445-3

2023-10-17

Exercises

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.*

    a) This regular expression matches: detects the string for the letter 'a' then it returns TRUE if the string contains the letter 'a' (alone or in a string and is lowercase) and 'FALSE' if the string does not. # strings <- c('a') # result = 'TRUE'

strings <- c('cat')

result = 'TRUE'

strings <- c('dog')

result = 'FALSE'

strings <- c('d')

result = 'FALSE'

strings <- c('A')

result = 'FALSE'

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

```
##   string result
## 1      A  FALSE
```

```
b) This regular expression matches: detects the string for 'ab' and returns
TRUE if the string contains the 'ab' (alone or in a string and is
lowercase) and 'FALSE' if the string does not.
```

strings <- c(‘ab’)

result = ‘TRUE’

strings <- c(‘Cab’)

result = ‘TRUE’

strings <- c(‘a’)

result = ‘FALSE’

strings <- c(‘Truck’)

result = ‘FALSE’

strings <- c(‘Able’)

result = ‘FALSE’

```
strings <- c('Able')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'ab') )
```

```
##   string result
## 1   Able  FALSE
```

c) This regular expression matches:detects the string for ’a’ and/or ’b’ in any order, with repetition, but is case sensitive

strings <- c('ab')

result = 'TRUE'

strings <- c('a')

result = 'TRUE'

strings <- c('A')

result = 'FALSE'

strings <- c('bark')

result = 'TRUE'

strings <- c('aab')

result = 'TRUE'

strings <- c('Bad')

result = 'TRUE'

```
strings <- c('Bad')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '[ab]') )
```

```
##   string result
## 1    Bad   TRUE
```

d) This regular expression matches:detects whether 'a' or 'b' is located at the beginning of the string.

strings <- c('bed')

result = 'TRUE'

strings <- c('rad')

result = 'FALSE'

strings <- c('Bed')

result = 'FALSE'

strings <- c('able')

result = 'TRUE'

strings <- c('aab')

result = 'TRUE'

```
strings <- c('aab')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^[ab]') )
```

```
##   string result
## 1    aab   TRUE
```

e) This regular expression matches:in order, detects any number of digits, then one white space, then detects the string for 'a' and/or 'A' in any order, with repetition and returns 'TRUE' if all satisfied and 'FALSE' otherwise.

strings <- c('0 aA')

result = 'TRUE'

strings <- c(' aA0')

result = 'FALSE'

strings <- c('01 aA')

result = 'TRUE'

strings <- c('01 aA')

result = 'FALSE'

strings <- c('01 aAAAAA')

result = 'TRUE'

```
strings <- c('01 aAAAAA')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s[aA]') )
```

```
##      string result
## 1 01 aAAAAA   TRUE
```

f) This regular expression matches:in order, detects any number of digits, then 0 or more white spaces(placed anywhere in string), then detects the string for 'a' and/or 'A' in any order, with repetition and returns 'TRUE' if all satisfied and 'FALSE' otherwise.

strings <- c('0 aA')

returns = 'TRUE'

strings <- c('0aA')

returns = 'TRUE'

strings <- c(' 0aA')

returns = 'TRUE'

strings <- c(' aA0')

returns = 'FALSE'

strings <- c('01 aA')

returns = 'TRUE'

```
strings <- c('01 aA')
data.frame( string = strings ) %>% # any # of digs,0+ whitespaces,
  mutate( result = str_detect(string, '\\d+\\s*[aA]') )
```

```
##   string result
## 1  01 aA   TRUE
```

g) This regular expression matches: detects any number and arrangement of any character of any kind including no input.

strings <- c(' ')

result = 'TRUE'

strings <- c('a')

result = 'TRUE'

strings <- c('0')

result = 'TRUE'

strings <- c(' ')

result = 'TRUE'

strings <- c('@')

result = 'TRUE'

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

```
##   string result
## 1      @   TRUE
```

h) This regular expression matches:detects string first for a number/letter (non case sensitive), then for a second letter/number, and then for the string 'bar' and returns 'TRUE' if satisfied and 'FALSE' otherwise.

strings <- c('ttbar')

returns = 'TRUE'

strings <- c('tttbar')

returns = 'FALSE'

strings <- c('bartt')

returns = 'FALSE'

strings <- c('ttar')

returns = 'FALSE'

strings <- c('twbar')

returns = 'TRUE'

strings <- c('TWbar')

returns = 'TRUE'

strings <- c('23bar')

returns = 'TRUE'

```
strings <- c('23bar')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^\\w{2}bar') )
```

```
##   string result
## 1  23bar   TRUE
```

```
i) This regular expression matches:detects first for the string 'foo', then
'.', then 'bar' or two alphanumerics first followed by the string 'bar' and
returns 'TRUE' if satisfied and 'FALSE' otherwise.
```

strings <- c('foobar')

returns = 'FALSE'

strings <- c('aabar')

returns = 'TRUE'

strings <- c('foo.bar')

returns = 'TRUE'

strings <- c('foo3bar')

returns = 'FALSE'

strings <- c('AAbar')

returns = 'TRUE'

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

```
##   string result
## 1  AAbar   TRUE
```

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
```

```
output <- data.frame(file.names = file.names) %>%
  mutate(file.names = str_remove_all(file.names, '\\.jpg')) %>%
  # remove '.jpg' in each string
  mutate(file.names = str_replace_all(file.names, '\\.', '_')) %>%
  # replace all chars with an underscore
  separate(file.names, sep='_', into=c('Site','Plot','Camera','YMD','HMS'))
  # separate df at '_' into 'Site','Plot','Camera','YMD','HMS'
Year = str_sub(output$YMD, start=1, end=4)  # positions of dates and time in
Month = str_sub(output$YMD, start=5, end=6) # each string by categoory
Day = str_sub(output$YMD, start=7, end=8)
Hour = str_sub(output$HMS, start=1, end=2)
Minute = str_sub(output$HMS, start=3, end=4)
Second = str_sub(output$HMS, start=5, end=6)
cbind(output[-1*c(4,5)], Year, Month, Day, Hour, Minute, Second)
```

```
##   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
```

```
# bind previously established date/time cols but removes prior 'YMS' & 'HMS'
```

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.'

output = NULL
Gettysburg2 = str_remove_all(Gettysburg,'(\\.|\\,|\\-\\-|\\-)') # remove chars
Gettysburg3 = str_split(Gettysburg2, pattern='\\s+') # split string at spaces
for( i in Gettysburg3 ) {
  output <- str_length(i) # length of all words saved to 'output'
```

```
}
mean(output) # mean word length
```

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