# STA 445 S24 Assignment 5

Sophia Kubisiak

03/21/2023

```
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: This expression is looking for words that contain the letter ‘a’ in them.

```
        strings <- c("apple" , "fruit" , "banana" , "blueberry")
        data.frame( string = strings ) %>%
          mutate( result = str_detect(string, 'a') )
```

```
##      string result
## 1     apple   TRUE
## 2     fruit  FALSE
## 3    banana   TRUE
## 4 blueberry  FALSE
```

b. This regular expression matches: This expression is looking for words that contain the ‘ab’ in them, in that order.

```
        strings <- c("banana" , "abba" , "apple" , "absolutely")
        data.frame( string = strings ) %>%
          mutate( result = str_detect(string, 'ab') )
```

```
##       string result
## 1     banana  FALSE
## 2       abba   TRUE
## 3      apple  FALSE
## 4 absolutely   TRUE
```

c. This regular expression matches: This expression is looking for words that contain the letters ‘a’ and/or ‘b’ in them, and the ‘a’ and ‘b’ don’t necessarily have to be together.

```
        strings <- c("apple" , "banana" , "fruit" , "kiwi")
        data.frame( string = strings ) %>%
          mutate( result = str_detect(string, '[ab]') )
```

```
##   string result
## 1  apple   TRUE
## 2 banana   TRUE
## 3  fruit  FALSE
## 4   kiwi  FALSE
```

d. This regular expression matches: This expression is looking for words that DO not NOT contain (so that do contain) the letters ‘a’ and/or ‘b’ in them. (based on my result, but if we moved ˆ inside like so: ‘[1]’ , then it looks for words that do not contain a or b b)

```
        strings <- c("apple" , "banana" , "fruit" , "kiwi")
        data.frame( string = strings ) %>%
          mutate( result = str_detect(string, '^[ab]') )
```

```
##   string result
## 1  apple   TRUE
## 2 banana   TRUE
## 3  fruit  FALSE
## 4   kiwi  FALSE
```

e. This regular expression matches: This expression is looking for a string that starts with any digit, followed by any white space, and the followed by ‘aA’ or any character that starts with ‘aA’.

```
        strings <- c("2 aApples" , "1 aA-bananas" , "fruit" , "3 graApes" , "4 aA")
        data.frame( string = strings ) %>%
          mutate( result = str_detect(string, '\\d+\\s[aA]') )
```

```
##         string result
## 1    2 aApples   TRUE
## 2 1 aA-bananas   TRUE
## 3        fruit  FALSE
## 4    3 graApes  FALSE
## 5         4 aA   TRUE
```

f. This regular expression matches: This expression is looking for a string that starts with any digit, followed by any white space, and the followed by ‘aA’ or any character that starts with ‘aA’ that has zero or more repetitions of the previous letter.

```
        strings <- c("10 aApples" , "5 aApples" , "fruit aA 2" , "3 kiwi")
        data.frame( string = strings ) %>%
          mutate( result = str_detect(string, '\\d+\\s*[aA]') )
```

```
##       string result
## 1 10 aApples   TRUE
## 2  5 aApples   TRUE
## 3 fruit aA 2  FALSE
## 4     3 kiwi  FALSE
```

[1] ˆab

g. This regular expression matches: This expression is looking for any character any number of times.

```
        strings <- c("2 aa" , "" , "banana" , "fruit" , ".-4")
        data.frame( string = strings ) %>%
      mutate( result = str_detect(string, '.*') )
```

```
##   string result
## 1   2 aa   TRUE
## 2          TRUE
## 3 banana   TRUE
## 4  fruit   TRUE
## 5    .-4   TRUE
```

h. This regular expression matches: This expression is looking for the beginning of the that has any alphanumeric character with 2 repetitions follow by the string ‘bar’ in that order.

```
        strings <- c("22bar" , "aabar" , "hi" , "bar")
        data.frame( string = strings ) %>%
          mutate( result = str_detect(string, '^\\w{2}bar') )
```

```
##   string result
## 1  22bar   TRUE
## 2  aabar   TRUE
## 3     hi  FALSE
## 4    bar  FALSE
```

i. This regular expression matches: This expression is looking for the string ‘foo’ followed by ‘.bar’ OR for the beginning of the that has any alphanumeric character with 2 repetitions follow by the string ‘bar’ in that order.

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

```
##    string result
## 1   aabar   TRUE
## 2 foo.bar   TRUE
## 3      hi  FALSE
## 4     foo  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
```

```
str_split(file.names , pattern = "[._]")%>%

  map_dfr(setNames, c("Site", "Plot", "Camera", "Date", "Time", "ext")) %>%
  separate(Date, into = c("Year", "Month", "Day"), sep = c(4, 6)) %>%
  separate(Time, into = c("Hour", "Minute", "Second"), sep = c(2, 4)) %>%
  select(-ext)
```

```
## # A tibble: 3 x 9
##   Site  Plot  Camera Year  Month Day   Hour  Minute Second
##   <chr> <chr> <chr>  <chr> <chr> <chr> <chr> <chr>  <chr>
## 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*).

```
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.'
```

```
NewGB1 <- str_replace_all(Gettysburg , pattern = "[,]" , "")
NewGB2 <- str_replace_all(NewGB1 , pattern = "[.]" , "")
NewGB3 <- str_replace_all(NewGB2 , pattern = "[\n]" , "")
NewGB4 <- str_replace_all(NewGB3 , pattern = "[-]" , "")
NewGB5 <- str_squish(NewGB4)

words <- str_split(NewGB5, pattern = " ")

mean(str_length((words[[1]])))
```

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