STA 445 Assignment #4

Chip Haskins

2024-10-18

Exercise 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. Make sure that your test set of strings has several examples that match as well as several that do not. Show at least two examples that return TRUE and two examples that return FALSE. *If you copy the Rmarkdown code for these exercises directly from my source pages, make sure to remove the `eval=FALSE` from the R-chunk headers.*

Here is an example of what a solution might look like.

q) This regular expression matches:

Any string that contains the lower-case letter "a".

```
strings <- c('Adel', 'Mathematics', 'able', 'cheese')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'a') )
```

```
##         string result
## 1         Adel  FALSE
## 2 Mathematics   TRUE
## 3         able   TRUE
## 4       cheese  FALSE
```

Please complete the questions below.

a) This regular expression matches:

Any string with the substring "ab", specifically the lowercase 'a' followed by a lowercase 'b'.

```
strings <- c("laboratory", "ABle", "thread", "stab")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'ab') )
```

```
##        string result
## 1 laboratory   TRUE
## 2       ABle  FALSE
## 3     thread  FALSE
## 4       stab   TRUE
```

b) This regular expression matches:

Any string that contains *either* a lowercase 'a' or a lowercase 'b'.

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

```
##   string result
## 1  shiny  FALSE
## 2  crate   TRUE
## 3 banana   TRUE
## 4 dABBle  FALSE
```

c) This regular expression matches:

Any string that *begins* with either a lowercase ‘a’ or a lowercase ‘b’.

```
strings <- c("stab", "able", "Abet", "basic")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^[ab]') )
```

```
##   string result
## 1   stab  FALSE
## 2   able   TRUE
## 3   Abet  FALSE
## 4  basic   TRUE
```

d) This regular expression matches:

Any string that contains one or more digits followed by a white space character and either a lower- or uppercase ‘a’.

```
strings <- c("8204 Avery Drive", "Stung by 340 bees", "2 apples")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s[aA]') )
```

```
##              string result
## 1  8204 Avery Drive   TRUE
## 2 Stung by 340 bees  FALSE
## 3          2 apples   TRUE
```

e) This regular expression matches:

Any string that contains one or more digits followed by zero or more white space characters and either a lower- or uppercase ‘a’.

```
strings <- c("123a", "Apple", "12 arithmetic operations", "8 sunflowers")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s*[aA]') )
```

```
##                     string result
## 1                     123a   TRUE
## 2                    Apple  FALSE
## 3 12 arithmetic operations   TRUE
## 4             8 sunflowers  FALSE
```

f) This regular expression matches:

Any string that contains zero or more characters of any kind.

```
strings <- c("", "ansdjpgn;akds2738,&^", "especially", "88.43")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '.*') )
```

```
##                  string result
## 1                         TRUE
## 2 ansdjpgn;akds2738,&^   TRUE
## 3           especially   TRUE
## 4                88.43   TRUE
```

g) This regular expression matches:

Any string that begins with 2 alphanumeric characters followed by the substring "bar".

```
strings <- c("lumbar", "foobaz", "rebar", "unbar")
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^\\w{2}bar') )
```

```
##   string result
## 1 lumbar  FALSE
## 2 foobaz  FALSE
## 3  rebar   TRUE
## 4  unbar   TRUE
```

h) This regular expression matches:

Any string that contains one or both of two patterns. The first pattern matches the string "foo.bar", and the second pattern matches any string that begins with 2 alphanumeric characters followed by the substring "bar".

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

```
##    string result
## 1 foo.bar   TRUE
## 2   rebar   TRUE
## 3 foo_bar  FALSE
## 4  lumbar  FALSE
```

Exercise 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
```

```
data <- data.frame(filenames = file.names) %>%
  separate_wider_regex(filenames, patterns=c(
    Site = "S\\d+", ".",
    Plot = "P\\d", ".",
    Camera = "C\\d+", ".",
    Year = "\\d{4}",
    Month = "\\d{2}",
    Day = "\\d{2}", ".",
    Hour = "\\d{2}",
    Minute = "\\d{2}",
    Second = "\\d{2}", ".jpg"
  ))

data
```

```
## # 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
```

Exercise 3

The full text from Lincoln's Gettysburg Address is given below. It has been provided in a form that includes lots of different types of white space. Your goal is to calculate the mean word length of Lincoln's Gettysburg Address! *Note: you may consider 'battle-field' as one word with 11 letters or as two words 'battle' and 'field'. The first option a bit more difficult and technical!.*

```
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.'
```

```
str_extract_all(Gettysburg, "\\w+", simplify = TRUE) %>% str_length %>% mean()
```

```
## [1] 4.224265
```

Optional Exercises

Exercise 4

Variable names in R may be any combination of letters, digits, period, and underscore. However, variables within a data frame may not start with a digit and if they start with a period, they must not be followed by a digit.

```
strings <- c('foo15', 'Bar', '.resid', '_14s',
             '99_Bottles', '.9Arggh', 'Foo!','HIV Rate')
```

The first four are valid variable names, but the last four are not.

a) First write a regular expression that determines if the string starts with a character (upper or lower case) or underscore and then is followed by zero or more numbers, letters, periods or underscores. *Notice below the use of start/end of string markers. This is important so that we don't just match somewhere in the middle of the variable name.*

```
data.frame( string=strings ) %>%
  mutate( result = str_detect(string, '^[A-Za-z_][\\w\\.]*$' ))
```

```
##       string result
## 1      foo15   TRUE
## 2        Bar   TRUE
## 3     .resid  FALSE
## 4       _14s   TRUE
## 5 99_Bottles  FALSE
## 6    .9Arggh  FALSE
## 7       Foo!  FALSE
## 8   HIV Rate  FALSE
```

b) Modify your regular expression so that the first group could be either `[a-zA-Z_]` as before or it could be a period followed by letters or an underscore.

```
data.frame( string=strings ) %>%
  mutate( result = str_detect(string, '^\\.?[A-Za-z_][\\w\\.]*$' ))
```

```
##       string result
## 1      foo15   TRUE
## 2        Bar   TRUE
## 3     .resid   TRUE
## 4       _14s   TRUE
## 5 99_Bottles  FALSE
## 6    .9Arggh  FALSE
## 7       Foo!  FALSE
## 8   HIV Rate  FALSE
```