Lab 7

Ashwin Dev

Math 241, Week 9

```
# Put all necessary libraries here
library(tidyverse)
library(tidytext)
library(wordcloud)

# Ensure the textdata package is installed
if (!requireNamespace("textdata", quietly = TRUE)) {
  install.packages("textdata")
}
# Load the textdata package
library(textdata)

# Before knitting your document one last time, you will have to download the AFINN lexicon explicitly
lexicon_afinn()
```

```
## # A tibble: 2,477 x 2
##    word       value
##    <chr>      <dbl>
##  1 abandon       -2
##  2 abandoned     -2
##  3 abandons      -2
##  4 abducted      -2
##  5 abduction     -2
##  6 abductions    -2
##  7 abhor         -3
##  8 abhorred      -3
##  9 abhorrent     -3
## 10 abhors        -3
## # i 2,467 more rows
```

```
lexicon_nrc()
```

```
## # A tibble: 13,872 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 abacus      trust
##  2 abandon     fear
##  3 abandon     negative
##  4 abandon     sadness
##  5 abandoned   anger
##  6 abandoned   fear
```

```
##  7 abandoned   negative
##  8 abandoned   sadness
##  9 abandonment anger
## 10 abandonment fear
## # i 13,862 more rows
```

Due: Friday, March 29th at 5:30pm

Goals of this lab

1. Practice matching patterns with regular expressions.
2. Practice manipulating strings with `stringr`.
3. Practice tokenizing text with `tidytext`.
4. Practice looking at word frequencies.
5. Practice conducting sentiment analysis.

Problem 1: What's in a Name? (You'd Be Surprised!)

1. Load the `babynames` dataset, which contains yearly information on the frequency of baby names by sex and is provided by the US Social Security Administration. It includes all names with at least 5 uses per year per sex. In this problem, we are going to practice pattern matching!

```
library(babynames)
data("babynames")
#?babynames
```

a. For 2000, find the ten most popular female baby names that start with the letter Z.

```
FZ <- babynames %>%
  filter(year == 2000, sex == 'F', grepl("^Z", name)) %>%
  top_n(10, n) %>%
  arrange(desc(n)) %>%
  select(name, n)

FZ
```

```
## # A tibble: 10 x 2
##    name       n
##    <chr>  <int>
##  1 Zoe     3785
##  2 Zoey     691
##  3 Zaria    568
##  4 Zoie     320
##  5 Zariah   168
##  6 Zion     156
##  7 Zainab   142
##  8 Zara     121
##  9 Zahra    113
## 10 Zaira    103
```

b. For 2000, find the ten most popular female baby names that contain the letter z.

```
z10 <- babynames %>%
  filter(year == 2000, sex == 'F', grepl("z", name, ignore.case = TRUE)) %>%
  arrange(desc(n)) %>%
  slice_head(n = 10) %>%
  select(name, n)

z10
```

```
## # A tibble: 10 x 2
##    name          n
##    <chr>     <int>
##  1 Elizabeth 15094
##  2 Mackenzie  6348
##  3 Zoe        3785
##  4 Mckenzie   2526
##  5 Makenzie   1613
##  6 Jazmin     1391
##  7 Jazmine    1353
##  8 Lizbeth     817
##  9 Eliza       759
## 10 Litzy       722
```

c. For 2000, find the ten most popular female baby names that end in the letter z.

```
endZ<- babynames %>%
  filter(year == 2000, sex == 'F', grepl("z$", name, ignore.case = TRUE)) %>%
  arrange(desc(n)) %>%
  slice_head(n = 10) %>%
  select(name, n)

endZ
```

```
## # A tibble: 10 x 2
##    name         n
##    <chr>    <int>
##  1 Luz        489
##  2 Beatriz    357
##  3 Mercedez   141
##  4 Maricruz    96
##  5 Liz         72
##  6 Inez        69
```

```
##  7 Odaliz      24
##  8 Marycruz    23
##  9 Cruz        19
## 10 Deniz       16
```

---

d. Between your three tables in 1.a - 1.c, do any of the names show up on more than one list? If so, which ones? (Yes, I know you could do this visually but use some joins!)

```
# joins FZ and z10
common_FZ_z10 <- inner_join(FZ, z10, by = "name")

# joins FZ and endZ
common_FZ_endZ <- inner_join(FZ, endZ, by = "name")

# Join z10 and endZ
common_z10_endZ <- inner_join(z10, endZ, by = "name")

# combines
all_common_names <- bind_rows(common_FZ_z10, common_FZ_endZ, common_z10_endZ) %>%
  distinct(name)

all_common_names
```

```
## # A tibble: 1 x 1
##   name
##   <chr>
## 1 Zoe
```

---

e. Verify that none of the baby names contain a numeric (0-9) in them.

```
names_with_digits <- babynames %>%
  filter(year == 2000, grepl("[0-9]", name)) %>%
  select(year, sex, name, n)

if(nrow(names_with_digits) > 0) {
  print("names contain digits:")
  print(names_with_digits)
} else {
  print("No names contain digits")
}
```

```
## [1] "No names contain digits"
```

---

f. While none of the names contain 0-9, that doesn't mean they don't contain "one", "two", . . . , or "nine". Create a table that provides the number of times a baby's name contained the word "zero", the word "one", . . . the word "nine".

Notes:

- I recommend first converting all the names to lower case.
- If none of the baby's names contain the written number, there you can leave the number out of the table.
- Use `str_extract()`, not `str_extract_all()`. (We will ignore names where more than one of the words exists.)

Hint: You will have two steps that require pattern matching: 1. Subset your table to only include the rows with the desired words. 2. Add a column that contains the desired word.

```
names_with_numbers <- babynames %>%
  mutate(name = tolower(name),
         number_word = str_extract(name, "zero|one|two|three|four|five|six|seven|eight|nine")) %>%
  filter(!is.na(number_word)) %>%
  group_by(number_word) %>%
  summarise(count = n())

names_with_numbers
```

```
## # A tibble: 9 x 2
##   number_word count
##   <chr>       <int>
## 1 eight         356
## 2 four            2
## 3 nine          807
## 4 one         10142
## 5 seven          50
## 6 six           106
## 7 three          58
## 8 two           288
## 9 zero            4
```

---

g. Which written number or numbers don't show up in any of the baby names?

It looks like there isn't any written number that doesn't show up in any of the baby names at least once.

---

h. Create a table that contains the names and their frequencies for the two least common written numbers.

```
# identifies names with number-words and their frequencies
names_with_numbers <- babynames %>%
  mutate(name_lower = tolower(name),
         number_word = str_extract(name_lower, "zero|one|two|three|four|five|six|seven|eight|nine")) %>%
  filter(!is.na(number_word)) %>%
  select(name, number_word, n) %>%
  group_by(number_word) %>%
  summarise(total_frequency = sum(n))

# identifies the two least common number-words
least_common_numbers <- names_with_numbers %>%
  arrange(total_frequency) %>%
  slice_head(n = 2)

least_common_numbers
```

```
## # A tibble: 2 x 2
##   number_word total_frequency
##   <chr>                 <int>
## 1 four                     10
## 2 zero                     24
```

```
# names and frequencies for these two least common numbers
names_for_least_common <- babynames %>%
  mutate(name_lower = tolower(name),
         number_word = str_extract(name_lower, paste(least_common_numbers$number_word, collapse="|"))) %
  filter(!is.na(number_word)) %>%
  select(name, number_word, n)

names_for_least_common
```

```
## # A tibble: 6 x 3
##   name    number_word     n
##   <chr>   <chr>       <int>
## 1 Balfour four            5
## 2 Balfour four            5
## 3 Luzero  zero            6
## 4 Luzero  zero            5
## 5 Zeron   zero            6
## 6 Zero    zero            7
```

---

i. List out the names that contain no vowels (consider "y" to be a vowel).

```
names_without_vowels <- babynames %>%
  mutate(name_lower = tolower(name)) %>%
  filter(!grepl("[aeiouy]", name_lower)) %>%
  select(name, n)

names_without_vowels
```

```
## # A tibble: 1,260 x 2
##    name      n
##    <chr> <int>
##  1 Wm       14
##  2 Wm       15
##  3 Wm       18
##  4 Wm       16
##  5 Wm       22
##  6 Wm       23
##  7 Ng        5
##  8 Wm       15
##  9 Wm       15
## 10 Wm       25
## # i 1,250 more rows
```

Problem 2: Tidying the "Call of the Wild"

Did you read "Call of the Wild" by Jack London? If not, read the first paragraph of its wiki page for a quick summary and then let's do some text analysis on this classic! The following code will pull the book into R using the `gutenbergr` package.

```
library(gutenbergr)
wild <- gutenberg_download(215)
```

a. Create a tidy text dataset where you tokenize by words.

```
wild <- gutenberg_download(215)

tidy_wild <- wild %>%
  unnest_tokens(word, text)

head(tidy_wild)
```

```
## # A tibble: 6 x 2
##   gutenberg_id word
##          <int> <chr>
## 1          215 cover
## 2          215 the
## 3          215 call
## 4          215 of
## 5          215 the
## 6          215 wild
```

b. Find the frequency of the 20 most common words. First, remove stop words.

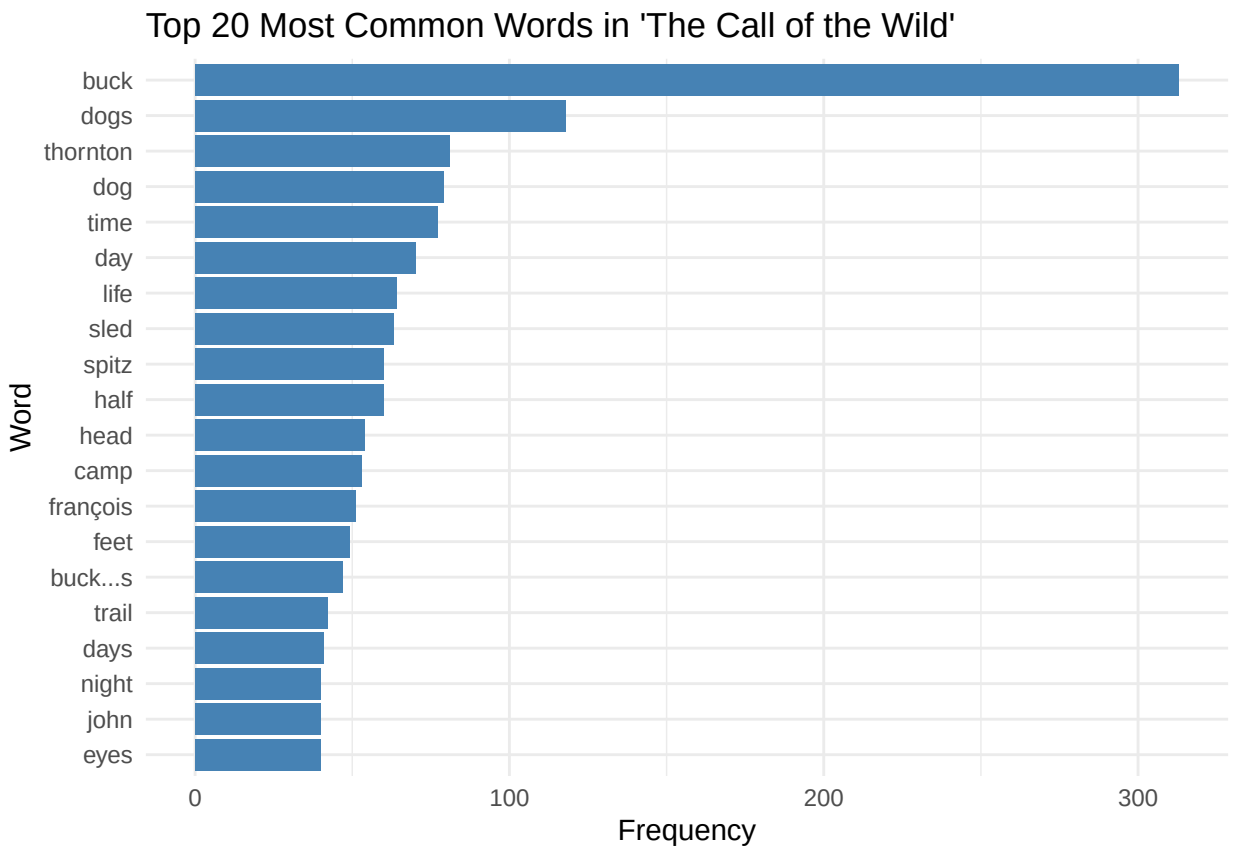
```
data("stop_words")

# removes stop words
filtered_wild <- tidy_wild %>%
  anti_join(stop_words, by = "word")

word_frequencies <- filtered_wild %>%
  count(word, sort = TRUE) %>%
  top_n(20, n)

word_frequencies
```

```
## # A tibble: 20 x 2
##    word         n
##    <chr>    <int>
##  1 buck       313
##  2 dogs       118
##  3 thornton    81
##  4 dog         79
##  5 time        77
##  6 day         70
##  7 life        64
##  8 sled        63
##  9 half        60
## 10 spitz       60
## 11 head        54
## 12 camp        53
## 13 françois    51
## 14 feet        49
## 15 buck's      47
## 16 trail       42
## 17 days        41
## 18 eyes        40
## 19 john        40
## 20 night       40
```

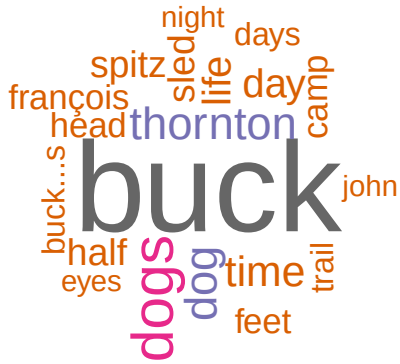
c. Create a bar graph and a word cloud of the frequencies of the 20 most common words.

```
ggplot(word_frequencies, aes(x = reorder(word, n), y = n)) +
  geom_col(fill = "steelblue") +
  labs(title = "Top 20 Most Common Words in 'The Call of the Wild'",
```

```
       x = "Word",
       y = "Frequency") +
  coord_flip() +
  theme_minimal()
```




```
wordcloud(words = word_frequencies$word,
          freq = word_frequencies$n,
          min.freq = 1,
          max.words = 20,
          random.order = FALSE,
          rot.per = 0.35,
          colors = brewer.pal(8, "Dark2"))
```



d. Explore the sentiment of the text using three of the sentiment lexicons in `tidytext`. What does your analysis say about the sentiment of the text?

Notes:

- Make sure to NOT remove stop words this time.
- `afinn` is a numeric score and should be handled differently than the categorical scores.

```
tidy_text <- wild %>%
  unnest_tokens(word, text)
```

```
afinn <- get_sentiments("afinn") %>%
  filter(word %in% tidy_text$word)

sentiment_afinn <- tidy_text %>%
  inner_join(afinn, by = "word") %>%
  summarise(total_score = sum(value))

sentiment_afinn
```

```
## # A tibble: 1 x 1
##   total_score
##         <dbl>
## 1        -591
```

```
bing <- get_sentiments("bing") %>%
  filter(word %in% tidy_text$word)

sentiment_bing <- tidy_text %>%
  inner_join(bing, by = "word") %>%
  count(sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment_balance = positive - negative)

sentiment_bing
```

```
## # A tibble: 1 x 3
##   negative positive sentiment_balance
##      <dbl>    <dbl>             <dbl>
## 1     1513      838              -675
```

```
nrc <- get_sentiments("nrc") %>%
  filter(word %in% tidy_text$word)

sentiment_nrc <- tidy_text %>%
  inner_join(nrc, by = "word") %>%
  count(sentiment) %>%
  spread(sentiment, n, fill = 0)

sentiment_nrc
```

```
## # A tibble: 1 x 10
##   anger anticipation disgust  fear   joy negative positive sadness surprise
##   <dbl>        <dbl>   <dbl> <dbl> <dbl>    <dbl>    <dbl>   <dbl>    <dbl>
## 1   671          620     417  1167   469     1749     1483     606      696
## # i 1 more variable: trust <dbl>
```

The AFINN, NRC, and Bing lexicons reveal a predominantly negative sentiment, underscored by a significant presence of words related to fear, anger, sadness, and disgust, which aligns with the book's themes of survival, conflict, and the harsh realities of nature. Despite this, it looks like the text also contains substantial positive and neutral emotions like joy, trust, and anticipation, hinting at moments of triumph and resilience amidst adversity.

---

e. If you didn't do so in 2.d, compute the average sentiment score of the text using `afinn`. Which positive words had the biggest impact? Which negative words had the biggest impact?

```
afinn <- get_sentiments("afinn")

average_sentiment <- tidy_text %>%
  inner_join(afinn, by = "word") %>%
  summarise(average_score = mean(value))

average_sentiment
```

```
## # A tibble: 1 x 1
##   average_score
##           <dbl>
## 1        -0.322
```

```
# positive words with the biggest impact
positive_impact <- tidy_text %>%
  inner_join(afinn, by = "word") %>%
  filter(value > 0) %>%
  arrange(desc(value)) %>%
  distinct(word, .keep_all = TRUE)

# negative words with the biggest impact
negative_impact <- tidy_text %>%
  inner_join(afinn, by = "word") %>%
  filter(value < 0) %>%
  arrange(value) %>%
  distinct(word, .keep_all = TRUE)

print(head(positive_impact, n = 5))
```

```
## # A tibble: 5 x 3
##   gutenberg_id word      value
##          <int> <chr>     <dbl>
```

```
## 1          215 miracle       4
## 2          215 wonderful     4
## 3          215 triumph       4
## 4          215 win           4
## 5          215 terrific      4
```

```
print(head(negative_impact, n = 5))
```

```
## # A tibble: 5 x 3
##   gutenberg_id word   value
##          <int> <chr>  <dbl>
## 1          215 cock      -5
## 2          215 hell      -4
## 3          215 angry     -3
## 4          215 anger     -3
## 5          215 bloody    -3
```

Miracle, wonderful, triumph, win, and terrific seemed to be the positive words with the biggest impact.

Cock, hell, angry, anger, and bloody seemed to be the negative words with the biggest impact.

---

f. You should have found that "no" was an important negative word in the sentiment score. To know if that really makes sense, let's turn to the raw lines of text for context. Pull out all of the lines that have the word "no" in them. Make sure to not pull out extraneous lines (e.g., a line with the word "now").

```
lines_with_no <- grep("\\bno\\b", wild$text, value = TRUE, ignore.case = TRUE)

head(lines_with_no, n = 20)
```

```
##  [1] "Manuel's treachery. No one saw him and Buck go off through the orchard"
##  [2] "solitary man, no one saw them arrive at the little flag station known"
##  [3] "that it was the club, but his madness knew no caution. A dozen times he"
##  [4] ""He's no slouch at dog-breakin', that's wot I say," one of the men on"
##  [5] "all, that he stood no chance against a man with a club. He had learned"
##  [6] "in the red sweater. "And seem' it's government money, you ain't got no"
##  [7] "animal. The Canadian Government would be no loser, nor would its"
##  [8] "while he developed no affection for them, he none the less grew"
##  [9] "The other dog made no advances, nor received any; also, he did not"
## [10] "heart of civilization and flung into the heart of things primordial. No"
## [11] "knew no law but the law of club and fang."
## [12] "full-grown wolf, though not half so large as she. There was no warning,"
## [13] "Buck to trouble him in his sleep. So that was the way. No fair play."
## [14] "tail appeasingly, turned to run when he saw that appeasement was of no"
## [15] "his flank. But no matter how Spitz circled, Joe whirled around on his"
## [16] "their comradeship had no more trouble. His only apparent ambition, like"
## [17] "again he returned. Were they in the tent? No, that could not be, else"
## [18] "unduly civilized dog, and of his own experience knew no trap and so"
## [19] "no ice at all."
## [20] "him of his unfinished ration. There was no defending it. While he was"
```

---

g. Draw some conclusions about how "no" is used in the text.

It looks like the word "no" often highlights the absence or lack of something, such as visibility or chances of success. It also highlights the limitations faced by characters, especially Buck, in the harsh environment.

---

h. We can also look at how the sentiment of the text changes as the text progresses. Below, I have added two columns to the original dataset. Now I want you to do the following wrangling:

- Tidy the data (but don't drop stop words).
- Add the word sentiments using `bing`.
- Count the frequency of sentiments by index.
- Reshape the data to be wide with the count of the negative sentiments in one column and the positive in another, along with a column for index.
- Compute a sentiment column by subtracting the negative score from the positive.

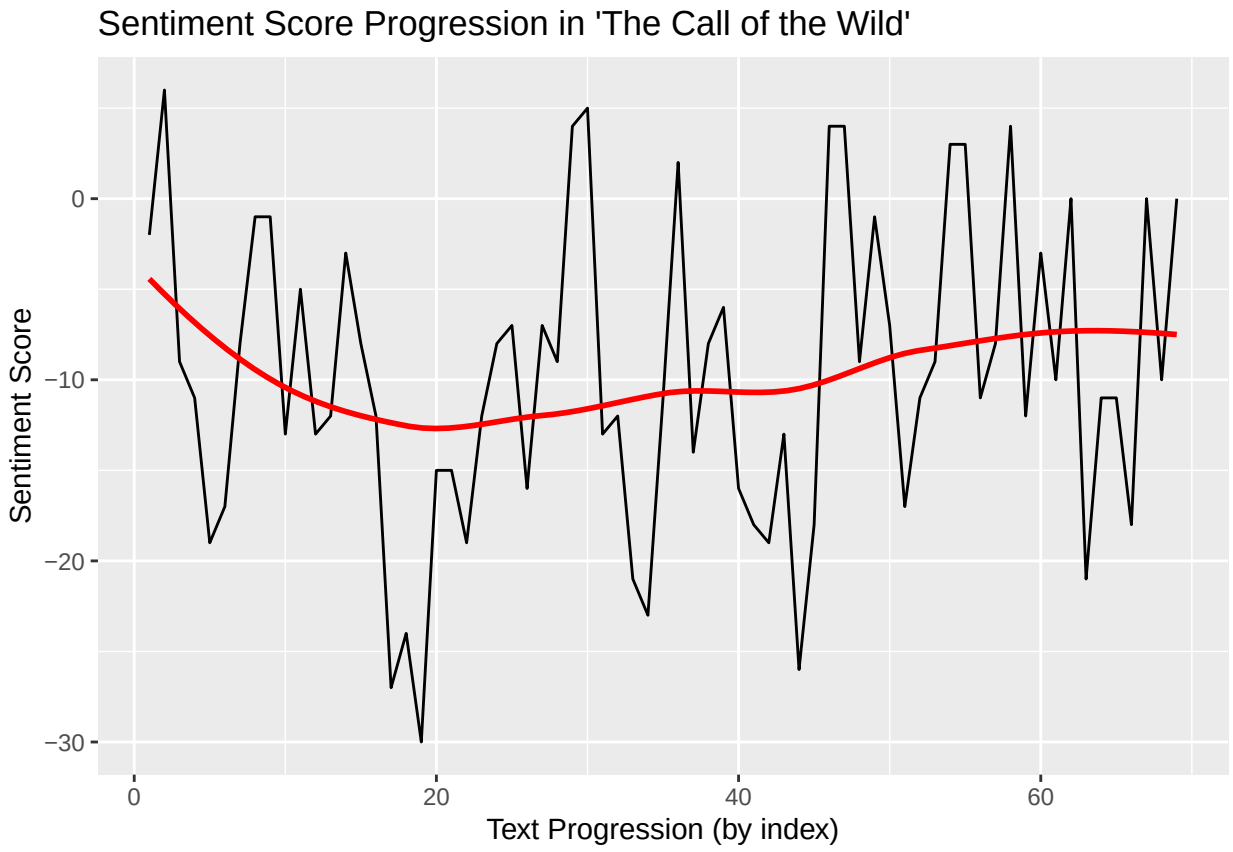
```
wild_time <- wild %>%
  mutate(line = row_number(), index = floor(line/45) + 1) %>%
  unnest_tokens(word, text) %>%
  inner_join(get_sentiments("bing"), by = "word")
```

```
sentiment_counts <- wild_time %>%
  count(index, sentiment)

sentiment_wide <- sentiment_counts %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = list(n = 0))

sentiment_wide <- sentiment_wide %>%
  mutate(sentiment_score = positive - negative)
```

---

i. Create a plot of the sentiment scores as the text progresses.

```
ggplot(sentiment_wide, aes(x = index, y = sentiment_score)) +
  geom_line() +
  geom_smooth(se = FALSE, colour = "red", method = "loess") +
  labs(title = "Sentiment Score Progression in 'The Call of the Wild'",
       x = "Text Progression (by index)", y = "Sentiment Score")
```



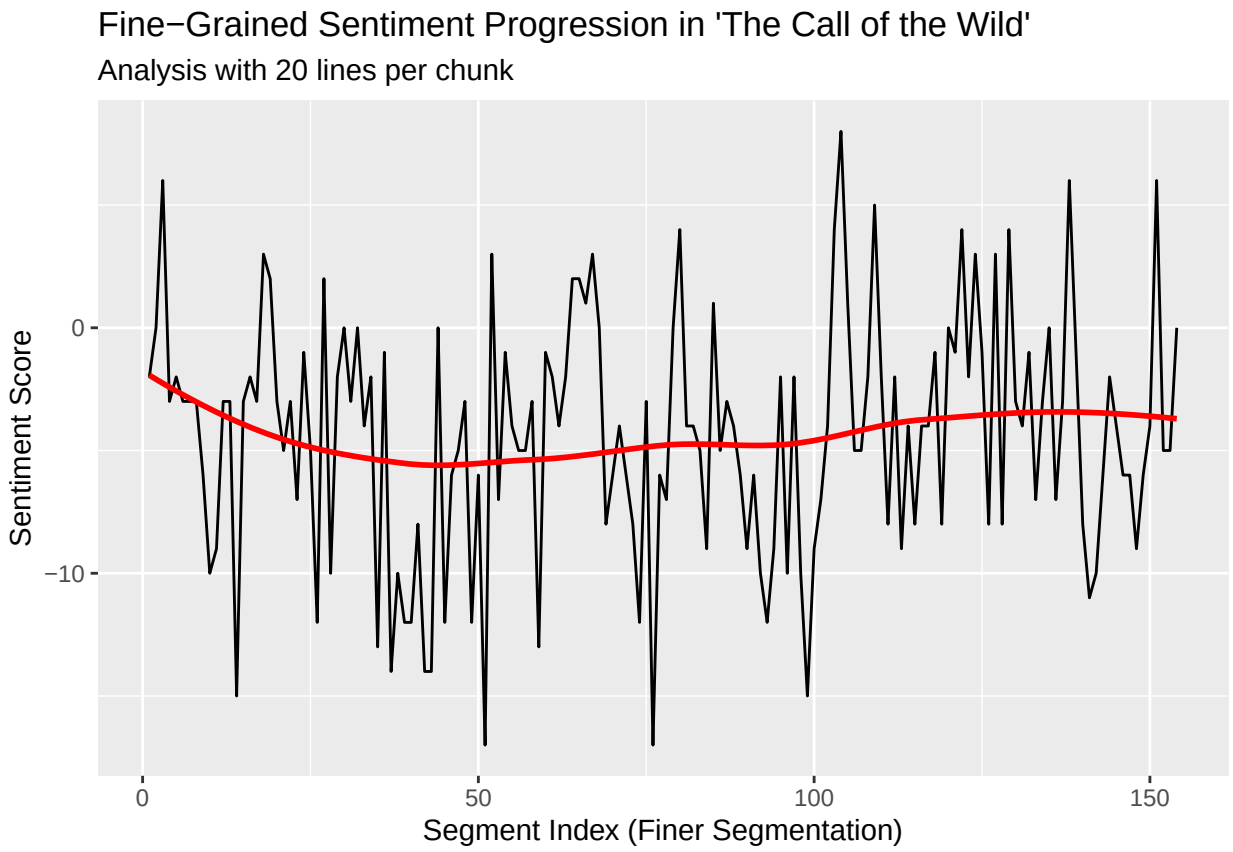
---

j. The choice of 45 lines per chunk was pretty arbitrary. Try modifying the index value a few times and recreating the plot in i. Based on your plots, what can you conclude about the sentiment of the novel as it progresses?

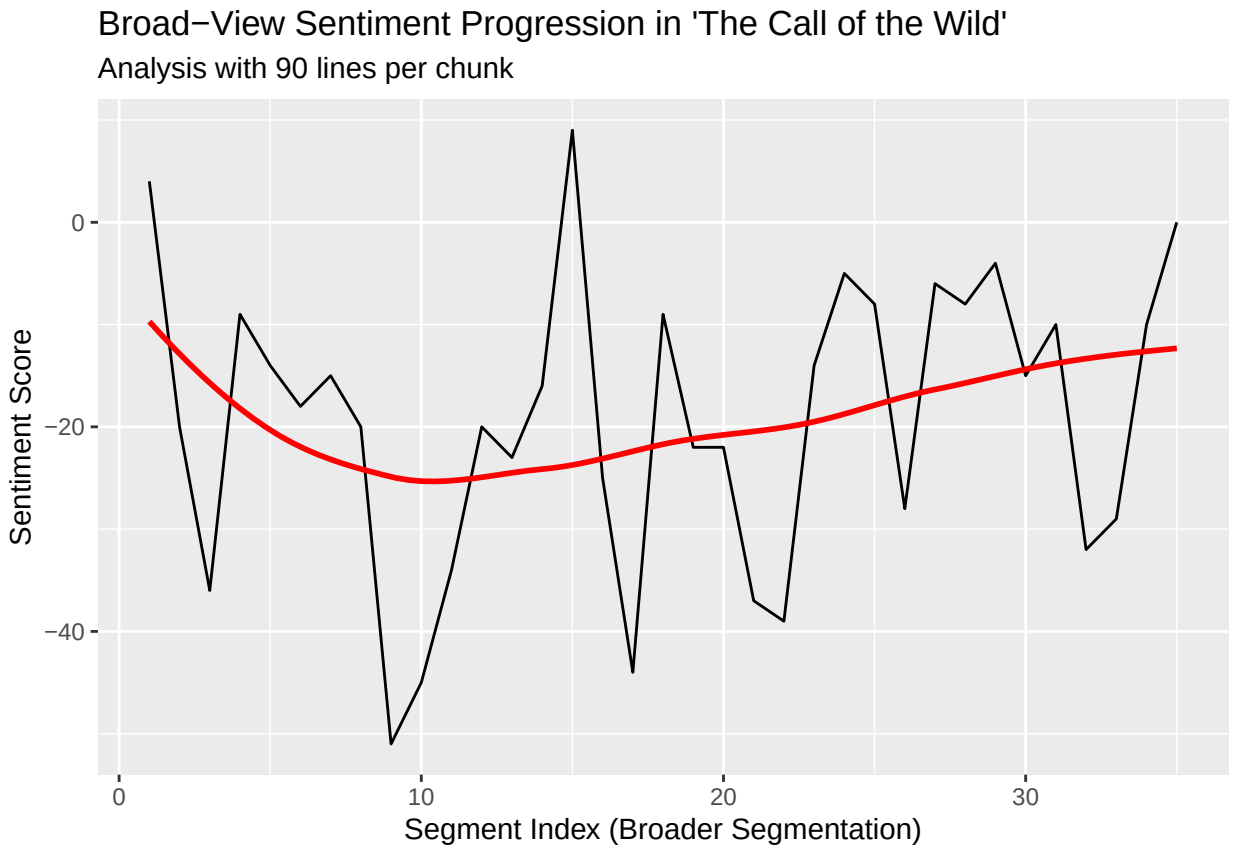
```
wild_time_fine <- wild %>%
  mutate(line = row_number(), index = floor(line/20) + 1) %>%
  unnest_tokens(word, text) %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  count(index, sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = list(n = 0)) %>%
  mutate(sentiment_score = positive - negative)
```

```
wild_time_coarse <- wild %>%
  mutate(line = row_number(), index = floor(line/90) + 1) %>%
  unnest_tokens(word, text) %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  count(index, sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = list(n = 0)) %>%
  mutate(sentiment_score = positive - negative)
```

```
ggplot(wild_time_fine, aes(x = index, y = sentiment_score)) +
  geom_line() +
  geom_smooth(se = FALSE, colour = "red", method = "loess") +
  labs(title = "Fine-Grained Sentiment Progression in 'The Call of the Wild'",
       x = "Segment Index (Finer Segmentation)", y = "Sentiment Score",
       subtitle = "Analysis with 20 lines per chunk")
```




```
ggplot(wild_time_coarse, aes(x = index, y = sentiment_score)) +
  geom_line() +
  geom_smooth(se = FALSE, colour = "red", method = "loess") +
  labs(title = "Broad-View Sentiment Progression in 'The Call of the Wild'",
       x = "Segment Index (Broader Segmentation)", y = "Sentiment Score",
       subtitle = "Analysis with 90 lines per chunk")
```



The fine-grained plot of sentiment progression possibly suggests detailed emotional fluctuations, corresponding closely with specific events and narrative nuances. The broad-view plot smooths these details into a depiction of general sentiment trends, aligning with major plot developments or thematic shifts. Both have a similar trendline despite their granularity differences, indicating consistent thematic sentiments throughout the novel.

---

k. Let's look at the bigrams (2 consecutive words). Tokenize the text by bigrams.

```
wild_bigrams <- wild %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

head(wild_bigrams)
```

```
## # A tibble: 6 x 2
##   gutenberg_id bigram
##          <int> <chr>
## 1          215 <NA>
## 2          215 <NA>
## 3          215 <NA>
## 4          215 <NA>
## 5          215 <NA>
## 6          215 the call
```

---

l. Produce a sorted table that counts the frequency of each bigram and notice that stop words are still an issue.

```
bigram_counts <- wild_bigrams %>%
  count(bigram, sort = TRUE)

head(bigram_counts)
```

```
## # A tibble: 6 x 2
##   bigram     n
##   <chr>  <int>
## 1 <NA>     408
## 2 of the   233
## 3 in the   172
## 4 he was   127
## 5 to the   116
## 6 it was   107
```
