Lab7

October 25, 2016

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Some standard R functions for text

Review the R string manipulators:

- `nchar`:
  - takes: a character vector
  - returns length of each string.
- `paste`:
  - takes a character vector
  - concatenate them.
- `substr`:
  - takes a character vector, start (int), and stop (int)
  - extract or replace substrings (inclusively from start to stop).
- `strsplit`:
  - takes a character vector, seperator (regex)
  - returns a list of subparts split by the seperator.
- `tolower`:
  - takes a character vector
  - convert all upper cases into lower.
- `toupper`:
  - takes a character vector
  - convert all lower cases into upper.
- `sub`:
  - takes in pattern (regex)
  - replacement (character), and a character vector, replaces the FIRST match with replacement.

Regular expressions

A regular expression is a sequence of characters that defines a pattern; most characters in the pattern simply "match" themselves in a target string.

Basics of regex

A plain text string is just a string.

Power of regex comes from the meta-characters. The commonly uesd meta-chatacters are:

```
.  \ | ( ) [ ] ^ $ { } * + ?
```

Character classes

- `.` matches anything (wildcard)
- `\w` a word (a single character)
- `\W` not a word
- `\s` a whitespace
- `\S` not a whitespace
- `\d` a digit
- `\D` not a digit
- `[aeiou]` is a character set, matches all vowels
- `[^aeiou]` is a negated character set, matches all but vowels
- `[a-e]` is a range, same as [abcde]

Escaped characters

- `\` is escape character, used when you need character-literals of the meta-characters, or when you use all escaped characters classes, anchors, etc. example: `\+` is literal `+`
- `\t` tab
- `\n` new line

Anchors

- `^` is the start of the string
- `$` is the end of the string
- `\b` word boundary
- `\B` not word boundary

Quantifiers and alternations

- `|` is the OR operator
- `{}` is a quantifier, it dictates how many times the preceeding character (group) must occur
  - `{m}` The preceding element or subexpression must occur exactly m times.
  - `{m,n}` The preceding element or subexpression must occur between m and n times, inclusive.
  - `{m,}` The preceding element or subexpression must occur at least m times.
- `*` means the immediate preceeding character (group) can appear multiple times (including 0 times)
- `+` means the immediate preceeding character (group) can appear multiple times, but at least once.
- `?` means the immediate preceeding character (group) can appear 0 or 1 time.

Capturing groups

- `()` is a capture group, can be used to limit the scope

An example with R function calls

This was in the news yesterday (Barcelona stars snub La Liga awards as Atletico claim FIVE of the big prizes at glamorous ceremony, *Mirror*, 25 Oct 2016):

```
txt <- paste("FC Barcelona's stars swerved La Liga's glamorous awards ceremony",
             " on Monday night, despite netting two gongs. Lionel Messi won the",
             " award for best forward while Luis Suarez was chosen as La Liga ",
             "World Player of the Year. But with coach Luis Enrique not even ",
             "nominated for best coach, despite winning the league, no Barca ",
             "players were in attendance even though they were on a day off. ",
             "Atletico Madrid were the story of the night, however, winning five",
             " major awards including best player, best coach, best goalkeeper ",
             "and best defender.",sep = "")
```

Now break into sentences using **strsplit**. Note that **strsplit** also uses regular expression, so you have to include escape sequences!

```
txt = strsplit(txt, split = "\\. ")[[1]]
substr(txt,1,70)
```

```
## [1] "FC Barcelona's stars swerved La Liga's glamorous awards ceremony on Mo"
## [2] "Lionel Messi won the award for best forward while Luis Suarez was chos"
## [3] "But with coach Luis Enrique not even nominated for best coach, despite"
## [4] "Atletico Madrid were the story of the night, however, winning five maj"
```

Why two \'s? Because when passing the string as an argument to **regexpr** in R you have to use the first \ to escape the second one.

You do NOT have to do this if you are calling the regex engine directly.

I want to find out how many **best** there are in each sentence.

```
pattern <- "best"
```

grep returns the indices vector with same length as "txt" and giving the starting position of the first match or -1 if there is none.

```
grep(pattern,txt)
```

```
## [1] 2 3 4
```

grepl returns a logical (match or not for each element of the text).

```
grepl(pattern,txt)
```

```
## [1] FALSE  TRUE  TRUE  TRUE
```

regexpr returns an integer vector with same length as "txt" and giving the starting position of the first match or -1 if there is none.

```
regexpr(pattern,txt)
```

```
## [1] -1 32 52 91
## attr(,"match.length")
## [1] -1  4  4  4
## attr(,"useBytes")
## [1] TRUE
```

`gregexpr` returns a list vector with same length as "txt" and giving the starting position of EVERY match or -1 if there is none.

```
gregexpr(pattern,txt)
```

```
## [[1]]
## [1] -1
## attr(,"match.length")
## [1] -1
## attr(,"useBytes")
## [1] TRUE
##
## [[2]]
## [1] 32
## attr(,"match.length")
## [1] 4
## attr(,"useBytes")
## [1] TRUE
##
## [[3]]
## [1] 52
## attr(,"match.length")
## [1] 4
## attr(,"useBytes")
## [1] TRUE
##
## [[4]]
## [1]  91 104 116 136
## attr(,"match.length")
## [1] 4 4 4 4
## attr(,"useBytes")
## [1] TRUE
```

What kind of `best`'s are there in the 4th sentence?

```
last.sentence <- txt[4]
best.something <- gregexpr("best \\w+", last.sentence, ignore.case = T)[[1]]

start <- best.something
end <- best.something+attr(best.something,"match.length")-1

substr(rep(last.sentence,length(start)),start,end)
```

```
## [1] "best player"     "best coach"      "best goalkeeper" "best defender"
```

`"best \w+"` matches the word `best` as well as the word that follows:

`\w` matches any word character, equivalent to `[A-Za-z0-9_]`, `+` asks regex to repeatedly find such characters as long as it succeeds in doing so.

Excercise 1: Match multiple strings with one pattern

`pattern` should match the following strings:

- regular expression
- regular expressions
- regex
- regexp
- regexes

```
text <- c("regular expression",
          "regular expressions",
          "regex",
          "regexp",
          "regexes")
```

```
regexpr(pattern, text)
```

```
## [1] 1 1 1 1 1
## attr(,"match.length")
## [1] 18 19  5  6  7
## attr(,"useBytes")
## [1] TRUE
```

Excercise 2: Validate email patterns

We will try to determine whether an email is valid.

Rather than giving a set of rules to define a valid email address, we will go by examples (from this post on Microsoft Developer Network):

Valid Email address	Reason
email@domain.com	Valid email
firstname.lastname@domain.com	Email contains dot in the address field
email@subdomain.domain.com	Email contains dot with subdomain
firstname+lastname@domain.com	Plus sign is considered valid character
email@123.123.123.123	Domain is valid IP address
1234567890@domain.com	Digits in address are valid
email@domain-one.com	Dash in domain name is valid
_______@domain.com	Underscore in the address field is valid
email@domain.name	.name is valid Top Level Domain name
email@domain.co.jp	Dot in Top Level Domain name also considered valid (use co.jp as e xample here)
firstname-lastname@domain.com	Dash in address field is valid

Invalid Email address	Reason
plainaddress	Missing @ sign and domain
#@%^%#$@#$@#.com	Garbage
@domain.com	Missing username
Joe Smith <email@domain.com>	Encoded html within email is invalid
email.domain.com	Missing @
email@domain@domain.com	Two @ sign
.email@domain.com	Leading dot in address is not allowed
email@domain.com (Joe Smith)	Text followed email is not allowed
email@domain	Missing top level domain (.com/.net/.org/etc)

Invalid Email address	Reason
email@-domain.com	Leading dash in front of domain is invalid
email@11.222.333.44444	Invalid IP format
email@domain..com	Multiple dot in the domain portion is invalid

The list of examples are typed here for convenience, copy & paste into R:

```
valid.emails <- c(
  "email@domain.com",
  "firstname.lastname@domain.com",
  "email@subdomain.domain.com",
  "firstname+lastname@domain.com",
  "email@123.123.123.123",
  "1234567890@domain.com",
  "email@domain-one.com",
  "_______@domain.com",
  "email@domain.name",
  "email@domain.co.jp",
  "firstname-lastname@domain.com"
  )

regexpr(pattern, valid.emails, ignore.case = T)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1
## attr(,"match.length")
##  [1] 16 29 26 29 21 21 20 18 17 18 29
## attr(,"useBytes")
## [1] TRUE
```

```
invalid.emails <- c(
  "plainaddress",
  "#@%^%#$@#$@#.com",
  "@domain.com",
  "Joe Smith <email@domain.com>",
  "email.domain.com",
  "email@domain@domain.com",
  ".email@domain.com",
  "email@domain.com (Joe Smith)",
  "email@domain",
  "email@-domain.com",
  "email@11.222.333.44444",
  "email@domain..com"
  )

regexpr(pattern, invalid.emails, ignore.case = T)
```

```
##  [1] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
## attr(,"match.length")
##  [1] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
## attr(,"useBytes")
## [1] TRUE
```