Working with stringr and regular expression

Merlin Shi

12/3/2017

Introductions

This post is going into detail on how to make the most of **stringr** package. Most functions in **stringr** package starts with "str_", which is easy to distinguish. **stringr** provides a clean, modern alternative to common string operations, and is sometimes easier to remember and use than R basic string functions. In the latter half, we are going through some details on

Examples

Property

- All function and argument names are consistent
- All functions deal with "NA"'s and zero length vectors in the same way
- The output from one function is easy to feed into the input of another.

installation

To get the current released version from CRAN:

```
install.packages("stringr")
```

To get the current development version from github:

```
# install.packages("devtools")
devtools::install_github("hadley/stringr")
```

To load the library: (Never forget to do this before anything)

```
library(stringr)
```

Manipulating individual characters, and string as a whole

We can easily get the length of a string with str_length():

```
str_length("abc")
```

```
## [1] 3
```

```
str_length(c("abc", "cdee"))
```

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

Get a substring with str_sub():

```
str_sub("abcde", -4, 5)
```

```
## [1] "bcde"
```

Note that string is indexed from 1. Any position can be referenced by a positive number if counted from the left, or be referenced by a negative number if counted from the right.

Duplicate and concatenate strings with str_dup(string, times):

```
str_dup(c("abcde", "pro"), c(2, 4))
```

```
## [1] "abcdeabcde"    "propropropro"
```

Note that "string" and "times" are vectorized.

Pattern matching

Many stringr functions work with patterns. Two parameters are needed: a character vector of strings to process, and a single pattern to match.

```
strings <- c(
  "apple pie",
  "219 733 8965",
  "329-293-8753",
  "Work: 579-499-7527; Home: 543.355.3679"
)
phone_pattern <- "([2-9][0-9]{2})[- .]([0-9]{3})[- .]([0-9]{4})"
```

str_detect() returns a logical vector showing which strings match the given pattern. str_subset() returns those matching strings.

```
# Which strings contain phone numbers?
str_detect(strings, phone_pattern)
```

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

```
#> [1] FALSE  TRUE  TRUE  TRUE
str_subset(strings, phone_pattern)
```

```
## [1] "219 733 8965"
## [2] "329-293-8753"
## [3] "Work: 579-499-7527; Home: 543.355.3679"
```

Regular expressions

Let's now focus on quantifiers and boundaries in regex, which is really worth knowing.

Quantifiers

```
<td>Quantifier</td>
<td>Legend</td>
<td>Example</td>
<td>Sample Match</td>
```

```
<td>+</td>
<td>One or more</td>
<td>Version \w-\w+</td>
<td>Version A-b1_1</td>
```

```
<td>{2,4}</td>
<td>Two to four times</td>
<td>\d{2,4}</td>
<td>156</td>
```

```
<td>*</td>
<td>Zero or more times</td> <td>A*B*C*</td> <td>AAACC</td>
```

```
<td>Once or none</td>   <td>plurals?</td>   <td>plural</td></tr>
```

?
Anchors and Boundaries These tokens are assertions about the engine's current position in the string. Therefore, none of them consume characters.

```
<td>^</td>
<td>Start of string or start of line depending on multiline mode. (But when [^inside brackets], it means "not")</t
d>
<td>^abc.*</td><td>abc (line start)</td>
```

```
<td>$</td>
<td>End of string or end of line</td><td>.*?the end$</td><td>this is the end</td>
```

```
<td>\A</td> <td>Beginning of string (all major engines except JS)</td>
<td>\Aabc[\d\D]*</td><td>abc (string...start)</td>
```

Word boundary Bob.*Bob ate the cat
Word boundary Bob.*Bob ate the кошка
Not a word boundaryc.. copycats

Conclusions

Regular expression is pervasively used in R and many other programming languages (and editors too, such as Sublime). We can make the most of stringr package if we know basic principles of regex. The idea never goes obsolete is that the process of learning comes along with practicing, so the best way to internalize the field is to do your own data processing tasks.

References

Four Column layout Cheat Sheet - RStudio https://www.rstudio.com/wp-content/uploads/2016/09/RegExCheatsheet.pdf

Quick-Start: Regex Cheat Sheet http://www.rexegg.com/regex-quickstart.html

Which characters need to be escaped on HTML? https://stackoverflow.com/questions/7381974/which-characters-need-to-be-escaped-on-html

Regular Expressions as used in R https://stat.ethz.ch/R-manual/R-devel/library/base/html/regex.html

Regular expressions in R vs RStudio https://www.r-bloggers.com/regular-expressions-in-r-vs-rstudio/

Regular Expression in R - Stat 545 https://www.regular-expressions.info/rlanguage.html

New RStudio cheat sheet: Strings in R https://www.r-bloggers.com/new-rstudio-cheat-sheet-strings-in-r/