Lab 7

Sydney Stitt

Math 241, Week 9

```r
# Put all necessary libraries here
library(tidyverse)
library(tidytext)
library(dplyr)
library(wordcloud)
library(RColorBrewer)
library(tm)

# 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
```

```r
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!

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

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

```r
#Filtering for the ten most popular female names starting with Z
start_z <- babynames %>%
  filter(year == "2000") %>%
  filter(sex == "F") %>%
  filter(substr(name, 1, 1) == "Z") %>%
  top_n(10, wt = n)
```

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

```r
#Filtering for the ten most popular female names containing Z
has_z_names <- babynames %>%
  filter(year == "2000") %>%
  filter(sex == "F") %>%
  filter(grepl("z", name)) %>%
  top_n(10, wt = n)
```

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

```
#Filtering for the ten most popular female names ending with Z
ends_z_names <- babynames %>%
  filter(year == "2000") %>%
  filter(sex == "F") %>%
  filter(grepl("z$", name)) %>%
  top_n(10, wt = n)
```

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!)

No, there are no overlapping names in the three tables

```
#Running anti joins to see if there are overlapping names
join_start_has <- anti_join(start_z, has_z_names)
join_start_ends <- anti_join(start_z, ends_z_names)
join_has_ends <- anti_join(has_z_names, ends_z_names)
```

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

```
#Filtering for names that contain numerics
babynames %>%
  filter(grepl("[0-9]", name)) %>%
  top_n(10, wt = n)
```

```
## # A tibble: 0 x 5
## # i 5 variables: year <dbl>, sex <chr>, name <chr>, n <int>, prop <dbl>
```

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".

```
#Creating the list of names with numbers spelled alphabetically
babynames_lowercase <- babynames %>%
  mutate(name = tolower(name))
df_numbers <- c("one", "two", "three", "four", "five", "six", "seven", "eight", "nine")
```

```
#Creating the table using the list of names as reference
extracted_babynames <- str_extract(babynames_lowercase$name, paste(df_numbers, collapse = "|"))
flattened_babynames <- unlist(extracted_babynames)
phrase_count_table <- table(flattened_babynames)
phrase_count_table
```

```
## flattened_babynames
## eight  four  nine   one seven   six three   two
##   356     2   807 10142    50   106    58   288
```

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

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

```r
#Making a table of names that start with "four"
four_babynames <- babynames_lowercase %>%
  filter(grepl("four", name)) %>%
  group_by(name)

#Making a table of names that start with "seven"
seven_babynames <- babynames_lowercase %>%
  filter(grepl("seven", name)) %>%
    group_by(name)

#Joining the seven and four name tables together
seven_four_names <- full_join(four_babynames, seven_babynames) %>%
  group_by(name) %>%
  summarize(n = n())
```

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

```r
#Filtering for names that do not contain aeiouy
babynames_no_vowels <- babynames_lowercase %>%
  filter(!grepl("[aeiouy]", name)) %>%
  group_by(name) %>%
  summarize(n = n())
babynames_no_vowels$name
```

```
##  [1] "bb"  "bg"  "bj"  "cj"  "dj"  "jb"  "jc"  "jd"  "jj"  "jl"  "jm"  "jp"
## [13] "jr"  "jt"  "jw"  "kc"  "kd"  "kj"  "kt"  "lb"  "lc"  "ld"  "lg"  "lj"
## [25] "mc"  "md"  "mj"  "mr"  "mrk" "ng"  "pj"  "rb"  "rc"  "rd"  "rj"  "rl"
## [37] "sj"  "st"  "tc"  "tj"  "tr"  "wc"  "wm"
```

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.

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

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

```r
#Tidying and tokenizing the text
wild_words <- wild %>%
  unnest_tokens(output = word, input = text)
```

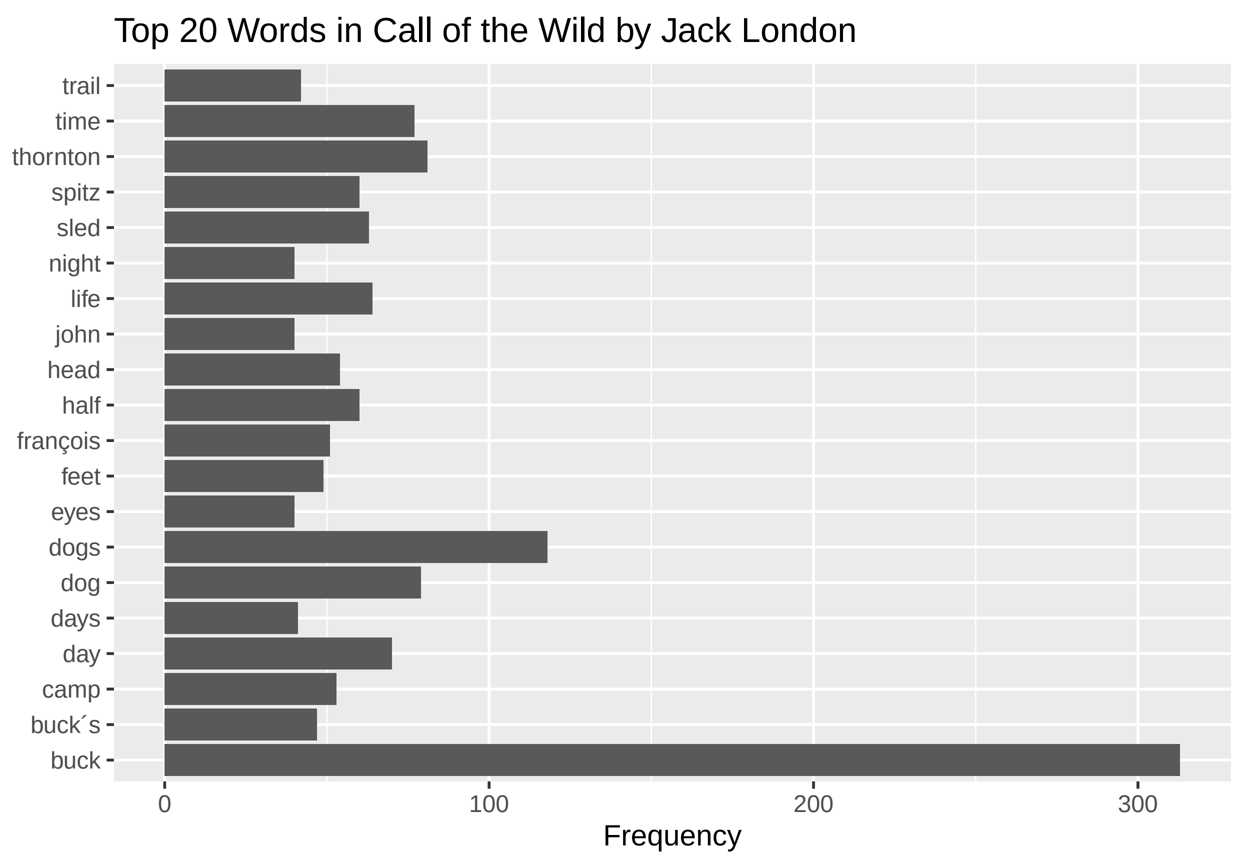
b. Find the frequency of the 20 most common words. First, remove stop words.

```r
#Removing stop words and finding frequencies
wild_words_freq <- wild_words %>%
  anti_join(stop_words) %>%
  group_by(word) %>%
  summarize(n = n()) %>%
  arrange(desc(n)) %>%
  slice_head(n = 20)
wild_words_freq
```

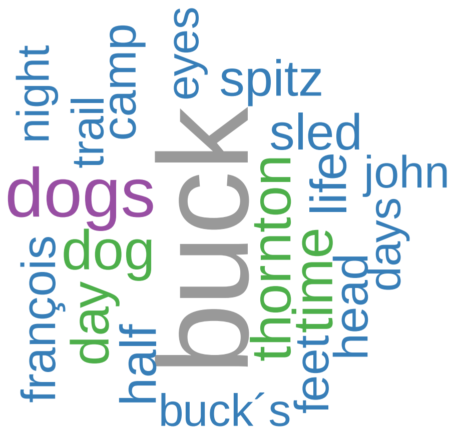
```
## # A tibble: 20 x 2
##    word         n
##    <chr>    <int>
##  1 buck       313
##  2 dogs       118
##  3 thornton    81
##  4 dog         79
##  5 time        77
##  6 day         70
##  7 life        64
##  8 sled        63
##  9 half        60
## 10 spitz       60
## 11 head        54
## 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.

```
#Making the bar graph
wild_words_freq %>%
  ggplot(aes(y = word, x = n)) +
  geom_col(show.legend = FALSE) +
  labs(title = "Top 20 Words in Call of the Wild by Jack London", x = "Frequency", y = NULL)
```

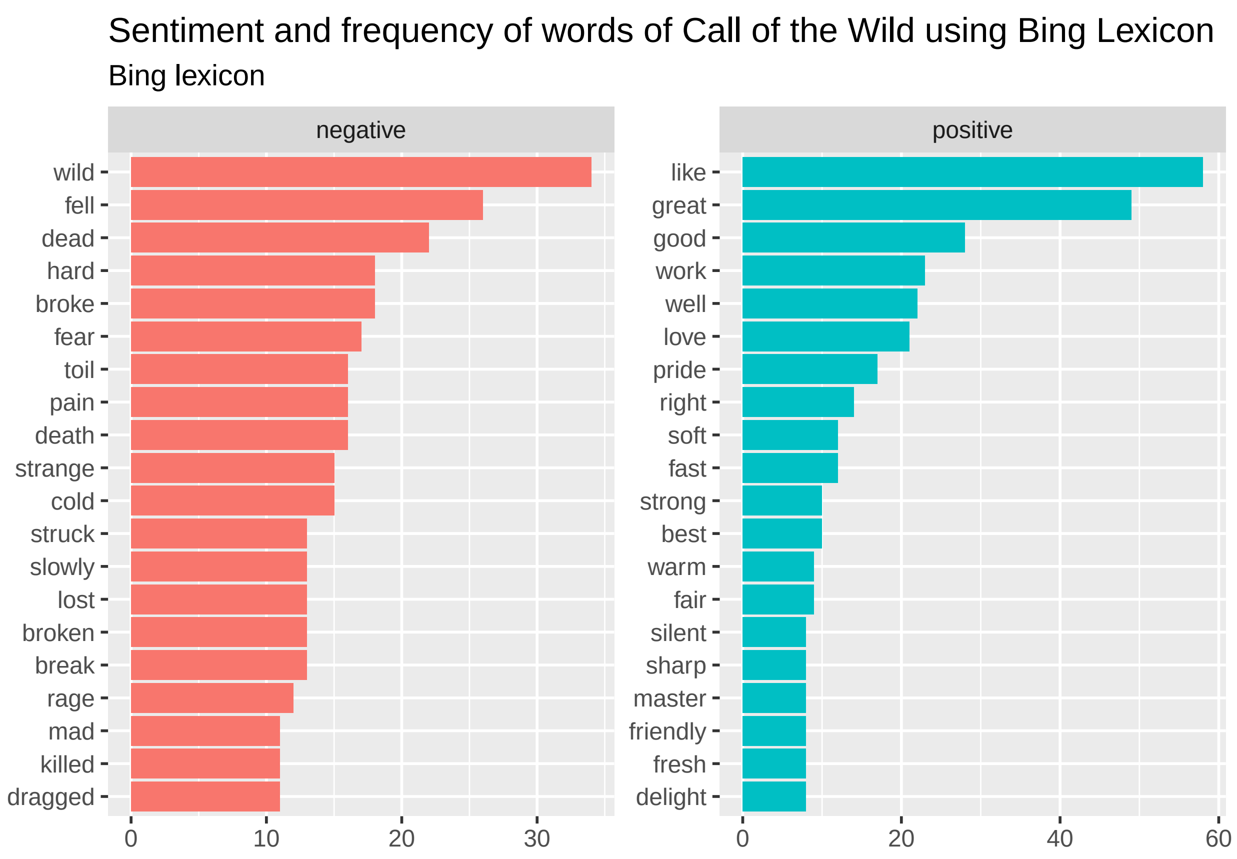


```
#Making the word cloud
pal <- brewer.pal(9, "Set1")
wordcloud(wild_words_freq$word, wild_words_freq$n,
          scale = c(4, 1),
          rot.per = .5, colors = pal,
          min.freq = 1, random.order = FALSE)
```

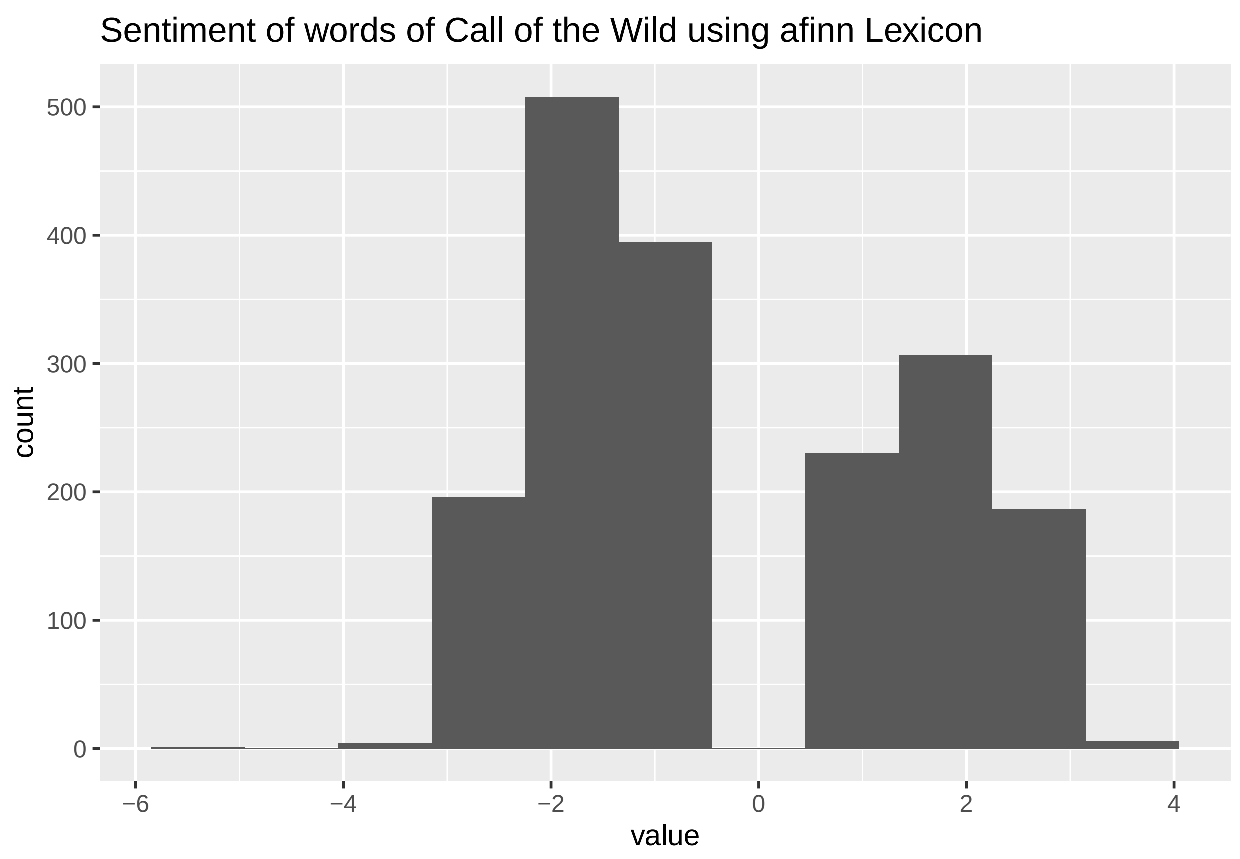


d. Explore the sentiment of the text using three of the sentiment lexicons in `tidytext`. What does your analysis say about the sentiment of the text?

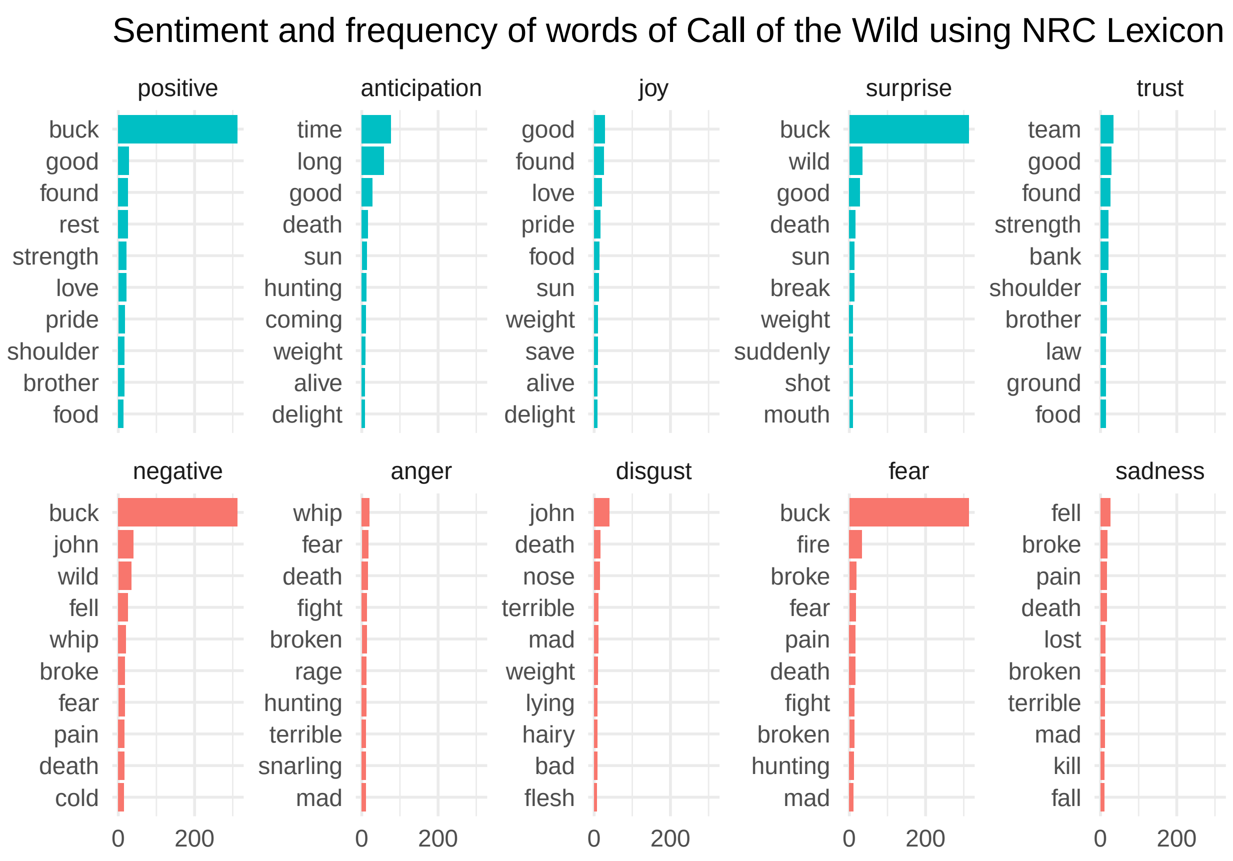
```r
#Using bing sentiment lexicon
sentiments_wild_bing <- wild_words %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  count(sentiment, word, sort = TRUE) %>%
  group_by(sentiment) %>%
  slice_head(n = 20) %>%
  ggplot(aes(y = fct_reorder(word, n), x = n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free") +
  labs(
    title = "Sentiment and frequency of words of Call of the Wild using Bing Lexicon",
    subtitle = "Bing lexicon",
    y = NULL, x = NULL
  )
sentiments_wild_bing
```



```
#Using afinn sentiment lexicon
sentiment_wild <- wild_words %>%
  left_join(get_sentiments("afinn"), by = c("word" = "word")) %>%
  filter(!is.na(value))
ggplot(sentiment_wild, aes(x = value)) +
  geom_histogram(binwidth = 0.9) +
  labs(title = "Sentiment of words of Call of the Wild using afinn Lexicon")
```



```
#Using NRC sentiment lexicon
wild_words %>%
  inner_join(get_sentiments("nrc"), by = "word") %>%
  mutate(
    sentiment = fct_relevel(
      sentiment, "positive", "anticipation", "joy", "surprise", "trust",
      "negative", "anger", "disgust", "fear", "sadness"
    ),
    sentiment_binary = if_else(sentiment %in% c("positive", "anticipation", "joy", "surprise", "trust")
  ) %>%
  count(sentiment_binary, sentiment, word, sort = TRUE) %>%
  group_by(sentiment) %>%
  slice_head(n = 10) %>%
  ggplot(aes(y = fct_reorder(word, n), x = n, fill = sentiment_binary)) +
  geom_col() +
  guides(fill = FALSE) +
  facet_wrap(~sentiment, scales = "free_y", ncol = 5) +
  labs(
    title = "Sentiment and frequency of words of Call of the Wild using NRC Lexicon",
    y = NULL, x = NULL
  ) +
  scale_x_continuous(breaks = c(0, 200)) +
  theme_minimal(base_size = 11)
```



My analysis shows that words in Call of the Wild tend to be more frequently associated with negative afinn scores. The histogram of the afinn scores shows that words in Call of the Wild tend to skew to the right, meaning words with engative afinn scores occur more frequently. There are more frequent negative-ly associated words in Call of the Wild, and those words are associated with fear, disgust, and sadness, whereas there are fewer frequent positive associated words as seen using the NRC lexicon to perform sentiment analysis. In general, as seen inn the Bing lexicon analysis, there are more negatively associated words that appear more frequently.

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?

```
#Computing average score
sentiment_wild %>%
  summarize(mean = mean(value))
```

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

```
sentiment_wild %>%
  group_by(word) %>%
  summarize(mean = mean(value),
            n = n()) %>%
  arrange(desc(n))
```

```
## # A tibble: 528 x 3
##    word      mean     n
##    <chr>    <dbl> <int>
##  1 no          -1    95
##  2 like         2    58
##  3 great        3    49
##  4 fire        -2    33
##  5 good         3    28
##  6 dead        -3    22
##  7 cried       -2    21
##  8 love         3    21
##  9 strength     2    21
## 10 broke       -1    18
## # i 518 more rows
```

The average sentiment score for the text is -0.3222465. Positive words like "like", "great", "good", and "love" have the biggest imact, and negative words like "no", "fire", "dead", "cried", and "broke" have the biggest impact on the sentiment

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").

```r
#Creating a table of lines that contain the word no
no_wild <- wild %>%
  filter(str_detect(text, "\\bno\\b"))
no_wild
```

```
## # A tibble: 83 x 2
##    gutenberg_id text
##           <int> <chr>
##  1          215 solitary man, no one saw them arrive at the little flag station~
##  2          215 that it was the club, but his madness knew no caution. A dozen ~
##  3          215 "He's no slouch at dog-breakin', that's wot I say," one of the ~
##  4          215 all, that he stood no chance against a man with a club. He had ~
##  5          215 in the red sweater. "And seem' it's government money, you ain't~
##  6          215 animal. The Canadian Government would be no loser, nor would its
##  7          215 while he developed no affection for them, he none the less grew
##  8          215 The other dog made no advances, nor received any; also, he did ~
##  9          215 knew no law but the law of club and fang.
## 10          215 full-grown wolf, though not half so large as she. There was no ~
## # i 73 more rows
```

g. Draw some conclusions about how "no" is used in the text. No is used to describe an action, person, or object that isn't something (adjective, adverb, etc). It isn't commonly used as a way to negate something in dialogue, which may be negative in sentiment, so I do not think it creates a negative sentiment for the book.

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).

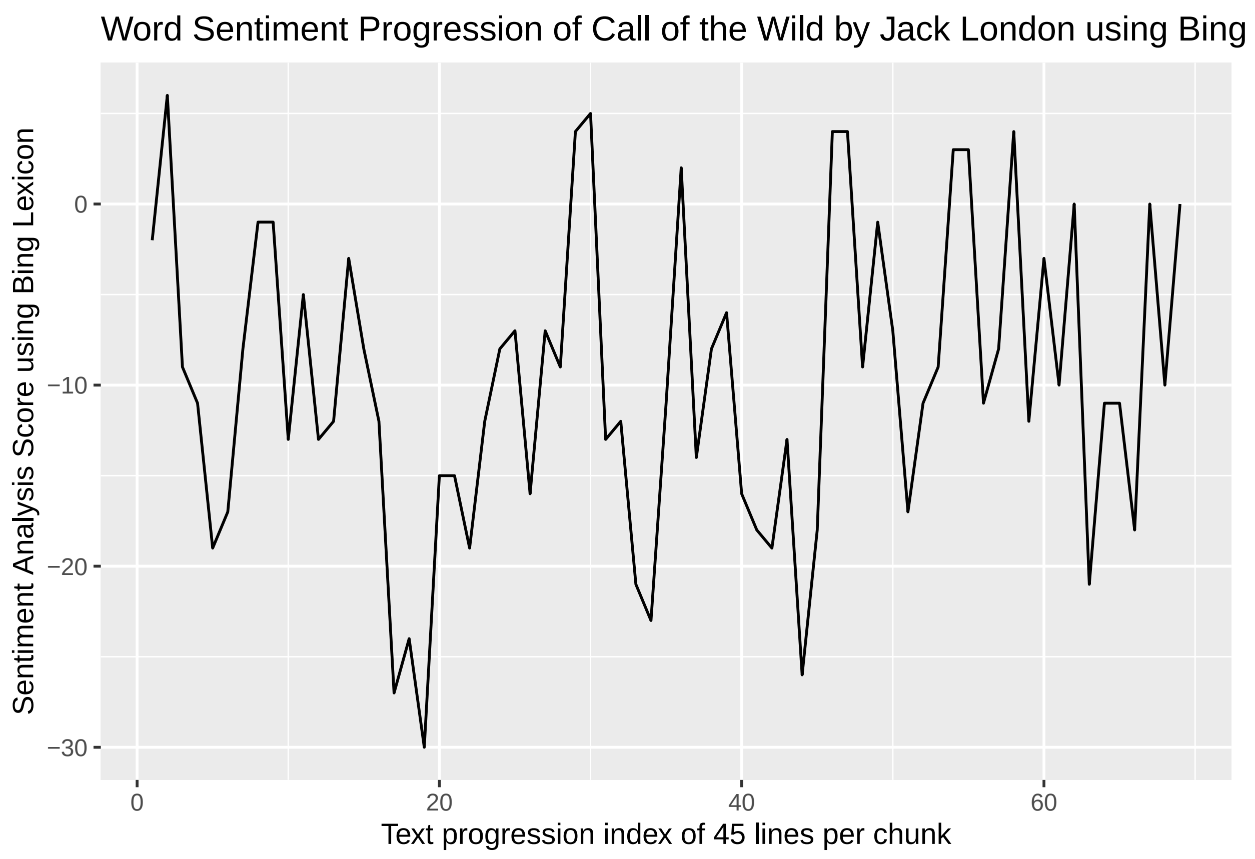
- Add the word sentiments using `bing`.
- Count the frequency of sentiments by index.
- Reshape the data to be wide with the count of the negative sentiments in one column and the positive in another, along with a column for index.
- Compute a sentiment column by subtracting the negative score from the positive.

```r
#Creating bing sentiments
bing_sentiments <- get_sentiments("bing")

#Joining bing sentiments dataframe grouping by index, pivoting
wild_time <- wild %>%
  mutate(line = row_number(), index = floor(line/45) + 1) %>%
  unnest_tokens(output = word, input = text) %>%
  left_join(bing_sentiments, by = "word") %>%
  group_by(index, sentiment) %>%
  summarise(count = n()) %>%
  pivot_wider(names_from = sentiment, values_from = count, values_fill = 0) %>%
  mutate(sentiment = positive - negative)
```

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

```r
#Making the plot of scores over time
ggplot(wild_time, aes(x = index, y = sentiment)) +
  geom_line() +
  labs(x = "Text progression index of 45 lines per chunk", y = "Sentiment Analysis Score using Bing Lex
```

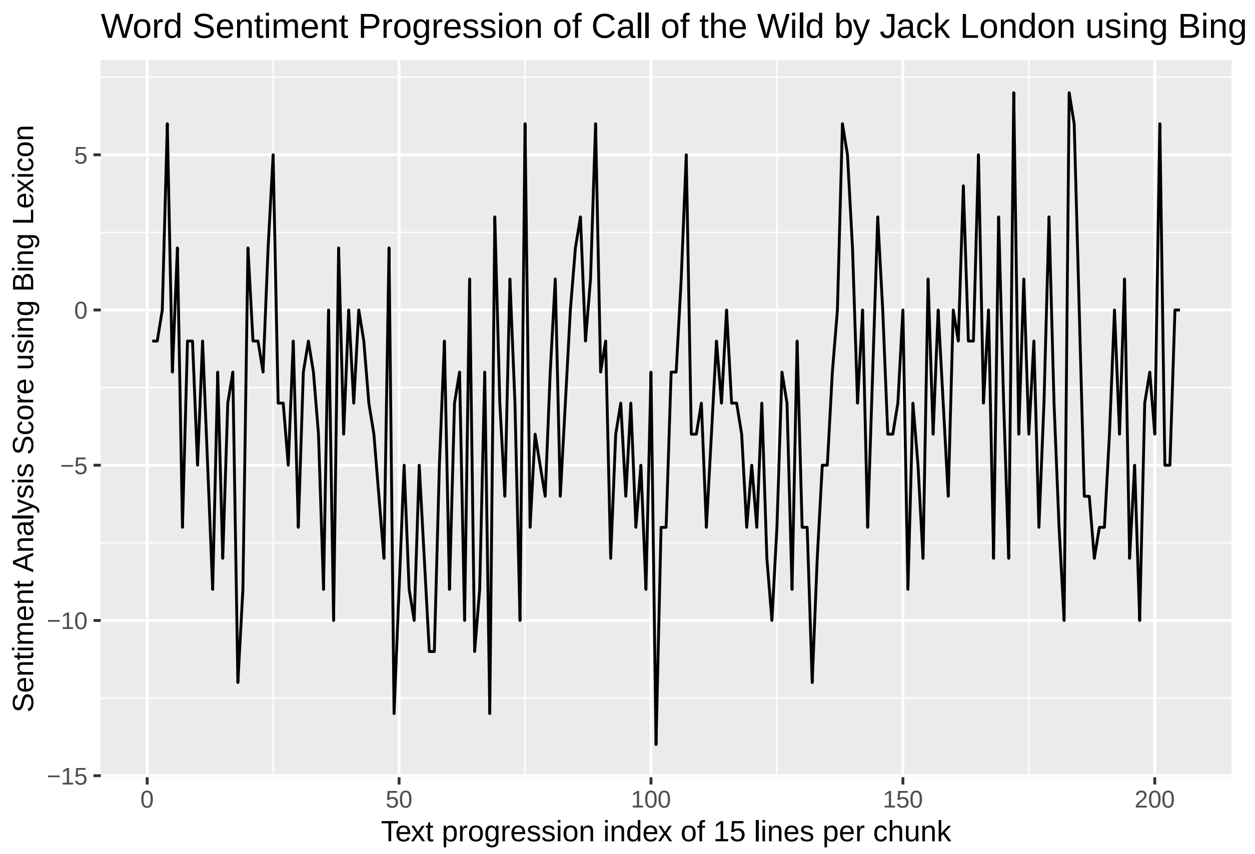


j. The choice of 45 lines per chunk was pretty arbitrary. Try modifying the index value a few times and recreating the plot in i. Based on your plots, what can you conclude about the sentiment of the novel as it progresses?

Generally, the sentiment of the novel based on Bing Lexicon is relatively negative, and skews more negative towards the middle of the novel, and slightly increases in positivity at the end of the novel.

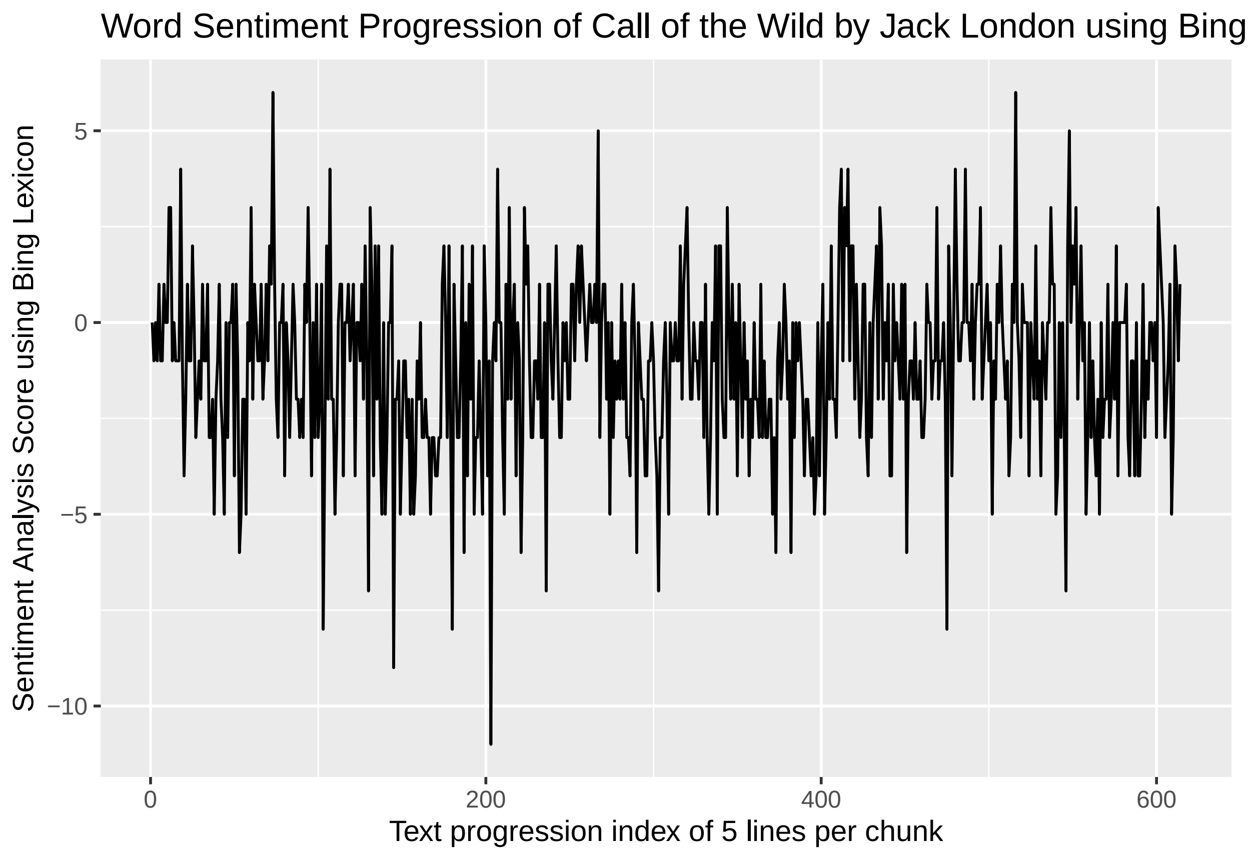
```r
#Changing the index length to 15 lines
wild_time2 <- wild %>%
  mutate(line = row_number(), index = floor(line/15) + 1) %>%
  unnest_tokens(output = word, input = text) %>%
  left_join(bing_sentiments, by = "word") %>%
  group_by(index, sentiment) %>%
  summarise(count = n()) %>%
  pivot_wider(names_from = sentiment, values_from = count, values_fill = 0) %>%
  mutate(sentiment = positive - negative)

#Graphing the 15 line index length
ggplot(wild_time2, aes(x = index, y = sentiment)) +
  geom_line() +
  labs(x = "Text progression index of 15 lines per chunk", y = "Sentiment Analysis Score using Bing Lex
```



```r
#Changing the index length to 5 lines
wild_time3 <- wild %>%
  mutate(line = row_number(), index = floor(line/5) + 1) %>%
  unnest_tokens(output = word, input = text) %>%
  left_join(bing_sentiments, by = "word") %>%
  group_by(index, sentiment) %>%
  summarise(count = n()) %>%
  pivot_wider(names_from = sentiment, values_from = count, values_fill = 0) %>%
  mutate(sentiment = positive - negative)

#Graphing the 5 line index length
ggplot(wild_time3, aes(x = index, y = sentiment)) +
  geom_line() +
  labs(x = "Text progression index of 5 lines per chunk", y = "Sentiment Analysis Score using Bing Lexi
```



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

```
#Tokenize by bigrams
wild_bigrams <- wild %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
  filter(!is.na(bigram))
wild_bigrams
```

```
## # A tibble: 29,442 x 2
##    gutenberg_id bigram
##           <int> <chr>
##  1          215 the call
##  2          215 call of
##  3          215 of the
##  4          215 the wild
##  5          215 by jack
##  6          215 jack london
##  7          215 chapter i
##  8          215 i into
##  9          215 into the
## 10          215 the primitive
## # i 29,432 more rows
```

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

```
#Counting for bigrams
bigram_freq <- wild_bigrams %>%
  count(bigram)
#Sorting for bigrams
bigram_freq <- bigram_freq %>%
  arrange(desc(n))
bigram_freq
```

```
## # A tibble: 18,907 x 2
##    bigram       n
##    <chr>    <int>
##  1 of the     233
##  2 in the     172
##  3 he was     127
##  4 to the     116
##  5 it was     107
##  6 and the     95
##  7 on the      80
##  8 he had      77
##  9 at the      68
## 10 into the    65
## # i 18,897 more rows
```