Assignment 6

Feiyu Zheng (fz114)

2022/3/20

Problem 1

1. Find the gutenberg IDs of Treasure Island and Kidnapped by Robert Louis Stevenson using the gutenberg_metadata data frame avaiable in the gutenberg package.

Import the gutenbergr package

```
# install.packages("gutenbergr") # install gutenbergr package

library(gutenbergr) # import gutenbergr package
```

Find the gutenberg IDs of two books

```
gutenberg_metadata %>%
  filter((title == "Treasure Island" | title == "Kidnapped")
         & author == "Stevenson, Robert Louis")
```

```
## # A tibble: 4 x 8
##   gutenberg_id title  author  gutenberg_autho~ language gutenberg_booksh~ rights
##          <int> <chr>  <chr>              <int> <chr>    <chr>             <chr>
## 1          120 Treas~ Steven~               35 en       Best Books Ever ~ Publi~
## 2          421 Kidna~ Steven~               35 en       Best Books Ever ~ Publi~
## 3        23936 Treas~ Steven~               35 en       <NA>              Publi~
## 4        27780 Treas~ Steven~               35 en       <NA>              Publi~
## # ... with 1 more variable: has_text <lgl>
```

2. Download the texts of these two books from the gutenberg package.

Download Treasure Island text

```
treasureIslandText <- gutenberg_download(120, mirror = "http://mirrors.xmission.com/gutenberg/")
treasureIslandText
```

```
## # A tibble: 7,491 x 2
##    gutenberg_id text
##           <int> <chr>
##  1          120 "TREASURE ISLAND"
##  2          120 ""
##  3          120 "by Robert Louis Stevenson"
##  4          120 ""
##  5          120 ""
## 6           120 ""
## 7           120 ""
## 8           120 "TREASURE ISLAND"
## 9           120 ""
## 10          120 "To S.L.O., an American gentleman in accordance with whose clas~
## # ... with 7,481 more rows
```

Download Kidnapped Text

```
kidnappedText <- gutenberg_download(421, mirror = "http://mirrors.xmission.com/gutenberg/")
kidnappedText
```

```
## # A tibble: 8,418 x 2
##    gutenberg_id text
##           <int> <chr>
##  1          421 "     KIDNAPPED"
##  2          421 "     BEING"
##  3          421 "     MEMOIRS OF THE ADVENTURES OF"
##  4          421 "     DAVID BALFOUR"
##  5          421 "     IN THE YEAR 1751"
##  6          421 ""
##  7          421 ""
##  8          421 ""
##  9          421 "     HOW HE WAS KIDNAPPED AND CAST AWAY; HIS SUFFERINGS IN"
## 10          421 "     A DESERT ISLE; HIS JOURNEY IN THE WILD HIGHLANDS;"
## # ... with 8,408 more rows
```

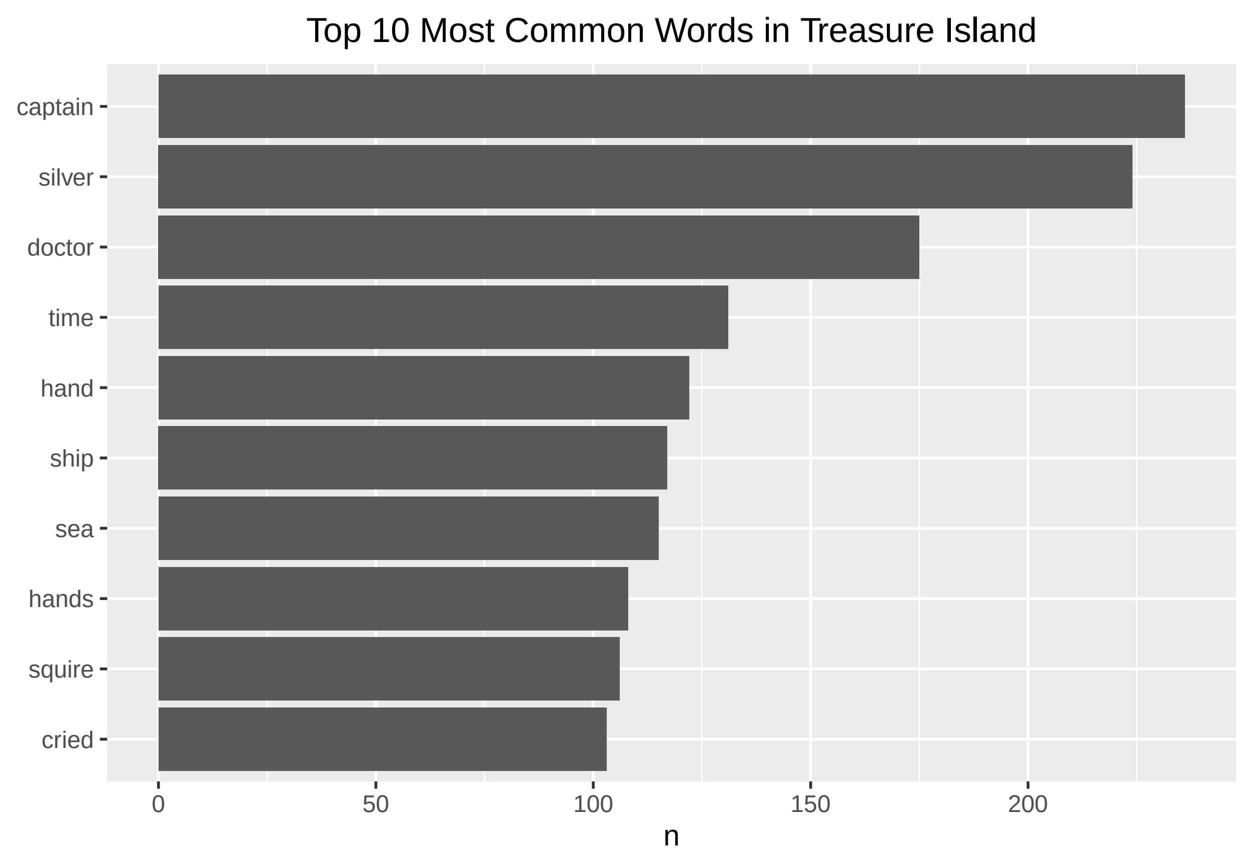
3. Find the 10 most common words (that are not stop words) in each novel.

Treasure Island top 10 most common words

```
tidyTreasureIslandWords <- treasureIslandText %>%
  unnest_tokens(word, text) %>% # convert line to word
  anti_join(stop_words) %>% # exclude stop words
  mutate(word = str_extract(word, "[a-z]+")) %>% # exclude non-letter words
  anti_join(stop_words) %>% # exclude stop words again
  filter(!is.na(word)) # filter out empty value
```

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

```
top10TreasureIslandWords <- tidyTreasureIslandWords %>%
  count(word, sort = TRUE) %>% # count word and sort in descending
  head(10) # show top 10

# visualization
top10TreasureIslandWords %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = NULL, title = "Top 10 Most Common Words in Treasure Island") +
  theme(plot.title = element_text(hjust = 0.5))
```

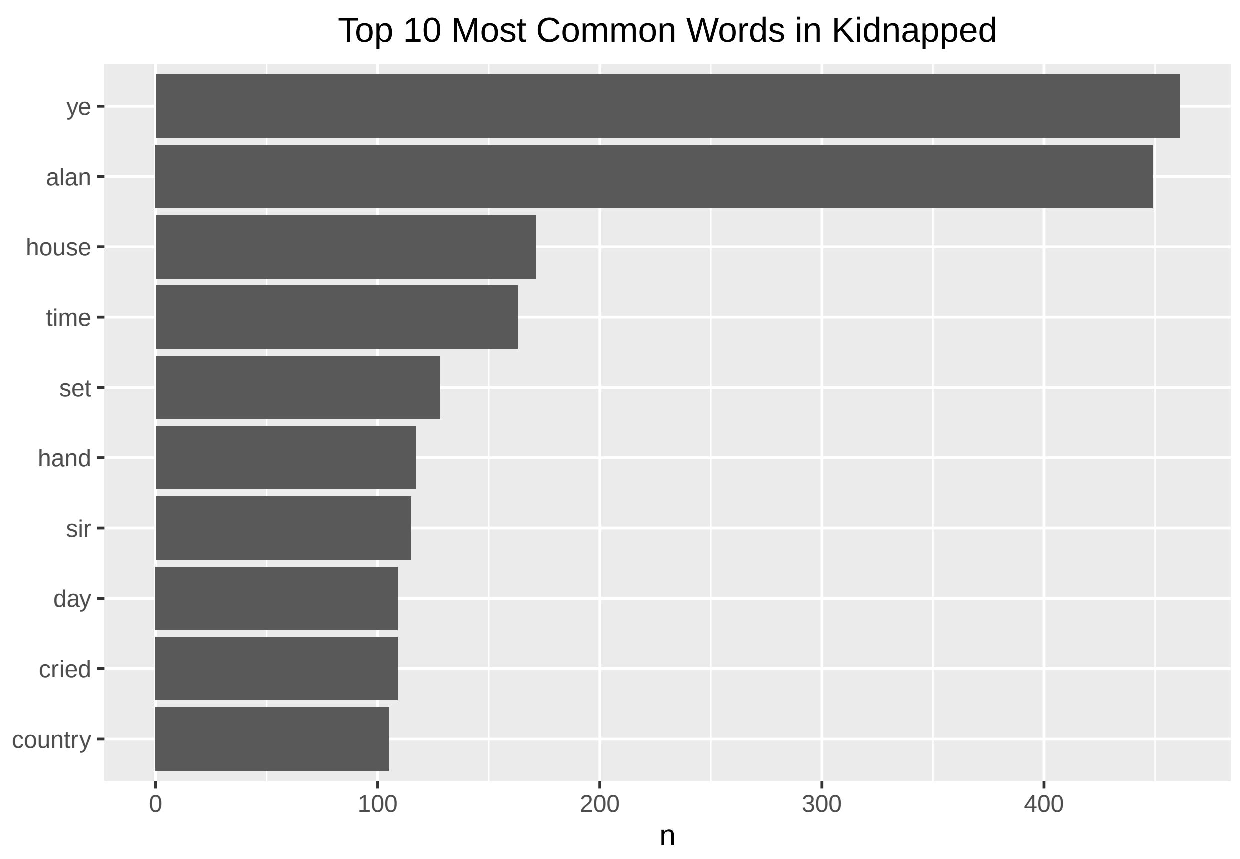


Kidnapped top 10 most common words

```
tidyKidnappedWords <- kidnappedText %>%
  unnest_tokens(word, text) %>% # convert line to word
  anti_join(stop_words) %>% # exclude stop words
  mutate(word = str_extract(word, "[a-z]+")) %>% # exclude non-letter words
  anti_join(stop_words) %>% # exclude stop words again
  filter(!is.na(word)) # filter out empty value
```

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

```
top10KidnappedWords <- tidyKidnappedWords %>%
  count(word, sort = TRUE) %>% # count word and sort in descending order
  head(10) # show top 10

# visualization
top10KidnappedWords %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = NULL, title = "Top 10 Most Common Words in Kidnapped") +
  theme(plot.title = element_text(hjust = 0.5))
```



4.

(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.

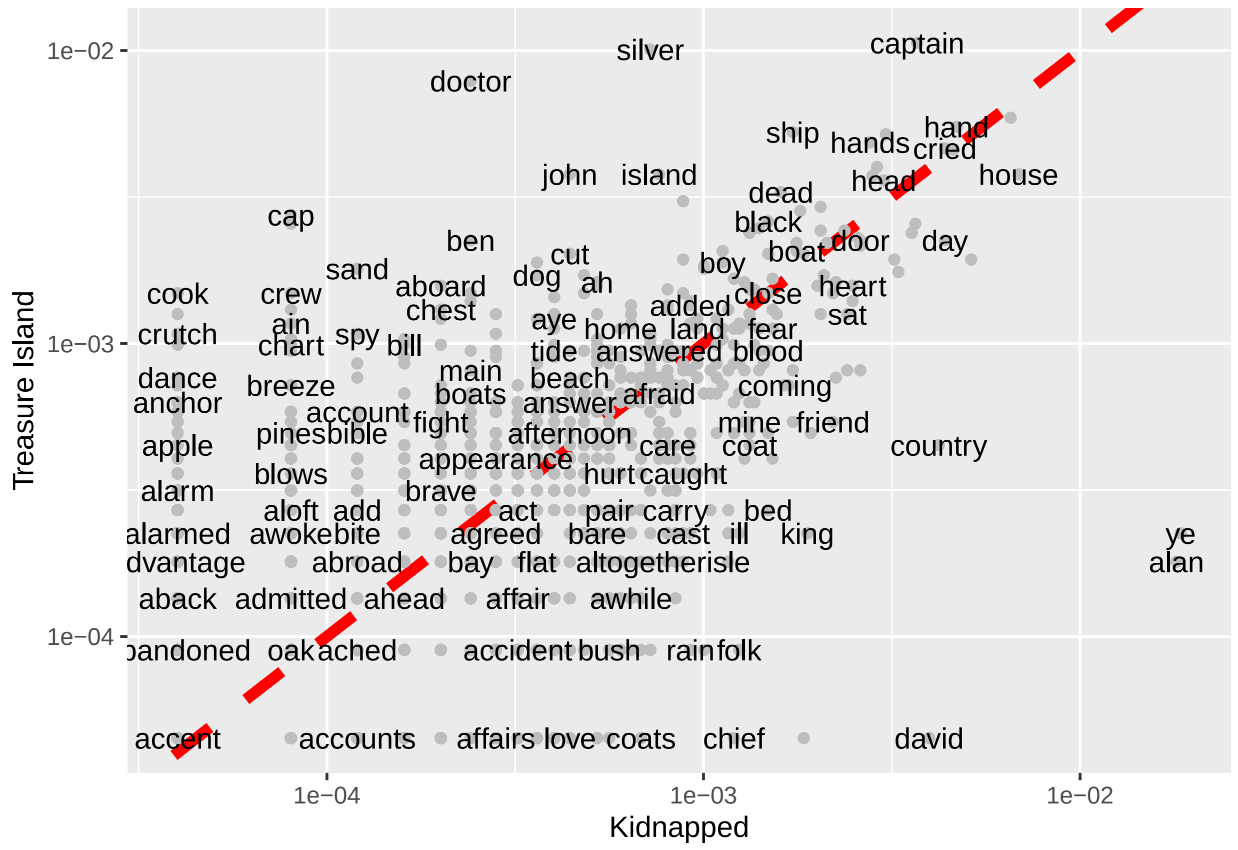
```
frequency <- bind_rows(
  mutate(tidyTreasureIslandWords, title = "Treasure Island"),
  mutate(tidyKidnappedWords, title = "Kidnapped")) %>%
  mutate(word = str_extract(word, "[a-z]+")) %>%
  count(title, word) %>%
  group_by(title) %>%
  mutate(proportion = n / sum(n)) %>%
  select(-n) %>%
  pivot_wider(names_from = "title", values_from = "proportion")
frequency
```

Visualization

```
## # A tibble: 8,566 x 3
##    word        Kidnapped `Treasure Island`
##    <chr>           <dbl>             <dbl>
##  1 ab          0.0000401        NA
##  2 aback       0.0000401         0.000135
##  3 abandoned   0.0000401         0.0000900
##  4 abashed     0.000120         NA
##   5 abate       0.0000401        NA
##   6 abated      0.0000401        NA
##   7 abhorred    0.0000401        NA
##   8 abhorrence  0.0000401        NA
##   9 abiding     0.0000401        NA
##  10 ability     0.0000401        NA
## # ... with 8,556 more rows
```

```
frequency %>%
  ggplot(aes(x = `Kidnapped`, y = `Treasure Island`)) +
  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 5819 rows containing missing values (geom_point).

## Warning: Removed 5819 rows containing missing values (geom_text).
```



```
frequency %>%
  filter(!(`Kidnapped` == "NA" | `Treasure Island` == "NA")) %>%
  select(, 2:3) %>%
  cor()
```

Correlation

```
##                 Kidnapped Treasure Island
## Kidnapped       1.0000000       0.4510147
## Treasure Island 0.4510147       1.0000000
```

(ii) Find two words that appear with a high frequency in Kidnapped but not in Treasure Island.

Based on the above visualization, we can find that “ye” and “alan” have a high frequency in Kidnapped but not in Treasure Island.

(iii) Find two words that appear with a high frequency in Treasure Island but not in Kidnapped.

Based on the above visualization, we can find that “doctor” and “cap” have a high frequency in Treasure Island but not in Kidnapped.

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

Based on the above visualization, we can find that “time” and “cried” have a high frequency in both novels.

5. Find the 10 most common bigrams in Treasure Island that do not include stop words.

```
top10TreasureIslandBigrams <- treasureIslandText %>%
  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 = " ") %>%
  count(bigram, sort = TRUE) %>%
  head(10)
top10TreasureIslandBigrams
```

```
## # A tibble: 10 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
```

6. Plot the sentiment for the two books using the bing lexicon, using 100 lines as the unit of length.

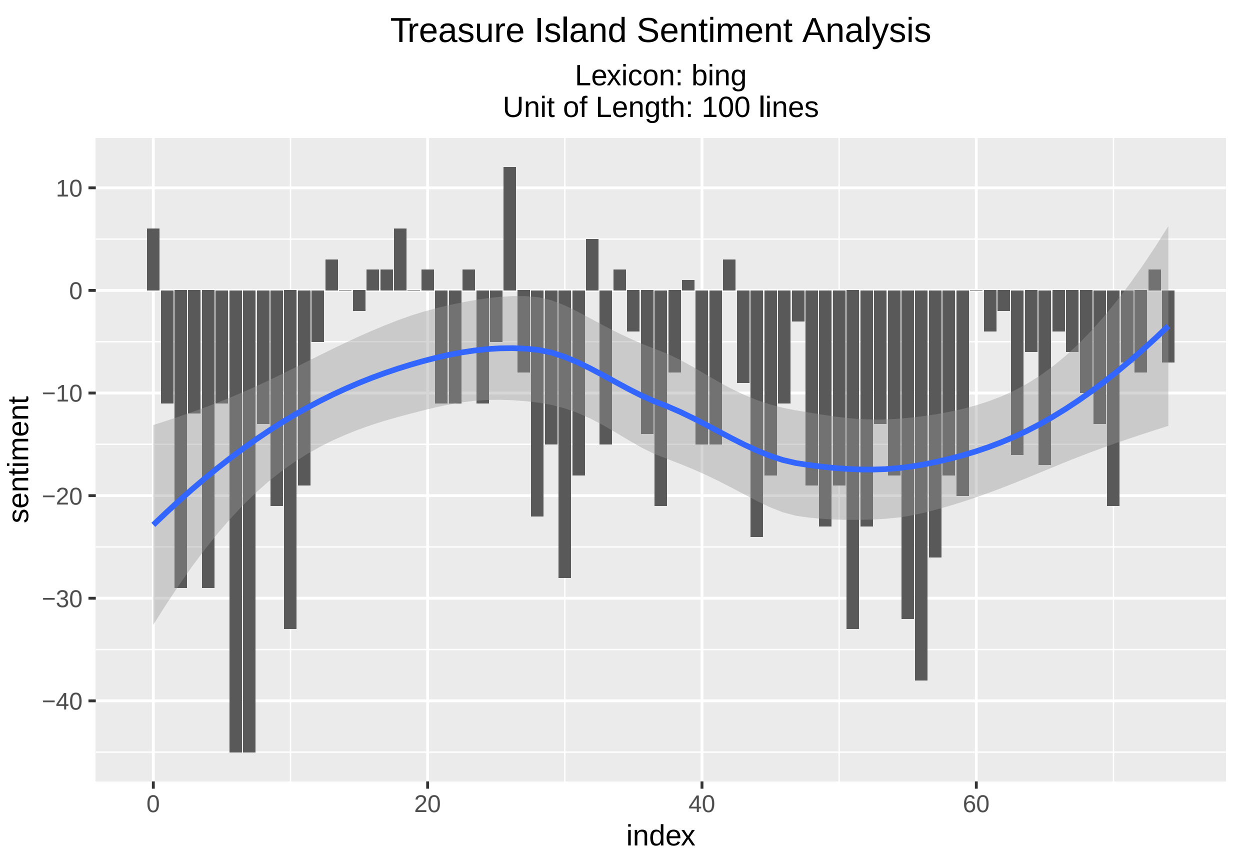
Define the lexicon and the unit of length

```
unitLength <- 100
bingLexicon <- get_sentiments("bing")
```

Plot the sentiment for Treasure Island

```
treasureIslandText %>%
  mutate(
    linenumber = row_number(),
    part = cumsum(
      str_detect(
        text,
        regex("^PART \\w+--", ignore_case = TRUE))),
    chapter = cumsum(
      str_detect(
        text,
        regex("^\\d+$")))) %>%
  unnest_tokens(word, text) %>%
  mutate(index = linenumber %/% unitLength) %>%
  anti_join(stop_words) %>%
  inner_join(bingLexicon) %>%
  count(index, sentiment) %>%
  pivot_wider(names_from = "sentiment", values_from = "n") %>%
  mutate(sentiment = positive - negative) %>%
  ggplot(aes(index, sentiment)) +
  geom_bar(stat = "identity") +
  geom_smooth() +
  labs(title = "Treasure Island Sentiment Analysis", subtitle = "Lexicon: bing\nUnit of Length: 100 lin
  theme(
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5))
```

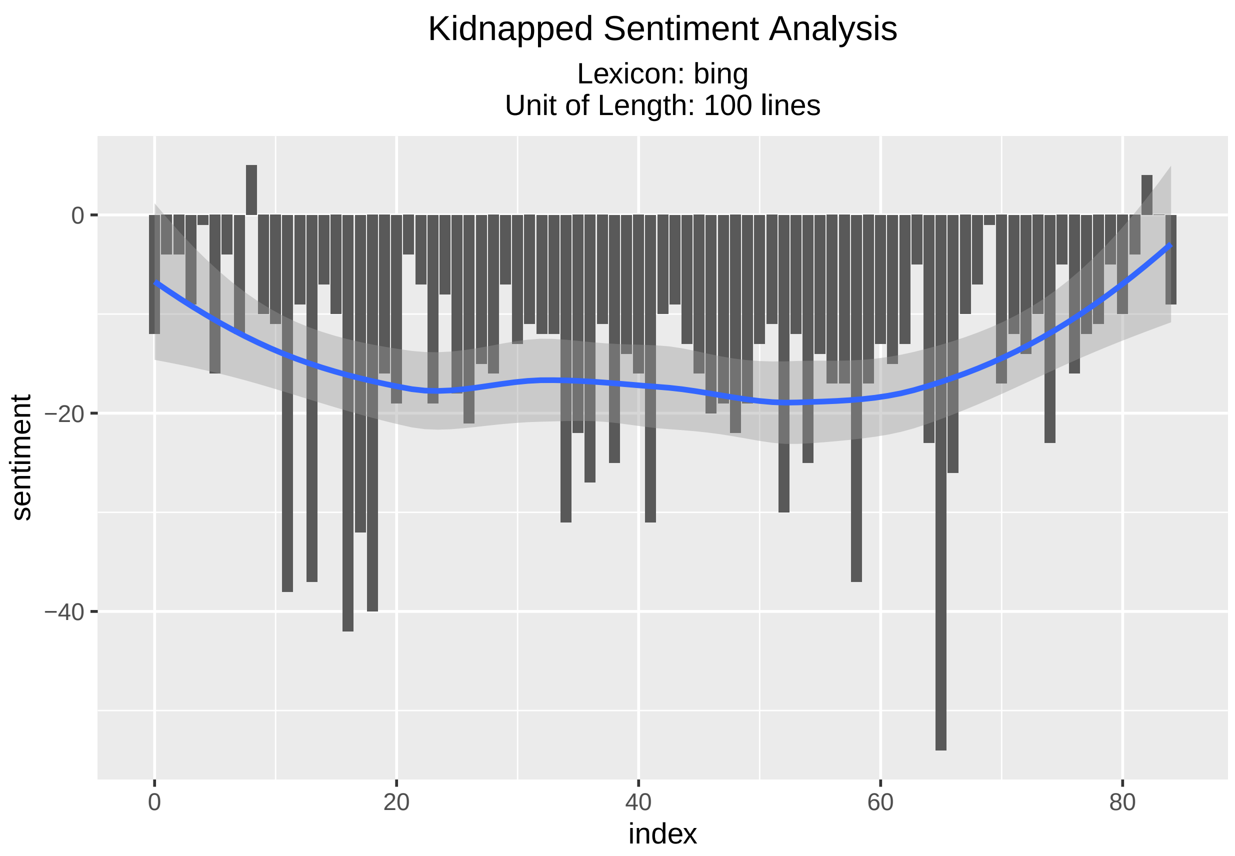
```
## Joining, by = "word"
## Joining, by = "word"

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Plot the sentiment for Kidnapped

```
kidnappedText %>%
  mutate(
    linenumber = row_number(),
    chapter = cumsum(
      str_detect(
        text,
        regex("^CHAPTER [\\divxlc]", ignore_case = TRUE)))) %>%
  unnest_tokens(word, text) %>%
  mutate(index = linenumber %/% unitLength) %>%
  anti_join(stop_words) %>%
  inner_join(bingLexicon) %>%
  count(index, sentiment) %>%
  pivot_wider(names_from = "sentiment", values_from = "n") %>%
  mutate(sentiment = positive - negative) %>%
  ggplot(aes(index, sentiment)) +
  geom_bar(stat = "identity") +
  geom_smooth() +
  labs(title = "Kidnapped Sentiment Analysis", subtitle = "Lexicon: bing\nUnit of Length: 100 lines") +
  theme(
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5))
```

```
## Joining, by = "word"
## Joining, by = "word"
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Problem 2

1. For the AssociatedPress dataset provided by the topicmodels packages, 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.

Install and import tm and topicmodels packages

```
# install tm and topicmodels packages
# install.packages("tm")
# install.packages("topicmodels")

# import tm and topicmodels packages
library(tm)
library(topicmodels)
```

Import the AssociatedPress dataset from topicmodels package

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

```
## <<DocumentTermMatrix (documents: 2246, terms: 10473)>>
## Non-/sparse entries: 302031/23220327
## Sparsity           : 99%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

Fit the dataset with three-topic LDA model using the Gibbs method

```
ap_lda <- LDA(AssociatedPress, k = 3, method = "Gibbs")
ap_lda
```

```
## A LDA_Gibbs topic model with 3 topics.
```

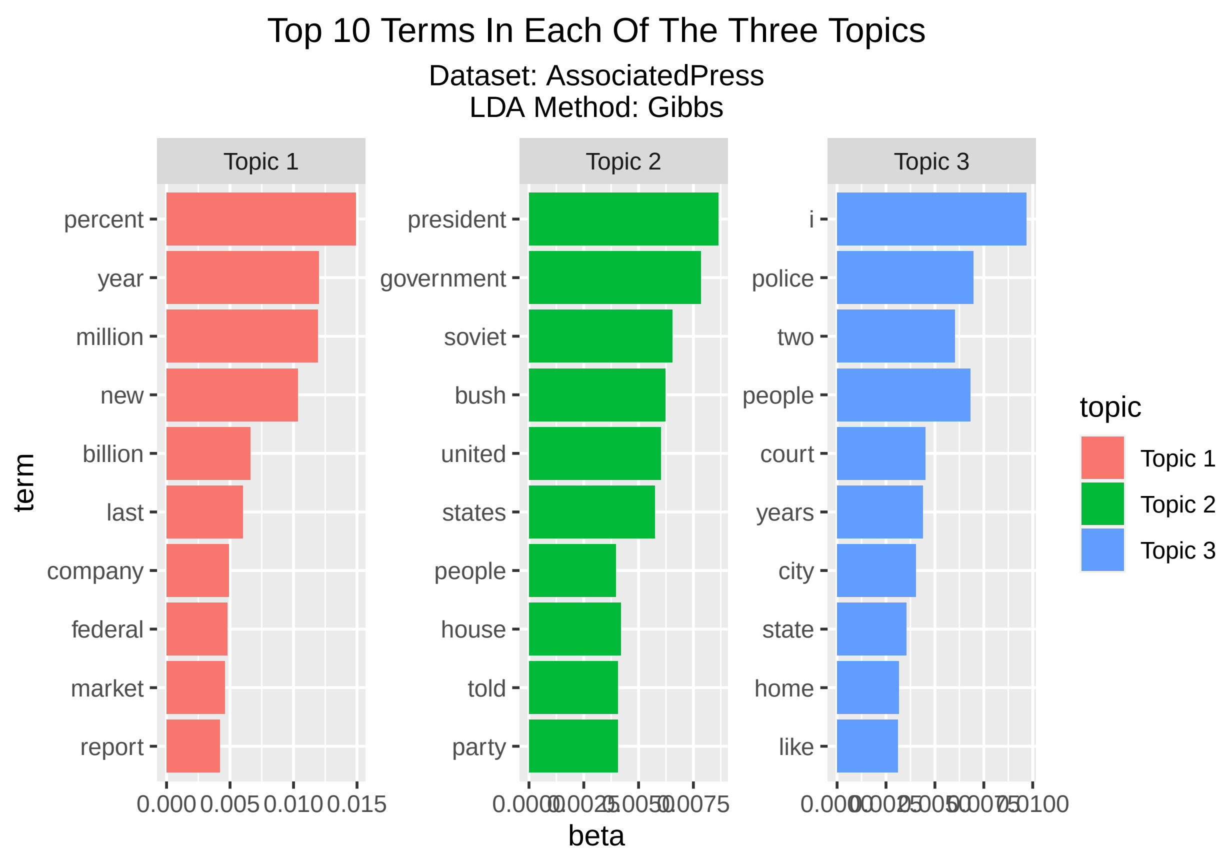
Tidy the LDA output

```
ap_topics <- tidy(ap_lda)
ap_topics
```

```
## # A tibble: 31,419 x 3
##    topic term              beta
##    <int> <chr>            <dbl>
##  1     1 aaron      0.000000765
##  2     2 aaron      0.0000661
##  3     3 aaron      0.000000643
##  4     1 abandon    0.0000543
##  5     2 abandon    0.0000530
##  6     3 abandon    0.000000643
##  7     1 abandoned  0.00000842
##  8     2 abandoned  0.000000655
##  9     3 abandoned  0.000251
## 10     1 abandoning 0.000000765
## # ... with 31,409 more rows
```

List the top 10 terms in each of the three topics in the fitted model

```
ap_topics %>%
  group_by(topic) %>%
  mutate(beta_rank = min_rank(desc(beta))) %>%
  filter(beta_rank <= 10) %>%
  ungroup() %>%
  arrange(beta_rank) %>%
  mutate(topic = recode(topic, "1" = "Topic 1", "2" = "Topic 2", "3" = "Topic 3")) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(beta, term, fill = topic)) +
  geom_col(position = "dodge") +
  facet_wrap(~topic, scales = "free") +
  labs(
    title = "Top 10 Terms In Each Of The Three Topics",
    subtitle = "Dataset: AssociatedPress\nLDA Method: Gibbs") +
  theme(
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5))
```



The topic with terms like “percent”, “year”, “million”, “company”, etc., might be about economy.

The topic with terms like “i”, “people”, “police”, “court”, “city”, etc., might be about law or lawsuit.

The topic with terms like “president”, “government”, “soviet”, “military”, etc., might be about politics or military.

(Please note the topic number might not be cohere with the terms due to the fitting process so I am using terms to represent the topic instead of the topic number)

2. Find the documents (by numbers) for which there is maximum uncertainty about the classification, with the maximum probability of being assigned to a group not exceeding 0.35.

```
ap_lda %>%
  tidy(matrix = "gamma") %>%
  group_by(document) %>%
  slice_max(gamma, n = 1,
            with_ties = FALSE) %>% # select one row with the largest gamma
  ungroup() %>%
  filter(gamma <= 0.35) # find those with the largest gamma less than or equal to 0.35
```

```
## # A tibble: 13 x 3
##    document topic gamma
##       <int> <int> <dbl>
##  1       23     1 0.345
##  2      217     2 0.347
## 3        326      1 0.339
## 4        339      2 0.339
## 5        444      2 0.345
## 6        514      2 0.344
## 7        663      1 0.348
## 8        709      1 0.340
## 9       1094      1 0.345
## 10      1661      2 0.339
## 11      1973      3 0.349
## 12      2080      3 0.343
## 13      2213      1 0.333
```

The documents listed above are those for which there is maximum uncertainty about the classification.