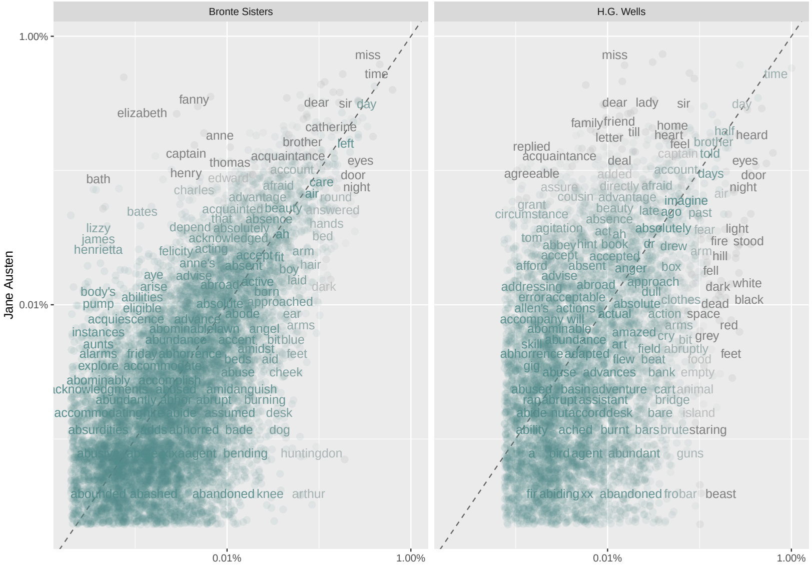
- gutenbergr package
- https://ropensci.org/tutorials/gutenbergr_tutorial/
- Search gutenberg_metadata

```
head(gutenberg_metadata %>%
  filter(author == "Wells, H. G. (Herbert George)"))
head(gutenberg_works(author == "Wells, H. G. (Herbert George)"))
```

- Download by id

```
hgwells <- gutenberg_download(c(35, 36, 5230, 159))
bronte <- gutenberg_download(c(1260, 768, 969, 9182, 767))
```

Word frequencies for Jane Austen, the Brontë sisters, and H.G. Wells



Sentiment analysis

- aka "opinion mining"
- computationally identifying & categorizing sentiment in text
- Some keywords
    - **lexicon** = inventory of words
    - **sentiment** = emotional content

The sentiments dataset (tidytext)

- contains several sentiment lexicons

```
unique(sentiments$lexicon)
```

```
## [1] "nrc"      "bing"     "AFINN"    "loughran"
```

- Limitations of this approach?
    - appropriateness
    - unigrams
        - *"It was not good"*

The lexicons

- **AFINN** (Finn Årup Nielsen)
    - -5 to 5
    - manually labelled

```r
head(get_sentiments("afinn"))
```

```
## # A tibble: 6 x 2
##   word       score
##   <chr>      <int>
## 1 abandon       -2
## 2 abandoned     -2
## 3 abandons      -2
## 4 abducted      -2
## 5 abduction     -2
## 6 abductions    -2
```

The lexicons

- **bing** (Bing Liu et al)
    - positive/negative

```r
head(get_sentiments("bing"))
```

```
## # A tibble: 6 x 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
```

The lexicons

- **nrc** (Saif Mohammad and Peter Turney)
    - positive/negative
    - anger, fear, anticipation, trust, surprise, sadness, joy, disgust
    - crowdsourced manual annotations

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

```
## # A tibble: 6 x 2
##   word      sentiment
##   <chr>     <chr>
## 1 abacus    trust
## 2 abandon   fear
## 3 abandon   negative
## 4 abandon   sadness
## 5 abandoned anger
## 6 abandoned fear
```

Most frequent words by sentiment

- Most frequent “joy” words in Emma

```
nrc_joy <- get_sentiments("nrc") %>%
  filter(sentiment == "joy")
tidy_books %>%
  filter(book == "Emma") %>%
  inner_join(nrc_joy) %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 298 x 2
##    word          n
##    <chr>     <int>
##  1 friend      166
##  2 hope        143
##  3 happy       125
##  4 love        117
##  5 deal         92
##  6 found        92
##  7 happiness    76
##  8 pretty       68
##  9 true         66
## 10 comfort      65
## # ... with 288 more rows
```

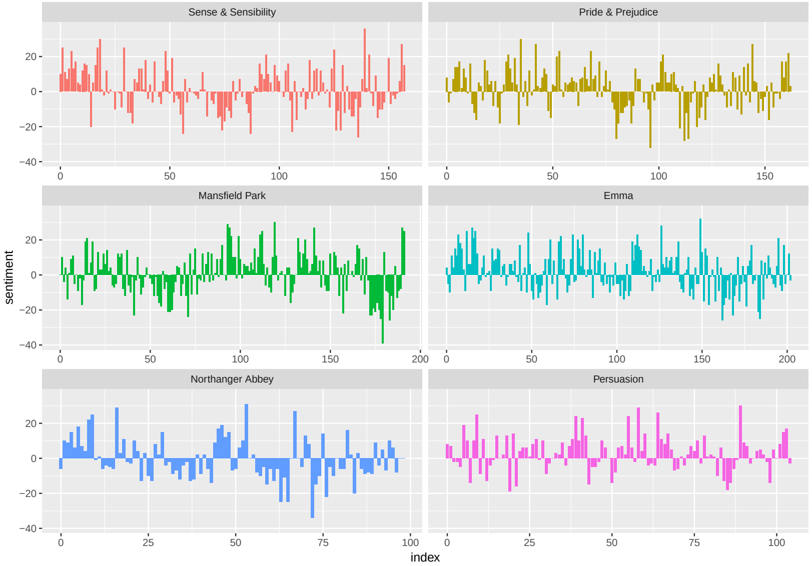
Sentiment change across text

- 80 line chunks

```
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)
jane_austen_sentiment
```

```
## # A tibble: 920 x 5
##    book                index negative positive sentiment
##    <fct>               <dbl>    <dbl>    <dbl>     <dbl>
##  1 Sense & Sensibility    0.      16.      26.       10.
##  2 Sense & Sensibility    1.      19.      44.       25.
##  3 Sense & Sensibility    2.      12.      23.       11.
##  4 Sense & Sensibility    3.      15.      22.        7.
##  5 Sense & Sensibility    4.      16.      29.       13.
##  6 Sense & Sensibility    5.      16.      39.       23.
##  7 Sense & Sensibility    6.      24.      37.       13.
##  8 Sense & Sensibility    7.      22.      39.       17.
##  9 Sense & Sensibility    8.      30.      35.        5.
## 10 Sense & Sensibility    9.      14.      18.        4.
## # ... with 910 more rows
```

Sentiment change across Austen novels



Most frequent positive versus negative words

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

head(bing_word_counts)
```

```
## # A tibble: 6 x 3
##   word      sentiment     n
##   <chr>     <chr>     <int>
## 1 miss      negative   1855
## 2 happy     positive    534
## 3 love      positive    495
## 4 pleasure  positive    462
## 5 poor      negative    424
## 6 happiness positive    369
```

Custom stop words

- "miss" is currently misanalysed...

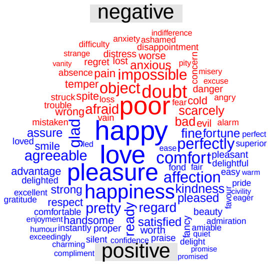
```
custom_stop_words <- bind_rows(data_frame(word = c("miss"),
                                          lexicon = c("custom")),
                               stop_words)

head(custom_stop_words)
```

```
## # A tibble: 6 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
```

Pipe data into a wordcloud plot...

- wordcloud package

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



In small groups

- Work through Section 1.5 (Chapter 1), especially the code chunk below:

```
frequency <- bind_rows(mutate(tidy_bronte, author = "Bronte 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, `Bronte Sisters`:`H.G. Wells`)
```

- Pick a novel from the Gutenberg collection, and discover. . .
  - the most frequent words
  - the most frequent “trust” words using the nrc lexicon
  - how the sentiment changes across the novel using the bing lexicon
- If you have time, create a data visualisation for one of these text mining exercises

References

- Silge, Julia and David Robinson (2017), *Text Mining with R: A Tidy Approach*, O'Reilly / `https://www.tidytextmining.com`
  - and references therein: `https://www.tidytextmining.com/references.html`

Dracula: the most frequent words

```r
gutenberg_works(title == "Dracula")$gutenberg_id
```

```
## [1] 345
```

```r
dracula <- gutenberg_download(345)
dracula_words <- dracula %>%
  mutate(linenumber = row_number(),
         chapter = cumsum(str_detect(text,
                          regex("^chapter [\\divxlc]",
                                ignore_case = TRUE)))  %>%
  unnest_tokens(word, text)
head(dracula_words %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE))
```

```
## # A tibble: 6 x 2
##   word        n
##   <chr>   <int>
## 1 time      390
## 2 van       323
## 3 night     310
## 4 helsing   301
## 5 dear      224
## 6 lucy      223
```

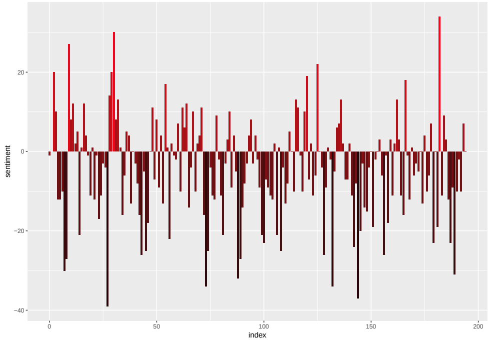
Dracula: the most frequent “trust” words using the nrc lexicon

```
nrc_trust <- get_sentiments("nrc") %>%
  filter(sentiment == "trust")
dracula_words %>%
  inner_join(nrc_trust) %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 471 x 2
##    word          n
##    <chr>     <int>
##  1 good        258
##  2 friend      184
##  3 professor   155
##  4 count       153
##  5 found       153
##  6 god         150
##  7 diary       110
##  8 white       107
##  9 lord         79
## 10 hope         66
## # ... with 461 more rows
```

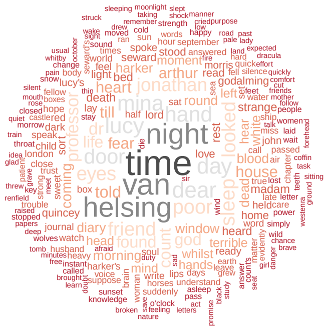
Dracula: how the sentiment changes across the novel using the bing lexicon

```
dracula_sentiment <- dracula_words %>%
  inner_join(get_sentiments("bing")) %>%
count(index = linenumber%/%80, sentiment) %>%
  spread(sentiment, n,
fill = 0) %>% mutate(sentiment = positive - negative)
ggplot(dracula_sentiment, aes(index, sentiment, fill=sentiment)) +
  scale_fill_gradient(low = "#230105", high = "#ff0019") +
  geom_col(show.legend = FALSE)
```



Dracula: a wordcloud

```r
red_palette <- brewer.pal(8,"RdGy")
dracula_words %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, min.freq=6, max.words = 250,
          random.order=FALSE, rot.per=.15, colors=red_palette))
```



Work through each line in the code chunk below from Section 1.5:

- h/t Alice!
  - bind_rows is like rbind() but in dplyr, add an author variable to each tibble
  - mutate is changing the word variable by looking for the regex starting with lower case letters
  - counting words by author
  - dplyr group_by gives each observation a group and performs next functions on those groups
  - mutate creates proportion variable (count of each word divided by the total counts)
  - select(-n) removes the n (count) variable from the tibble
  - spread() is creating a wide format table where each author has a variable
  - gather() is pulling back together so Austen has a variable, but Bronte and Wells are in long format

Packages

```r
library(dplyr)
library(tidytext)
library(janeaustenr)
library(dplyr)
library(stringr)
library(ggplot2)
library(gutenbergr)
library(tidyr)
library(scales)
library(wordcloud)
library(reshape2)
```

Potential problems

- Problem loading wordcloud with R version 3.3.3
- `https://stackoverflow.com/questions/39885408/dependency-slam-is-not-available-when-installing-tm-package`
- Probably better to update R?

```
# install.packages('devtools')
# library(devtools)
# slam_url <- "https://cran.r-project.org/src/contrib/...
# ...Archive/slam/slam_0.1-37.tar.gz"
# install_url(slam_url)
```

Potential problems

- "Error in summarise_impl(.data, dots) : invalid argument type"
- Update dplyr

```
head(data_frame(text = prideprejudice) %>%
  unnest_tokens(sentence, text, token = "sentences"))
```

```
## # A tibble: 6 x 1
##   sentence
##   <chr>
## 1 pride and prejudice  by jane austen    chapter 1   it is a truth univer~
## 2 however little known the feelings or views of such a man may be on his ~
## 3 "\"my dear mr."
## 4 "bennet,\" said his lady to him one day, \"have you heard that netherfi~
## 5 mr.
## 6 bennet replied that he had not.
```