Lab 7

Omar Youssif

Math 241, Week 9

```r
# Put all necessary libraries here
library(tidyverse)
library(tidytext)
library(stringr)
library(wordcloud)
library(RColorBrewer)
# 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
Fbaby2000 <- babynames %>% filter(year == 2000, sex == "F")
dfa <- top_n(Fbaby2000 %>% filter(Fbaby2000$name %in% str_subset(Fbaby2000$name, "^Z")),
      n = 10,
      wt = n)
dfa$dataset <- "dfa"
dfa
```

```
## # A tibble: 10 x 6
##      year sex   name       n      prop dataset
##     <dbl> <chr> <chr>  <int>     <dbl> <chr>
##   1  2000 F     Zoe     3785 0.00190   dfa
##   2  2000 F     Zoey     691 0.000346  dfa
##   3  2000 F     Zaria    568 0.000285  dfa
##   4  2000 F     Zoie     320 0.000160  dfa
##   5  2000 F     Zariah   168 0.0000842 dfa
##   6  2000 F     Zion     156 0.0000782 dfa
##   7  2000 F     Zainab   142 0.0000712 dfa
##   8  2000 F     Zara     121 0.0000607 dfa
##   9  2000 F     Zahra    113 0.0000566 dfa
## 10  2000 F     Zaira    103 0.0000516 dfa
```

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

```r
dfb <- top_n(Fbaby2000 %>% filter(Fbaby2000$name %in% str_subset(Fbaby2000$name, "z")),
      n = 10,
      wt = n)
dfb$dataset <- "dfb"
dfb
```

```
## # A tibble: 10 x 6
##      year sex   name           n     prop dataset
```

```
##    <dbl> <chr> <chr>     <int>    <dbl> <chr>
##  1  2000 F     Elizabeth 15094 0.00757  dfb
##  2  2000 F     Mackenzie  6348 0.00318  dfb
##  3  2000 F     Mckenzie   2526 0.00127  dfb
##  4  2000 F     Makenzie   1613 0.000809 dfb
##  5  2000 F     Jazmin     1391 0.000697 dfb
##  6  2000 F     Jazmine    1353 0.000678 dfb
##  7  2000 F     Lizbeth     817 0.000410 dfb
##  8  2000 F     Eliza       759 0.000380 dfb
##  9  2000 F     Litzy       722 0.000362 dfb
## 10  2000 F     Esperanza   499 0.000250 dfb
```

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

```r
dfc <- top_n(Fbaby2000 %>% filter(Fbaby2000$name %in% str_subset(Fbaby2000$name, "z$")),
      n = 10,
      wt = n)
dfc$dataset <- "dfc"
dfc
```

```
## # A tibble: 11 x 6
##     year sex   name          n       prop dataset
##    <dbl> <chr> <chr>     <int>      <dbl> <chr>
##  1  2000 F     Luz         489 0.000245   dfc
##  2  2000 F     Beatriz     357 0.000179   dfc
##  3  2000 F     Mercedez    141 0.0000707  dfc
##  4  2000 F     Maricruz     96 0.0000481  dfc
##  5  2000 F     Liz          72 0.0000361  dfc
##  6  2000 F     Inez         69 0.0000346  dfc
##  7  2000 F     Odaliz       24 0.0000120  dfc
##  8  2000 F     Marycruz     23 0.0000115  dfc
##  9  2000 F     Cruz         19 0.00000952 dfc
## 10  2000 F     Deniz        16 0.00000802 dfc
## 11  2000 F     Taiz         16 0.00000802 dfc
```

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!)

Nope, we can also check this through inner joining all of them and doing all possible inner joins to catch all intersects, and we can see in the dataset "dfd" that we have 0 overlaps between all of them as it is empty.

```r
dfd1 <- inner_join(dfa, dfb, by = "name")
dfd2 <- inner_join(dfa, dfc, by = "name")
dfd3 <- inner_join(dfb, dfc, by = "name")

dfd <- full_join(dfd1, dfd2, dfd3, by = "name")
dfd
```

```
## # A tibble: 0 x 21
## # i 21 variables: year.x.x <dbl>, sex.x.x <chr>, name <chr>, n.x.x <int>,
## #   prop.x.x <dbl>, dataset.x.x <chr>, year.y.x <dbl>, sex.y.x <chr>,
## #   n.y.x <int>, prop.y.x <dbl>, dataset.y.x <chr>, year.x.y <dbl>,
## #   sex.x.y <chr>, n.x.y <int>, prop.x.y <dbl>, dataset.x.y <chr>,
## #   year.y.y <dbl>, sex.y.y <chr>, n.y.y <int>, prop.y.y <dbl>,
## #   dataset.y.y <chr>
```

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

This returns an empty set, so we know that none of the names have a numeric in them.

```
babynames %>% filter(babynames$name %in% str_subset(babynames$name, "[0-9]"))
```

```
## # 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".

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.

```
lower_baby <- babynames
lower_baby$name <- lower_baby$name %>% str_replace_all(pattern = c("A" = "a", "B" = "b", "C" = "c", "D"
```

```
num_baby <- lower_baby %>% filter(lower_baby$name %in% str_subset(lower_baby$name, "one|two|three|four|

dfd <- data.frame(
one = sum(str_detect(num_baby$name, pattern = "one")),
two = sum(str_detect(num_baby$name, pattern = "two")),
three = sum(str_detect(num_baby$name, pattern = "three")),
four = sum(str_detect(num_baby$name, pattern = "four")),
five = sum(str_detect(num_baby$name, pattern = "five")),
six = sum(str_detect(num_baby$name, pattern = "six")),
seven = sum(str_detect(num_baby$name, pattern = "seven")),
eight = sum(str_detect(num_baby$name, pattern = "eight")),
nine = sum(str_detect(num_baby$name, pattern = "nine"))
)

print(dfd)
```

```
##     one two three four five six seven eight nine
## 1 10210 288    58    2    0 106    50   356  807
```

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

The written number "five" does not show up at all with "four" being very close to zero in terms of count too.

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

```
num_baby %>% filter(str_detect(num_baby$name, pattern = "four"))
```

```
## # A tibble: 2 x 5
##    year sex   name        n       prop
##   <dbl> <chr> <chr>   <int>      <dbl>
## 1  1914 M     balfour     5 0.00000732
## 2  1928 M     balfour     5 0.00000438
```

```
num_baby %>% filter(str_detect(num_baby$name, pattern = "seven"))
```

```
## # A tibble: 50 x 5
```

```
##     year sex   name      n       prop
##    <dbl> <chr> <chr> <int>      <dbl>
##  1  1968 M     seven     5 0.00000282
##  2  1993 M     seven     7 0.00000339
##  3  1994 F     seven     5 0.00000257
##  4  1994 M     seven    10 0.00000491
##  5  1995 M     seven    11 0.00000547
##  6  1996 F     seven     7 0.00000365
##  7  1996 M     seven    10 0.00000499
##  8  1997 F     seven     7 0.00000367
##  9  1997 M     seven    22 0.0000110
## 10  1998 F     seven    14 0.00000722
## # i 40 more rows
```

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

```r
lower_baby %>% filter(str_detect(lower_baby$name, pattern = "^[^aeiouy]+$"))
```

```
## # A tibble: 1,260 x 5
##     year sex   name      n      prop
##    <dbl> <chr> <chr> <int>     <dbl>
##  1  1880 M     wm       14 0.000118
##  2  1881 M     wm       15 0.000139
##  3  1882 M     wm       18 0.000148
##  4  1883 M     wm       16 0.000142
##  5  1884 M     wm       22 0.000179
##  6  1885 M     wm       23 0.000198
##  7  1885 M     ng        5 0.0000431
##  8  1886 M     wm       15 0.000126
##  9  1887 M     wm       15 0.000137
## 10  1888 M     wm       25 0.000192
## # i 1,250 more rows
```

(I'm not sure what dataset to use here, the question is a little unclear to me. Can't tell whether all baby names or the written number one. I checked and the written number one yields 0 names).

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.

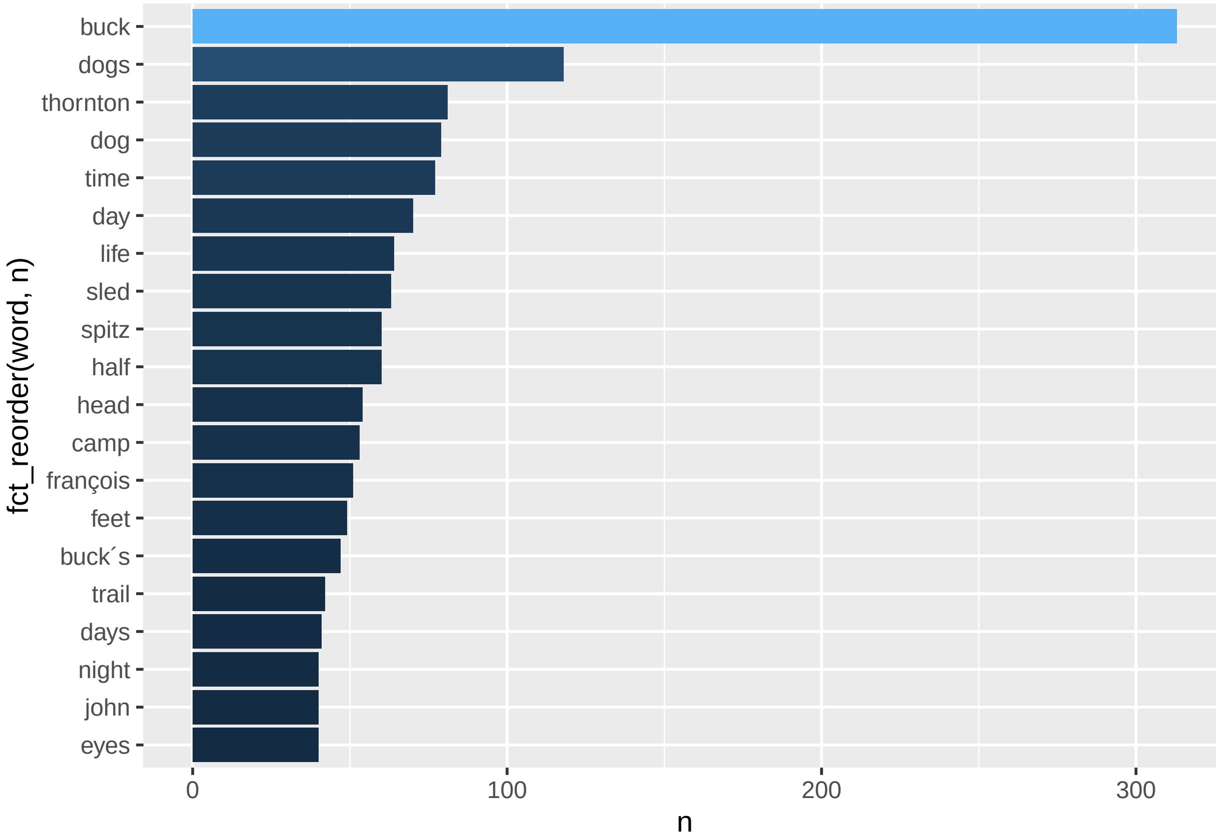
```r
library(gutenbergr)
wild <- gutenberg_download(215)
```

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

```r
p2dfa <- wild %>% unnest_tokens(output = word, input = text) %>% select(-gutenberg_id)
```

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

```r
p2dfa %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  filter(n > 39) %>%
  ggplot(aes(y = fct_reorder(word, n), x = n, fill = n)) +
  geom_col() +
  guides(fill = FALSE)
```




```
p2df <- p2dfa %>% anti_join(stop_words) %>% count(word, sort = TRUE) %>% filter(n > 39)
```

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

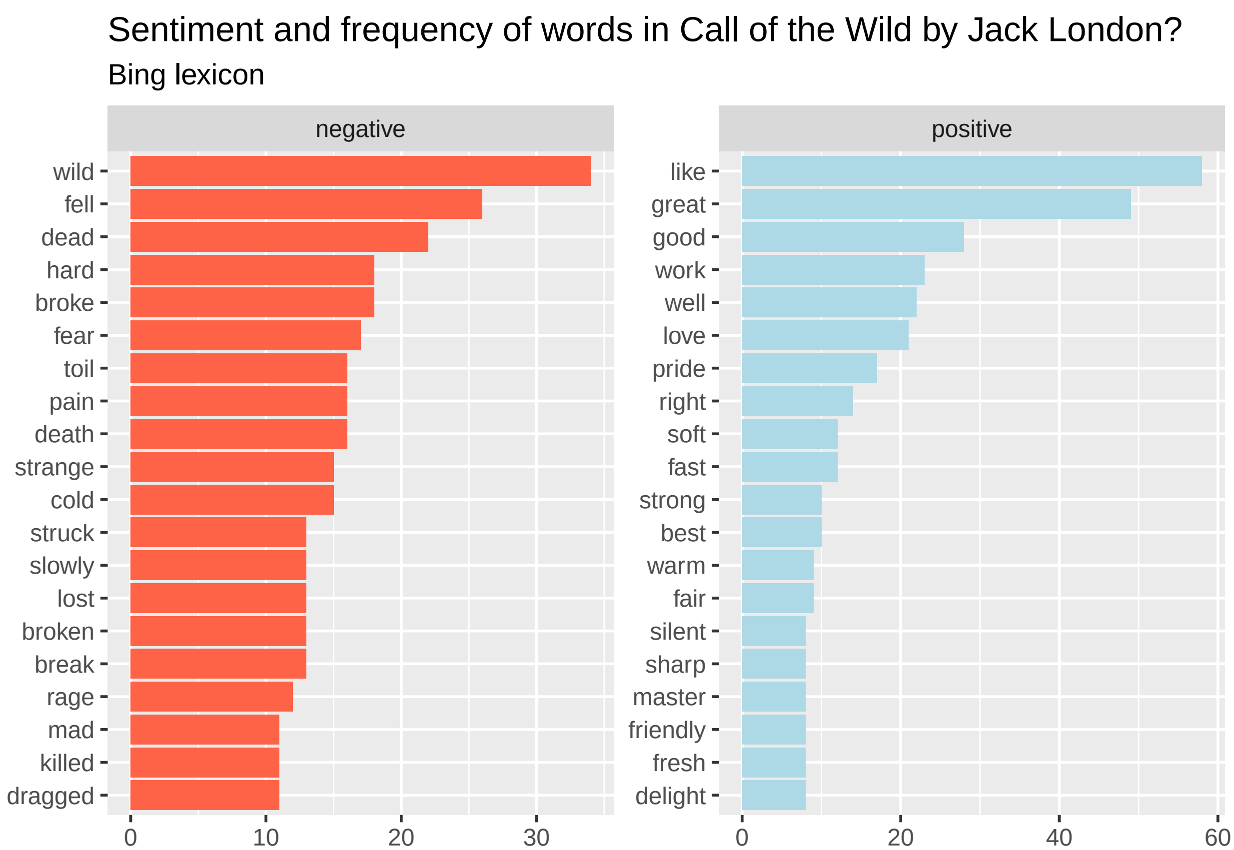
```
pal <- brewer.pal(9, "Set1")

wordcloud(p2df$word,
          scale = c(3, 1),
          rot.per = .5, colors = pal,
          min.freq = 0, 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?

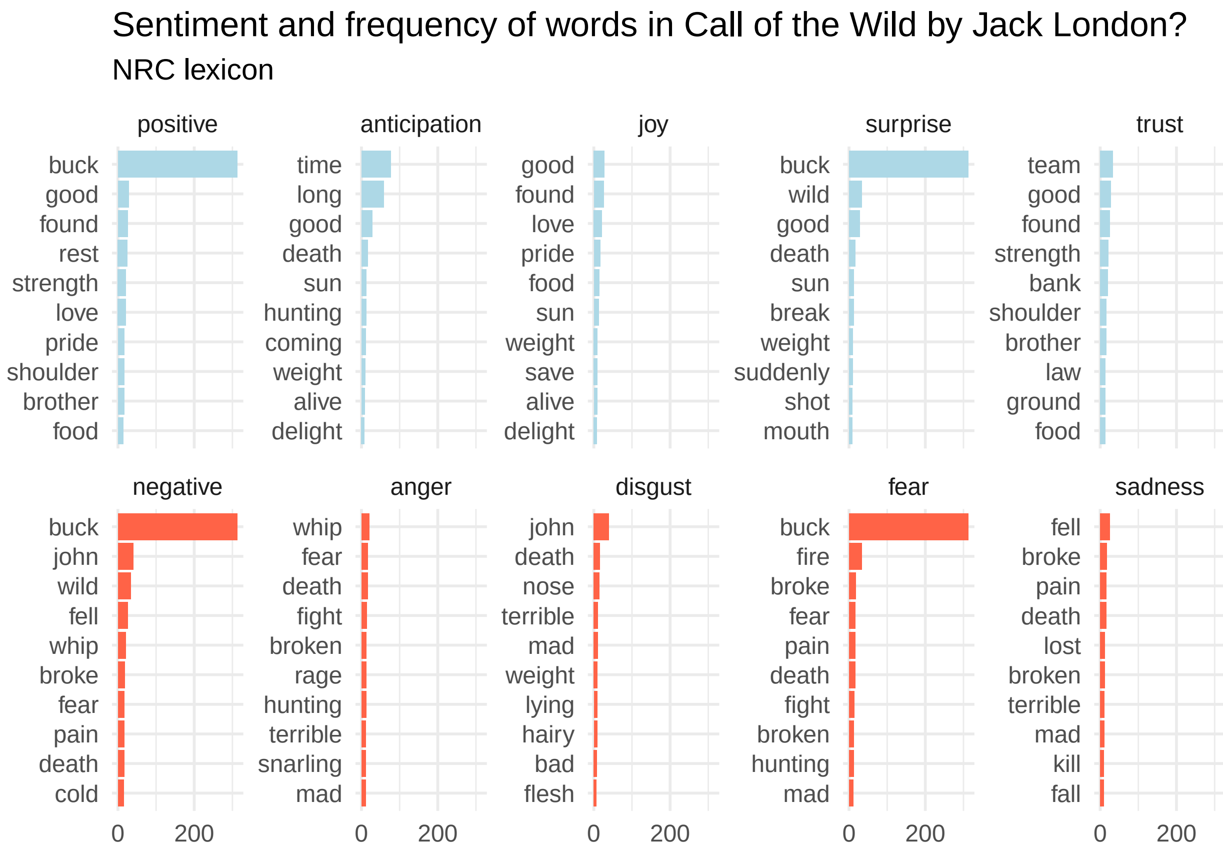
Notes:

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

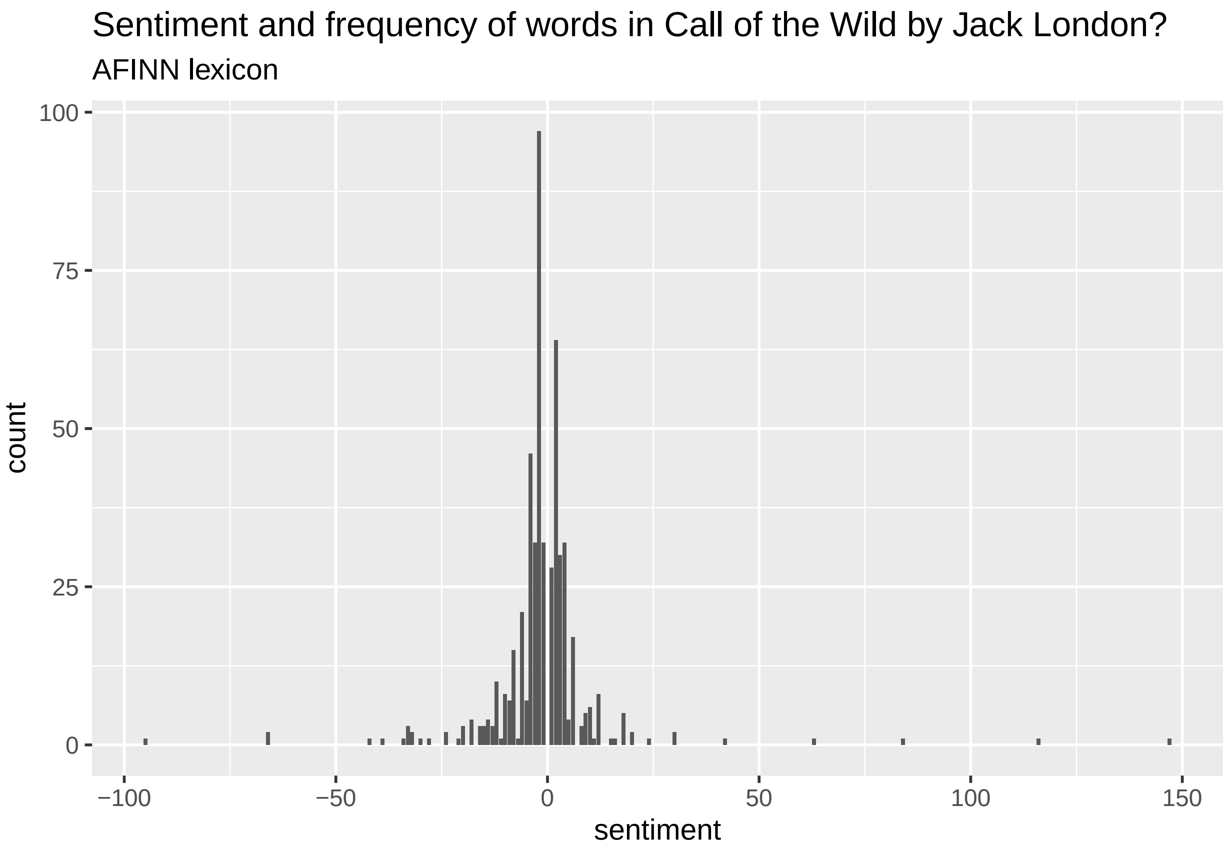
```
p2dfa %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  count(sentiment, word, sort = TRUE) %>%
  group_by(sentiment) %>%
  slice_head(n = 20) %>%
  ggplot(aes(y = fct_reorder(word, n), x = n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free") +
  labs(
    title = "Sentiment and frequency of words in Call of the Wild by Jack London?",
    subtitle = "Bing lexicon",
    y = NULL, x = NULL
  ) +
  scale_fill_manual(values = c("tomato", "lightblue"))
```




```r
p2dfa %>%
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 in Call of the Wild by Jack London?",
    subtitle = "NRC lexicon",
    y = NULL, x = NULL
  ) +
  scale_x_continuous(breaks = c(0, 200)) +
  theme_minimal(base_size = 11) +
  scale_fill_manual(values = c("tomato", "lightblue"))
```




```
p2dfe <- p2dfa %>% inner_join(get_sentiments("afinn")) %>%
  group_by(word) %>%
  summarise(sentiment = sum(value))

ggplot(data = p2dfe, mapping = aes(x = sentiment)) +
    geom_bar() +
  labs(
    title = "Sentiment and frequency of words in Call of the Wild by Jack London?",
    subtitle = "AFINN lexicon")
```



The text seems to lean a little more towards the negative side, particularly when looking at the positive words and some possibly being connector words, not actual positive words.

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?

```
p2dfe %>% summarize(avg = mean(sentiment))
```

```
## # A tibble: 1 x 1
##     avg
##   <dbl>
## 1 -1.12
```

```
p2dfe %>%  arrange(desc(sentiment)) %>% slice_head(n = 5)
```

```
## # A tibble: 5 x 2
##   word     sentiment
##   <chr>        <dbl>
## 1 great          147
## 2 like           116
## 3 good            84
## 4 love            63
## 5 strength        42
```

```
p2dfe %>%  arrange(sentiment) %>% slice_head(n = 5)
```

```
## # A tibble: 5 x 2
##   word  sentiment
##   <chr>     <dbl>
## 1 no          -95
```

```
## 2 dead         -66
## 3 fire         -66
## 4 cried        -42
## 5 lost         -39
```

Positive: great, like, good, love, strength Negative: no, dead, fire, cried, lost

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").

```
dff <- wild %>%
  unnest_tokens(output = words, input = text, token = "lines") %>%
  select(-gutenberg_id) %>%
  mutate(id = row_number())

dff %>% group_by(id) %>%
  filter(str_count(words, "\\s") > 1) %>%
  filter(str_detect(words, "\\bno\\b"))
```

```
## # A tibble: 94 x 2
## # Groups:   id [94]
##    words                                                                  id
##    <chr>                                                               <int>
##  1 manuel's treachery. no one saw him and buck go off through the orchard    86
##  2 solitary man, no one saw them arrive at the little flag station known     88
##  3 that it was the club, but his madness knew no caution. a dozen times he   217
##  4 "he's no slouch at dog-breakin', that's wot i say," one of the men on     233
##  5 all, that he stood no chance against a man with a club. he had learned    252
##  6 in the red sweater. "and seem' it's government money, you ain't got no    280
##  7 animal. the canadian government would be no loser, nor would its          284
##  8 while he developed no affection for them, he none the less grew           297
##  9 the other dog made no advances, nor received any; also, he did not       311
## 10 heart of civilization and flung into the heart of things primordial. no   338
## # i 84 more rows
```

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

It seems to be used mostly as a synonym of "regardless of" but always in a negative connotation, so I see why the lexicons thought it's a very negative word. Though this is a overgenerlization and there are some cases where that's not the case, but it can't separate them, so it's incredibly boosted.

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)
```

```
dfh1 <- wild_time %>%
  unnest_tokens(output = word, input = text) %>%
  select(-gutenberg_id) %>%
```

```r
  inner_join(get_sentiments("bing"), by = "word") %>%
  count(sentiment, index, sort = TRUE)

dfh2 <- wild_time %>%
  unnest_tokens(output = word, input = text) %>%
  select(-gutenberg_id) %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  count(sentiment, index, sort = TRUE) %>%
  pivot_wider(names_from = sentiment, values_from = n) %>%
  mutate(vibe = positive - negative)

dfh1
```

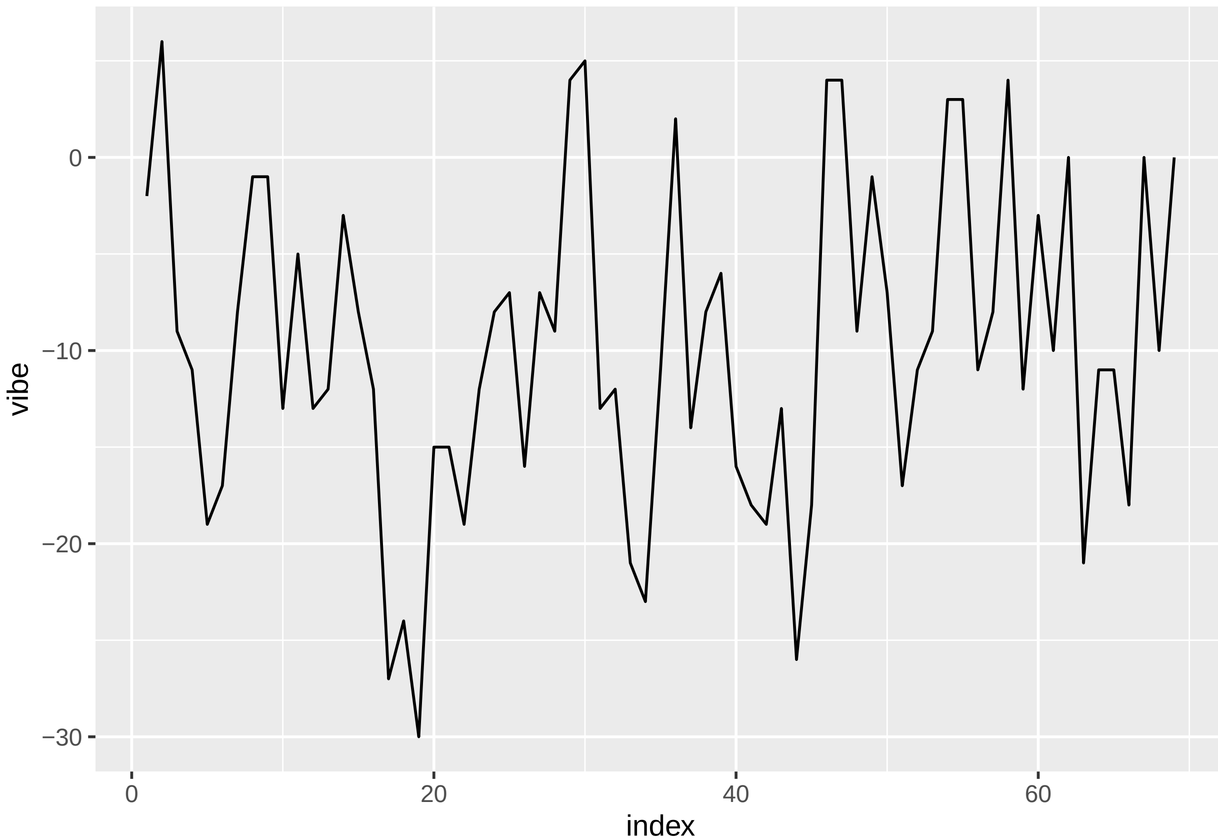
```
## # A tibble: 138 x 3
##    sentiment index     n
##    <chr>     <dbl> <int>
##  1 negative     17    38
##  2 negative     19    35
##  3 negative     22    35
##  4 negative     63    34
##  5 negative     44    33
##  6 negative     21    32
##  7 negative     18    30
##  8 negative     33    30
##  9 negative     40    30
## 10 negative     42    30
## # i 128 more rows
```

```r
dfh2
```

```
## # A tibble: 69 x 4
##    index negative positive  vibe
##    <dbl>    <int>    <int> <int>
##  1    17       38       11   -27
##  2    19       35        5   -30
##  3    22       35       16   -19
##  4    63       34       13   -21
##  5    44       33        7   -26
##  6    21       32       17   -15
##  7    18       30        6   -24
##  8    33       30        9   -21
##  9    40       30       14   -16
## 10    42       30       11   -19
## # i 59 more rows
```

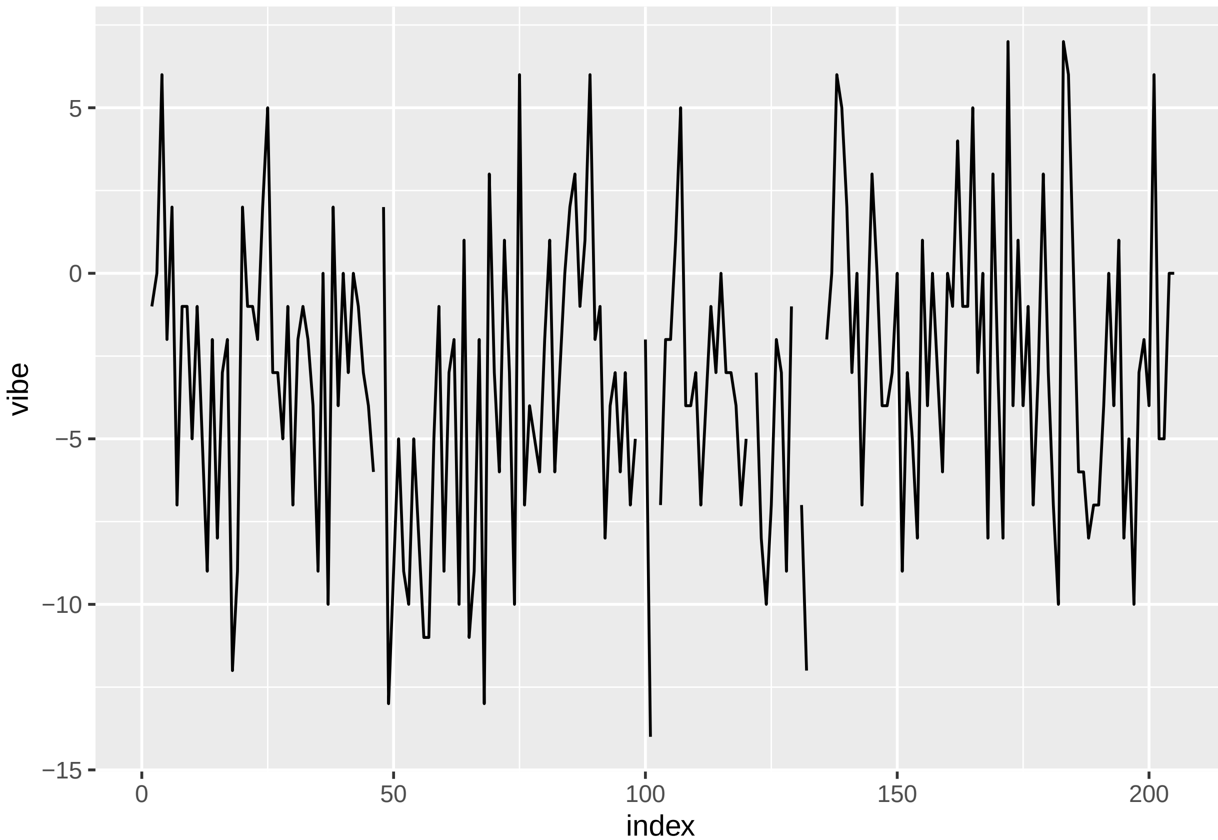
i. Create a plot of the sentiment scores as the text progresses.

```r
ggplot(data = dfh2, mapping = aes(x = index, y = vibe)) +
  geom_line()
```



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_time2 <- wild %>%
  mutate(line = row_number(), index = floor(line/15) + 1)

dfj <- wild_time2 %>%
  unnest_tokens(output = word, input = text) %>%
  select(-gutenberg_id) %>%
  inner_join(get_sentiments("bing"), by = "word") %>%
  count(sentiment, index, sort = TRUE) %>%
  pivot_wider(names_from = sentiment, values_from = n) %>%
  mutate(vibe = positive - negative)

ggplot(data = dfj, mapping = aes(x = index, y = vibe)) +
  geom_line()
```



It seems generally stagnant (trend-wise) though. it doesn't look as extremely negative as the other line makes it out to be. Though it seems like it goes up and down significantly in short periods of times, which makes a large index division lean negative.

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

```
dfk <- wild %>%
  unnest_tokens(output = bigrams, input = text, token = "ngrams", n = 2) %>%
  select(-gutenberg_id) %>%
  mutate(id = row_number()) %>%
  drop_na()

dfk
```

```
## # A tibble: 29,442 x 2
##    bigrams          id
##    <chr>         <int>
##  1 the call          6
##  2 call of           7
##  3 of the            8
##  4 the wild          9
##  5 by jack          11
##  6 jack london      12
##  7 chapter i        19
##  8 i into           20
##  9 into the         21
## 10 the primitive    22
## # 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.

```r
dfk %>%
  unnest_tokens(word, bigrams, drop = FALSE) %>%
  anti_join(stop_words) %>%
  group_by(bigrams) %>%
  summarize(n = n()) %>%
  arrange(desc(n))
```

```
## # A tibble: 15,351 x 2
##    bigrams            n
##    <chr>          <int>
##  1 john thornton     60
##  2 sol leks          56
##  3 the sled          44
##  4 the dogs          35
##  5 buck was          28
##  6 his feet          26
##  7 and buck          23
##  8 half breed        22
##  9 red sweater       22
## 10 the team          21
## # i 15,341 more rows
```