STA 445 S24 Assignment 5

Connor Phillips

3/19/2024

```
library(tidyverse)
```

Problem 1

For the following regular expression, explain in words what it matches on. Then add test strings to demonstrate that it in fact does match on the pattern you claim it does. Do at least 4 tests. Make sure that your test set of strings has several examples that match as well as several that do not. Make sure to remove the `eval=FALSE` from the R-chunk options.

a. This regular expression matches: Matches strings that contain a lowercase "a"

```
strings <- c("house", "leaves", "Ramble", "Soggy Rat", "RASCAL")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'a') )
```

```
##      string result
## 1     house  FALSE
## 2    leaves   TRUE
## 3    Ramble   TRUE
## 4 Soggy Rat   TRUE
## 5    RASCAL  FALSE
```

b. This regular expression matches: Matches strings containing both the lowercase characters "a" and "b"

```
        strings <- c("Annabelle", "alphabet", "benny", "gargoyle")
        data.frame( string = strings ) %>%
          mutate( result = str_detect(string, 'ab') )
```

```
##      string result
## 1 Annabelle   TRUE
## 2  alphabet   TRUE
## 3     benny  FALSE
## 4  gargoyle  FALSE
```

c. This regular expression matches: Matches strings containing the lowercase characters "a" or "b"

```
        strings <- c("Annabelle", "alphabet", "benny", "gargoyle")
        data.frame( string = strings ) %>%
          mutate( result = str_detect(string, '[ab]') )
```

```
##      string result
## 1 Annabelle   TRUE
## 2  alphabet   TRUE
## 3     benny   TRUE
## 4  gargoyle   TRUE
```

d. This regular expression matches: Matches strings beginning with lowercase characters "a" or "b"

```
        strings <- c("abel", "alphabet", "benny", "gargoyle")
        data.frame( string = strings ) %>%
          mutate( result = str_detect(string, '^[ab]') )
```

```
##     string result
## 1     abel   TRUE
## 2 alphabet   TRUE
## 3    benny   TRUE
## 4 gargoyle  FALSE
```

e. This regular expression matches: Matches strings with one or more digits, any white space, and contains an upper or lowercase "a"

```
        strings <- c("123 Anna", "4567 Matt", "8 Tom", "9and10 Astrid")
        data.frame( string = strings ) %>%
          mutate( result = str_detect(string, '\\d+\\s[aA]') )
```

```
##          string result
## 1      123 Anna   TRUE
## 2     4567 Matt  FALSE
## 3         8 Tom  FALSE
## 4 9and10 Astrid   TRUE
```

f. This regular expression matches: Matches strings with one or more digits, any or zero white space, and contains an upper or lowercase "a".

```
        strings <- c("33 Anna", "444anna", "anna 33", "anna444")
        data.frame( string = strings ) %>%
          mutate( result = str_detect(string, '\\d+\\s*[aA]') )
```

```
##    string result
## 1 33 Anna   TRUE
## 2 444anna   TRUE
## 3 anna 33  FALSE
## 4 anna444  FALSE
```

g. This regular expression matches: Matches any character (wildcard) that occurs zero or more times.

```
    strings <- c("1234", "abcd", "1 a 2 b", "", "Tim\n Robinson")
    data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '.*') )
```

```
##           string result
## 1           1234   TRUE
## 2           abcd   TRUE
## 3        1 a 2 b   TRUE
## 4                  TRUE
## 5 Tim\n Robinson   TRUE
```

h. This regular expression matches: Matches strings with two characters, digits, etc (but not whitespace) followed by the characters 'bar'.

```
    strings <- c("11bar", "aabar", "1_bar", "  bar", "bar11", "baraa")
    data.frame( string = strings ) %>%
      mutate( result = str_detect(string, '^\\w{2}bar') )
```

```
##   string result
## 1  11bar   TRUE
## 2  aabar   TRUE
## 3  1_bar   TRUE
## 4    bar  FALSE
## 5  bar11  FALSE
## 6  baraa  FALSE
```

i. This regular expression matches: Matches two different string patterns using the OR operator. The first matches any string containing 'foo.bar'. The second matches any string with two characters, digits, etc (but not whitespace) followed by the characters 'bar' (same as above.) The string only has to match one of these requirements.

```
    strings <- c("11bar", "bar11", "aaafoo.baraaa", "food.bar")
    data.frame( string = strings ) %>%
      mutate( result = str_detect(string, '(foo\\.bar)|(^\\w{2}bar)') )
```

```
##          string result
## 1         11bar   TRUE
## 2         bar11  FALSE
## 3 aaafoo.baraaa   TRUE
## 4      food.bar  FALSE
```

Problem 2

The following file names were used in a camera trap study. The S number represents the site, P is the plot within a site, C is the camera number within the plot, the first string of numbers is the YearMonthDay and the second string of numbers is the HourMinuteSecond.

```
file.names <- c( 'S123.P2.C10_20120621_213422.jpg',
                 'S10.P1.C1_20120622_050148.jpg',
               'S187.P2.C2_20120702_023501.jpg')
```

Produce a data frame with columns corresponding to the `site`, `plot`, `camera`, `year`, `month`, `day`, `hour`, `minute`, and `second` for these three file names. So we want to produce code that will create the data frame:

```
 Site Plot Camera Year Month Day Hour Minute Second
 S123   P2    C10 2012    06  21   21     34     22
  S10   P1     C1 2012    06  22   05     01     48
  S187   P2     C2 2012    07  02   02     35     01
```

```
file.names = str_replace_all(file.names, pattern = ".jpg", replacement = "")
file.names = str_replace_all(file.names, pattern = "_", replacement = ".")
prob2data = data.frame(file.names)
prob2data = separate(prob2data, col=file.names, into = c("Site", "Plot", "Camera", "Date", "Time"), sep
  mutate(Year = str_sub(Date, start=1, end=4)) %>%
  mutate(Month = str_sub(Date, start=5, end=6)) %>%
  mutate(Day = str_sub(Date, start=7, end=8)) %>%
  mutate(Hour = str_sub(Time, start=1, end=2)) %>%
  mutate(Minute = str_sub(Time, start=3, end=4)) %>%
  mutate(Second = str_sub(Time, start=5, end=6))
prob2data$Date = NULL
prob2data$Time = NULL
prob2data
```

```
##   Site Plot Camera Year Month Day Hour Minute Second
## 1 S123   P2    C10 2012    06  21   21     34     22
## 2  S10   P1     C1 2012    06  22   05     01     48
## 3 S187   P2     C2 2012    07  02   02     35     01
```

3. The full text from Lincoln's Gettysburg Address is given below. Calculate the mean word length *Note: consider 'battle-field' as one word with 11 letters*).

***Should get a mean word length of 4.23(ish)

```
Gettysburg <- 'Four score and seven years ago our fathers brought forth on this
continent, a new nation, conceived in Liberty, and dedicated to the proposition
that all men are created equal. Now we are engaged in a great civil war, testing
whether that nation, or any nation so conceived and so dedicated, can long
endure. We are met on a great battle-field of that war. We have come to dedicate
a portion of that field, as a final resting place for those who here gave their
lives that that nation might live. It is altogether fitting and proper that we
should do this. But, in a larger sense, we can not dedicate -- we can not
consecrate -- we can not hallow -- this ground. The brave men, living and dead,
who struggled here, have consecrated it, far above our poor power to add or
detract. The world will little note, nor long remember what we say here, but it
can never forget what they did here. It is for us the living, rather, to be
dedicated here to the unfinished work which they who fought here have thus far
so nobly advanced. It is rather for us to be here dedicated to the great task
remaining before us -- that from these honored dead we take increased devotion
to that cause for which they gave the last full measure of devotion -- that we
here highly resolve that these dead shall not have died in vain -- that this
nation, under God, shall have a new birth of freedom -- and that government of
the people, by the people, for the people, shall not perish from the earth.'
```

```r
getty1 = str_replace_all(Gettysburg, pattern = "\\. ", replacement = " ")
getty2 = str_replace_all(getty1, pattern = "\\.", replacement = "")
getty3 = str_replace_all(getty2, pattern = "-- ", replacement = "")
getty4 = str_replace_all(getty3, pattern = "-", replacement = "")
getty5 = str_replace_all(getty4, pattern = ",", replacement = "")
getty6 = str_trim(str_replace_all(getty5, pattern = "\n", replacement = " "))

getty_split = str_split(getty6, pattern = " ")
mean_length = mean(str_length(getty_split[[1]]))
mean_length
```

```
## [1] 4.239852
```