Sentiment analysis

1 Learning objectives

Workflow for sentiment analysis with `tidytext`:

- **Step 1**: Read in the text
- **Step 2**: Tokenize text into individual words: `tidytext::unnest_tokens()`
- **Step 3**: Remove stop words: `tidytext::stop_words` + `anti_join()`
- **Step 4**: Associate individual words with positive or negative sentiment: `tidytext::sentiments`, `inner_join()`
- **Step 5**: Compute the proportion of positive sentiment words in each chapter: `dplyr` verbs
- **Step 6**: Plot the positive sentiment across chapters: `ggplot`

1.1 Load libraries

```
library(tidyverse)
library(tidytext)
```

1.2 Sentiment analysis of *Life of Pi*

```
# this will load an object called `life_of_pi`
load(here::here("data/life_of_pi.rda"))
dt <- tibble(text = read_lines(life_of_pi))

dt
```

A few things to do in cleaning:

- assign each line a chapter number
- remove all the title lines: "Yann Martel: Life of Pi"
- remove all the page lines: "Page 1"
- remove empty lines

```
dt_clean <- dt |>
  mutate(
    # how can you locate the lines with new chapter?
    chapter = str_...(text, "..."),
    # how can you locate the page lines: "Page 1"?
    page = str_...(text, "..."),
    # how can you locate the title lines: "Yann Martel: Life of Pi"?
    title = str_...(text, "...")) |>
  # assign each line a chapter number
  mutate(chapter = ...(chapter)) |>
  # remove empty lines:
  mutate(blank = str_detect(text, "[a-zA-Z0-9]+", negate = ...)) |>
  # remove all the page and title line, empty line, and lines before chapter 1
  filter(...) |>
  select(-page, -title, -blank)
```

1.3 Unnest token

`unnest_tokens`:

- `tbl`: A data frame
- `output`: Output column to be created as string or symbol.
- `input`: Input column that gets split as string or symbol. The output/input arguments are passed by expression and support quasiquotation; you can unquote strings and symbols.

- `token`: Unit for tokenizing, or a custom tokenizing function. Built-in options are "words" (default), "characters", "character_shingles", "ngrams", "skip_ngrams", "sentences", "lines", "paragraphs", "regex", and "ptb" (Penn Treebank). If a function, should take a character vector and return a list of character vectors of the same length.

```
dt_tokens <- dt_clean |>
  unnest_tokens(output = word, input = ..., token = "...")
dt_tokens
```

1.4 Stop words

What are the most common words in Life of Pi?

```
dt_tokens |> count(word, sort = TRUE)
```

Are they useful for the analysis?

In the `tidytext` package, there is a built-in dataset called `stop_words` that contains common stop words from three lexicons: "onix", "SMART", and "snowball".

```
tidytext::stop_words
stop_words |> count(lexicon)
```

We want to merge the two datasets together so that only the token NOT in the `stopword_df` is preserved - which join should we use?

```
stopword_df <- stop_words |>
  filter(lexicon == "SMART") |>
  select(word)

dt_joined <- dt_tokens |> ..._join(stopword_df, by = "word")
dt_joined
```

1.5 Sentiment data

In the `tidytext` package, there is a built-in dataset called `sentiments` that contains words and their associated sentiments from three lexicons: "AFINN", "bing", and "nrc".

We want to attach each word with a sentiment label (positive or negative). We can only do that for words that's in the `sentiments` dictionary - Which join should we use?

```
tidytext::sentiments
dt_joined2 <- dt_joined |>
  ..._join(sentiments, by = "word")
dt_joined2
```

What happen if you use `left_join()`?

1.6 Visualization

It can be useful to look at how the positive/ negative sentiment changes as the story progresses. How would you calculate the proportion of positive sentiment words in each chapter?

```
sentiment_df <- dt_joined2 |>
  group_by(chapter, sentiment) |>
  summarise(n = n(), .groups = "drop") |>
  pivot_wider(names_from = sentiment, values_from = n) |>
  mutate(prop = positive / (positive + negative))
```

```
sentiment_df2 <- dt_joined2 |>
  group_by(chapter, sentiment) |>
  summarise(n = n(), .groups = "drop") |>
  group_by(chapter)|>
  mutate(prop = n / sum(n)) |>
  filter(sentiment == "positive")
```

```
sentiment_df2 |>
  ggplot(aes(x = chapter, y = prop)) +
  geom_line() +
  geom_point() +
  geom_hline(yintercept = 0.5, linetype = "dashed")
```

Does any part of this plot prompt further investigation?

```
sentiment_df2 |> arrange(-prop)
dt_clean |> filter(chapter == 18)
```

How come we have Chapter 18 labelled for Chapter 19?

```
dt_clean |> filter(chapter == 7)
dt_clean |> filter(chapter == 8)
dt_clean |> filter(chapter == 10)
dt_clean |> filter(chapter == 13)
dt_clean |> filter(chapter == 15)
dt_clean |> filter(chapter == 14)
```

Something is funny at Chapter 14 and 15.

```
dt_clean |> filter(chapter == 14) |> View()
```

What do you find when you `View()` Chapter 14? Can you tell why this is the case? What would you do to fix it?