

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

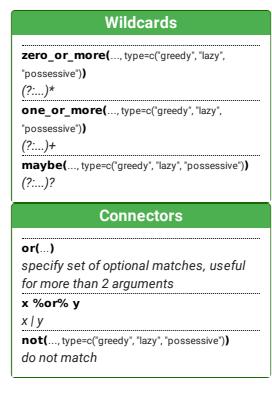
Shortcuts
start
end
\$
any
·
anything
.*
something
.+
letter
[[:alpha:]]
number
[[:digit:]]
letters
[[:alpha:]]+
numbers
[[:digit:]]+
names(shortcuts)
a complete list of shortcuts

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]+



Friendly Regular Expressions

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



```
Counts

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

n_times("abc", 5) \rightarrow (?:abc){5}

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

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

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

at_least("abc", 5) \rightarrow (?:abc){5, }

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

at_most("abc", 5) \rightarrow (?:abc){5}
```