Lab 7

Morgen Kaufman

Math 241, Week 9

```r
# Put all necessary libraries herey
library(tidyverse)
library(tidytext)

# 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
```

```
library(dplyr)
library(stringr)
```

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.

```
#Hint: Use
t1 <- babynames %>%
  filter(year == "2000",
         sex == "F",
         str_detect(name, "Z")) %>%
  top_n(10)

t1
```

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

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

```
t2 <- babynames %>%
  filter(year == "2000",
         sex == "F",
         str_detect(name, "[Zz]")) %>%
  top_n(10)

t2
```

```
## # A tibble: 10 x 5
##     year sex   name          n     prop
##    <dbl> <chr> <chr>     <int>    <dbl>
##  1  2000 F     Elizabeth 15094 0.00757
##  2  2000 F     Mackenzie  6348 0.00318
##  3  2000 F     Zoe        3785 0.00190
##  4  2000 F     Mckenzie   2526 0.00127
##  5  2000 F     Makenzie   1613 0.000809
##  6  2000 F     Jazmin     1391 0.000697
##  7  2000 F     Jazmine    1353 0.000678
##  8  2000 F     Lizbeth     817 0.000410
##  9  2000 F     Eliza       759 0.000380
## 10  2000 F     Litzy       722 0.000362
```

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

```
t3 <- babynames %>%
  filter(year == "2000",
         sex == "F",
         str_detect(name, "z$")) %>%
  top_n(10)

t3
```

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

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!)

```r
names_1_2 <- inner_join(t1, t2, by = "name")
names_1_3 <- inner_join(t1, t3, by = "name")
names_2_3 <- inner_join(t2, t3, by = "name")
name_z_all <- bind_rows(names_1_2, names_1_3, names_2_3)
name_z_all
```

```
## # A tibble: 1 x 9
##   year.x sex.x name    n.x  prop.x year.y sex.y   n.y  prop.y
##    <dbl> <chr> <chr> <int>   <dbl>  <dbl> <chr> <int>   <dbl>
## 1   2000 F     Zoe    3785 0.00190   2000 F      3785 0.00190
```

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

```r
numeric_names <- babynames %>%
  filter(str_detect(name, "[0-9]"))

numeric_names
```

```
## # 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.

```r
numbernames <- babynames %>%
  mutate(name = tolower(name),
         zero = str_extract(name, "zero"),
         one = str_extract(name, "one"),
         two = str_extract(name, "two"),
         three = str_extract(name, "three"),
         four = str_extract(name, "four"),
         five = str_extract(name, "five"),
         six = str_extract(name, "six"),
         seven = str_extract(name, "seven"),
         eight = str_extract(name, "eight"),
         nine = str_extract(name, "nine")
         )

numbernames_sub <- numbernames %>%
```

```r
  filter(rowSums(!is.na(select(., zero, one, two, three, four, five, six, seven, eight, nine))) > 0)

numbernames_count <- numbernames_sub %>%
  summarise(
    zero = sum(!is.na(zero)),
    one = sum(!is.na(one)),
    two = sum(!is.na(two)),
    three = sum(!is.na(three)),
    four = sum(!is.na(four)),
    five = sum(!is.na(five)),
    six = sum(!is.na(six)),
    seven = sum(!is.na(seven)),
    eight = sum(!is.na(eight)),
    nine = sum(!is.na(nine))
  )

numbernames_count
```

```
## # A tibble: 1 x 10
##    zero   one   two three  four  five   six seven eight  nine
##   <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1     4 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?

Five does not show up at all.

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

```r
least_common_numbers_count <- numbernames %>%
  filter(rowSums(!is.na(select(., zero, four))) > 0) %>%
  count(name)

least_common_numbers_count
```

```
## # A tibble: 4 x 2
##   name        n
##   <chr>   <int>
## 1 balfour     2
## 2 luzero      2
## 3 zero        1
## 4 zeron       1
```

```r
least_common_numbers <- numbernames %>%
  filter(rowSums(!is.na(select(., zero, four))) > 0)

least_common_numbers
```

```
## # A tibble: 6 x 15
##    year sex   name        n       prop zero  one   two   three four  five  six
##   <dbl> <chr> <chr>   <int>      <dbl> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
```

```
## 1  1914 M     balfour     5 0.00000732 <NA>  <NA>  <NA>  <NA>  four  <NA>  <NA>
## 2  1928 M     balfour     5 0.00000438 <NA>  <NA>  <NA>  <NA>  four  <NA>  <NA>
## 3  1990 F     luzero      6 0.00000292 zero  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
## 4  1991 F     luzero      5 0.00000246 zero  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
## 5  2007 M     zeron       6 0.00000271 zero  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
## 6  2017 M     zero        7 0.00000357 zero  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
## # i 3 more variables: seven <chr>, eight <chr>, nine <chr>
```

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

```
# Filter names containing no vowels
names_no_vowels <- babynames %>%
  filter(!str_detect(name, "[aeiouyAEIOUY]")) %>%
  count(name)

names_no_vowels
```

```
## # A tibble: 43 x 2
##    name      n
##    <chr> <int>
##  1 Bb        5
##  2 Bg        6
##  3 Bj       49
##  4 Cj       62
##  5 Dj       49
##  6 Jb       69
##  7 Jc       97
##  8 Jd       85
##  9 Jj       44
## 10 Jl        6
## # i 33 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_token <- wild %>%
  unnest_tokens(output = word, input = text)
```

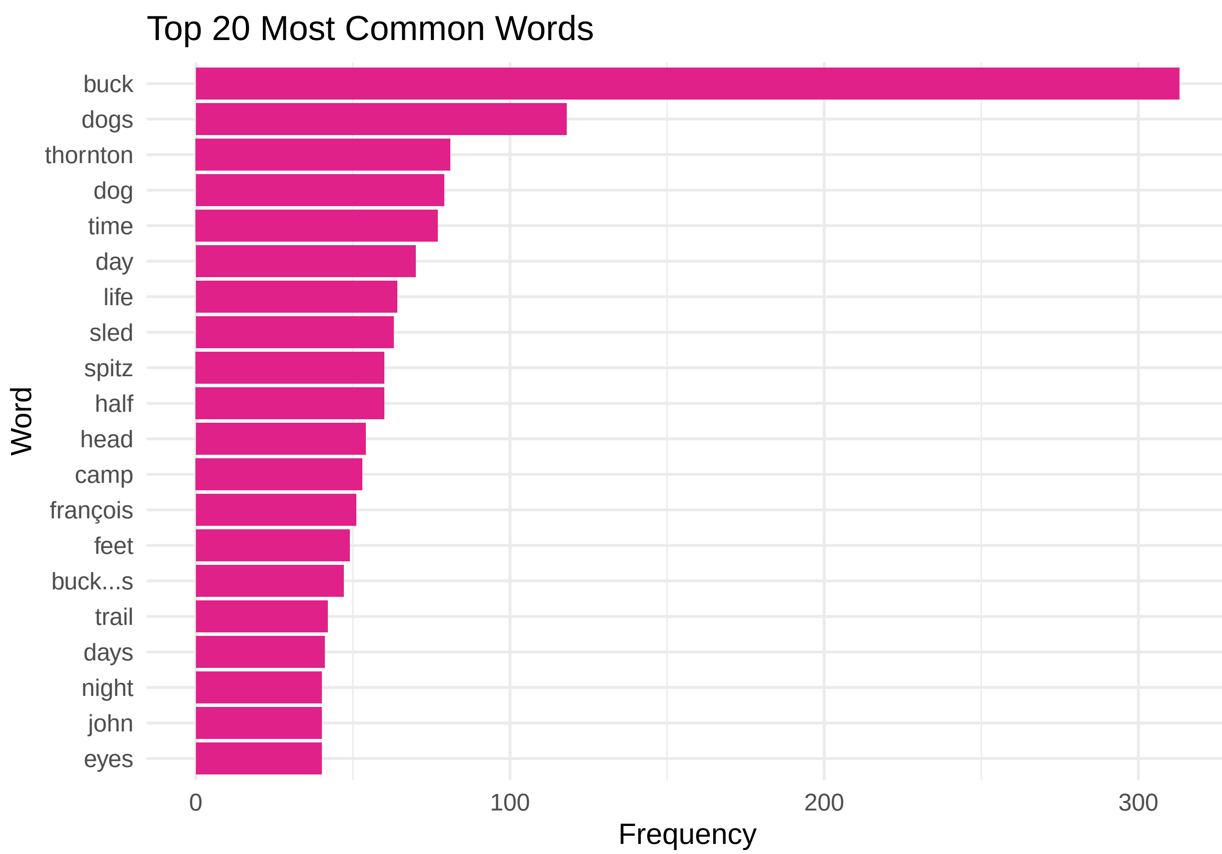
b. Find the frequency of the 20 most common words. First, remove stop words.

```r
data("stop_words")

wild20 <- wild_token %>%
  anti_join(stop_words) %>%
  group_by(word) %>%
  summarise(frequency = n()) %>%
  arrange(desc(frequency)) %>%
  top_n(20)

wild20
```

```
## # A tibble: 20 x 2
##    word     frequency
##    <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.

```r
library(wordcloud)

ggplot(data = wild20, aes(x = reorder(word, frequency), y = frequency)) +
  geom_bar(stat = "identity", fill = "#e0218a") +
  labs(x = "Word", y = "Frequency") +
  coord_flip() +
  ggtitle("Top 20 Most Common Words") +
  theme_minimal()
```




```
wordcloud(words = wild20$word,
          freq = wild20$frequency,
          scale = c(4, 1),
          rot.per = .5,
          colors = "#e0218a",
          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.

Afinn

```
wild_group <- wild_token %>%
  group_by(word) %>%
  summarise(frequency = n())

afinn_wild <- wild_group %>%
  left_join(get_sentiments("afinn")) %>%
  filter(!is.na(value)) %>%
  arrange(desc(value)) %>%
  group_by(word)
afinn_wild
```

```
## # A tibble: 528 x 3
## # Groups:   word [528]
```

```
##    word      frequency value
##    <chr>         <int> <dbl>
##  1 miracle           2     4
##  2 terrific          1     4
##  3 triumph           1     4
##  4 win               1     4
##  5 wonderful         1     4
##  6 adore             1     3
##  7 affection         2     3
##  8 beautiful         4     3
##  9 best             10     3
## 10 cheery            1     3
## # i 518 more rows
```

```
afinn_wild %>%
  group_by(value) %>%
  summarise(n())
```

```
## # A tibble: 9 x 2
##   value `n()`
##   <dbl> <int>
## 1    -5     1
## 2    -4     1
## 3    -3    56
## 4    -2   175
## 5    -1    80
## 6     1    71
## 7     2   101
## 8     3    38
## 9     4     5
```

```
afinn_wild %>%
  ungroup()%>%
  summarise(
    mean = mean(value))
```

```
## # A tibble: 1 x 1
##     mean
##    <dbl>
## 1 -0.379
```

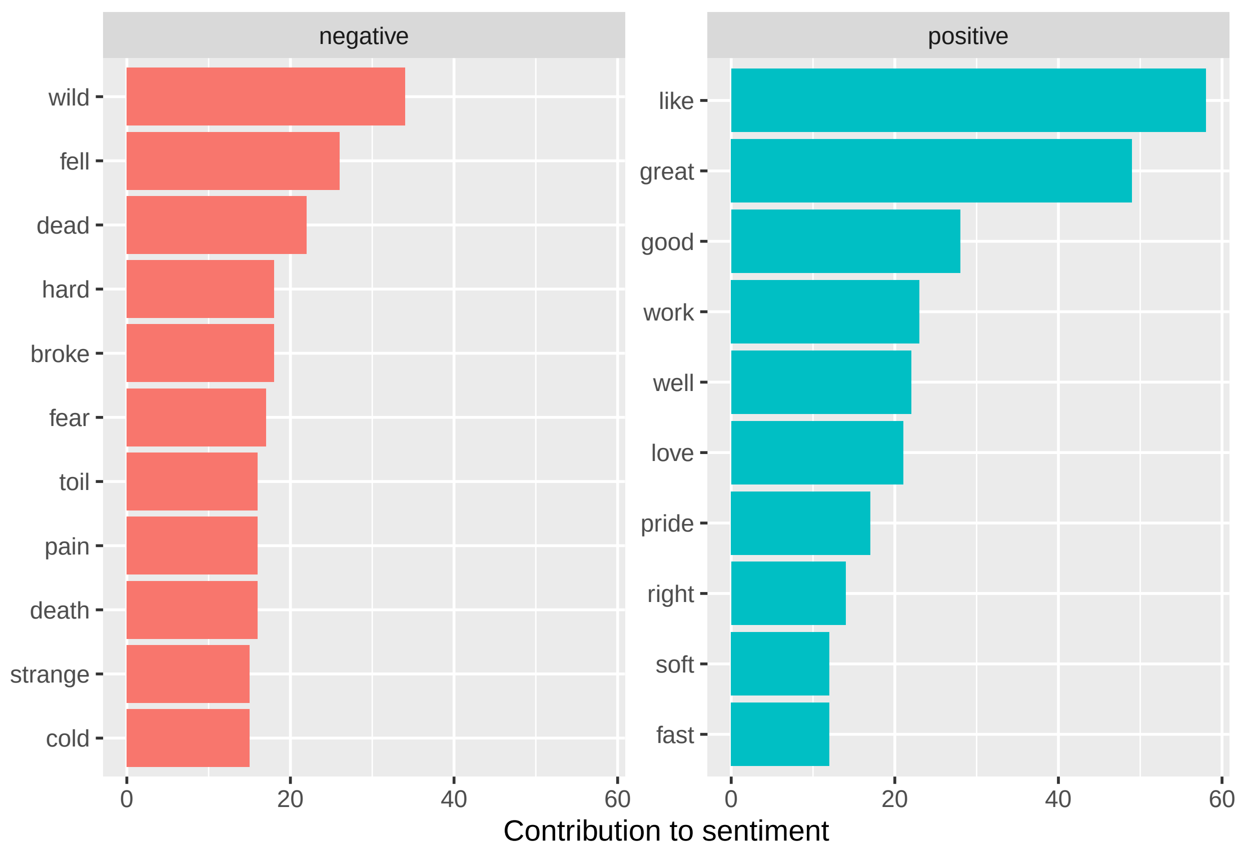
Bing

```
bing_wild <- wild_token %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
bing_wild
```

```
## # A tibble: 892 x 3
##    word  sentiment     n
##    <chr> <chr>     <int>
```

```
##    1 like  positive     58
##    2 great positive     49
##    3 wild  negative     34
##    4 good  positive     28
##    5 fell  negative     26
##    6 work  positive     23
##    7 dead  negative     22
##    8 well  positive     22
##    9 love  positive     21
##   10 broke negative     18
## # i 882 more rows
```

```
bing_wild %>%
  group_by(sentiment) %>%
  slice_max(n, n = 10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(x = "Contribution to sentiment",
       y = NULL)
```



NRC

```r
nrc_wild <- wild_token %>%
  inner_join(get_sentiments("nrc")) %>%
  count(word, sentiment) %>%
  ungroup()
nrc_wild
```

```
## # A tibble: 2,613 x 3
##    word        sentiment        n
##    <chr>       <chr>        <int>
##  1 abandonment anger            1
##  2 abandonment fear             1
##  3 abandonment negative         1
##  4 abandonment sadness          1
##  5 abandonment surprise         1
##  6 ability     positive         2
##  7 absent      negative         1
##  8 absent      sadness          1
##  9 abundance   anticipation     2
## 10 abundance   disgust          2
## # i 2,603 more rows
```

```r
nrc_wild %>%
  group_by(sentiment) %>%
  slice_max(n, n = 10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(x = "Contribution to sentiment",
       y = NULL)
```

The general sentiument seems to be quite neutral. While there looks like a slight negative skew with afinn and a slight positive skew with bing in general the sentiment is relatively neutral with a good balance of negative and 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?

```
wild_token %>%
  group_by(word) %>%
  summarise(frequency = n()) %>%
  left_join(get_sentiments("afinn")) %>%
  filter(!is.na(value)) %>%
  arrange(desc(frequency)) %>%
  top_n(5, wt = frequency)
```

```
## # A tibble: 5 x 3
##   word  frequency value
##   <chr>     <int> <dbl>
## 1 no           95    -1
## 2 like         58     2
## 3 great        49     3
## 4 fire         33    -2
## 5 good         28     3
```

Top Positive:like, great, good

Top Negative: no, fire

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").

```
nowild <- wild %>%
  filter(str_detect(text, "\\bno\\b"))
print(nowild)
```

```
## # 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 within this text seems to be used as a modifier for adjectives and verbs. In some cases when it occurs while it does have a slight negative connotation in others it is actullt positibe such as "no more trouble" and "no matter what the odds." As such it seems that no while greatkly contributing to the calculated sentiment is quite nuetral in most cases and works purley as a modifier.

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.

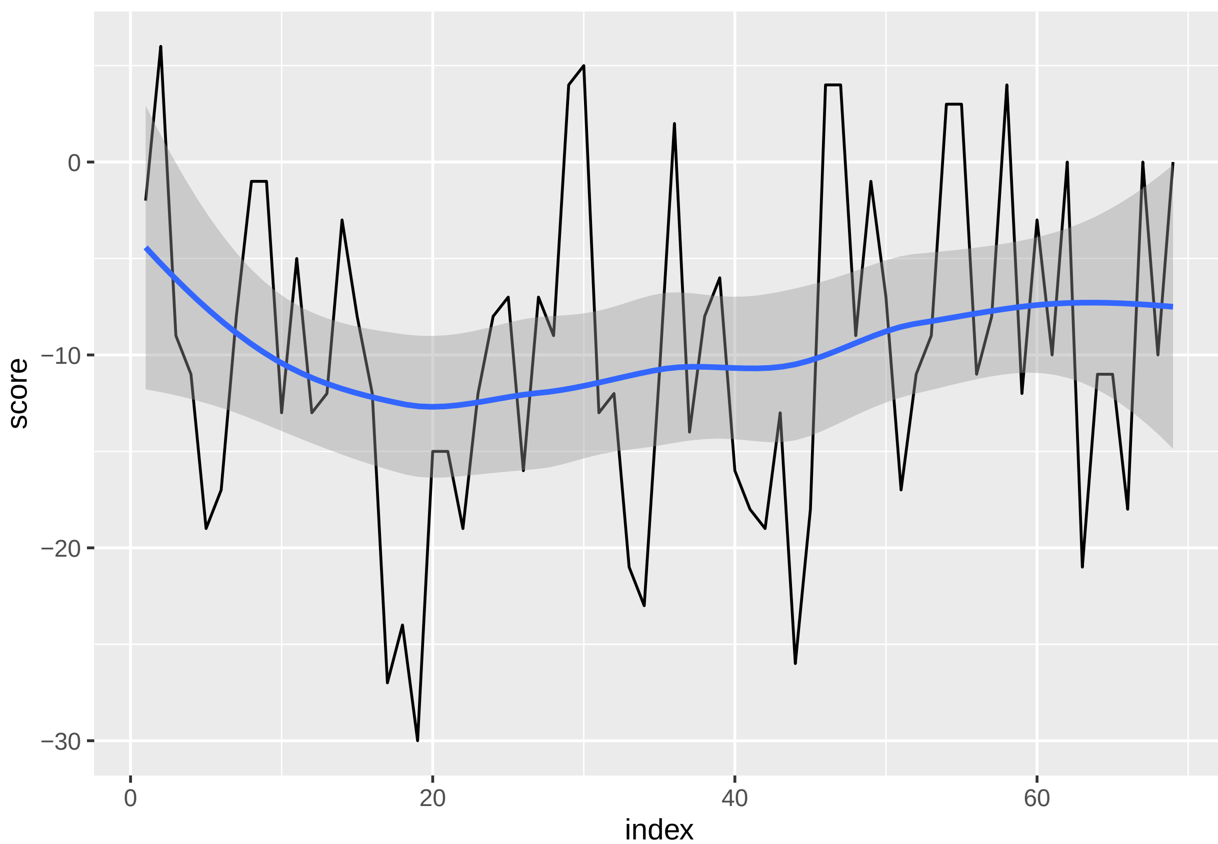
```
wild_time <- wild %>%
  mutate(line = row_number(), index = floor(line/45) + 1)

wild_token <- wild_time %>%
  unnest_tokens(output = word, input = text)

wild_sent <- wild_token %>%
  inner_join(get_sentiments("bing"))
wild_sent
```

```
## # A tibble: 2,351 x 5
##    gutenberg_id  line index word      sentiment
##           <int> <int> <dbl> <chr>     <chr>
##  1          215     6     1 wild      negative
##  2          215    15     1 primitive negative
##  3          215    18     1 won       positive
##  4          215    19     1 toil      negative
##  5          215    20     1 love      positive
##  6          215    26     1 primitive negative
##  7          215    32     1 strain    negative
##  8          215    35     1 trouble   negative
##  9          215    37     1 strong    positive
## 10          215    37     1 warm      positive
## # i 2,341 more rows
```

```
wild_score <- wild_sent %>%
  group_by(index) %>%
  summarise(count = n(), total_pos = sum(sentiment == "positive"), total_neg = sum(sentiment == "negati
  mutate(score = total_pos - total_neg)
```

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

```
wild_score %>%
  ggplot(aes(x = index, y = score)) +
  geom_line() +
  geom_smooth()
```

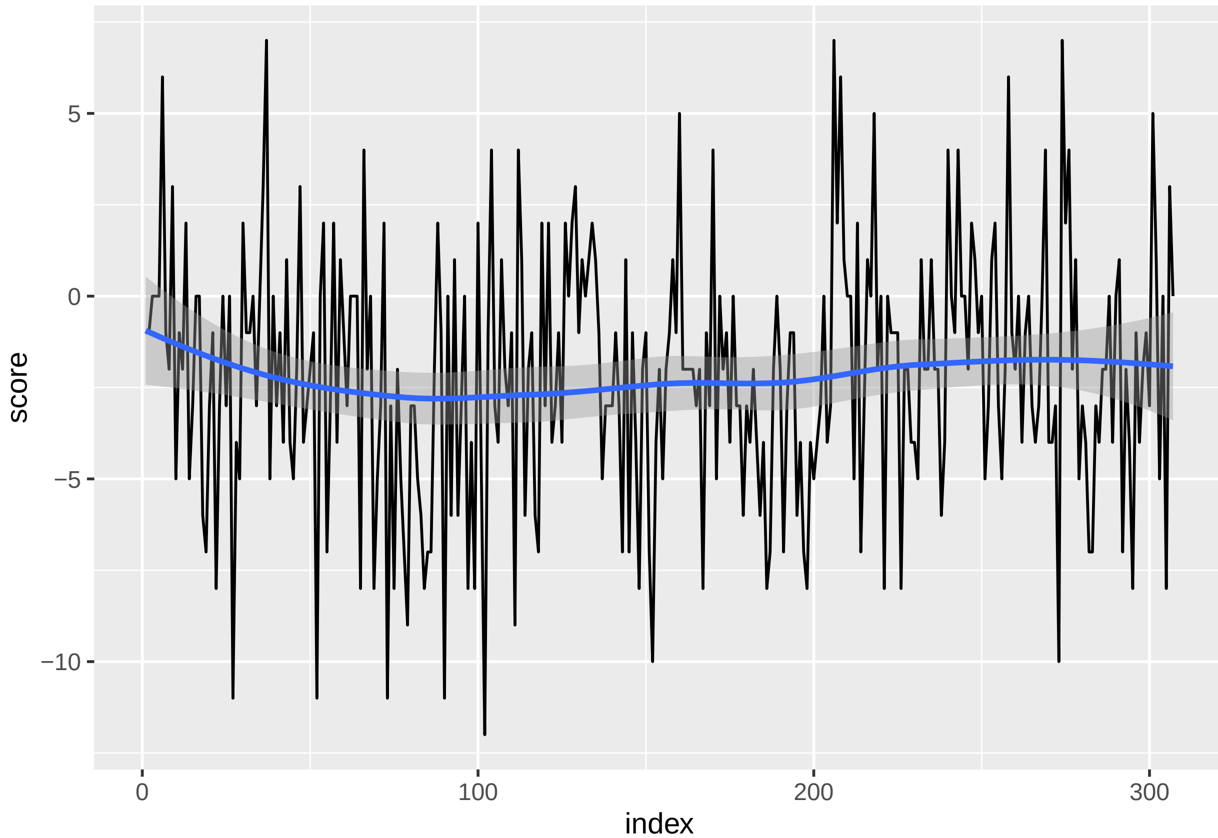


j. The choice of 45 lines per chunk was pretty arbitrary. Try modifying the index value a few times and recreating the plot in i. Based on your plots, what can you conclude about the sentiment of the novel as it progresses?

```
wild_time <- wild %>%
  mutate(line = row_number(), index = floor(line/10) + 1)

wild_token <- wild_time %>%
  unnest_tokens(output = word, input = text)

wild_sent <- wild_token %>%
  inner_join(get_sentiments("bing"))
wild_sent
```

```
## # A tibble: 2,351 x 5
##    gutenberg_id  line index word      sentiment
##           <int> <int> <dbl> <chr>     <chr>
##  1          215     6     1 wild      negative
##  2          215    15     2 primitive negative
##  3          215    18     2 won       positive
##  4          215    19     2 toil      negative
##  5          215    20     3 love      positive
##  6          215    26     3 primitive negative
##  7          215    32     4 strain    negative
##  8          215    35     4 trouble   negative
##  9          215    37     4 strong    positive
## 10          215    37     4 warm      positive
## # i 2,341 more rows
```

```
wild_score <- wild_sent %>%
  group_by(index) %>%
  summarise(count = n(), total_pos = sum(sentiment == "positive"), total_neg = sum(sentiment == "negati
  mutate(score = total_pos - total_neg)

wild_score %>%
  ggplot(aes(x = index, y = score)) +
  geom_line() +
  geom_smooth()
```



Throughout the novel there is a constant and drastic shift from positive to negaiuve in scores. However, in the case of negative, the lowest score is much more than the highest positive score. Additionally while there are near constant fluctuations, when plotted and averagered out the score is fairly nuetral.

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

```r
wild_bigrams <- wild %>%
  unnest_tokens(output = bigram, input = text, token = "ngrams", n = 2) %>%
  na.omit()

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.

```
wild_bigrams %>%
  group_by(bigram) %>%
  summarise(frequency = n()) %>%
  arrange(desc(frequency))
```

```
## # A tibble: 18,907 x 2
##    bigram   frequency
##    <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
```