Module11

Joe Vargovich

10/3/2020

Exercise 1 - Regular expression analysis

```
#a. This matches strings that contain a.
strings <- c('a','b','ba','c', 'cccba')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'a') )
```

```
##   string result
## 1      a   TRUE
## 2      b  FALSE
## 3     ba   TRUE
## 4      c  FALSE
## 5  cccba   TRUE
```

```
#b. This matches strings with the substring ab in them
strings <- c('cc','ab', 'b', 'abba', 'ba')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, 'ab') )
```

```
##   string result
## 1     cc  FALSE
## 2     ab   TRUE
## 3      b  FALSE
## 4   abba   TRUE
## 5     ba  FALSE
```

```
#c. This matches strings that contain a or b.
strings <- c('abab', 'ab', 'cb', 'cba', 'cab', 'd', 'cc', 'cfcds')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '[ab]') )
```

```
##   string result
## 1   abab   TRUE
## 2     ab   TRUE
## 3     cb   TRUE
## 4    cba   TRUE
## 5    cab   TRUE
## 6      d  FALSE
## 7     cc  FALSE
## 8  cfcds  FALSE
```

```
#d. This matches strings that begin with a or b.
strings <- c('a', 'b', 'cb', 'bc', 'c', 'cd')
data.frame( string = strings ) %>%
```

```
  mutate( result = str_detect(string, '^[ab]') )
```

```
##   string result
## 1      a   TRUE
## 2      b   TRUE
## 3     cb  FALSE
## 4     bc   TRUE
## 5      c  FALSE
## 6     cd  FALSE
```

```
#e. Begins with one or more digits and has a space separating them from an a or A.
strings <- c('ab', '1a', '1 a', '121221323123434325324 a', 'a 1221', '12 Aa', '12 b')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s[aA]') )
```

```
##                     string result
## 1                       ab  FALSE
## 2                       1a  FALSE
## 3                      1 a   TRUE
## 4 121221323123434325324 a   TRUE
## 5                    a 1221  FALSE
## 6                     12 Aa   TRUE
## 7                      12 b  FALSE
```

```
#f. Begins with one or more digits and has 0 or more spaces seperating digits from a or A.
strings <- c('1212134a', '12231 a', '121323 A', '12213434A', '223443 B', 'aasdf 12', '12 aa')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '\\d+\\s*[aA]') )
```

```
##      string result
## 1  1212134a   TRUE
## 2   12231 a   TRUE
## 3  121323 A   TRUE
## 4 12213434A   TRUE
## 5  223443 B  FALSE
## 6  aasdf 12  FALSE
## 7     12 aa   TRUE
```

```
#g. Captures anything as . is a wildcard general capture. Can have any length due to the Kleene star op
strings <- c('Whatever', 'I', 'Want', '', 'dasfasdfsdffsdafdsfsadafds')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '.*') )
```

```
##                       string result
## 1                   Whatever   TRUE
## 2                          I   TRUE
## 3                       Want   TRUE
## 4                              TRUE
## 5 dasfasdfsdffsdafdsfsadafds   TRUE
```

```
#h. Begins with two alphanumeric characters and ends with bar.
strings <- c('aabar', 'dasdfbar', 'bbar', 'bbbar', 'xyz', 'xyzbar', 'xybar')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^\\w{2}bar') )
```

```
##     string result
## 1    aabar   TRUE
```

```
## 2 dasdfbar  FALSE
## 3     bbar  FALSE
## 4    bbbar   TRUE
## 5      xyz  FALSE
## 6   xyzbar  FALSE
## 7    xybar   TRUE
```

```
#i. Captures foo.bar OR two alphanumeric chars followed by bar
strings <- c('foo.bar', 'foobar', 'aabar', 'aabar', 'bar.foo', 'sdaf', 'gxxxbar', 'gzbar')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '(foo\\.bar)|(^\\w{2}bar)') )
```

```
##    string result
## 1 foo.bar   TRUE
## 2  foobar  FALSE
## 3   aabar   TRUE
## 4   aabar   TRUE
## 5 bar.foo  FALSE
## 6    sdaf  FALSE
## 7 gxxxbar  FALSE
## 8   gzbar   TRUE
```

Exercise 2 - File name parsing for camera images

```
file.names <- c( 'S123.P2.C10_20120621_213422.jpg',
                 'S10.P1.C1_20120622_050148.jpg',
                 'S187.P2.C2_20120702_023501.jpg')
#Produce a dataframe with columns corresponding to the site, plot, camera, year, month, day, hour, minu

#I tried to do it more elegantly with string.split, but I couldn't get the numbers formatted correctly.
splitStrings = str_replace_all(file.names, pattern='\\.', replacement="_")

for(i in splitStrings){
  finalSplit = str_split(splitStrings, pattern='_')
}

df = ldply(finalSplit, data.frame)

df2 = do.call(rbind.data.frame, finalSplit)

#Brute forceish, but this is how I will build it using the substring range function
site_vec = NULL
plot_vec = NULL
camera_vec = NULL
year_vec = NULL
month_vec = NULL
day_vec = NULL
hour_vec = NULL
minute_vec = NULL
second_vec = NULL

#Build sites column
site_vec = site_vec %>%
    append(str_sub(file.names[1], start=1, end=4)) %>%
```

```
    append(str_sub(file.names[2], start=1, end=3)) %>%
    append(str_sub(file.names[3], start=1, end=4))

site_vec
```

```
## [1] "S123" "S10"  "S187"
```

```
#Build plot_vec
plot_vec = plot_vec %>%
    append(str_sub(file.names[1], start=6, end=7)) %>%
    append(str_sub(file.names[2], start=5, end=6)) %>%
    append(str_sub(file.names[3], start=6, end=7))

plot_vec
```

```
## [1] "P2" "P1" "P2"
```

```
#Build Camera
camera_vec = camera_vec %>%
    append(str_sub(file.names[1], start=9, end=11)) %>%
    append(str_sub(file.names[2], start=8, end=9)) %>%
    append(str_sub(file.names[3], start=9, end=10))

camera_vec
```

```
## [1] "C10" "C1"  "C2"
```

```
#Build Year
year_vec = year_vec %>%
    append(str_sub(file.names[1], start=13, end=16)) %>%
    append(str_sub(file.names[2], start=11, end=14)) %>%
    append(str_sub(file.names[3], start=12, end=15))

year_vec
```

```
## [1] "2012" "2012" "2012"
```

```
#Build Month
month_vec = month_vec %>%
    append(str_sub(file.names[1], start=17, end=18)) %>%
    append(str_sub(file.names[2], start=15, end=16)) %>%
    append(str_sub(file.names[3], start=16, end=17))

month_vec
```

```
## [1] "06" "06" "07"
```

```
#Build Day
day_vec = day_vec %>%
    append(str_sub(file.names[1], start=19, end=20)) %>%
    append(str_sub(file.names[2], start=17, end=18)) %>%
    append(str_sub(file.names[3], start=18, end=19))

day_vec
```

```
## [1] "21" "22" "02"
```

```r
#Build Hour
hour_vec = hour_vec %>%
    append(str_sub(file.names[1], start=22, end=23)) %>%
    append(str_sub(file.names[2], start=20, end=21)) %>%
    append(str_sub(file.names[3], start=21, end=22))

hour_vec
```

```
## [1] "21" "05" "02"
```

```r
#Build Minute
minute_vec = minute_vec %>%
    append(str_sub(file.names[1], start=24, end=25)) %>%
    append(str_sub(file.names[2], start=22, end=23)) %>%
    append(str_sub(file.names[3], start=23, end=24))

minute_vec
```

```
## [1] "34" "01" "35"
```

```r
#Build Second
second_vec = second_vec %>%
    append(str_sub(file.names[1], start=26, end=27)) %>%
    append(str_sub(file.names[2], start=24, end=25)) %>%
    append(str_sub(file.names[3], start=25, end=26))

second_vec
```

```
## [1] "22" "48" "01"
```

```r
result = data.frame(
  Site = site_vec,
  Plot = plot_vec,
  Camera = camera_vec,
  Year = year_vec,
  Month = month_vec,
  Day = day_vec,
  Hour = hour_vec,
  Minute = minute_vec,
  Second = second_vec
)

result
```

```
##   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 - Gettysberg address parsing

```r
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.'

#Pipe the gettysburg string and remove invalid characters
Gettysburg %>%
  str_remove_all(pattern = '-') %>%
  str_remove_all(pattern = '\\.|,|\\?|!') %>%
  str_split(pattern='\\s+') %>%
  .[[1]] %>%
    str_length() %>%
    mean()
```

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

Exercise 4 - Creating a variable name regular expression.

```
#a. Write a regex to determine if a string starts with a character (upper or lower case)
# or underscore and then is followed by zero or more numbers, letters, periods, or underscores.
strings <- c('foo15', 'Bar', '.resid', '_14s',
             '99_Bottles', '.9Arggh', 'Foo!','HIV Rate')
data.frame( string = strings ) %>%
  mutate( result = str_detect(string, '^[\\w|\\W|_][0-9a-zA-Z\\._]*$') )
```

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

```
#b. Modify regex so the first group can only be [a-zA-Z_]
strings <- c('foo15', 'Bar', '.resid', '_14s',
             '99_Bottles', '.9Arggh', 'Foo!','HIV Rate')
data.frame( string = strings ) %>%
```

```r
  mutate( result = str_detect(string, '^[a-zA-Z_][0-9a-zA-Z\\._]*$') )
```

```
##       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
```