Assignment 6

Yaniv Bronshtein

3/5/2021

Load the required libraries

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 4.0.4
```

```
library(gutenbergr)
```

```
## Warning: package 'gutenbergr' was built under R version 4.0.4
```

```
library(textdata)
```

```
## Warning: package 'textdata' was built under R version 4.0.4
```

```
library(topicmodels)
```

```
## Warning: package 'topicmodels' was built under R version 4.0.4
```

```
library(broom)
```

PROBLEM 1:

1.Download the texts of Treasure Island and Kidnapped by Robert Louis Stevenson, using the gutenberg package. You can find the numbers of the two books by searching the website https://www.gutenberg.org/ (Links to an external site.). Type the name of the book in the quick search window and click "go". Then you should be able to find the web page for the book, and the e-book number can be found in the bibliographic record section.

Use gutenbergr to download and store each novel as a tibble, noting the linenumbers

```
treasure_island_t <- gutenberg_download(c(120)) %>% as_tibble() %>%
  mutate(linenumber = row_number())
```

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

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

```
kidnapped_t <- gutenberg_download(c(421)) %>% as_tibble() %>%
  mutate(linenumber = row_number())
```

For each novel tibble do the following: 1. Call unnest_tokens() to break down each line in the novel into word tokens These will be stored separately on each line 2. Perform an anti_join() between the unnested_token tibble and the list of stop words to remove the useless words that detract from the meaning of each novel 3. Filter out the NA words

```
tidy_treasure_island_t <- treasure_island_t %>%
  unnest_tokens(word,
                text,
                token = "words") %>%
  anti_join(stop_words) %>%
  filter(word != "NA")
```

```
## Joining, by = "word"
```

```
tidy_kidnapped_t <- kidnapped_t %>%
  unnest_tokens(word,
                text,
                token = "words") %>%
  anti_join(stop_words) %>%
  filter(word != "NA")
```

```
## Joining, by = "word"
```

2.Find the 10 most common words (that are not stop words) in each novel.

For each novel do the following: 1. Call count() with the sort flag set to true to count the frequency of the words 2. Call select(word) to only view the words 2. Call head(10) to only display the first 10 results

```
top10_words_treasure_island <- tidy_treasure_island_t %>%
  count(word, sort = TRUE) %>%
  select(word) %>%
  head(10)

top10_words_treasure_island
```

```
## # A tibble: 10 x 1
##    word
##    <chr>
##  1 captain
##  2 silver
##  3 doctor
##  4 time
##  5 hand
##  6 sea
##  7 hands
##  8 i'll
##  9 cried
## 10 sir
```

```
top10_words_kidnapped <- tidy_kidnapped_t %>%
  count(word, sort = TRUE) %>%
  select(word) %>%
  head(10)

top10_words_kidnapped
```

```
## # A tibble: 10 x 1
##    word
##    <chr>
##  1 alan
##  2 ye
##  3 house
##  4 time
##  5 set
##  6 hand
##  7 sir
##  8 cried
##  9 day
## 10 country
```

3.

(i) Create a visualization on the similarity/dissimilarity between the proportions of the non stop words (i.e., words that are not stop words) in the two books, and calculate the correlation between them.

To create the frequency table perform the following steps: 1. For each tidy-d tibble, create an additional column for the title 2. Call bind_rows() to stack the tibbles on top of each other 3. Create a new column using mutate() with the extracted words 4. Call count() to tally the title,word combination 5. Call group_by(title) to create the grouping as a precursor to calculating the word proportions 6. Call select(-n) to remove the n column from the tibble 7. Call pivot wider to generate a tibble with two columns for each of the novels containing only the proportion values

```
frequency_t <- bind_rows(mutate(tidy_treasure_island_t, title = "Treasure_Island"),
                         mutate(tidy_kidnapped_t, title = "Kidnapped")) %>%
  mutate(word = str_extract(word, "[a-z']+")) %>%
  count(title, word) %>%
```

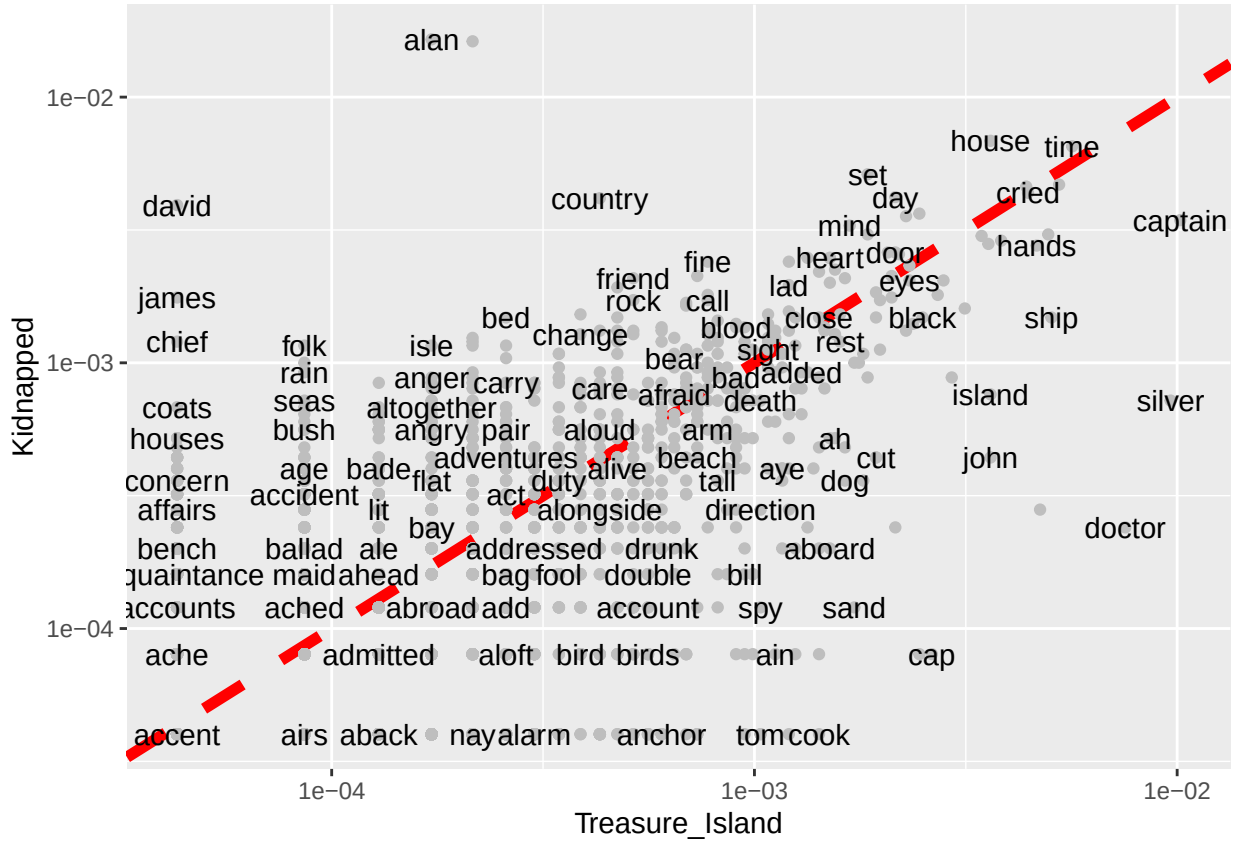
```
  group_by(title) %>%
  mutate(proportion = n / sum(n)) %>%
  select(-n) %>%
  pivot_wider(names_from = "title", values_from = "proportion")


frequency_t
```

```
## # A tibble: 8,748 x 3
##    word       Kidnapped Treasure_Island
##    <chr>          <dbl>           <dbl>
##  1 ab         0.0000400      NA
##  2 aback      0.0000400       0.000129
##  3 abandoned  0.0000400       0.0000862
##  4 abashed    0.000120       NA
##  5 abate      0.0000400      NA
##  6 abated     0.0000400      NA
##  7 abhorred   0.0000400      NA
##  8 abhorrence 0.0000400      NA
##  9 abiding    0.0000400      NA
## 10 ability    0.0000400      NA
## # ... with 8,738 more rows
```

Generate the ggplot for the similarity/dissimilarity on a data frame, but first remove the null values. Scale the results to reasonable values by performing a log transform on the proportions

```
ggplot(data = frequency_t %>% filter(!(Treasure_Island=="NA"|Kidnapped=="NA")), mapping = aes(x = Treasu
                      y = Kidnapped)) +
  geom_abline(color = "red", lty = 2,
              lwd=2) +
  geom_point(color="grey")+
  geom_text(aes(label = word),
            check_overlap = TRUE) +
  scale_x_log10() +
  scale_y_log10()
```

```
## Warning: Removed 1 rows containing missing values (geom_text).
```



Calculate and display the correlation

```
corr <- frequency_t %>% filter(!(Treasure_Island=="NA"|Kidnapped=="NA")) %>% select(-word) %>% cor()
corr
```

```
##                 Kidnapped Treasure_Island
## Kidnapped       1.0000000       0.4677002
## Treasure_Island 0.4677002       1.0000000
```

(ii) Find two words that appear with a high frequency in Kidnapped but not in Treasure Island. **We achieve this result by removing NA's from Kidnapped but not from Treasure Island In such a way, we are able to find popular words in Kidnapped that are non-existent in Treasure Island**

```
high_kidnapped_low_treasure_freq <-
  frequency_t %>%
  filter(Kidnapped != "NA") %>%
  arrange(desc(Kidnapped)) %>%
  filter(is.na(Treasure_Island)) %>%
  head(2) %>%
  select(word)

high_kidnapped_low_treasure_freq
```

```
## # A tibble: 2 x 1
```

```
##   word
##   <chr>
## 1 uncle
## 2 ay
```

(iii) Find two words that appear with a high frequency in Treasure Island but not in Kidnapped. **We apply the same logic as in (ii)

```
high_treasure_low_kidnapped_freq <-
  frequency_t %>%
  filter(Treasure_Island != "NA") %>%
  arrange(desc(Treasure_Island)) %>%
  filter(is.na(Kidnapped)) %>%
  head(2) %>%
  select(word)

high_treasure_low_kidnapped_freq
```

```
## # A tibble: 2 x 1
##   word
##   <chr>
## 1 i
## 2 you
```

(iv) Find two words that appear with high frequency in both novels.

**For each novel, perform the same steps as in (i) and (ii) without caring about the other column. Perform an inner join to find matching words

```
highest_freq_treasure_island <-
  frequency_t %>%
  filter(Treasure_Island != "NA") %>%
  arrange(desc(Treasure_Island)) %>%
  select(word)

highest_freq_kidnapped <-
  frequency_t %>%
  filter(Kidnapped != "NA") %>%
  arrange(desc(Kidnapped)) %>%
  select(word)

both_highest_freq <-
  inner_join(highest_freq_treasure_island, highest_freq_kidnapped) %>%
  head(2)
```

```
## Joining, by = "word"
```

```
both_highest_freq
```

```
## # A tibble: 2 x 1
##   word
##   <chr>
## 1 captain
## 2 silver
```

4.Find the 10 most common bigrams in Treasure Island that do not include stop words. **1.Call unnest_tokens() like before, but instead of using token as "words", change the parameter to ngrams, specifying n=2 for a bigram 2.Call filter() to remove the NA bigrams 3.Call separate() to split the bigram column for each word as a separate column 4.Remove the stop words from each column using filter() 5.Call combine to regenerate the bigram as a single column**

```
treasure_bigram <- treasure_island_t %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
  filter(bigram != "NA") %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  unite(bigram, word1, word2, sep=" ")

top_10_treasure_bigrams <- treasure_bigram %>%
  count(bigram, sort = TRUE)

top_10_treasure_bigrams
```

```
## # A tibble: 4,485 x 2
##    bigram               n
##    <chr>            <int>
##  1 dr livesey          38
##  2 ben gunn            31
##  3 captain smollett    29
##  4 spy glass           24
##  5 black dog           19
##  6 block house         17
##  7 admiral benbow      15
##  8 cried silver        15
##  9 john silver         15
## 10 log house           14
## # ... with 4,475 more rows
```

5.Plot the sentiment for the two books using the bing lexicon, using 100 words as the unit of length. **Call get_sentiments() specifying the bing lexicon**

```
bing_sentiments <- get_sentiments("bing")
```

Create tidy_books tibble by taking the tidy tibbles for each novel, using bind rows() to stack them into one tibble, and then generating the wordnumber Call ungroup() so that the data is restored to its ungrouped state

```
tidy_books <- bind_rows(tidy_treasure_island_t %>% select(-linenumber) %>%
                      mutate(title = "Treasure_Island"),
                    tidy_kidnapped_t %>% select(-linenumber) %>%
                      mutate(title = "Kidnapped")) %>%
  group_by(title) %>%
  mutate(wordnumber = row_number()) %>%
  ungroup()
```

Generate the stevenson_sentiment tibble by applying the following steps: 1.Perform an inner join between the books the books and the bing_sentiments 2.Create a new column containing an index for every 100 words 3.Count the combination of title, index, and sentiment 4.Use pivot_wider() to aid in the creation of a tibble calculating the net sentiment, Here if the result is negative, we assume a negative sentiment, 0 for neutral and positive for positive sentiment

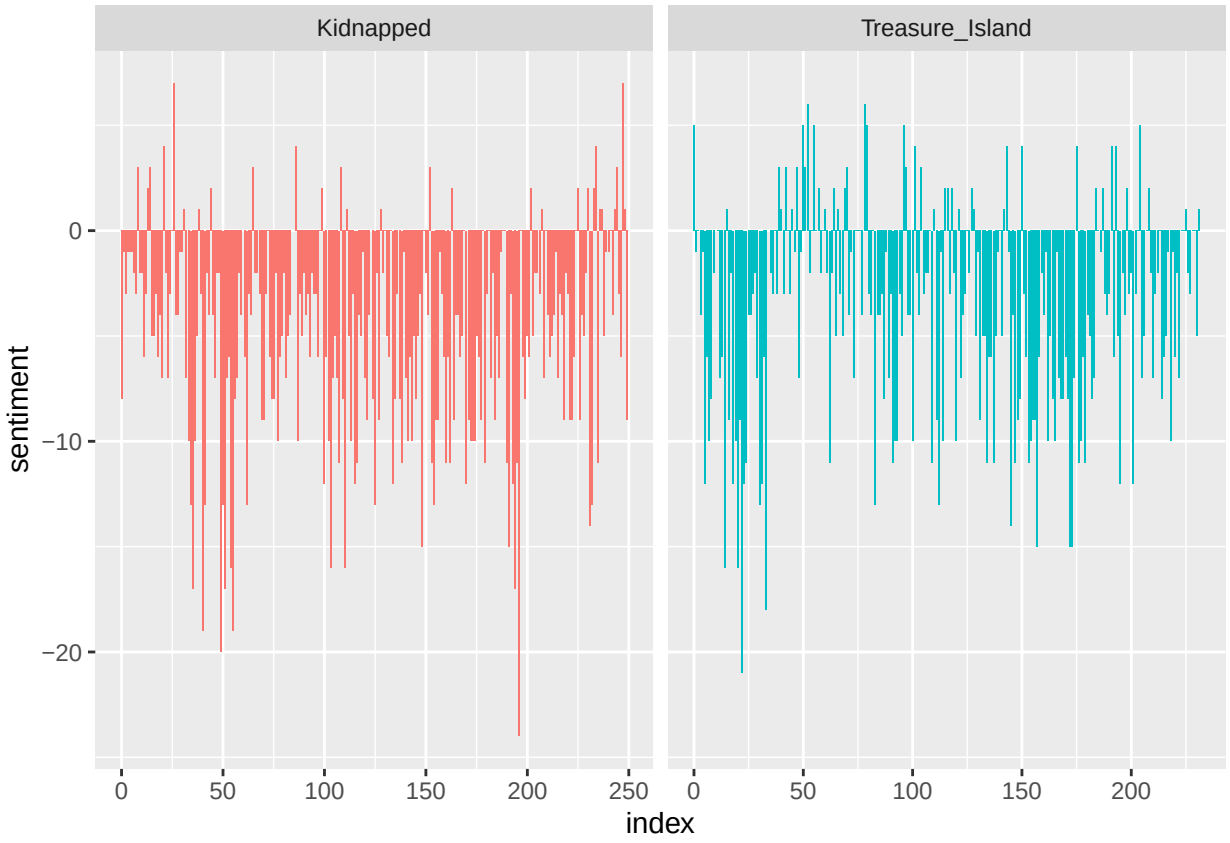
```
stevenson_sentiment <- tidy_books %>%
  inner_join(bing_sentiments) %>%
  mutate(index = wordnumber %/% 100) %>%
  count(title, index, sentiment) %>%
  pivot_wider(names_from = "sentiment", values_from = "n") %>%
  mutate(sentiment = positive - negative)
```

```
## Joining, by = "word"
```

Generate the ggplot() using a facet_wrap to display the sentiment for both novels

```
ggplot(data = stevenson_sentiment,
       mapping = aes(x = index, y = sentiment, fill = title)) +
  geom_bar(stat="identity") +
  facet_wrap(~title, ncol = 2, scales = "free_x") +
  theme(legend.position = "none")
```

```
## Warning: Removed 7 rows containing missing values (position_stack).
```



PROBLEM 2:

For the AssociatedPress dataset provided by the topicmodels package, create a three-topic LDA model using the "Gibbs" method instead of the default VEM method. List the top 10 terms in each of the three topics in the fitted model, and suggest what these topics might be. **Fix the data**

```
data("AssociatedPress", package = "topicmodels")
```

Train an LDA model with the Gibbs model and 3 topics. Specify the seed for reproducible results

```
ap_lda <- LDA(AssociatedPress, k = 3, method = "Gibbs",
              control = list(seed = 1234))
```

Apply the tidy() function from broom to the results to extract the topics, terms and scores from the model

```
ap_topics <- broom::tidy(ap_lda, matrix = "beta")

ap_topics
```

```
## # A tibble: 31,419 x 3
##    topic term              beta
##    <int> <chr>            <dbl>
##  1     1 aaron      0.000000748
##  2     2 aaron      0.0000594
##  3     3 aaron      0.00000723
##  4     1 abandon    0.000000748
##  5     2 abandon    0.000000653
##  6     3 abandon    0.0000993
##  7     1 abandoned  0.0000232
##  8     2 abandoned  0.000242
##  9     3 abandoned  0.000000657
## 10     1 abandoning 0.000000748
## # ... with 31,409 more rows
```

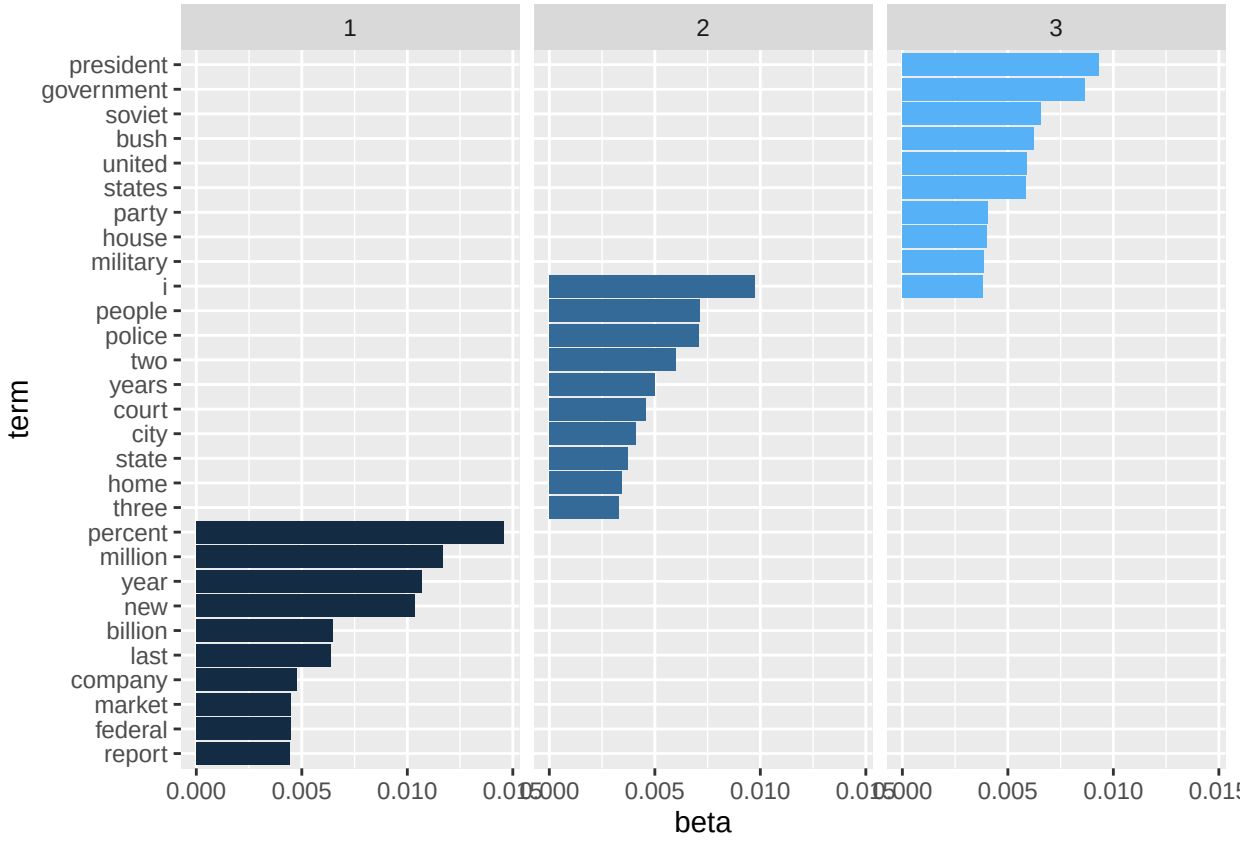
For each topic, perform the following steps: 1.Fitler ap_topics for the specific topic number(1,2 or 3) 2.Create a column for beta_rank and compute it by order the beta in descending order and then using the window function min_rank() 3.Filter for only bet ranks under 10 4.Order the frame by the bet_rank) 5.Create a column term that is a reorder based on the term,beta tuple 6.Save only the first 10 rows using head(10) Finally, combine the 3 topics into one tibble using bind_rows()

```
topic1 <- ap_topics %>% filter(topic==1) %>%
  mutate(beta_rank = min_rank(desc(beta))) %>%
  filter(beta_rank <= 10) %>%
  arrange(beta_rank) %>%
  mutate(term = reorder(term, beta)) %>%
  head(10)

topic2 <- ap_topics %>% filter(topic==2) %>%
  mutate(beta_rank = min_rank(desc(beta))) %>%
```

```
  filter(beta_rank <= 10) %>%
  arrange(beta_rank) %>%
  mutate(term = reorder(term, beta)) %>%
  head(10)

topic3 <- ap_topics %>% filter(topic==3) %>%
  mutate(beta_rank = min_rank(desc(beta))) %>%
  filter(beta_rank <= 10) %>%
  arrange(beta_rank) %>%
  mutate(term = reorder(term, beta)) %>%
  head(10)

topics_t <- rbind(topic1, topic2, topic3)
```

Generate the ggplot

```
ggplot(data = topics_t,
       mapping = aes(x = beta, y = term, fill = topic)) +
  geom_bar(stat="identity") +
  facet_wrap(~topic, ncol = 3) +
  theme(legend.position = "none")
```



Discussion: I believe that topic 1 represents finance/stocks, topic 2 represents a criminal trial/sentencing, and topic 3 represents presidential elections