rex CHEAT SHEET

Friendly Regular Expressions

Regex usage

rex(..., env=parent.frame()**)**

generate a regular expression

re_matches(data, pattern, global=FALSE, options=NULL, locations=FALSE, ...**)**

match function

re_substitutes(data, pattern, replacement, global=FALSE, options=NULL, ...**)**

substitute regular expressions in a string with another string

rex_mode()

While within rex mode, functions used within the rex function are attached, so one can get e.g. auto-completion within editors

Groups

capture(..., name=NULL**)**

create a capture group

group(...**)**

similar to capture except that it does not store the value of the group

capture_group(name**)**

use a captured value

Shortcuts

start

^

end

$

any

.

anything

.*

something

.+

letter

[[:alpha:]]

number

[[:digit:]]

letters

[[:alpha:]]+

numbers

[[:digit:]]+

alnum

[[:alnum:]]

space

[[:space:]]

names(shortcuts)

a complete list of shortcuts

Character classes

character_class("abc123")
one_of("abc123")

[abc123]

range("a", "j")
"a":"j"

[a-j]

any_of("abc")

*[abc]**

some_of("abc")

[abc]+

none_of("abc")

[^abc]

except_any_of("abc")

*[^abc]**

except_some_of("abc")

[^abc]+

Good to know

Rex is based on PCRE, thus if you want to use a rex regular expression in grepl, you have to use flag perl=TRUE explicitly

More at: https://github.com/kevinushey/rex

rex CHEAT SHEET

Friendly Regular Expressions

Counts

n_times(x, n, type=c("greedy", "lazy", "possessive")**)**

n_times("abc", 5) → (?:abc){5}

between(x, low, high, type=c("greedy", "lazy", "possessive")**)**

between("abc", 5, 10) → (?:abc){5, 10}

at_least(x, n, type=c("greedy", "lazy", "possessive")**)**

at_least("abc", 5) → (?:abc){5, }

at_most(x, n, type=c("greedy", "lazy", "possessive")**)**

at_most("abc", 5) → (?:abc){,5}

Wildcards

zero_or_more(..., type=c("greedy", "lazy", "possessive")**)**

*(?:...)**

one_or_more(..., type=c("greedy", "lazy", "possessive")**)**

(?:...)+

maybe(..., type=c("greedy", "lazy", "possessive")**)**

(?:...)?

Connectors

or(...**)**

specify set of optional matches, useful for more than 2 arguments

x %or% y

x | y

not(..., type=c("greedy", "lazy", "possessive")**)**

do not match

Lookarounds

x %if_next_is% y

TRUE if x follows y

x — a regex pattern

y — a regex pattern

x %if_next_isnt% y

TRUE if x does not follow y

x %if_prev_is% y

TRUE if y comes before x

x %if_prev_isnt% y

TRUE if y does not come before x

More at: https://github.com/kevinushey/rex