More Regular Expressions

Stat 220

Bastola

February 01 2022

Quantifiers and Special Characters

Preceding characters are matched

- `*` = 0 or more
- `?` = 0 or 1
- `+` = 1 or more
- `{n}` = exactly n times

Matching character types

- `\\d` = digit
- `\\s` = white space
- `\\w` = word
- `\\t` = tab
- `\\n` = newline

Repitition using ?

```
aboutMe <- c("my SSN 536-76-9423 and my age is 55")
```

```
str_view_all(aboutMe, "\\s\\d?") # space followed by 0 or 1 digit
```

```
my SSN 536-76-9423 and my age is 55
```

Repitition using +

```
aboutMe <- c("my SSN 536-76-9423 and my age is 55")
```

```
str_view_all(aboutMe, "\\s\\d+")  # space followed by 1 or more digits
```

```
my SSN 536-76-9423 and my age is 55
```

Repitition using *

```r
aboutMe <- c("my SSN 536-76-9423 and my age is 55")
```

```r
str_view_all(aboutMe, "\\s\\d*")  # space followed by 0 or more digits
```

```
my SSN 536-76-9423 and my age is 55
```

More quantifiers

> useful when you want to match a pattern a specific number of times

- `{n, }` = n or more times
- `{, m}` = at most m times
- `{n, m}` = between n & m times

Alternatives

useful for matching patterns more flexibly

- `[abc]` = one of **a**, **b**, or **c**
- `[e-z]` = a letter from **e** to **z**
- `[^abc]` = anything other than **a**, **b**, or **c**

Recap: Extract Substrings

Using `str_sub()`

```
cc <- "Carleton College"
str_sub(cc, start = 1, end = 8)
[1] "Carleton"
```

```
str_sub(cc, 10) # end defaults to the last character
[1] "College"
```

```
# match the elements of each vector for positions
str_sub(cc, c(1, 10), c(8, 16))
[1] "Carleton" "College"
```

Extract substrings

Can also extract the end..

```
cc <- "Carleton College"
str_sub(cc, -4)
[1] "lege"
```

```
str_sub(cc, -10, -4)
[1] "on Coll"
```

Your Turn 1

Please git clone the repository on advanced string manipulations to your local folder.

Make changes to the code to ...

1. Isolate the last letter of every name
2. Create a logical variable that displays whether the last letter is one of "a", "e", "i", "o", "u", or "y".
3. Use a weighted mean to calculate the proportion of children whose name ends in a vowel by year (see `?weighted.mean`)
4. and then display the results as a line plot.

04:00

str_count()

Tells you how many times a pattern appears in each entry of a string/character vector

```
days <- rep(c("Monday", "Tuesday", "Wednesday",
              "Thursday", "Friday", "Saturday", "Sunday"),2)
```

```
#number of times the letter "e" appears in each entry
days %>% str_count("e")
 [1] 0 1 2 0 0 0 0 0 1 2 0 0 0 0
```

```
# number of times the letter "a" or "e" appears in each entry
days %>% str_count("[ae]")
 [1] 1 2 3 1 1 2 1 1 2 3 1 1 2 1
```

Your Turn 2

Determine how many baby names in 2017 contain "ders".

```
babynames %>%
  filter(year == 2017) %>%
 head(4)
# A tibble: 4 × 5
   year sex   name         n    prop
  <dbl> <chr> <chr>    <int>   <dbl>
1  2017 F     Emma     19738 0.0105
2  2017 F     Olivia   18632 0.00994
3  2017 F     Ava      15902 0.00848
4  2017 F     Isabella 15100 0.00805
```

02:00

str_extract()

pull all set of values matching the specified pattern

```
name_phone <- c("Moly Robins: 250-999-8878",
                "Ali Duluth: 416-908-2044",
                "Eli Mitchell: 204-192-9829",
                "May Flowers: 250-209-7047")
```

```
str_extract(name_phone, "[:alpha:]*")
[1] "Moly" "Ali"  "Eli"  "May"
```

```
str_extract(name_phone, "[:digit:]{3}-[:digit:]{3}-[:digit:]{4}")
[1] "250-999-8878" "416-908-2044" "204-192-9829" "250-209-7047"
```

str_extract_all()

pull all set of values matching the specified pattern

```r
name_phone <- c("Moly Robins: 250-999-8878",
                "Ali Duluth: 416-908-2044",
                "Eli Mitchell: 204-192-9829",
                "May Flowers: 250-209-7047")
```

```r
str_extract_all(name_phone, "[:alpha:]{2,}", simplify = TRUE)
     [,1]   [,2]
[1,] "Moly" "Robins"
[2,] "Ali"  "Duluth"
[3,] "Eli"  "Mitchell"
[4,] "May"  "Flowers"
```

Your Turn 3

Consider the 2017 babynames data

- Fill in the code to determine how many baby names in 2017 began with a vowel.
- Fill in the code to determine how many baby names in 2017 started and/or ended with a vowel.

04:00

Modify the case of a string

```
str_to_lower("BEAUTY is in the EYE of the BEHOLDER")
[1] "beauty is in the eye of the beholder"
```

```
str_to_upper("one small step for man, one giant leap for mankind")
[1] "ONE SMALL STEP FOR MAN, ONE GIANT LEAP FOR MANKIND"
```

```
str_to_title("Aspire to inspire before we expire")
[1] "Aspire To Inspire Before We Expire"
```

```
str_to_sentence("everything you can imagine is real")
[1] "Everything you can imagine is real"
```

Your Turn 4

Consider the 2017 babynames data

- how many baby names contain "illie`?
- how many baby names start with "M"?

04:00

Visualizing matches

```r
str_view(name_phone, "[5-8]")
```

```
Moly Robins: 250-999-8878

Ali Duluth: 416-908-2044

Eli Mitchell: 204-192-9829

May Flowers: 250-209-7047
```

```r
str_view_all(name_phone, "[5-8]")
```

```
Moly Robins: 250-999-8878

Ali Duluth: 416-908-2044

Eli Mitchell: 204-192-9829

May Flowers: 250-209-7047
```

Visualizing matches

```
str_view(name_phone, "[5|8]")
```

```
Moly Robins: 250-999-8878

Ali Duluth: 416-908-2044

Eli Mitchell: 204-192-9829

May Flowers: 250-209-7047
```

```
str_view_all(name_phone, "[5-8|a-e]")
```

```
Moly Robins: 250-999-8878

Ali Duluth: 416-908-2044

Eli Mitchell: 204-192-9829

May Flowers: 250-209-7047
```

Extract using repitition

```
str_view_all(name_phone, pattern = "[2-9][0-9]{2}-[0-9]{3}-[0-9]{4}")
```

```
Moly Robins: 250-999-8878

Ali Duluth: 416-908-2044

Eli Mitchell: 204-192-9829

May Flowers: 250-209-7047
```

str_locate()

tell you the character positions the pattern characters are found in

```
days %>%
  str_locate("day") %>% # start and end positions of the pattern "day"
  head(n=7)
     start end
[1,]     4   6
[2,]     5   7
[3,]     7   9
[4,]     6   8
[5,]     4   6
[6,]     6   8
[7,]     4   6
```

str_flatten()

collapses a string vector into a single string (i.e. a character vector of length 1)

```
str_flatten(c("a_3", "d_54"), # vector to collapse
            collapse =  " ") # insert in between
[1] "a_3 d_54"
```

str_glue()

allows one to interpolate strings and values that have been assigned to names in R

```
y <- Sys.Date() # current date
str_glue("today is {y}")
today is 2022-02-01
```

```
# base R equivalent
paste0("today is ", y)
[1] "today is 2022-02-01"
```

```
nm <- "Alex"
str_glue("Hi, my name is {nm}.")
Hi, my name is Alex.
```

```
a <- 5
str_glue("a = {a}")
a = 5
```

Lengths of strings

We can manage the lengths of strings using the following set of functions:

- `str_length()`
- `str_pad()`
- `str_trunc()`
- `str_trim()`

str_length()

tells you how many characters are in each entry of a character vector

```
gapminder %>% names()
[1] "country"   "continent" "year"      "lifeExp"   "pop"       "gdpPercap"
```

```
# length of each column names
gapminder %>% names() %>% str_length()
[1] 7 9 4 7 3 9
```

str_pad()

standardizes the length of strings in a character vector by padding it on the left or right ends with a specified character (by default, a space)

```
gapminder %>% names() %>%
  str_pad(width = 9, # minimum width of the string to fill/pad positions up to
          side = "both", # side to insert the extra characters on
          pad = "+")
[1] "+country+" "continent" "++year+++" "+lifeExp+" "+++pop+++" "gdpPercap"
```

str_trunc()

standardizes string lengths by controlling the maximum width and truncating strings longer than this

```
gapminder %>% names() %>%
  str_trunc(width = 7, #the maximum width to allow for strings
          side= "right",
          # entries which have been truncated will show this to indicate that
          # something has been removed
          ellipsis = "...")
[1] "country" "cont..." "year"    "lifeExp" "pop"     "gdpP..."
```

str_trim()

removes empty spaces on the ends of a string

```
#add some whitespace
padded_names <- gapminder %>% names() %>% str_pad(12, "both")
padded_names
[1] "  country   " " continent  " "    year    " "  lifeExp   " "    pop     "
[6] " gdpPercap  "
```

```
#remove it
str_trim(padded_names)
[1] "country"   "continent" "year"      "lifeExp"   "pop"       "gdpPercap"
```

Disambiguate your regexps

```
name_phone <- c("Moly Robins: 250-999-8878",
       "Ali Duluth: 416-908-2044",
       "Eli Mitchell: 204.192.9829",
       "May Flowers: 250.209.7047")
```

```
str_view_all(name_phone,
             pattern = "([2-9][0-9]{2})[.-]([0-9]{3})[.-]([0-9]{4})")
```

```
Moly Robins: 250-999-8878

Ali Duluth: 416-908-2044

Eli Mitchell: 204.192.9829

May Flowers: 250.209.7047
```

Replacing strings

```
str_replace_all(name_phone,
pattern = "([2-9][0-9]{2})[.-]([0-9]{3})[.-]([0-9]{4})",
replacement = "XXX-XXX-XXXX"
)
[1] "Moly Robins: XXX-XXX-XXXX"  "Ali Duluth: XXX-XXX-XXXX"
[3] "Eli Mitchell: XXX-XXX-XXXX" "May Flowers: XXX-XXX-XXXX"
```

Duplicating Groups

Use escaped numbers (\1, \2, etc) to repeat a group based on position

Which numbers have the same 1st and 3rd digits?

```r
phone_numbers <- c("515 111 2244", "310 549 6892", "474 234 7548")
str_view(phone_numbers, "(\\d)\\d\\1")
```

```
515 111 2244

310 549 6892

474 234 7548
```

Your Turn 5

```
x <- c("scuttlebutt", "Stetson", "Scattter", "Scatter")
str_detect(x, "^[Ss](.*)(t+)(.+)(t+)")
[1]  TRUE  TRUE  TRUE FALSE
```

The regular expression `"^[Ss](.*)(t+)(.+)(t+)"` detects "scuttlebutt", "Stetson", and "Scattter", but not "Scatter." Why?

05:00