# Tidy Text

Evan Bowman

2023-03-28

## Contents

## 1 Sentiment Analysis

1. Download the following two complete works from the early 20th century from Project Gutenberg:

- Upton Sinclair: "*The Jungle*" (1906)
- W.E.B. Du Bois: "*The Quest of the Silver Fleece*" (1911)

```
# The Jungle
gutenberg_works() %>%
  filter(title == "The Jungle")
```

```
## # A tibble: 1 x 8
##   gutenberg_id title      author           guten~1 langu~2 guten~3 rights has_t~4
##          <int> <chr>      <chr>              <int> <chr>   <chr>   <chr>  <lgl>
## 1          140 The Jungle Sinclair, Upton       88 en      Contem~ Publi~ TRUE
## # ... with abbreviated variable names 1: gutenberg_author_id, 2: language,
## #   3: gutenberg_bookshelf, 4: has_text
```

```
jungle <- gutenberg_download(140)
```

```
## Determining mirror for Project Gutenberg from http://www.gutenberg.org/robot/harvest
```

```
## Using mirror http://aleph.gutenberg.org
```

```
# The Quest of the Silver Fleece
gutenberg_authors[(str_detect(gutenberg_authors$author, "Du Bois")),]
```

```
## # A tibble: 4 x 7
##   gutenberg_author_id author                alias birth~1 death~2 wikip~3 aliases
##                 <int> <chr>                 <chr>   <int>   <int> <chr>   <chr>
## 1                 226 Du Bois, W. E. B. (~ <NA>     1868    1963 https:~ Bois, ~
## 2                7579 Du Boisgobey, Fortu~ <NA>     1821    1891 https:~ Boisgo~
## 3               34882 Du Bois, Louis       <NA>     1773    1855 https:~ Du Boi~
## 4               44429 Du Bois, Gaylord     <NA>     1899    1993 <NA>    Bois, ~
## # ... with abbreviated variable names 1: birthdate, 2: deathdate, 3: wikipedia
```

```
gutenberg_works(gutenberg_author_id == 226)
```

```
## # A tibble: 10 x 8
##    gutenberg_id title              author guten~1 langu~2 guten~3 rights has_t~4
##           <int> <chr>              <chr>    <int> <chr>   <chr>   <chr>  <lgl>
##  1          408 "The Souls of Bla~ Du Bo~     226 en      Africa~ Publi~ TRUE
##  2         5685 "The Conservation~ Du Bo~     226 en      <NA>    Publi~ TRUE
##  3        15210 "Darkwater: Voice~ Du Bo~     226 en      Africa~ Publi~ TRUE
##  4        15265 "The Quest of the~ Du Bo~     226 en      Africa~ Publi~ TRUE
##  5        15359 "The Negro"        Du Bo~     226 en      Africa~ Publi~ TRUE
##  6        17700 "The Suppression ~ Du Bo~     226 en      Africa~ Publi~ TRUE
##  7        31254 "The Conservation~ Du Bo~     226 en      Africa~ Publi~ TRUE
##  8        62582 "The social evolu~ Du Bo~     226 en      <NA>    Publi~ TRUE
##  9        62799 "John Brown"       Du Bo~     226 en      <NA>    Publi~ TRUE
## 10        66398 "The Gift of Blac~ Du Bo~     226 en      <NA>    Publi~ TRUE
## # ... with abbreviated variable names 1: gutenberg_author_id, 2: language,
## #   3: gutenberg_bookshelf, 4: has_text
```

```
fleece <- gutenberg_download(15265)
```

2. Write a single function, `tidydoc()`, to take an argument of a downloaded book tibble and return it in tidy text format.

- The function must add line and chapter numbers as variables before removing any lines.
- The function must unnest tokens at the word level.
- The function must remove any Project Gutenberg formatting so only the words remain.
- The function must remove any stop_words and filter out any `NA`s.
- The function must remove any front matter (words before Chapter 1).
- Consider the following regex as an example
  - `c_pattern <- regex("(^chapter [\\divxlc]|^_(?!Contents)(?!Note)[:alpha:]+-?[:alpha:]*_$)", ignore_case = TRUE)`

```
tidyBook <- function(bookDF) {
  stopifnot(is.data.frame(bookDF))

  bookDF %>%
    mutate(linenumber = row_number(),
         chapter = cumsum(str_detect(text,
                                    regex("(^chapter [\\divxlc]|^_(?!Contents)(?!Note)[:alpha:]+-?[:al
                                          ignore_case = TRUE))),
         .before = text) %>%
    unnest_tokens(word, text) %>%
    select(-gutenberg_id) %>%
    mutate(word = str_extract(word, "[a-z']+")) %>%
    anti_join(stop_words, by = "word") %>%
    filter(!is.na(word), chapter !=0)
}
```

3. Use the function from step 2 to tidy each book

- Then add `book` and `author` as variables and save each tibble to a new tibble.
- Show the number of rows in each book's tibble? should be 41,606 for Fleece and over 48,306 for Jungle.

```
jungle_tidy <- jungle %>%
  tidyBook() %>%
  mutate(book = "The Jungle",
         author = "Sinclair",
         .before = linenumber)

fleece_tidy <- fleece %>%
  tidyBook() %>%
  mutate(book = "The Quest of the Silver Fleece",
         author = "Du Bois",
         .before = linenumber)

# Row Checking
nrow(jungle_tidy)
```

```
## [1] 47828
```

```
nrow(fleece_tidy)
```

```
## [1] 41606
```

4. Use a {dplyr} function to combine the two tibbles into a new tibble

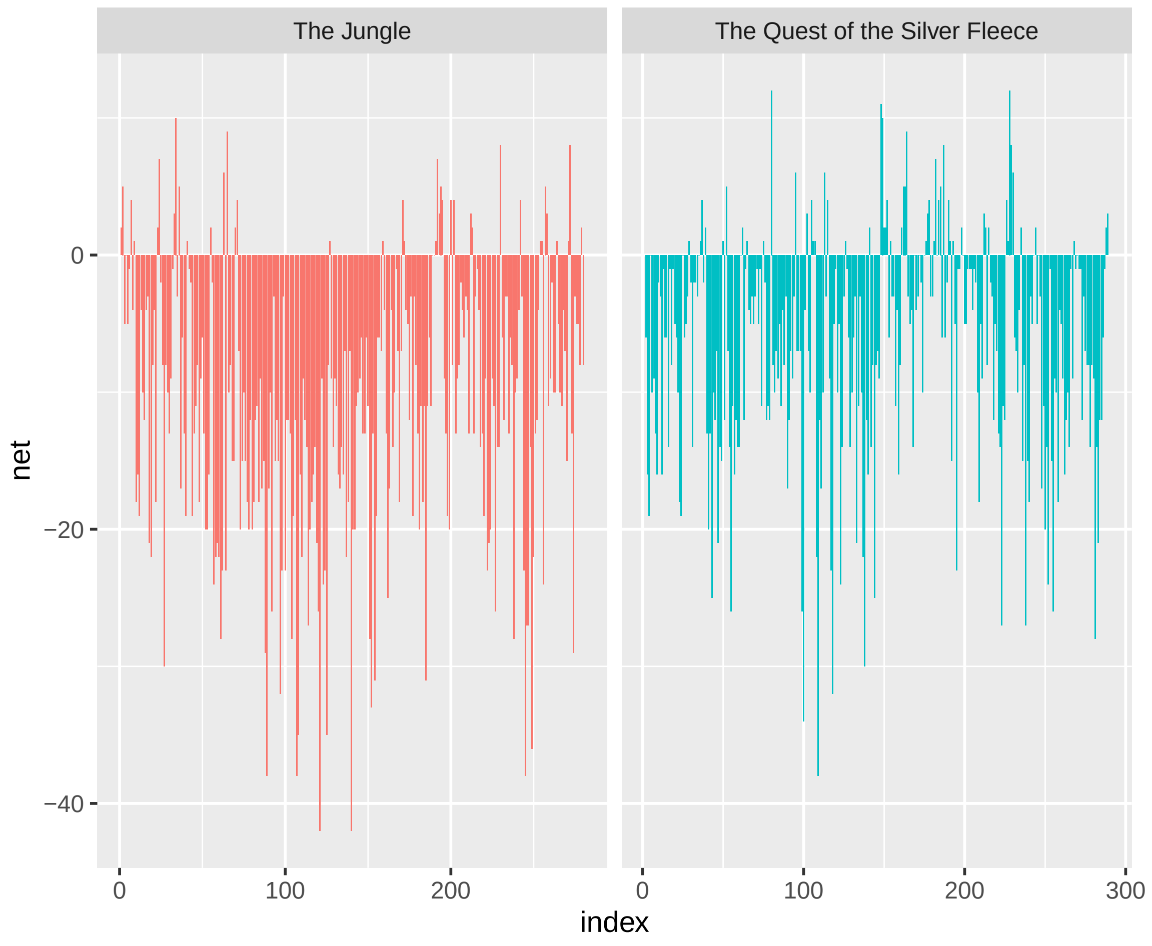
- Show the number of rows. It should be over 89K

```
combined_tidy <- bind_rows(jungle_tidy, fleece_tidy)
```

5. Measure the net sentiment using bing for each block of **50** lines

- Plot the sentiment for each book in an appropriate faceted plot - either line or column.
- Remove the legend.
- Save the plot to a variable and then show the plot.
- Interpret the plots for each book and compare them.

```
combined_tidy %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  count(index = linenumber %/% 50, sentiment, book, sort = TRUE) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = list(n=0)) %>%
  mutate(net = positive - negative) %>%
  ggplot(aes(index, net, fill = book)) +
  geom_col(show.legend = F) +
  facet_wrap(~book, scales = "free_x")
```



Both books are overwhelmingly negative in sentiment when broken into indexes of 50. The Jungle has larger indexes of negative sentiment than that of The Quest of the Silver Fleece.

6. Measure the total for each nrc sentiment in each block of **500** lines and then,

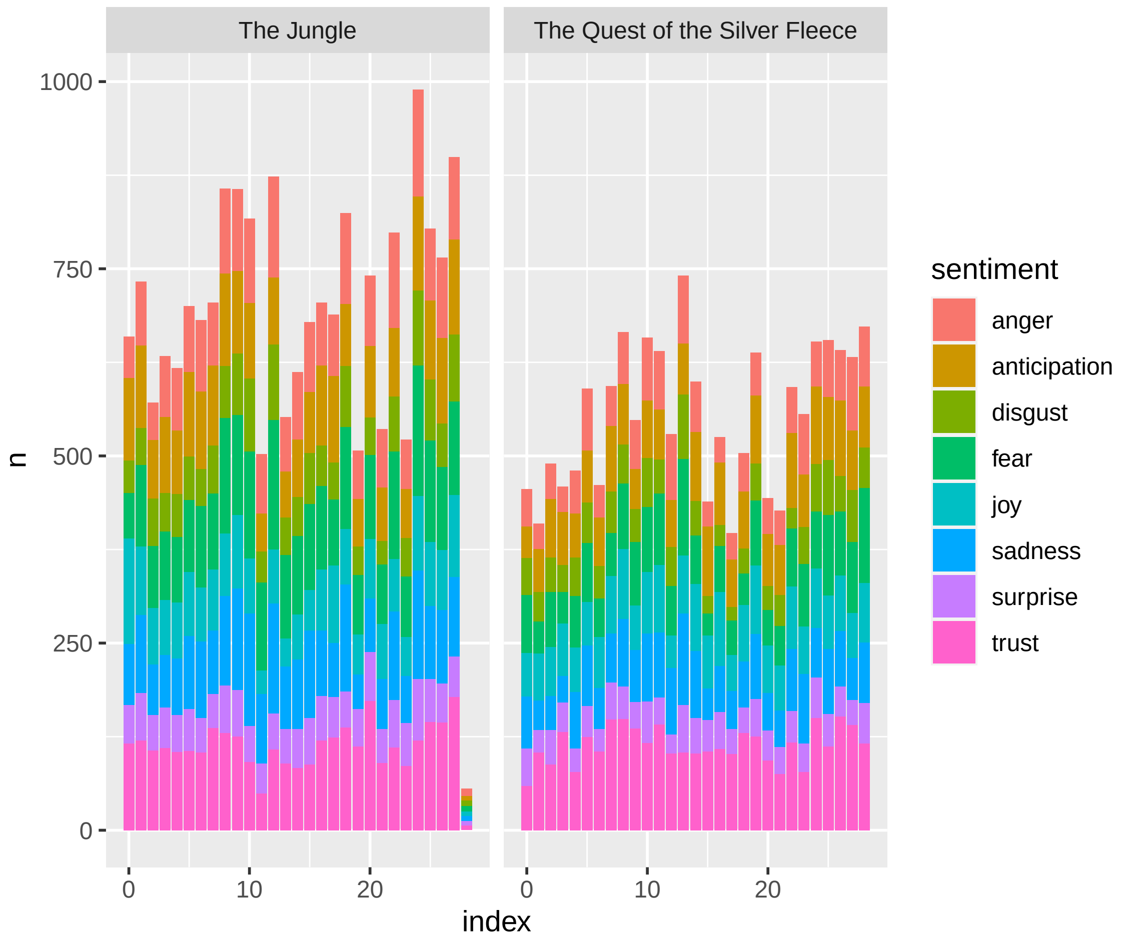
- Filter out the "positive" and "negative" and save to a new variable. You should have 464 observations.

```r
sentiment_words <- combined_tidy %>%
  inner_join(get_sentiments("nrc"), by = "word") %>%
  count(index = linenumber %/% 500, sentiment, book, sort = TRUE) %>%
  filter(sentiment != "positive" & sentiment != "negative")
nrow(sentiment_words)
```

```
## [1] 464
```

- Plot the count of the sentiments for each block in each book in an appropriate faceted plot.
- Interpret the plots for each book and then compare them.
- Why did the values drop off so suddenly at the end?

```
ggplot(sentiment_words, aes(index, n)) +
  geom_col(aes(fill = sentiment)) +
  facet_wrap(~book)
```



The sentiment structure of both books are relatively similar when blocked by 500. Values seem to drop off at the end because, as the story concludes, I feel words carry less sentiment as there is little plot building or explaining occurring. I would also hypothesize that there are fewer words in the final block given the value we assigned to each.

7. Using bing, create a new data frame with the counts of the positive and negative sentiment words for each book.

```
common_sentiment <- combined_tidy %>% group_by(book) %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

- Show the “top 20” most frequent words in a book (*looking across both books*) along with their book, sentiment, and count, in descending order by count.

```r
common_sentiment %>%
  arrange(desc(n)) %>%
  slice(1:20) %>%
  select(2, 1, 3, 4)
```

```
## # A tibble: 20 x 4
##    word    book                           sentiment     n
##    <chr>   <chr>                          <chr>     <int>
##  1 miss    The Quest of the Silver Fleece negative    467
##  2 slowly  The Quest of the Silver Fleece negative    124
##  3 dark    The Quest of the Silver Fleece negative     96
##  4 poor    The Jungle                     negative     80
##  5 cold    The Jungle                     negative     79
##  6 hard    The Jungle                     negative     61
##  7 lost    The Jungle                     negative     61
##  8 wild    The Jungle                     negative     55
##  9 love    The Quest of the Silver Fleece positive     54
## 10 fell    The Jungle                     negative     51
## 11 fell    The Quest of the Silver Fleece negative     50
## 12 mighty  The Quest of the Silver Fleece positive     48
## 13 death   The Jungle                     negative     47
## 14 free    The Jungle                     positive     44
## 15 killing The Jungle                     negative     43
## 16 silent  The Quest of the Silver Fleece positive     43
## 17 cry     The Jungle                     negative     41
## 18 agony   The Jungle                     negative     40
## 19 hard    The Quest of the Silver Fleece negative     40
## 20 dead    The Jungle                     negative     39
```

- What are the positive words in the list of "top 20"?

```r
common_sentiment %>%
  arrange(desc(n)) %>%
  slice(1:20) %>%
  filter(sentiment != "negative") %>%
  select(2, 1, 3, 4)
```

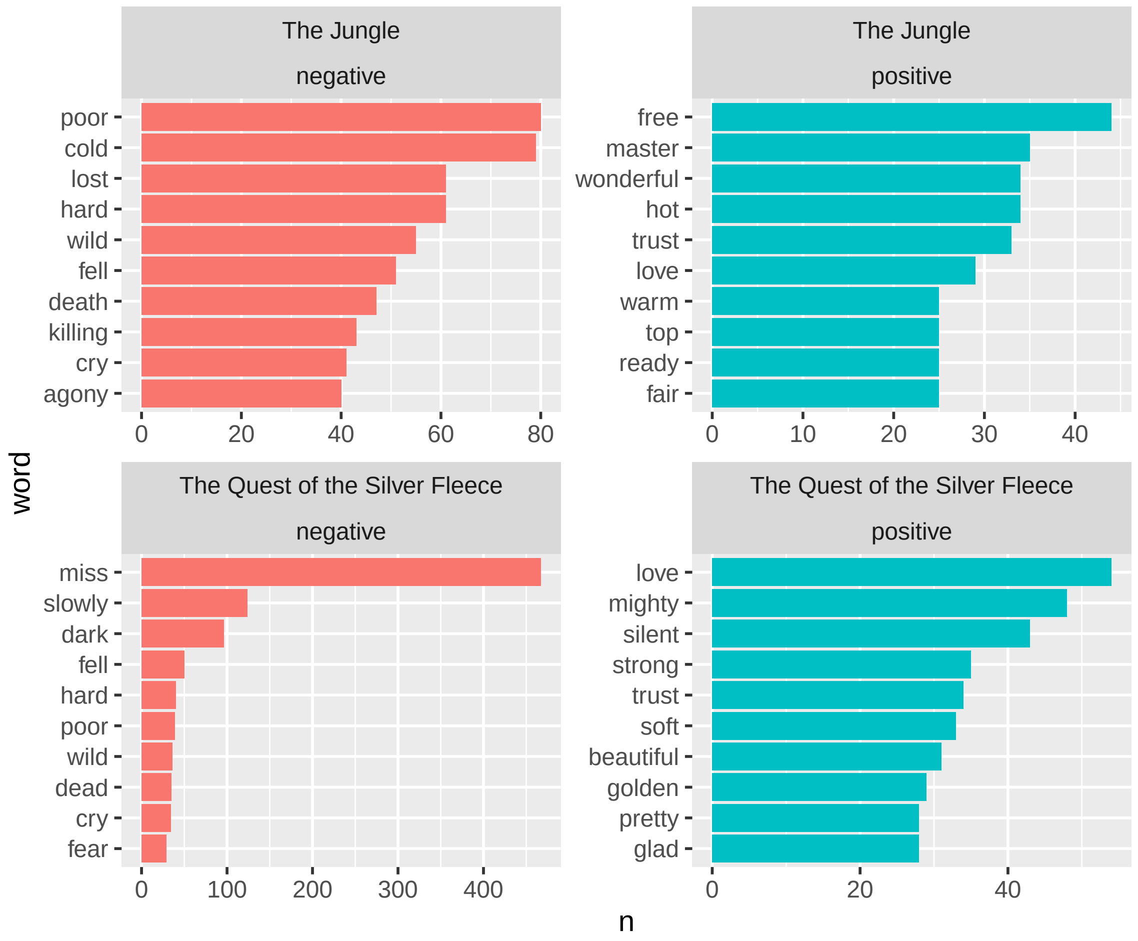
```
## # A tibble: 4 x 4
##   word   book                           sentiment     n
##   <chr>  <chr>                          <chr>     <int>
## 1 love   The Quest of the Silver Fleece positive     54
## 2 mighty The Quest of the Silver Fleece positive     48
## 3 free   The Jungle                     positive     44
## 4 silent The Quest of the Silver Fleece positive     43
```

8. Plot the top ten positive and top ten negative sentiments just for each book, faceting by book and sentiment (total 20 words per book).

- Ensure each facet has the words in the proper order for that book.
- Identify any that may be inappropriate for the context of the book and should be excluded from the sentiment analysis.

```r
# Problem with reorder_within inside the aesthetic. Will Try a mutate before plot
# common_sentiment %>% group_by(book, sentiment) %>%
#   slice_max(order_by = n, n = 10, with_ties = F) %>%
#   ggplot(aes(word, reorder_within(n, n, book), fill = book)) +
#   geom_col() +
#   scale_x_reordered() +
#   facet_wrap(book ~ sentiment, scales = "free_x") +
#   coord_flip()

common_sentiment %>% group_by(book, sentiment) %>%
  slice_max(order_by = n, n = 10, with_ties = F) %>% #Eliminates the word strange that is tied with fea
  mutate(word = reorder_within(word, n, book)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = F) +
  facet_wrap(book ~ sentiment, scales = "free") +
  coord_flip() +
  scale_x_reordered() # Eliminates book name by each word
```

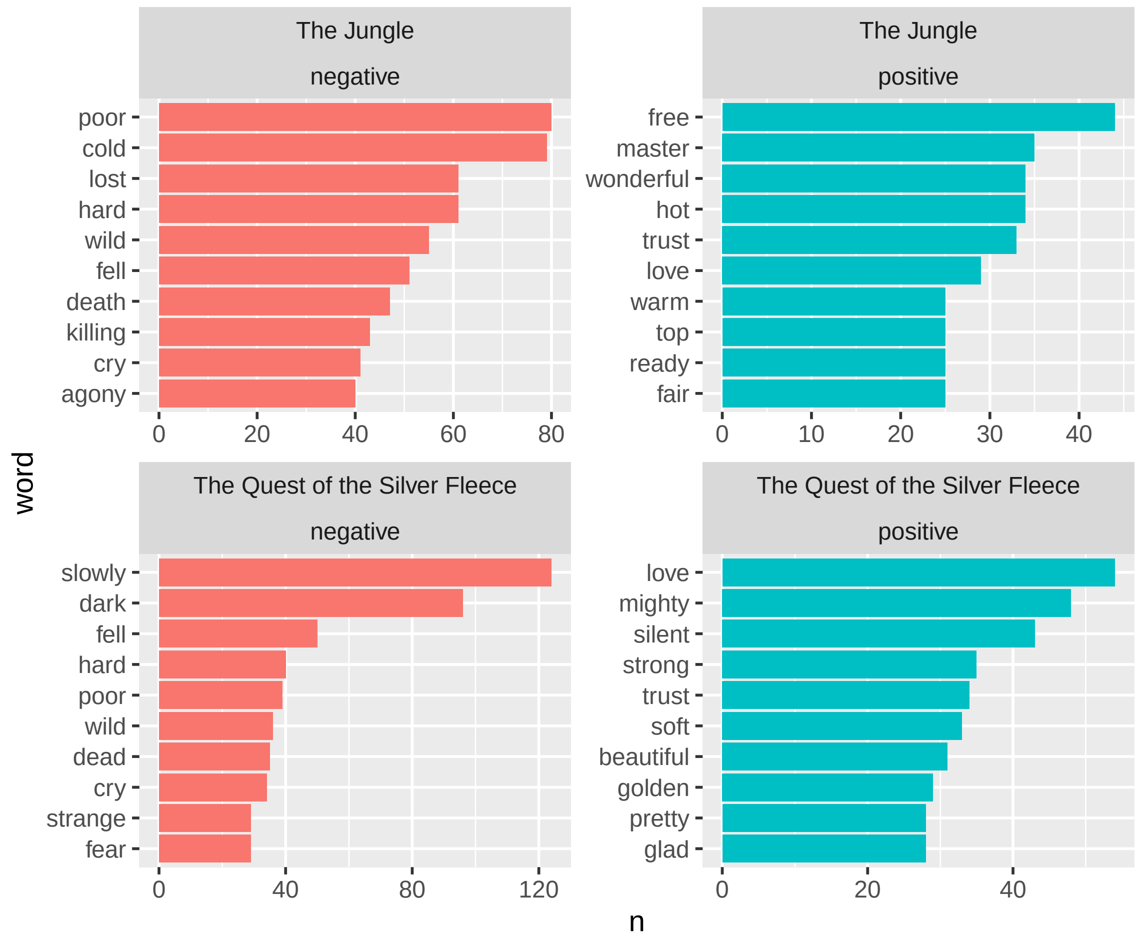


For negative sentiment in The Quest of the Silver Fleece, the word with the most repetition is "miss". I would suspect that this is conflated with the word miss that references a woman. We will remove this to ensure better accuracy of sentiment.

9. Remove the inappropriate word(s) from the analysis.

- Replot the top 10 for each sentiment per book from step 8 (total 20 words per book).
- Interpret the plots

```
common_sentiment %>% group_by(book, sentiment) %>%
  filter(word != "miss") %>% # Same code with added filter command to eliminate miss
  slice_max(order_by = n, n = 10, with_ties = F) %>%
  mutate(word = reorder_within(word, n, book)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = F) +
  facet_wrap(book ~ sentiment, scales = "free") +
  coord_flip() +
  scale_x_reordered()
```



The results of this plot further confirmed our hypothesis from the previous index plots. The top ten words identified as having negative sentiment for each book greatly outnumber those with positive sentiment.

10. Rerun the analysis from step 5 and recreate the plot with the title "Custom Bing".

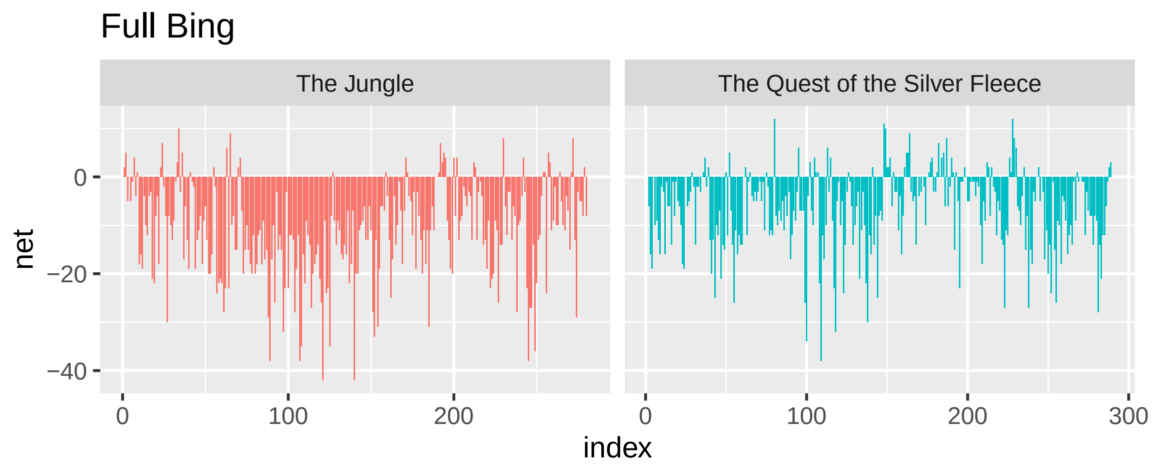
- Show both the original step 5 plot with the new plot in the same output graphic, one on top of the other so the books are above each other (in columns) with original in the top row and custom in the second row.

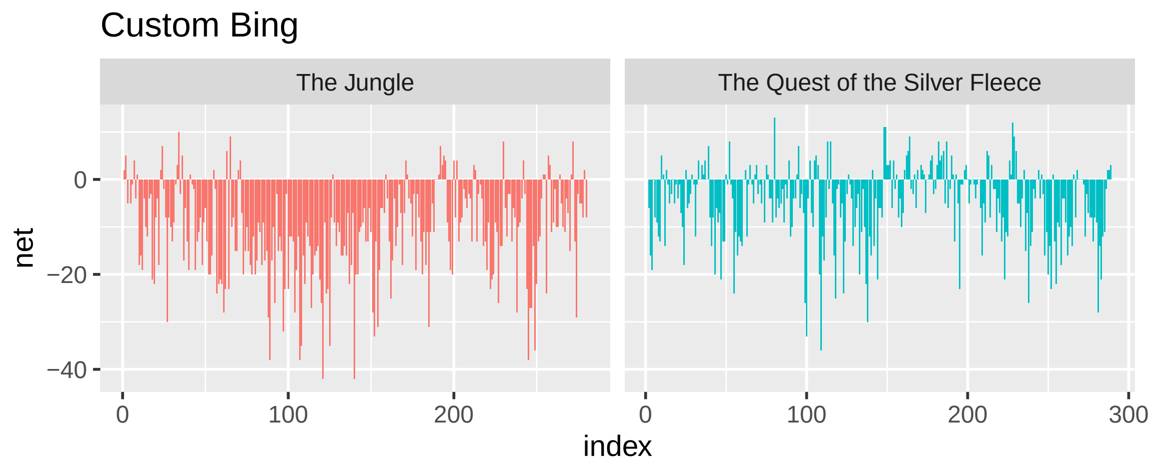
- Interpret the plots

```r
#With Miss
full_bing <- combined_tidy %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  count(index = linenumber %/% 50, sentiment, book, sort = TRUE) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = list(n=0)) %>%
  mutate(net = positive - negative) %>%
  ggplot(aes(index, net, fill = book)) +
  geom_col(show.legend = F) +
  facet_wrap(~book, scales = "free_x") +
  ggtitle("Full Bing")

custom_bing <- combined_tidy %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  filter(word != "miss") %>%
  count(index = linenumber %/% 50, sentiment, book, sort = TRUE) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = list(n=0)) %>%
  mutate(net = positive - negative) %>%
  ggplot(aes(index, net, fill = book)) +
  geom_col(show.legend = F) +
  facet_wrap(~book, scales = "free_x") +
  ggtitle("Custom Bing")

gridExtra::grid.arrange(full_bing, custom_bing)
```





There is relatively little change in the net sentiment for The Jungle's indexes. This makes sense as miss was not one of the negative words used most. There are some small differences in The Quest of the Silver Fleece's net indexes. As shown in the graph, the number of positive indexes has increased and, inversely, some of the negative indexes have decreased in size.

## 2 tf-idf for A Selection of Mark Twain's books

1. Use a single call to download all the following *complete* books by author Mark Twain from Project Gutenberg

- Use the `meta_fields` argument to include the book title as part of the download
- *Huckleberry Finn*, *Tom Sawyer* , *Connecticut Yankee in King Arthur's Court*, *Life on the Mississippi* , *Prince and the Pauper*, and *A Tramp Abroad*

```
twain <- gutenberg_download(c(76, 74, 86, 245, 1837, 119), meta_fields = "title")
```

2. Modify your earlier function to create a new one called `tidydoctf()` to output a tf-idf ready dataframe where now you **leave the the stop words in the text**.

- Unnest at the word level, remove any formatting, and get rid of any `NA`s.

- Add the count for each word by title.
- Use your function to tidy the downloaded texts and save to a variable. It should have 56,759 rows.

```r
tidydoctf <- function(bookDF) {
  stopifnot(is.data.frame(bookDF))

  bookDF %>%
    unnest_tokens(word, text) %>%
    mutate(word = str_extract(word, "[a-z']+")) %>%
    filter(!is.na(word)) %>%
    count(title, word, sort = TRUE) -> book_words

  book_words %>%
  group_by(title) %>%
  summarize(total = sum(n), .groups = "drop") ->
  total_words

  book_words %>%
  left_join(total_words, by = "title")
}
twain_tf <- tidydoctf(twain)
nrow(twain_tf)
```

```
## [1] 56478
```

3. Calculate the tf-idf

- Save back to the data frame.

```r
twain_tf_idf <- twain_tf  %>%
  bind_tf_idf(word, title, n)
```

4. Plot the tf for each book using a faceted graph.

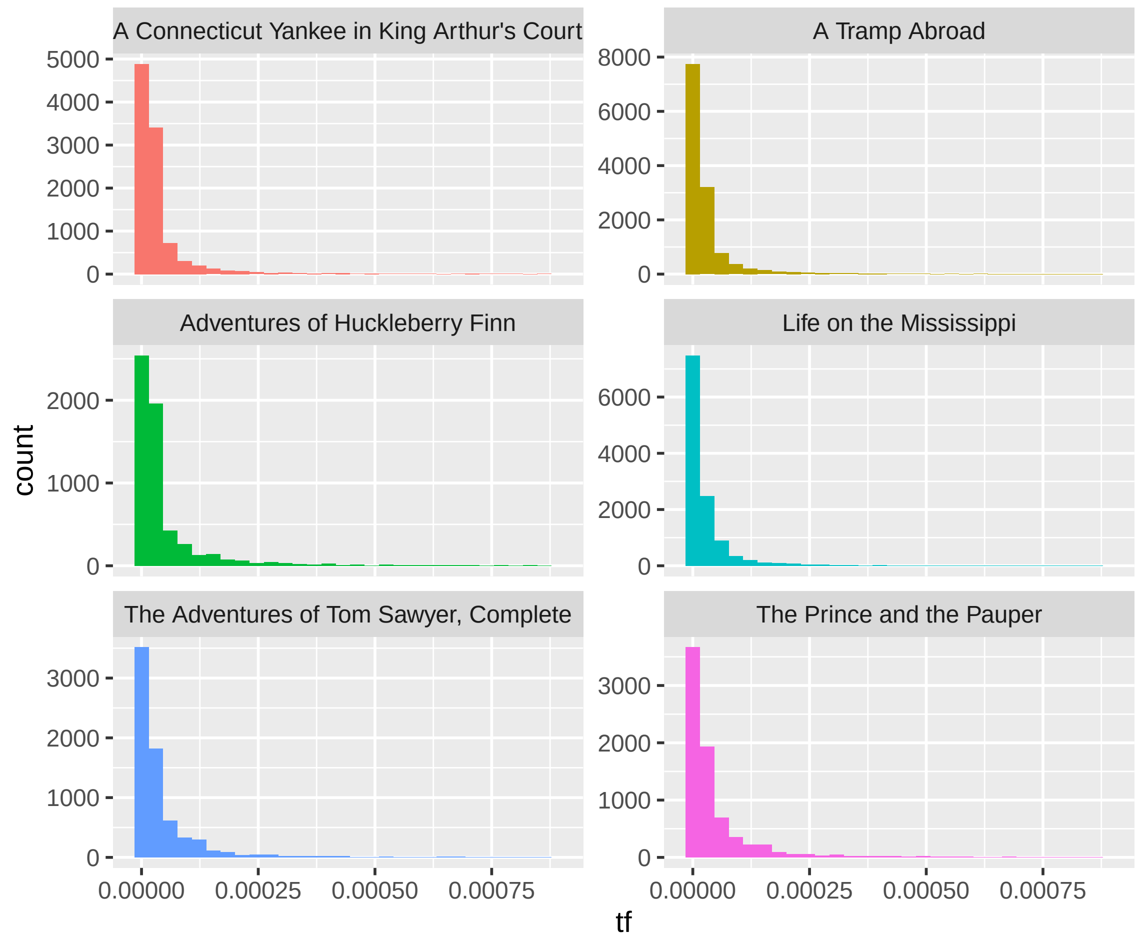
- Facet by book and *constrain the data or the X axis to see the shape of the distribution.*

```r
twain_tf_idf %>%
ggplot(aes(tf, fill = title)) +
  geom_histogram(show.legend = FALSE) +
  xlim(NA, 0.0009) +
  facet_wrap(~ title, ncol = 2, scales = "free_y")
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.

## Warning: Removed 846 rows containing non-finite values (stat_bin).

## Warning: Removed 6 rows containing missing values (geom_bar).
```



5. Show the words with the 15 highest tf-idfs across across all books

- Only show those rows.

```
twain_tf_idf %>%
  arrange(desc(tf_idf)) %>%
  select(word, title, tf_idf) %>%
  slice(1:15)
```

```
## # A tibble: 15 x 3
##    word   title                                  tf_idf
##    <chr>  <chr>                                   <dbl>
##  1 hendon The Prince and the Pauper             0.00406
##  2 becky  The Adventures of Tom Sawyer, Complete 0.00285
##  3 huck   The Adventures of Tom Sawyer, Complete 0.00248
##  4 canty  The Prince and the Pauper             0.00244
##  5 didn   Adventures of Huckleberry Finn        0.00221
##  6 couldn Adventures of Huckleberry Finn        0.00218
##  7 ain    Adventures of Huckleberry Finn        0.00193
##  8 prince The Prince and the Pauper             0.00186
##  9 wouldn Adventures of Huckleberry Finn        0.00184
## 10 joe    The Adventures of Tom Sawyer, Complete 0.00163
## 11 en     Adventures of Huckleberry Finn         0.00144
## 12 dey    Adventures of Huckleberry Finn         0.00130
## 13 don    The Adventures of Tom Sawyer, Complete 0.00126
## 14 don    Adventures of Huckleberry Finn         0.00123
## 15 sid    The Adventures of Tom Sawyer, Complete 0.00122
```

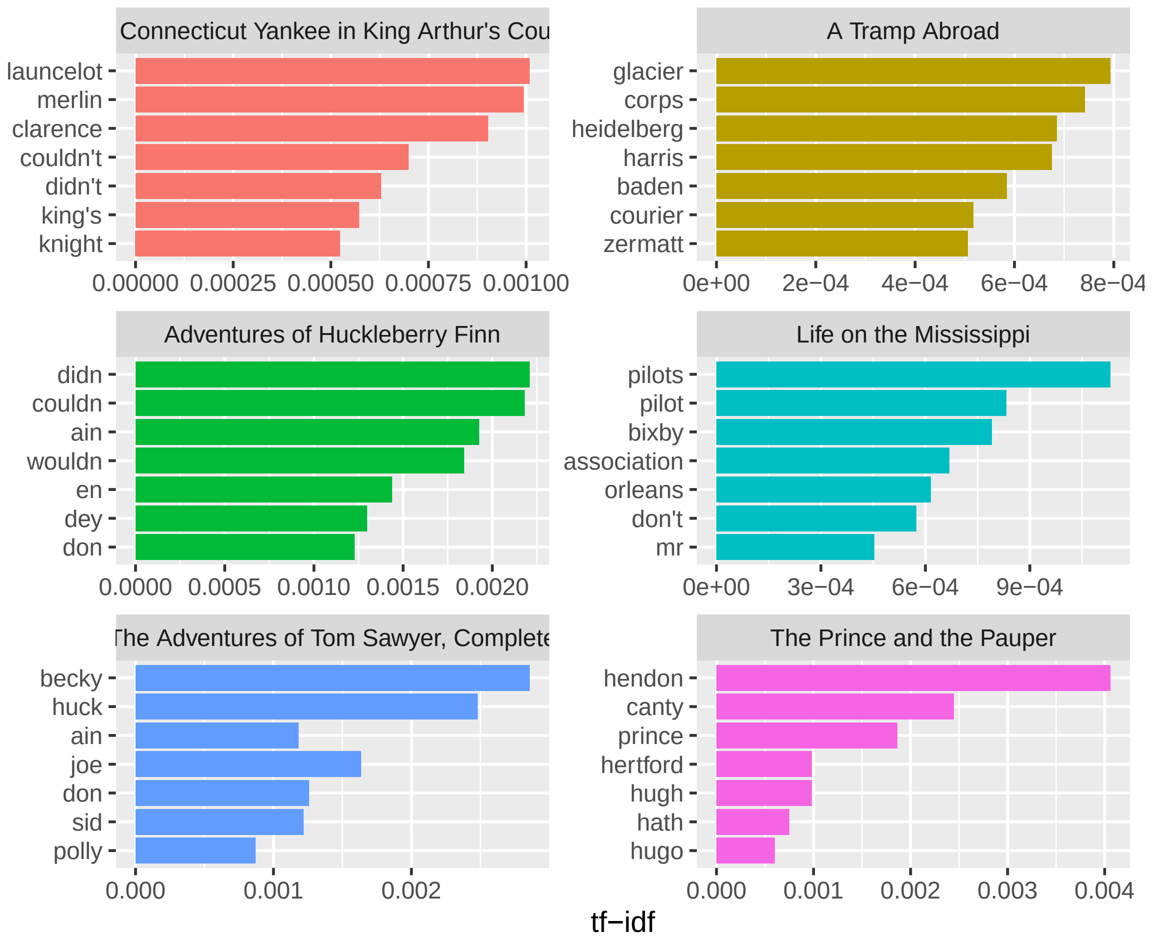
- From just looking at the output, which ones look like possible names?

From the top 15 words with the highest fr-idf, Hendon, Becky, Huck, Canty, Joe, and Sid seem like they could be names. I do not think Don was a popular name in this time period.

6. Plot the top 7 tf_idf words from each book again faceted by book.

- Sort in descending order of tf_idf

```r
twain_tf_idf %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = fct_rev(parse_factor(word))) %>%
  group_by(title) %>%
  slice_max(order_by = tf_idf, n = 7) %>%
  ungroup() %>%
  ggplot(aes(word, tf_idf, fill = title)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~title, ncol = 2, scales = "free") +
  coord_flip()
```



- Interpret the plots.

It seems the majority of words are names, titles, or prefix/family prefixes (Mr, aunt, etc.). The remaining words also seem to be overwhelmingly traditional nouns. There are few verbs or adjectives visualized within these graphs.