Homework 8

Sarah Cannon

Due: 10/29/24

Question 1 (4 points)

Consider the following list of example words, which are a subset of the entire words list.

```
words_example <- words[seq(19, length(words), 50)]
words_example
```

```
##  [1] "afternoon" "bag"       "business"  "colour"    "deep"
##  [6] "end"       "fine"      "ground"    "include"   "let"
## [11] "mile"      "off"       "picture"   "quid"      "sale"
## [16] "sit"       "stuff"     "through"   "village"   "worry"
```

For each part below, give a regular expression that will find the described patterns, and display these patterns using str_view.

(Be sure the regular expressions you give are general enough to work on the whole words data set, even though I'm not asking you to do that here. For example, If I ask for words containing the letter c, you should not simple match the strings "colour" and "include" and "picture", instead your command should be general enough to find any word containing the letter "c", even though there are no other such strings in the words_example list.)

a. **Words that start with the letter b.**

```
str_view(words_example, "^b")
```

```
## [2] │ <b>ag
## [3] │ <b>usiness
```

b. **Words containing the letter d before one of the six most popular letters in the English language, which are e, a, r, i, o, and t.**

```
str_view(words_example, "d[eariot]")
```

```
## [5] │ <de>ep
## [9] │ inclu<de>
```

c. **Words containing the letter d after a character that is not one of the six most popular letters in the English language (e, a, r, i, o, t).**

```
str_view(words_example, "[^eariot]d")
```

```
## [6] │ e<nd>
## [8] │ grou<nd>
## [9] │ incl<ud>e
```

d. **Words that end with the letter "d" but not with the string "nd"**

```
str_view(words_example, "(?<!n)d$")
```

```
## [14] │ qui<d>
```

Question 2 (6 points)

The question refers to the follow string

```
price_info <- "Apple: $51; Banana: $3; Coconut: $123 (all fruits are ripe!)"
```

a. **Use an appropriate str_ function to find/highlight/emphasize the dollar signs that appear in the price_info string.**

b. **Use an appropriate str_ function to find/highlight/emphasize the dollar signs and the digits that follow them in the price_info string, that is, it should highlight $51, $3, and $123. It should not highlight any of the semicolons in the sentence.**

```
str_extract_all(price_info, "\\$\\d+")
```

```
## [[1]]
## [1] "$51"  "$3"   "$123"
```

c. **Use an appropriate str_function to highlight the phrase appearing inside the parentheses in the price_info string (it is fine if the parentheses are also highlighted)**

```
str_extract(price_info, "\\(.*?\\)")
```

```
## [1] "(all fruits are ripe!)"
```

Question 3 (8 points)

In each of the examples below, explain in your own words in 1-2 sentences what pattern(s) the regular expression will match.

While you are welcome to try out these regular expressions on some examples to help you figure out the answers, I'd suggest first trying, as this is better practice for the test, where you won't be able to run code.

a.

```
regex_string <- "\\b[Dd]ogs?\\b"
```

`\\b[Dd]ogs?\\b` matches whole words that are either "Dog" or "dog", optionally followed by an "s" to account for the plural form "Dogs" or "dogs". The `\\b` at both ends ensures that the match is a complete word, not a substring of a longer word.

b.

```
regex_string <- "\\s\\w+\\s"
```

`\\s\\w+\\s` matches a sequence of one or more word characters that are surrounded by whitespace on both sides. This effectively captures whole words that are preceded and followed by spaces.

c.

```
regex_string <- " .+? "
```

`.+?` matches the shortest sequence of characters surrounded by spaces.

d.

```
regex_string <- "(.)\\1"
```

`(.)\\1` matches any character that is immediately followed by the same character.

Question 4 (8 points)

Genomic data can be specified as a string of the characters A, C, G, and T.

a. **Restriction enzymes are used to cut DNA strands at locations when specific sequences are identified. The BsaWI enzyme recognizes sequences that look like WCCGGW, where W could stand for A or T (a *weak* enzyme). Would the BsaWI enzyme find such a sequence in the following DNA example? Use regular expressions to find out. Your answer should include the example DNA sequence with the sequence WCCGGW colored/emphasized/highlighted, if it appears.**

```
DNAex1 <- "ACCGCTAGCTCGGCTAGATAGCTTCCGGAATCGTCGTCTATCGCTAGATCGATCGGCGGGCTCTAGATCGA"
```

```
str_view_all(DNAex1, "[AT]CCGG[AT]")
```

```
## [1] │ ACCGCTAGCTCGGCTAGATAGCT<TCCGGA>ATCGTCGTCTATCGCTAGATCGATCGGCGGGCTCTAGATCGA
```

> b. **In fact, this enzyme can recognize this sequence when it appears either forwards or backwards, that is, it also recognizes the sequence WGGCCW. How many times would the BsaWI enzyme find such a sequence in the following DNA example? Use regular expressions to find out. Your answer should include the example DNA sequence with the sequence(s) WCCGGW and WGGCCW colored/emphasized/highlighted, if they appear.**

```
DNAex2 <- "ACCGCTAGTCGCTCGCAGGCCATGGGCAGCTTCCGGAATCGTCGTCTACTTCGTAGA"
```

```
str_view_all(DNAex2, "[AT]CCGG[AT]|[AT]GGCC[AT]")
```

```
## [1] │ ACCGCTAGTCGCTCGC<AGGCCA>TGGGCAGCT<TCCGGA>ATCGTCGTCTACTTCGTAGA
```

> c. **Fragile X Syndrome can be characterized by having the three nucleotide sequence CGG repeating 200 or more times in a row in a particular gene (see https://www.cdc.gov/ncbddd/fxs/inherited.html (https://www.cdc.gov/ncbddd/fxs/inherited.html), for example). Write a command to check whether the following DNA sequence meets this criteria:**

```
DNAex3 <- "ACGTCAGCCGGCGGCGGCGGCGGCGGCGGCGGCGGGCTAGCTCGCTCCCCCAGCAGCAGCAGCAGCAGCCGGCG
GCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGG
CGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGA
GCGCGGACGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGG
CGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGTACTCTCTCACGCTAGCTCGATCGA
TCGCTCGATCGCCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGC
GGCGGCGGCGGCGGCGGCGGCGGACTGCTCGTAGTCGATGCTAGCTAGTGCTAGTACGCCGGCGGCGGCGGCGGCGGCGGCGGAC
GTACGTAGCCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGACGTACGTGCATGCACGGCGGCGGCGGC
GGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCG
GACGATCGATCGTAGCTAGCTACG"
```

```
str_detect(DNAex3, "(CGG){200,}")
```

```
## [1] FALSE
```

No it does not.

d. **Consider the (artificially generated) genomic patient data in the attached patient_genetic_data.csv file. Import this data, and determine which of the patients have the genetic marker for Fragile X Syndrome, which is CGG repeated more than 200 times in a row.**

```
patient_data <- read_csv("patient_genetic_data.csv")
```

```
## Rows: 6 Columns: 2
## ── Column specification ────────────────────────────────────────────────
## Delimiter: ","
## chr (1): PatientDNA
## dbl (1): PatientID
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Check for the Fragile X Syndrome marker
patient_data <- patient_data %>%
  mutate(has_fxs_marker = str_detect(PatientDNA, "(CGG){200,}"))

patients_with_marker <- patient_data %>%
  filter(has_fxs_marker)

# Print the patient
patients_with_marker
```

```
## # A tibble: 1 × 3
##   PatientID PatientDNA                                    has_fxs_marker
##       <dbl> <chr>                                         <lgl>
## 1         3 TTGTCCGTCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGG… TRUE
```

Patient 3 has the marker.

Question 5 (6 points)

a. **Suppose you have a vector l1 of length 10 and a vector tf1 of 10 True/False values. How would you find the elements of the list l1 that are at the same positions as the TRUE values in the vector tf1? Here is an example you can practice on; a correct command should output the words lizard, bear, and fish. Your command should be extremely simple; no if/else statements are needed.**

```
l1 <- c("cat", "dog", "bird", "lizard", "snake", "ferret", "bear", "turtle", "fish", "ha
mster")
tf1 <- c(FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE, TRUE, FALSE)
```

```
l1[tf1]
```

```
## [1] "lizard" "bear"   "fish"
```

b. **Suppose you have a vector l1 of length 10, a vector tf1 of 10 True/False values describing which words meet some Condition 1, and a vector tf2 of 10 True/False values describing which words meet some Condition 2. How would you find all the words that meet Condition 1 and Condition 2? Hint: First make a list of TRUE/FALSE values describing whether or not the words meet both conditions.**

```
l1 <- c("cat", "dog", "bird", "lizard", "snake", "ferret", "bear", "turtle", "fish", "ha
mster")
tf1 <- c(FALSE, FALSE, TRUE, TRUE, TRUE, FALSE, TRUE, FALSE, TRUE, FALSE)
tf2 <- c(FALSE, FALSE, TRUE, FALSE, TRUE, FALSE, TRUE, FALSE, FALSE, FALSE)
```

```
l1[tf1 & tf2]
```

```
## [1] "bird"  "snake" "bear"
```

c. **In the same setting as the previous part, how would you output all the words that meet Condition 1 or condition 2 (or both)?**

```
l1[tf1 | tf2]
```

```
## [1] "bird"   "lizard" "snake"  "bear"   "fish"
```

Question 6 (6 points)

Consider the following list of example words. For some parts, you may want to use the concepts in the previous question.

```
words_example <- words[seq(19, length(words), 50)]
words_example
```

```
##  [1] "afternoon" "bag"       "business"  "colour"    "deep"
##  [6] "end"       "fine"      "ground"    "include"   "let"
## [11] "mile"      "off"       "picture"   "quid"      "sale"
## [16] "sit"       "stuff"     "through"   "village"   "worry"
```

(Be sure any code you give is general enough to work on the whole words data set, even though I'm not asking you to do that here)

a. **Output all words that don't have the letter o in them. (Do not use str_view; instead, create an object with these words in them and output this object).**

```
words_example[!str_detect(words_example, "o")]
```

```
##  [1] "bag"      "business" "deep"     "end"      "fine"     "include"
##  [7] "let"      "mile"     "picture"  "quid"     "sale"     "sit"
## [13] "stuff"    "village"
```

b. **Output all words that have the letter o in them but do not have the letter f in them. (Do not use str_view; instead, create an object with these words in them and output this object)**

```
words_example[str_detect(words_example, "o") & !str_detect(words_example, "f")]
```

```
## [1] "colour"  "ground"  "through" "worry"
```

Question 7 (8 points)

Consider the words object in R, which has 980 common English words in it. At no point in this question (or ever) should you output a list of length 980, or anything longer than 10.

a. **Find the total number of consonants across all words in the words list, where a consonant is any letter other than a,e,i,o,u. (For the purposes of all parts in this question, you should consider y to be a consonant)**

```
sum(str_count(words, "[^aeiouAEIOU]"))
```

```
## [1] 3174
```

b. **Find the average number of consonants per word across all words in the words list.**

```
mean(str_count(words, "[^aeiouAEIOU]"))
```

```
## [1] 3.238776
```

c. **Find the proportion of words in the words list that have two consecutive consonants in them.**

```
mean(str_detect(words, "[^aeiouAEIOU]{2}"))
```

```
## [1] 0.6785714
```

d. **Count the number of words in the words list that have two consecutive consonants in them.**

```
sum(str_detect(words, "[^aeiouAEIOU]{2}"))
```

```
## [1] 665
```

Question 8 (9 points)

Suppose you want to find all sentences that mention one of the fruits in the fruit list.

```
fruit
```

```
##  [1] "apple"             "apricot"           "avocado"
##  [4] "banana"            "bell pepper"       "bilberry"
##  [7] "blackberry"        "blackcurrant"      "blood orange"
## [10] "blueberry"         "boysenberry"       "breadfruit"
## [13] "canary melon"      "cantaloupe"        "cherimoya"
## [16] "cherry"            "chili pepper"      "clementine"
## [19] "cloudberry"        "coconut"           "cranberry"
## [22] "cucumber"          "currant"           "damson"
## [25] "date"              "dragonfruit"       "durian"
## [28] "eggplant"          "elderberry"        "feijoa"
## [31] "fig"               "goji berry"        "gooseberry"
## [34] "grape"             "grapefruit"        "guava"
## [37] "honeydew"          "huckleberry"       "jackfruit"
## [40] "jambul"            "jujube"            "kiwi fruit"
## [43] "kumquat"           "lemon"             "lime"
## [46] "loquat"            "lychee"            "mandarine"
## [49] "mango"             "mulberry"          "nectarine"
## [52] "nut"               "olive"             "orange"
## [55] "pamelo"            "papaya"            "passionfruit"
## [58] "peach"             "pear"              "persimmon"
## [61] "physalis"          "pineapple"         "plum"
## [64] "pomegranate"       "pomelo"            "purple mangosteen"
## [67] "quince"            "raisin"            "rambutan"
## [70] "raspberry"         "redcurrant"        "rock melon"
## [73] "salal berry"       "satsuma"           "star fruit"
## [76] "strawberry"        "tamarillo"         "tangerine"
## [79] "ugli fruit"        "watermelon"
```

> a. (3 points) **First, use code to make a second list of fruits that makes all the fruits plural. If a fruit ends in y, you make it plural by replacing the y with “ies”; if a fruit is “physalis” or “peach”, you make it plural by adding “es”; and all other fruits are made plural by adding “s”.**

```
pluralize_fruit <- function(fruit_name) {
  if (fruit_name %in% c("physalis", "peach")) {
    return(paste0(fruit_name, "es"))
  } else if (grepl("y$", fruit_name)) {
    return(sub("y$", "ies", fruit_name))
  } else {
    return(paste0(fruit_name, "s"))
  }
}

plural_fruits <- sapply(fruit, pluralize_fruit)
head(plural_fruits)
```

```
##          apple        apricot        avocado         banana
##       "apples"     "apricots"     "avocados"      "bananas"
##    bell pepper       bilberry
## "bell peppers"   "bilberries"
```

b. (4 points) **Use code to make a regular expression that will find any fruits, singular or plural, that don't appear as a substring of a longer word. For this part, you should only look for the fruits when they appear lowercase. Test your regular expression on the sentences list using str_view(); you should find 11 sentences, and 13 total fruits.**

```
fruit_regex <- str_c("\\b(", str_c(fruit, collapse = "|"), "|", str_c(plural_fruits, col
lapse = "|"), ")\\b")

str_view(sentences, fruit_regex)
```

```
##   [6] │ The juice of <lemons> makes fine punch.
## [116] │ He ordered <peach> pie with ice cream.
## [191] │ The fruit of a <fig> tree is <apple> shaped.
## [260] │ Canned <pears> lack full flavor.
## [288] │ Hedge <apples> may stain your hands green.
## [357] │ A ripe <plum> is fit for a king's palate.
## [463] │ The bunch of <grapes> was pressed into wine.
## [464] │ He sent the <figs>, but kept the ripe <cherries>.
## [513] │ Ripe <pears> are fit for a queen's table.
## [644] │ The sky in the west is tinged with <orange> red.
## [705] │ The big red <apple> fell to the ground.
```

c. (2 points) **What if you want to also find fruits if they appear capitalized in a sentence? In addition to str_view(), use a new function we learned this week to change how you find patterns, so that when looking for a match for your fruits regular expression in the sentences list, the case of a letter (upper case vs. lowercase) doesn't matter. You should find one additional sentence compared to the previous part.**

```
fruit_regex_case_insensitive <- regex(fruit_regex, ignore_case = TRUE)

str_view(sentences, fruit_regex_case_insensitive)
```

```
##   [6] │ The juice of <lemons> makes fine punch.
## [116] │ He ordered <peach> pie with ice cream.
## [191] │ The fruit of a <fig> tree is <apple> shaped.
## [260] │ Canned <pears> lack full flavor.
## [288] │ Hedge <apples> may stain your hands green.
## [357] │ A ripe <plum> is fit for a king's palate.
## [383] │ <Grape> juice and water mix well.
## [463] │ The bunch of <grapes> was pressed into wine.
## [464] │ He sent the <figs>, but kept the ripe <cherries>.
## [513] │ Ripe <pears> are fit for a queen's table.
## [644] │ The sky in the west is tinged with <orange> red.
## [705] │ The big red <apple> fell to the ground.
```

Question 9 (18 points)

This question looks for colors that appear in sentences from the sentences list. Consider the following variable, which contains a list of the colors recognized by R (DO NOT print this entire color list in your knitted file).

```
color_list <- colors()
```

a. (2 points) **First, make a new list of colors that removes all digits that appear in a color's name. Print only the first five entries in your new list.**

```
color_list_no_digits <- str_remove_all(color_list, "\\d")

head(color_list_no_digits, 5)
```

```
## [1] "white"        "aliceblue"    "antiquewhite" "antiquewhite"
## [5] "antiquewhite"
```

b. (2 points) **Use code to make a regular expression that will recognize any color from the list you made in part (a). These colors should not appear as substrings of longer words. You do not need to (and should not) print this regular expression in your knitted file. (You can test your regex on the sentences list if you'd like, but this isn't required)**

```
color_regex <- str_c("\\b(", str_c(color_list_no_digits, collapse = "|"), ")\\b")
```

c. (2 points) **Make the sentences list into a tibble with a single column and a row for each sentence. The make the remaining parts easier, make every sentence entirely lowercase.**

```
sentences_tibble <- tibble(sentence = tolower(sentences))
```

d. (3 points) **Make a summary tibble with two columns, one displaying how many sentences contain one of the colors in the colors list (not as part of a longer word) and one displaying how many times a color from the colors list appears in a sentence.**

```
summary_tibble <- sentences_tibble %>%
  mutate(has_color = str_detect(sentence, color_regex),
         color_count = str_count(sentence, color_regex)) %>%
  summarize(sentences_with_color = sum(has_color),
            total_color_appearances = sum(color_count))
summary_tibble
```

```
## # A tibble: 1 × 2
##   sentences_with_color total_color_appearances
##                  <int>                   <int>
## 1                   68                      74
```

e. (3 points) **Make a table that only contains sentences a color from the colors list mentioned. Use str_extract to put which of these color(s) appear in a sentence in a new column. Explain in your own words why this new column contains things like < chr [1] > and < chr [2] >.**

```
color_sentences <- sentences_tibble %>%
  filter(str_detect(sentence, color_regex)) %>%
  mutate(colors_found = str_extract_all(sentence, color_regex))

color_sentences
```

```
## # A tibble: 68 × 2
##    sentence                                           colors_found
##    <chr>                                              <list>
##  1 glue the sheet to the dark blue background.        <chr [1]>
##  2 a rod is used to catch pink salmon.                <chr [2]>
##  3 two blue fish swam in the tank.                    <chr [1]>
##  4 cars and busses stalled in snow drifts.            <chr [1]>
##  5 the navy attacked the big task force.              <chr [1]>
##  6 a wisp of cloud hung in the blue air.              <chr [1]>
##  7 leaves turn brown and yellow in the fall.          <chr [2]>
##  8 the spot on the blotter was made by green ink.     <chr [1]>
##  9 mud was spattered on the front of his white shirt. <chr [1]>
## 10 the sofa cushion is red and of light weight.       <chr [1]>
## # ℹ 58 more rows
```

<chr [n]> indicates that the column contains a list of character vectors, where each vector can have multiple elements (colors found in a sentence).

f. (2 points) **Use an appropriate function to modify your outcome in the previous part so that there is a column displaying the first color found in each sentence and another column displaying the second color that is found in each sentences (if there is one).**

```
color_sentences <- color_sentences %>%
  mutate(first_color = map_chr(colors_found, ~ .x[1]),
         second_color = map_chr(colors_found, ~ .x[2], .default = NA))
color_sentences
```

```
## # A tibble: 68 × 4
##    sentence                            colors_found first_color second_color
##    <chr>                               <list>       <chr>       <chr>
##  1 glue the sheet to the dark blu… <chr [1]>    blue        <NA>
##  2 a rod is used to catch pink sa… <chr [2]>    pink        salmon
##  3 two blue fish swam in the tank. <chr [1]>    blue        <NA>
##  4 cars and busses stalled in sno… <chr [1]>    snow        <NA>
##  5 the navy attacked the big task… <chr [1]>    navy        <NA>
##  6 a wisp of cloud hung in the bl… <chr [1]>    blue        <NA>
##  7 leaves turn brown and yellow i… <chr [2]>    brown       yellow
##  8 the spot on the blotter was ma… <chr [1]>    green       <NA>
##  9 mud was spattered on the front… <chr [1]>    white       <NA>
## 10 the sofa cushion is red and of… <chr [1]>    red         <NA>
## # ℹ 58 more rows
```

g. (2 points) **Use an appropriate function to modify your outcome from part (c) so that when there are multiple colors in a sentence, each ends up in its own row.**

```
color_sentences_long <- color_sentences %>%
  unnest_longer(colors_found)
color_sentences_long
```

```
## # A tibble: 74 × 4
##    sentence                         colors_found first_color second_color
##    <chr>                            <chr>        <chr>       <chr>
##  1 glue the sheet to the dark blu… blue         blue        <NA>
##  2 a rod is used to catch pink sa… pink         pink        salmon
##  3 a rod is used to catch pink sa… salmon       pink        salmon
##  4 two blue fish swam in the tank. blue         blue        <NA>
##  5 cars and busses stalled in sno… snow         snow        <NA>
##  6 the navy attacked the big task… navy         navy        <NA>
##  7 a wisp of cloud hung in the bl… blue         blue        <NA>
##  8 leaves turn brown and yellow i… brown        brown       yellow
##  9 leaves turn brown and yellow i… yellow       brown       yellow
## 10 the spot on the blotter was ma… green        green       <NA>
## # ℹ 64 more rows
```

> h. (2 points) **Make a summary table with two columns: one has all the colors from the colors list that appear in at least one sentence, and another that counts how many times that colors appears. Hint: Use the table you made in the previous part.**

```
color_summary <- color_sentences_long %>%
  count(colors_found, name = "appearance_count")
color_summary
```

```
## # A tibble: 19 × 2
##    colors_found appearance_count
##    <chr>                   <int>
##  1 black                       5
##  2 blue                        8
##  3 brown                       7
##  4 gold                        8
##  5 gray                        2
##  6 green                      10
##  7 navy                        1
##  8 orange                      1
##  9 orchid                      1
## 10 pink                        3
## 11 plum                        1
## 12 purple                      1
## 13 red                        11
## 14 salmon                      1
## 15 snow                        2
## 16 tan                         3
## 17 wheat                       1
## 18 white                       6
## 19 yellow                      2
```

Question 10 (15 points)

For this question, use the sentences tibble you created in 9(c), which should have all sentences in a single column, entirely in lowercase.

a. **For all sentences that include the word "and", swap the word that appears after "and" and the word that appears before "and". For example, now the 8th sentence should read "the hogs were fed chopped garbage and corn." Make sure the column with these swapped words is displayed in your knitted file.**

```
sentences_tibble %>%
  mutate(swapped_sentence = str_replace_all(sentence, "(\\b\\w+\\b)\\s+and\\s+(\\b\\w+
\\b)", "\\2 and \\1"))
```

```
## # A tibble: 720 × 2
##    sentence                                    swapped_sentence
##    <chr>                                       <chr>
##  1 the birch canoe slid on the smooth planks.  the birch canoe slid on …
##  2 glue the sheet to the dark blue background. glue the sheet to the da…
##  3 it's easy to tell the depth of a well.      it's easy to tell the de…
##  4 these days a chicken leg is a rare dish.    these days a chicken leg…
##  5 rice is often served in round bowls.        rice is often served in …
##  6 the juice of lemons makes fine punch.       the juice of lemons make…
##  7 the box was thrown beside the parked truck. the box was thrown besid…
##  8 the hogs were fed chopped corn and garbage. the hogs were fed choppe…
##  9 four hours of steady work faced us.         four hours of steady wor…
## 10 a large size in stockings is hard to sell.  a large size in stocking…
## # ℹ 710 more rows
```

b. **Filter your sentences tibble to find all sentences with a repeated word.**

```
repeated_word_sentences <- sentences_tibble %>%
  filter(str_detect(sentence, "\\b(\\w+)\\b.*\\b\\1\\b"))

repeated_word_sentences
```

```
## # A tibble: 257 × 1
##    sentence
##    <chr>
##  1 the birch canoe slid on the smooth planks.
##  2 glue the sheet to the dark blue background.
##  3 these days a chicken leg is a rare dish.
##  4 the box was thrown beside the parked truck.
##  5 the boy was there when the sun rose.
##  6 the source of the huge river is the clear spring.
##  7 the soft cushion broke the man's fall.
##  8 the salt breeze came across from the sea.
##  9 the girl at the booth sold fifty bonds.
## 10 the small pup gnawed a hole in the sock.
## # ℹ 247 more rows
```

c. **Add a new column to this tibble that says what the repeated word is. Each entry in this new column should only be a single word. Make sure this new column is visible in your knitted file.**

```
repeated_word_sentences <- repeated_word_sentences %>%
  mutate(repeated_word = str_extract(sentence, "\\b(\\w+)\\b(?=.*\\b\\1\\b)"))

repeated_word_sentences
```

```
## # A tibble: 257 × 2
##    sentence                                            repeated_word
##    <chr>                                               <chr>
##  1 the birch canoe slid on the smooth planks.          the
##  2 glue the sheet to the dark blue background.         the
##  3 these days a chicken leg is a rare dish.            a
##  4 the box was thrown beside the parked truck.         the
##  5 the boy was there when the sun rose.                the
##  6 the source of the huge river is the clear spring. the
##  7 the soft cushion broke the man's fall.              the
##  8 the salt breeze came across from the sea.           the
##  9 the girl at the booth sold fifty bonds.             the
## 10 the small pup gnawed a hole in the sock.            the
## # ℹ 247 more rows
```

d. **Add another new column to your tibble that includes all text between the two instances of the repeated word. For example, for the first sentence, this new column should include ” birch canoe slid on “. Make sure this new column is visible in your knitted file.**

```
repeated_word_sentences <- repeated_word_sentences %>%
  mutate(text_between = str_extract(sentence, "\\b(\\w+)\\b.*\\b\\1\\b"))

repeated_word_sentences
```

```
## # A tibble: 257 × 3
##    sentence                                   repeated_word text_between
##    <chr>                                      <chr>         <chr>
##  1 the birch canoe slid on the smooth planks. the           the birch c…
##  2 glue the sheet to the dark blue backgroun… the           the sheet t…
##  3 these days a chicken leg is a rare dish.   a             a chicken l…
##  4 the box was thrown beside the parked truc… the           the box was…
##  5 the boy was there when the sun rose.       the           the boy was…
##  6 the source of the huge river is the clear… the           the source …
##  7 the soft cushion broke the man's fall.     the           the soft cu…
##  8 the salt breeze came across from the sea.  the           the salt br…
##  9 the girl at the booth sold fifty bonds.    the           the girl at…
## 10 the small pup gnawed a hole in the sock.   the           the small p…
## # ℹ 247 more rows
```

e. **Consider the sentence “the cat and the dog slept and played together.” This sentence contains two repeated words, “the” and “and”. Explain in your own words why your regular expression (or any correct regular expression finding pairs of repeated words) will only find one match in this sentence.**

A regular expression for repeated words will usually find only the first pair. Regular expressions match the first occurrence of a pattern and stop, unless designed to find multiple matches. So, it will match “the” first and stop, not finding “and” as a repeated word.

Question 11 (6 points)

a. (3 points) **Which words appear most frequently in the sentences list? Beginning with the tibble you made in 9(c), use the boundary() function with an appropriate argument and other functions we’ve learned to answer this question.**

Hint: First make a single column that contains all the words, and then use function(s) from previous weeks.

```
word_frequencies <- sentences_tibble %>%
  mutate(words = str_split(sentence, boundary("word"))) %>%
  unnest_longer(words) %>%
  count(words, sort = TRUE)

word_frequencies
```

```
## # A tibble: 1,890 × 2
##    words     n
##    <chr> <int>
##  1 the     744
##  2 a       213
##  3 of      132
##  4 to      123
##  5 and     118
##  6 in       87
##  7 is       81
##  8 was      67
##  9 on       59
## 10 with     52
## # ℹ 1,880 more rows
```

b. (3 points) **Which characters appear most frequently in the sentences list?**

```
character_frequencies <- sentences_tibble %>%
  mutate(characters = str_split(sentence, boundary("character"))) %>%
  unnest_longer(characters) %>%
  count(characters, sort = TRUE)

character_frequencies
```

```
## # A tibble: 31 × 2
##    characters     n
##    <chr>      <int>
##  1 " "         5025
##  2 "e"         3050
##  3 "t"         2350
##  4 "a"         1751
##  5 "h"         1655
##  6 "s"         1585
##  7 "o"         1562
##  8 "r"         1356
##  9 "n"         1220
## 10 "i"         1204
## # ℹ 21 more rows
```

Question 12 (6 points)

> **As we get farther into this semester, rather than providing you with specific ethics resources, I'm going to start asking you to do some of your own research! This more closely mimics what you might need to do in the future, where you won't have a professor guiding your exploration and learning.**

> a. (3 points) **Using any resources you'd like, research what the field of *natural language processing* entails, give some examples of tasks that natural language processing tools can accomplish, and explain how natural language processing is different from what we've been learning this week, which is looking for and working with patterns in strings using regular expressions. Your answer should be written in your own words. Include a list of the sources you consulted (urls are fine).**

NLP is a field of AI focused on enabling computers to understand and process human language.

Examples of NLP Tasks: - Sentiment Analysis: Identifying emotions in text. - Machine Translation: Translating text between languages. - Named Entity Recognition (NER): Detecting names of people, places, etc. - Speech Recognition: Converting speech to text. - Text Summarization: Creating summaries of documents.

Difference from Regular Expressions: - NLP handles complex language understanding, while regular expressions are for simple pattern matching. - NLP understands context and semantics. regular expressions do not do that but could be used to help with that.

Sources: - Wikipedia - NLP (https://en.wikipedia.org/wiki/Natural_language_processing) - Book Genome Project (https://bookgenomeproject.org/)

> b. (3 points) **Again using any resources you'd like, give some examples of ethical issues and concerns that may arise in natural language processing. Some places you may want to get started are https://github.com/uclanlp/awesome-fairness-papers (https://github.com/uclanlp/awesome-fairness-papers) or https://aclweb.org/aclwiki/Ethics_in_NLP (https://aclweb.org/aclwiki/Ethics_in_NLP). Be specific about some ethical benefits and ethical harms that could result from natural language processing.**

NLP presents several ethical issues and concerns. One major concern is bias in NLP models, which can arise from biased training data, leading to unfair or discriminatory outcomes. For example, language models might perpetuate stereotypes or exhibit gender and racial biases, impacting decisions in hiring or law enforcement. Another issue is privacy, as NLP systems often process sensitive personal data, raising concerns about data security and consent.

NLP can also enhance accessibility, such as through speech-to-text services for individuals with disabilities, and improve efficiency in information retrieval and customer service. However, ethical harms include the potential for misuse in surveillance or spreading misinformation. Its important to remember that "language is also a political instrument, though, and an instrument of power." (Hovy, Dirk and Spruit, Shannon L., 2016)

The ACM Code of Ethics emphasizes the importance of fairness, transparency, and accountability in technology development, which are crucial in addressing these ethical challenges in NLP.

Sources: - "The Social Impact of Natural Language Processing", Hovy, Dirk and Spruit, Shannon L., 2016 (https://aclanthology.org/P16-2096/) - ACM Code of Ethics (https://www.acm.org/code-of-ethics)