Week 8: Sentiment analysis

Stefan Veleski

13/5/2021

Tidytext sentiment analysis

Importing and wrangling the data

Let's start off by loading in the necessary packages

```
library(tidyverse)
library(syuzhet)
library(gutenbergr)
library(tidytext)
library(textdata)
```

Now let's download metadata for Hardy's works and manually select all novels (metadata about this not included on Project Gutenberg).

```
hardy_meta <- gutenberg_works(author == "Hardy, Thomas")
hardy_meta <- hardy_meta %>%
  slice(c(1,2,3,4,5,6,7,8,10,11,13,18,22,23,24))
```

This does the same for Austen's works.

```
austen_meta <- gutenberg_works(author == "Austen, Jane")
austen_meta <- austen_meta %>%
  slice(1:8)
```

Now let's download all of Hardy's and Austen's texts, and retain a column with the titles of the novels.

```
hardy_tidy_texts <- gutenberg_download(hardy_meta$gutenberg_id,
                                       meta_fields = "title")

austen_tidy_texts <- gutenberg_download(austen_meta$gutenberg_id,
                                       meta_fields = "title")
```

Tokenizing the text by individual words

The following code tokenizes the full text by individual words, but also provides additional metadata about the line and the chapter that the word is located in.

```
austen_tidy_texts <- austen_tidy_texts %>%
  group_by(title) %>%
  mutate(
    linenumber = row_number(),
    chapter = cumsum(str_detect(
      text, # cumulative sum, detect string, column analyzed
      regex("^chapter [\\divxlc]", # regular expression, chapter any char
```

```
        ignore_case = TRUE # include both lower and uppercase
      )
    ))
  ) %>%
  ungroup() %>% # removing the grouping by title set up above
  unnest_tokens(word, text) # tokenization of the text column by word (word per row)

nrc_joy <- get_sentiments("nrc") %>% # tidy text function that extracts the values of 3 sentiment lexic
  filter(sentiment == "joy") # out of the 8 general emotions that the nrc contains, this filters only j
```

The "Text Mining With R" book uses the janeaustenr package, which is not really necessary, as we can represent the full workflow from downloading the books to visualization, which can be used with books from other authors as well.

Most common words associated with the NRC emotion "joy"

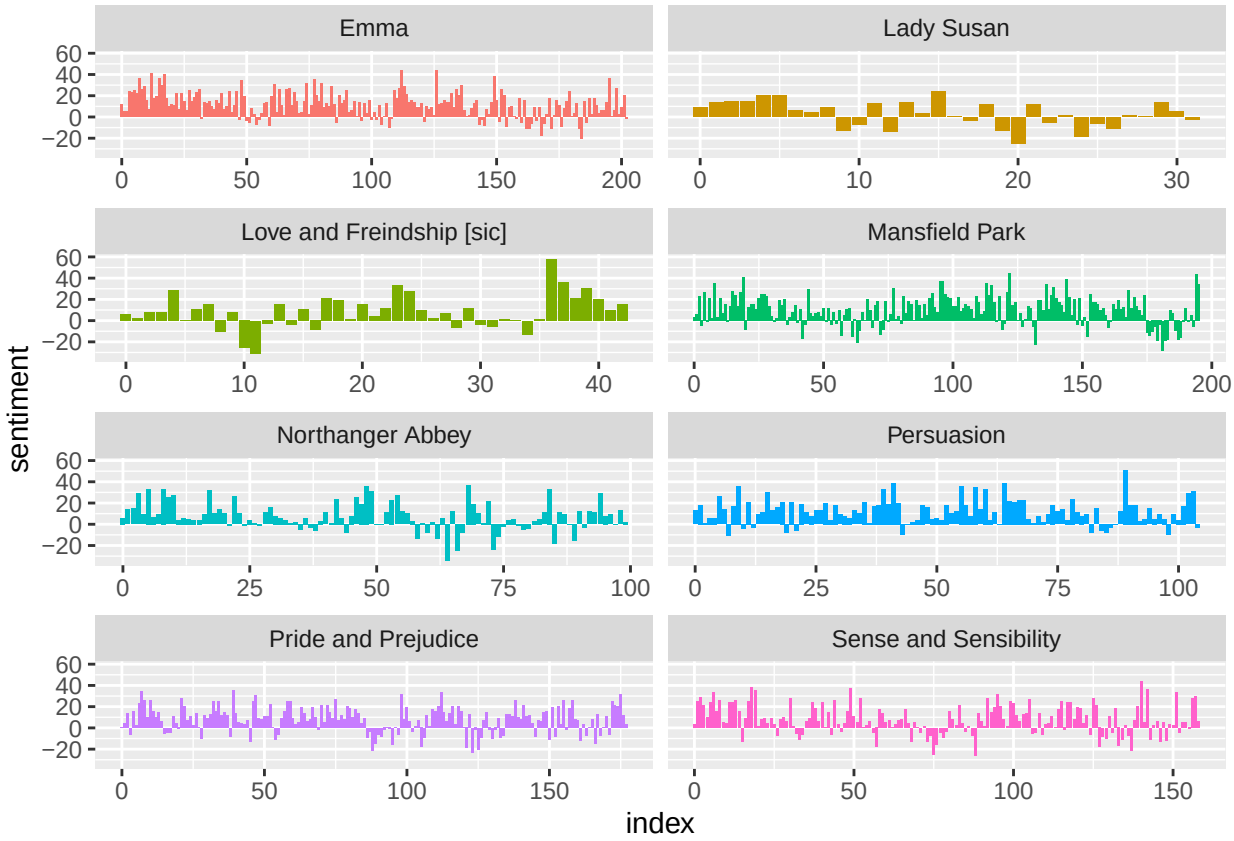
The code below simply counts the most common words in Persuasion that correspond to the joy emotion in the NRC sentiment lexicon

```
austen_tidy_texts %>%
  filter(title == "Persuasion") %>% # only selecting Persuasion from the rest of the books
  inner_join(nrc_joy) %>% # combining two tables together (see data wrangling cheat sheet)
  count(word, sort = TRUE) # counting the words
```

```
## # A tibble: 258 x 2
##    word        n
##    <chr>   <int>
##  1 good      187
##  2 young      84
##  3 found      83
##  4 friend     77
##  5 present    65
##  6 happy      64
##  7 hope       53
##  8 deal       45
##  9 love       42
## 10 spirits    41
## # ... with 248 more rows
```

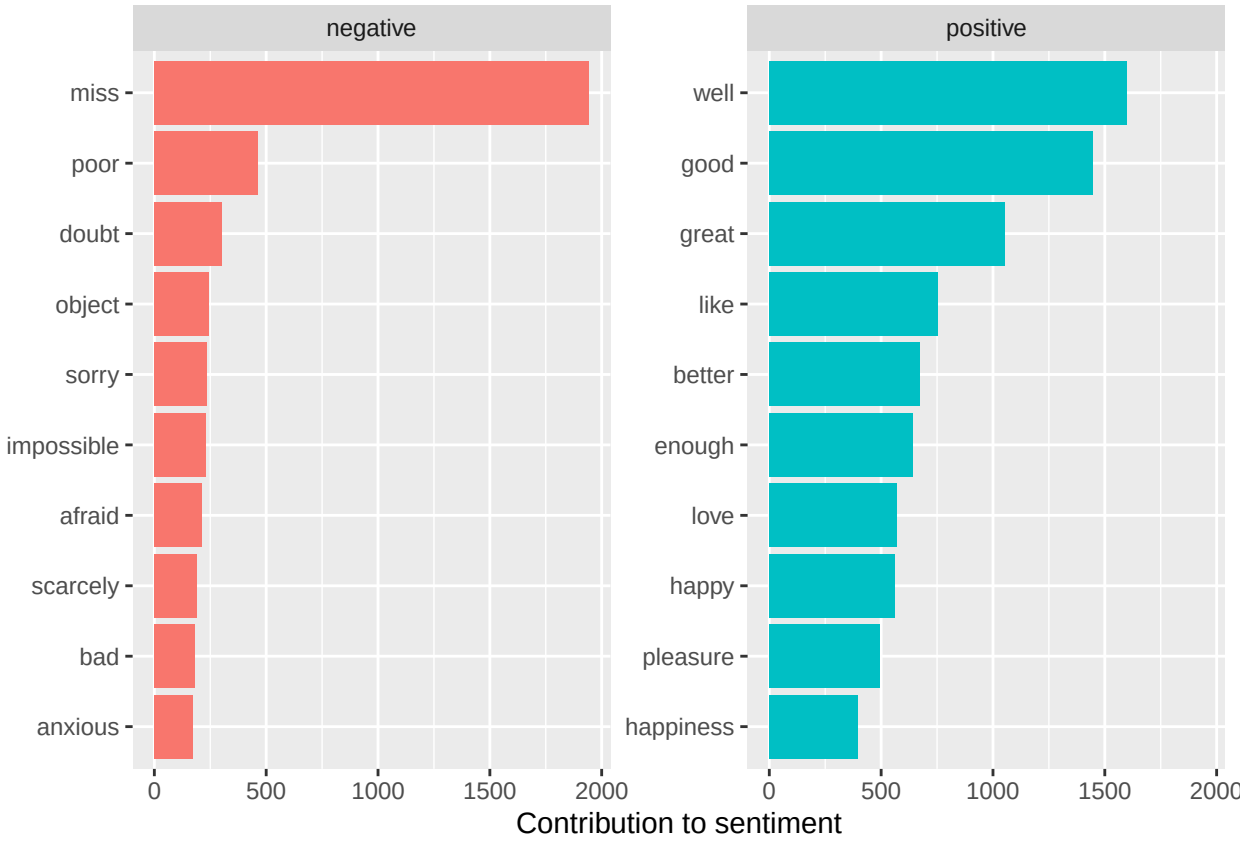
The sentiment plots of each of Austen's novels

```
jane_austen_sentiment <- austen_tidy_texts %>%
  inner_join(get_sentiments("bing")) %>%
  count(title, index = linenumber %/% 80, sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(sentiment = positive - negative)

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



The top ten most positive and most negative words in Austen's work

```
bing_word_counts <- austen_tidy_texts %>%
  inner_join(get_sentiments("bing")) %>% #retain only words that exist in both
  count(word, sentiment, sort = TRUE) %>%
  ungroup()

bing_word_counts %>%
  group_by(sentiment) %>%
  slice_max(n, n = 10) %>%  # slice n to retain top 10
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word, fill = sentiment)) + # n x axis, word y axis, color according to sentiment
  geom_col(show.legend = FALSE) + # barplot, no legends
  facet_wrap(~sentiment, scales = "free_y") + # facet wrapped along sentiment, free y scale
  labs(x = "Contribution to sentiment", # x label
       y = NULL) # no y label
```



Adding miss as a custom stopword to the stopwords lexicon contained in the tidytext package This will be removed from the visualization later on.

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

Sentiment wordcloud

This visualization uses the same package as the wordcloud visualization in week 7, but here the words are ordered according to their sentiment in addition to their frequency.

```
library(wordcloud)
library(reshape2)

set.seed(1234) # reproducibility

austen_tidy_texts %>%
  anti_join(custom_stop_words) %>% # remove all words that have a match in custom_stop_words
  inner_join(get_sentiments("bing")) %>% # retain only words that exist in both
  count(word, sentiment, sort = TRUE) %>% #
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("brown1", "cyan3"),
                   max.words = 150, # maximum number of words in the visualization
                   scale = c(2.5, 0.05), # largest and smallest words in the word cloud
                   min.freq = 1, # minimal frequency of the words in the visualizaion
```

```
                random.order = FALSE, # no random order, size dictated by frequency
               rot.per = 0.35) # percentage of words rotated (35%)
```

Syuzhet package

The syuzhet package is one of the more popular specialized sentiment analysis packages. You can use the bing, afinn, nrc, and syuzhet sentiment lexicons with this package.

Let's first load the required packages and download Austen novels from scratch, so that we have a clean slate.

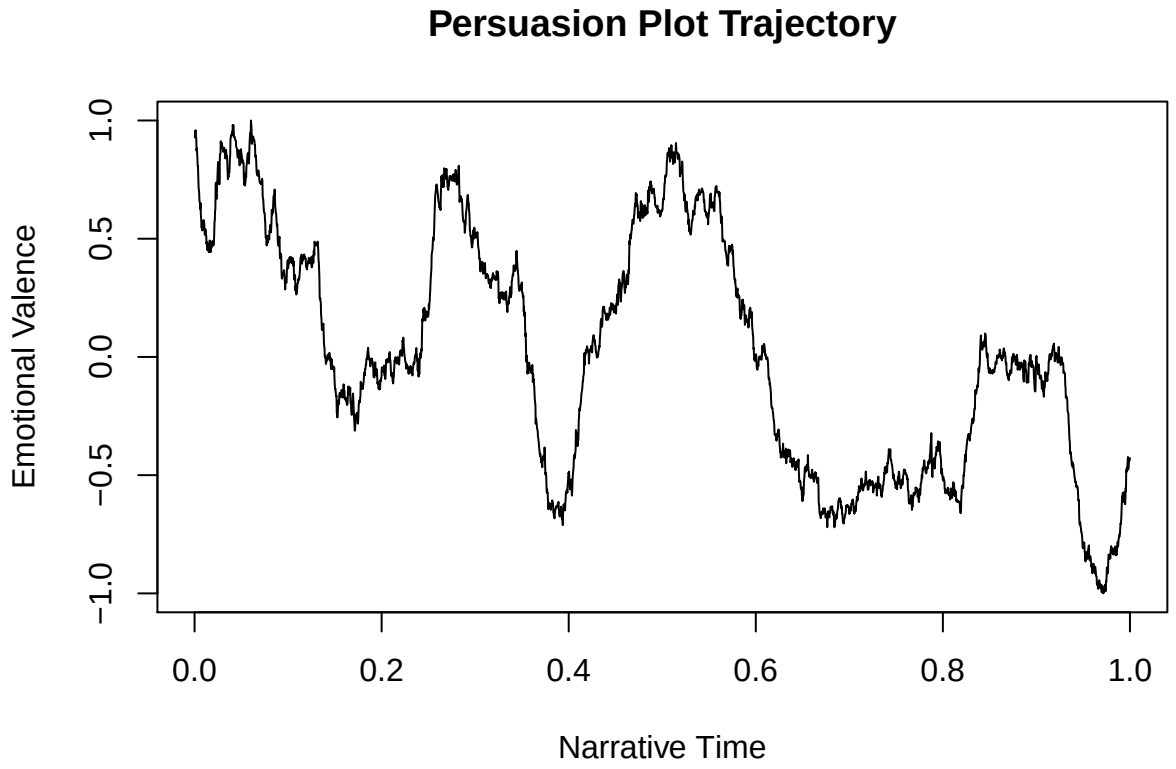
```
library(zoo)
library(syuzhet)

austen_tidy_texts <- gutenberg_download(austen_meta$gutenberg_id,
                                        meta_fields = "title")
```

We will apply the syuzhet package on a single novel (Persuasion), and in order to properly use the package, we will need to transform the text into a single character string.

```
persuasion_tidy <- austen_tidy_texts %>% #this extracts only the text of Persuasion
  filter(title == "Persuasion")

persuasion_string <- paste(persuasion_tidy$text, collapse = " ") # Extracting the text column as a char

persuasion_sentences <- tolower(get_sentences(persuasion_string)) # Extract sentences & lowercase

persuasion_sentiment <- get_sentiment(persuasion_sentences, method = "syuzhet") # Get sentiment for eac
```

The following code produces the first plot.

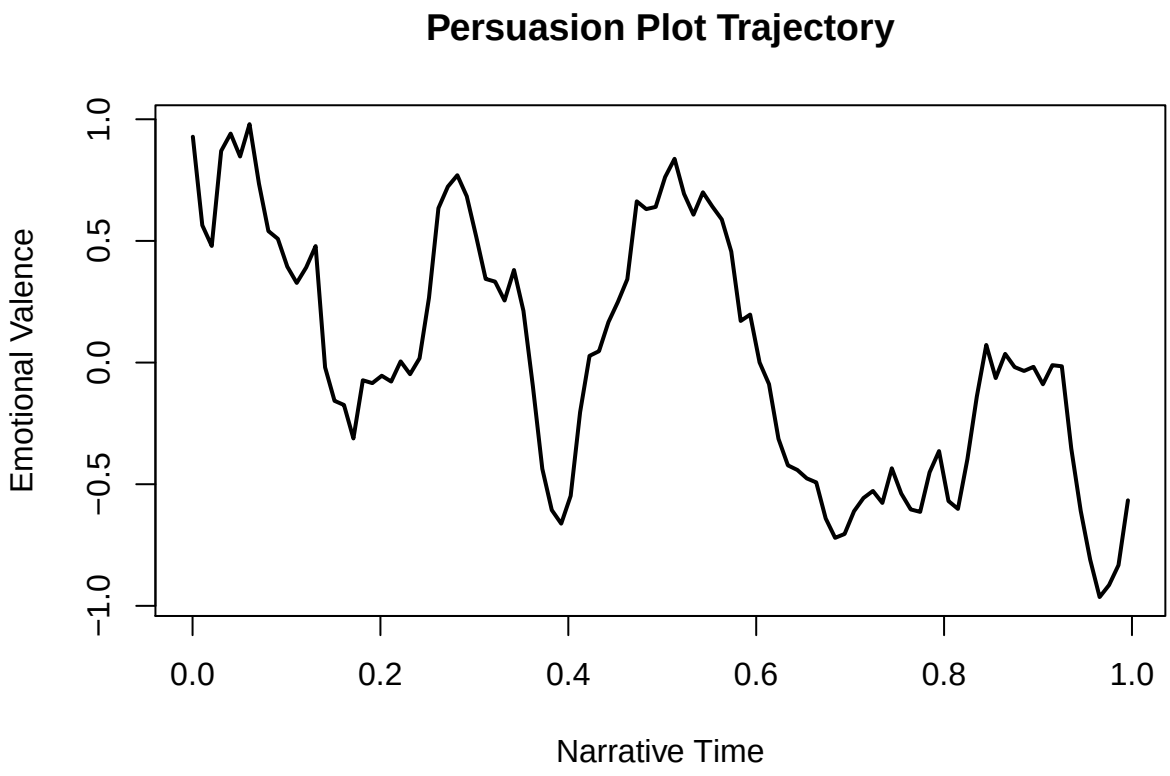
```
pwdw <- round(length(persuasion_sentiment)*.1)
persuasion_rolled <- rollmean(persuasion_sentiment, k=pwdw) #moving/rolling average (1/10 window)
persuasion_list <- rescale_x_2(persuasion_rolled) # rescaled so another novel can be compared
plot(persuasion_list$x,
     persuasion_list$z,
     type="l", # line plot
     main ="Persuasion Plot Trajectory", # title
     col="black", # color of the line
     xlab="Narrative Time", #
     ylab="Emotional Valence") #This is almost perfect, but it's not smoothed out just right.
```



I am not entirely satisfied with the appearance though, and I prefer the approach of dividing the novel into 100 equal chunks, and then plotting the mean sentiment of each chunk.

```
persuasion_sample <- seq(1, length(persuasion_list$x),
                         by=round(length(persuasion_list$x)/100)) # Taking 100 equal chunks of the text
plot(persuasion_list$x[persuasion_sample],
     persuasion_list$z[persuasion_sample],
     type="l", # line plot
     lwd = 2, # line width
     main = "Persuasion Plot Trajectory",# title
     col="black", # color of the line
     xlab="Narrative Time",#x axis name
     ylab="Emotional Valence",#y axis name
```

```
)
```



Package of the week - Sentimentr

A sentiment analysis package that is more sophisticated in several ways than both tidytext and syuzhet, because it takes valence shifters into consideration.

```
library(sentimentr)
library(magrittr)

austen_sentiment <- austen_tidy_texts %>%
  mutate(sentences = get_sentences(text)) %$%
  sentiment_by(sentences, title)

hardy_sentiment <- hardy_tidy_texts %>%
  mutate(sentences = get_sentences(text)) %$%
  sentiment_by(sentences, title)

author_column <- factor(c("Austen","Austen","Austen","Austen","Austen","Austen","Austen","Austen","Hardy
```

Fusing the two data frames together.

```
hardy_vs_austen <- rbind(austen_sentiment, hardy_sentiment)

hardy_vs_austen <- hardy_vs_austen %>%
  cbind(author_column)
```

The resulting dataframe is an ideal use case for a boxplot/violin plot visualization. Let's use the ggstatsplot package we used in week 5.

```
library(ggstatsplot)
options(scipen = 10000)

hardy_vs_austen_plot <- ggbetweenstats(
  data = hardy_vs_austen, # data
  x = author_column, # data for x axis
  y = ave_sentiment, # data for y axis
  title = "Comparison of the mean sentiment of Hardy's and Austen's novels", # Title
  xlab = "Author", # x axis label
  ylab = "Sentiment" # y axis label
)
hardy_vs_austen_plot
```