STA 445 - Assignment 4

Richard McCormick

2023-10-16

R Markdown

(1) Exercise 1 – 7 pts. In 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. 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.

- **Complete any 7 of the 9 scenarios from a – i. Ensure the scenarios are clearly identifiable.**
- **The string <- c() at the start of each chunk is where you will enter several strings to show that the 2 lines of code underneath do what you say they are doing.**
- **Your test string will need to include several expressions return a TRUE result and a FALSE result from the code. For example, if you state the code checks for the letter "e", your test string needs to have some expressions with the letter e , to return TRUE result, and some expressions without the letter e, to show the code returns FALSE when no "e" is found.**
- **If you need help coming up with good test strings (particularly as the scenarios get more complex), please ask.**

a. This regular expression matches: Any string with the lowercase letter 'a'.

```
strings <- c( 'do', 're', 'mi', 'so', 'fa', 'ti', 'lo', 'BIGMAC', 'a', 'A' )
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'a') )
```

```
##    string result
## 1      do  FALSE
## 2      re  FALSE
## 3      mi  FALSE
## 4      so  FALSE
## 5      fa   TRUE
## 6      ti  FALSE
## 7      lo  FALSE
## 8  BIGMAC  FALSE
## 9       a   TRUE
## 10      A  FALSE
```

b. This regular expression matches: Any string with lowercase letters 'a' and 'b' directly adjacent to each other, in alphabetical order.

```
# This regular expression matches:  Insert your answer here...
strings <- c( 'absolute', 'ABSOLUTE', 'barn', 'stacked bricks' )
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'ab') )
```

```
##           string result
## 1       absolute   TRUE
## 2       ABSOLUTE  FALSE
## 3           barn  FALSE
## 4 stacked bricks  FALSE
```

c. This regular expression matches: Any string which has the letters 'a' or 'b' in lowercase anywhere in the string.

```
strings <- c( 'absolute', 'ABSOLUTE', 'barn', 'stacked bricks', 'big mac',
              'BIG MAC', 'little mac', 'biggie smalls', 'eminem', 'canada' )
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '[ab]') )
```

```
##            string result
## 1        absolute   TRUE
## 2        ABSOLUTE  FALSE
## 3            barn   TRUE
## 4  stacked bricks   TRUE
## 5         big mac   TRUE
## 6         BIG MAC  FALSE
## 7      little mac   TRUE
## 8   biggie smalls   TRUE
## 9          eminem  FALSE
## 10         canada   TRUE
```

d. This regular expression matches: Any string which has 'a' or 'b' at the start of the line.

```
strings <- c( 'absolute', 'ABSOLUTE', 'barn', 'stacked bricks', 'big mac',
              'little mac', 'c a b', 'b a c', 'canada', 'biden' )
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^[ab]') )
```

```
##            string result
## 1        absolute   TRUE
## 2        ABSOLUTE  FALSE
## 3            barn   TRUE
## 4  stacked bricks  FALSE
## 5         big mac   TRUE
## 6      little mac  FALSE
```

```
## 7           c a b  FALSE
## 8           b a c   TRUE
## 9          canada  FALSE
## 10          biden   TRUE
```

e. This regular expression matches: A digit / number of any length, followed by a space, followed by either a lower or uppercase letter 'a', followed by any other additional text or digits.

```
strings <- c( '1 a', 'one a', '2 B', '31 A', '3a', '1234 a b', '1234a',
              '1234 a testing string', '1234 ABSOLUTE', '123 abc street unit 4' )
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s[aA]') )
```

```
##                    string result
## 1                     1 a   TRUE
## 2                    one a  FALSE
## 3                      2 B  FALSE
## 4                     31 A   TRUE
## 5                       3a  FALSE
## 6                  1234 a b   TRUE
## 7                    1234a  FALSE
## 8  1234 a testing string   TRUE
## 9           1234 ABSOLUTE   TRUE
## 10 123 abc street unit 4   TRUE
```

f. This regular expression matches: A digit / number of any length, followed by any number of white spaces (including zero), followed be a lower or capital letter 'a', with any amount of text or digits afterwards.

```
strings <- c( '1 a', 'one a', '2 B', '31 A', '3a', '1234 a b', '1234a',
              '1234 a testing string', '1234 ABSOLUTE', '123  atab' )
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s*[aA]') )
```

```
##                    string result
## 1                     1 a   TRUE
## 2                    one a  FALSE
## 3                      2 B  FALSE
## 4                     31 A   TRUE
## 5                       3a   TRUE
## 6                  1234 a b   TRUE
## 7                    1234a   TRUE
## 8  1234 a testing string   TRUE
## 9           1234 ABSOLUTE   TRUE
## 10              123  atab   TRUE
```

g. This regular expression matches: Any charcter of any length (including zero). This will match literally anything which is a string.

```
strings <- c( 'file.exe', '.exe', 'exe', 'exe.exe', 'file.', 'file exe', '',
              '123', '+-123' )
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '.*') )
```

```
##     string result
## 1 file.exe   TRUE
## 2     .exe   TRUE
## 3      exe   TRUE
## 4  exe.exe   TRUE
## 5    file.   TRUE
## 6 file exe   TRUE
## 7            TRUE
## 8      123   TRUE
## 9    +-123   TRUE
```

h. This regular expression matches: Exactly 2 non-alphanumeric characters, followed by the letters 'bar' (lowercase), with no spaces.

```
strings <- c( '%', '%%', '++bar', 'bar', '--', '__bar', '+=BAR' )
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^\\w{2}bar') )
```

```
##   string result
## 1      %  FALSE
## 2     %%  FALSE
## 3  ++bar  FALSE
## 4    bar  FALSE
## 5     --  FALSE
## 6  __bar   TRUE
## 7  +=BAR  FALSE
```

i. This regular expression matches: Any string which is either EXACTLY 'foo.bar', OR two non-alphanumeric characters followed by 'bar' with no space (section h).

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

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

```
## 5  foo.abar  FALSE
## 6          %  FALSE
## 7         %%  FALSE
## 8      __bar   TRUE
## 9        bar  FALSE
## 10  foo.%ar  FALSE
```

(2) Exercise 2 – 3 pts 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.frame( Site = str_extract( file.names, '\\w\\d*' ) ) %>%
  mutate( Plot = str_extract( file.names, 'P\\d*' ),
          Camera = str_extract( file.names, 'C\\d*' ),
          Year = str_extract( file.names, '\\d{4}' ),
          Month = str_extract( file.names, '(?<=\\d{4})(\\d{2})' ),
          Day = str_extract( file.names, '(?<=\\d{6})(\\d{2})' ),
          Hour = str_extract( file.names, '(?<=\\d{8}_)(\\d{2})' ),
          Minute = str_extract( file.names, '(?<=\\d{8}_\\d{2})(\\d{2})' ),
          Second = str_extract( file.names, '(?<=\\d{8}_\\d{4})(\\d{2})' ) )
```

```
##   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) Exercise 3 – 3 pts. 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.'
```

```
## Remove all the punctuation, newline characters, and double hyphens
## keep all words of any length, and all hyphenated words
strings_df <- data.frame( pog =
                          str_extract_all( Gettysburg,
                                           "[^\\.\\,\\\n\\ \\--]\\w*(\\-+\\w*|w*)" ) )

colnames( strings_df ) <- c( "Strings" )

head( strings_df )
```

```
##   Strings
## 1    Four
## 2   score
## 3     and
## 4   seven
## 5   years
## 6     ago
```

```
length = 0
sum = 0

for ( word in strings_df$Strings )
{
  length = length + nchar( word )
  sum = sum + 1
```

```
}

avg <- round( length / sum, 3 )

print( paste( "The average length of the words in the Gettysburg Address is:",
              avg ) )
```

```
## [1] "The average length of the words in the Gettysburg Address is: 4.244"
```

(4) Turned in by the due date/time – 2 points.