Regular Expressions and String manipulation in R

Cheat Sheet --*Chen DA*

Libraries

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

Character class / numeric class

[[:digit:]]	Digits; [0-9]
[[:alphabet:]]	Alphabets; [A-z]
[[:upper alpha:]]	Upper case letters
[[:lower alpha:]]	Upper case letters
[[:alnum:]]	Alphas and digits [A-z][0-9]
[[:punct:]]	Punctuation characters;!"#$%&'()*+,-./:;<=>?@[]^_`{|}~

Advanced pattern matching

Look ahead

(?=pattern) requires that the expression pattern must be matched after this position
(?!pattern) requires that the expression pattern cannot be matched after this position

```
> win <- c("Windows2000", "Windows", "Windows3.1")
> str_view(win, "Windows(?=95|98|NT|2000)")
Windows2000
Windows
Windows3.1
```

Look behind

(?<=pattern) requires that this position must be preceded by a matching expression pattern
(?<!pattern) requires that the expression pattern cannot be matched before this position

```
> win <- c("2000Windows", "Windows", "3.1Windows")
> str_view(win, "(?<!95|98|NT|2000)Windows")
2000Windows
Windows
3.1Windows
```

Special characters

[\b]	backspace
\c	control character
\d	any number
\D	any non-number
\f	form feed
\n	newline
\r	carriage return
\t	Tabs
\v	vertical tab
\x	a hexadecimal number
\0	matches an octal number

Character group

?	0 or 1 time
+	1 or more times
*	0 or more times
{n}	exactly n times
{n, }	n or more times
{ ,m}	at most m times
{n,m}	between n and m times
+?	Convert from "greedy" expression to "non-greedy" expression or minimal match
*?	Convert from "greedy" expression to "non-greedy" expression or minimal match

repeated match

?	0 or 1 time
+	1 or more times
*	0 or more times
{n}	exactly n times
{n, }	n or more times
{ ,m}	at most m times
{n,m}	between n and m times
+?	Convert from "greedy" expression to "non-greedy" expression or minimal match
*?	Convert from "greedy" expression to "non-greedy" expression or minimal match

anchors

^	The start of the string
$	The end of the string
\\b	character boundaries
\\B	characters that are not boundaries
\\s	empty including spaces, newlines, etc.
\\S	non-empty characters
\\<	characters starting with whitespace
\\>	characters ending with whitespace

Application of regular expression pattern matching

Match patterns

str_detect(string, pattern, negate = FALSE)
Detect if a character vector matches a pattern

```
> x <- c("apple", "banana", "pear")
> str_detect(x, "e")
[1]  TRUE FALSE  TRUE
```

stringr::words contains common words in the Oxford dictionary

```
> sum(str_detect(words, "^t"))
[1] 65
```

Extract matched patterns

str_extract(string, pattern)
extract the first matched group

str_extract_all(string, pattern, simplify = FALSE)
extract all matched groups
If simplify is FALSE, returns a list of character vectors.
If TRUE returns a character matrix.

```
> shopping_list <- c("apples x4", "bag of flour", "bag of sugar", "milk x2")
> str_extract(shopping_list, "[a-z]{1,4}")
[1] "appl" "bag"  "bag"  "milk"
```

Replace matched patterns

str_replace(string, pattern, replacement)
Replace first match

str_replace_all(string, pattern, replacement)
Replace all matches

str_replace_na() to turn missing values into "NA"

```
> fruits <- c("one apple", "two pears", "three bananas")
> str_replace(fruits, "[aeiou]", "-")
[1] "-ne apple"     "tw- pears"     "thr-e bananas"
```

Split or join string with a pattern

str_split(string, pattern, n = Inf, simplify = FALSE)
str_split_fixed(string, pattern, n)

```
> lines <- "I love my country"
> str_split(lines, " ")
[[1]]
[1] "I"      "love"    "my"      "country"
```

str_c(..., sep = "", collapse = NULL)

Locate matched patterns

str_locate(string, pattern)
Return a matrix. The first column gives the start position of the match, and the second column gives the end position.

str_locate_all(string, pattern)
Return a matrix. The first column gives the start position of all matches, and the second column gives the end position.

```
> fruit <- c("apple", "banana", "pear")
> str_locate(fruit, c("a", "b", "p"))
     start end
[1,]     1   1
[2,]     1   1
[3,]     1   1
> str_locate_all(fruit, "a")
[[1]]
     start end
[1,]     1   1

[[2]]
     start end
[1,]     2   2
[2,]     4   4
[3,]     6   6

[[3]]
     start end
[1,]     3   3
```