Lab 7

Insert Name

Math 241, Week 9

```
# Put all necessary libraries here
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
```

```
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.

```
#Hint: Use
top_n(___)
```

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

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

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!)

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

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.

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

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

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

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.

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

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

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.

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?

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").

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

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)
```

```
#Hint: fill = 0 will insert zero instead of NA
pivot_XXX(..., values_fill = 0)
```

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

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?

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

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