assignment10

Noah Plant

2024-10-25

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".

```
library(stringr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v purrr     1.0.2
## v forcats   1.0.0     v readr     2.1.5
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
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 strings that contain the lowercase ab in that order.

```
strings <- c('hello','after','before','about','ABOUT','abore','harbor')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'ab') )
```

```
##   string result
## 1  hello  FALSE
## 2  after  FALSE
## 3 before  FALSE
## 4  about   TRUE
## 5  ABOUT  FALSE
## 6  abore   TRUE
## 7 harbor  FALSE
```

b) This regular expression matches:

Any string that contains either a lower case a or a lower case b or both.

```
strings <- c('ABout','About','aBout','car','harbor','core')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '[ab]') )
```

```
##   string result
## 1  ABout  FALSE
## 2  About   TRUE
## 3  aBout   TRUE
## 4    car   TRUE
## 5 harbor   TRUE
## 6   core  FALSE
```

c) This regular expression matches:

Any string that starts with either a lowercase a or a lower case b.

```
strings <- c('About','ABout','Harbor','core','about','bore','after')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^[ab]') )
```

```
##   string result
## 1  About  FALSE
## 2  ABout  FALSE
## 3 Harbor  FALSE
## 4   core  FALSE
## 5  about   TRUE
## 6   bore   TRUE
## 7  after   TRUE
```

d) This regular expression matches:

Any string with the pattern: any digit number then a space then a word that starts with either upper or lowercase a.

```
strings <- c('12 abc','123 Abc','123abc','123 ','123 Harbor','12345  acorn','1 after')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s[aA]') )
```

```
##         string result
## 1       12 abc   TRUE
## 2      123 Abc   TRUE
## 3       123abc  FALSE
## 4         123   FALSE
## 5   123 Harbor  FALSE
## 6 12345  acorn  FALSE
## 7      1 after   TRUE
```

e) This regular expression matches:

This string matches string in pattern: any digit number then any number of spaces including 0, then a word that starts with either a or A.

```
strings <- c('123    After','123 After','123a','123B','123Ba','A','1A','12345 before')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s*[aA]') )
```

```
##         string result
## 1 123    After   TRUE
## 2    123 After   TRUE
## 3         123a   TRUE
## 4         123B  FALSE
## 5        123Ba  FALSE
## 6            A  FALSE
## 7           1A   TRUE
## 8 12345 before  FALSE
```

f) This regular expression matches:

Any string that contains any number of characters.

```
strings <- c('apple','aaron','becase','!@#%$%&^%$','','/','654654','','','','')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '.*') )
```

```
##        string result
## 1       apple   TRUE
## 2       aaron   TRUE
## 3      becase   TRUE
## 4  !@#%$%&^%$   TRUE
## 5               TRUE
## 6           /   TRUE
## 7      654654   TRUE
## 8               TRUE
## 9               TRUE
## 10              TRUE
## 11              TRUE
```

g) This regular expression matches:

Any string that starts with two alphanumeric values meaning any letter or a number0-9. And then the word bar in lowercase.

```
strings <- c('12bar','11bar','11rab','3abar','%6bar','//bar','12BAR')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^\\w{2}bar') )
```

```
##   string result
## 1  12bar   TRUE
## 2  11bar   TRUE
## 3  11rab  FALSE
## 4  3abar   TRUE
## 5  %6bar  FALSE
## 6  //bar  FALSE
## 7  12BAR  FALSE
```

h) This regular expression matches:

Any string that contains "foo.bar" or starts with any 2 alphanumeric combo and then proceeds with lowercase bar.

```
strings <- c('foobar','a2bar','foo.bar','12foo.bar','12foo')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '(foo\\.bar)|(^\\w{2}bar)') )
```

```
##      string result
## 1    foobar  FALSE
## 2     a2bar   TRUE
## 3   foo.bar   TRUE
## 4 12foo.bar   TRUE
## 5     12foo  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
```

```
fileData<-data.frame(input=file.names) %>%
  mutate(Site=str_extract(input,'\\w+')) %>%
  mutate(Plot=str_extract(input,'P\\d+')) %>%
  mutate(Camera=str_extract(input,'C\\d+')) %>%
  mutate(year=str_extract(input,'_\\w+')) %>% mutate(year=str_remove_all(year,'_')) %>% # I formatted t
  mutate(month=str_extract(year,'\\d{4,6}')) %>% mutate(month=str_remove(month,'\\d{4}'))%>%
  mutate(day=str_extract(year,'\\d{4,8}')) %>% mutate(day=str_remove(day,'\\d{6}'))%>%
  mutate(hour=str_extract(year,'\\d{4,10}')) %>% mutate(hour=str_remove(hour,'\\d{8}'))%>%
  mutate(minute=str_extract(year,'\\d{4,12}')) %>% mutate(minute=str_remove(minute,'\\d{10}'))%>%
  mutate(second=str_extract(year,'\\d{4,14}')) %>% mutate(second=str_remove(second,'\\d{12}'))%>%
  mutate(year=str_extract(year,'\\d{4}'))

  fileData<-subset(fileData,select=-input)

fileData
```

```
##   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
```

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!.*

```
library(stringr)
library(dplyr)
library(tidyverse)
```

```
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.'

myString<-str_replace_all(Gettysburg,pattern='\n',replacement='') # Removes the new lines

myString<-str_replace_all(myString,pattern='[^[:alnum:] ]',replacement=' ') # Removes all special chara


myString<-str_replace_all(myString,pattern='\\s+',replacement=' ')# Removes all double spaces

myString<-str_trim(myString,"right")



myString<-str_split(myString,pattern=' ')
strValue<-(myString[[1]]) # Accessing my string list

lengths<-str_length(strValue)

meanVal<-mean(lengths)

meanVal
```

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