Package ‘rex’

April 21, 2020

Type Package

Title Friendly Regular Expressions

Version 1.2.0

Maintainer Jim Hester <james.f.letcher@gmail.com>

URL https://github.com/kevinushey/rex

BugReports https://github.com/kevinushey/rex/issues

Description A friendly interface for the construction of regular expressions.

Imports lazyeval

License MIT + file LICENSE

Suggests testthat, knitr, markdown, dplyr, ggplot2, magrittr, Hmisc, stringr, rvest, roxygen2, covr

VignetteBuilder knitr


RoxygenNote 7.1.0

NeedsCompilation no

Author Kevin Ushey [aut],
Jim Hester [aut, cre],
Robert Krzyzanowski [aut]

Repository CRAN

Date/Publication 2020-04-21 18:20:06 UTC

R topics documented:

as.character.regex ........................................... 2
as.regex ...................................................... 3
capture ....................................................... 3
character_class ................................................. 4
character_class_escape .................................... 6
as.character.regex

Description

Specify an explicit regular expression. This expression must already be escaped.

Usage

```r
## S3 method for class 'regex'
as.character(x, ...)

## S3 method for class 'regex'
print(x, ...)

regex(x, ...)
```

Arguments

- `x` Object
- `...` further arguments

Methods (by generic)

- `as.character`: coerce regex object to a character
- `print`: Print regex object

See Also

`as.regex` to coerce to a regex object.
as.regex

**Description**

Coerce objects to a *regex*.

**Usage**

```r
as.regex(x, ...)
```

## Default S3 method:

```r
as.regex(x, ...)
```

**Arguments**

- `x` Object to coerce to *regex*.
- `...` further arguments passed to methods.

**Methods (by class)**

- **default**:
  - Simply escape the Object.

---

capture

**Create a capture group**

**Description**

Used to save the matched value within the group for use later in the regular expression or to extract the values captured. Both named and unnamed groups can later be referenced using `capture_group`.

**Usage**

```r
capture(..., name = NULL)
capture_group(name)
```

**Arguments**

- `...` shortcuts, R variables, text, or other *rex* functions.
- `name` of the group. Unnamed capture groups are numbers starting at 1 in the order they appear in the regular expression. If two groups have the same name, the leftmost group is the used in any reference.
See Also

- `group` for grouping without capturing. Perl 5 Capture Groups [http://perldoc.perl.org/perlre.html#Capture-groups](http://perldoc.perl.org/perlre.html#Capture-groups)
- Other regexes: `%or%`, `character_class()`, `counts`, `group()`, `lookarounds`, `not()`, `rex()`, `shortcuts`, `wildcards`

Examples

```r
# Match paired quotation marks
re <- rex(
  # first quotation mark
  capture(quotes),

  # match all non-matching quotation marks
  zero_or_more(except(capture_group(1))),

  # end quotation mark (matches first)
  capture_group(1)
)

# Named capture - don't match apples to oranges
re <- rex(
  capture(name = "fruit", or("apple", "orange"),
          ":=",
          capture_group("fruit")
)
)
```

---

**character_class**  
Create character classes

Description

There are multiple ways you can define a character class.

Usage

```r
character_class(x)

one_of(...)

any_of(..., type = c("greedy", "lazy", "possessive"))

some_of(..., type = c("greedy", "lazy", "possessive"))

none_of(...)

except_any_of(..., type = c("greedy", "lazy", "possessive"))
```
`character_class`

```r
except_some_of(..., type = c("greedy", "lazy", "possessive"))
range(start, end)
`:`(start, end)
exclude_range(start, end)
```

**Arguments**

- **x**
  - text to include in the character class (must be escaped manually)
- **...**
  - shortcuts, R variables, text, or other `rex` functions.
- **type**
  - the type of match to perform.
  - There are three match types
    1. `greedy`: match the longest string. This is the default matching type.
    2. `lazy`: match the shortest string. This matches the shortest string from the same anchor point, not necessarily the shortest global string.
    3. `possessive`: match and don’t allow backtracking
- **start**
  - beginning of character class
- **end**
  - end of character class

**Functions**

- `character_class`: explicitly define a character class
- `one_of`: matches one of the specified characters.
- `any_of`: matches zero or more of the specified characters.
- `some_of`: matches one or more of the specified characters.
- `none_of`: matches anything but one of the specified characters.
- `except_any_of`: matches zero or more of anything but the specified characters.
- `except_some_of`: matches one or more of anything but the specified characters.
- `range`: matches one of any of the characters in the range.
- `::`: matches one of any of the characters in the range.
- `exclude_range`: matches one of any of the characters except those in the range.

**See Also**

Other `rex`: `%or%`, `capture()`, `counts`, `group()`, `lookarounds`, `not()`, `rex()`, `shortcuts`, `wildcards`
Examples

# grey = gray
re <- rex("gr", one_of("a", "e"), "y")
grepl(re, c("grey", "gray")) # TRUE TRUE

# Match non-vowels
re <- rex(none_of("a", "e", "i", "o", "u"))
# They can also be in the same string
re <- rex(none_of("aeiou"))
grepl(re, c("k", "l", "e")) # TRUE TRUE FALSE

# Match range
re <- rex(range("a", "e"))
grepl(re, c("b", "d", "f")) # TRUE TRUE FALSE

# Explicit creation
re <- rex(character_class("abcd\["))
grepl(re, c("a", "d", ",", "]")) # TRUE TRUE FALSE

character_class_escape

Character class escapes

Description

Character class escapes

Usage

character_class_escape(x)

## S3 method for class 'regex'
character_class_escape(x)

## S3 method for class 'character_class'
character_class_escape(x)

## S3 method for class 'character'
character_class_escape(x)

## S3 method for class 'list'
character_class_escape(x)

## Default S3 method:
character_class_escape(x)

Arguments

x Object to escape.
counts

Methods (by class)
- regex: objects are passed through unchanged.
- character_class: objects are passed through unchanged.
- character: objects properly escaped for character classes.
- list: call character_class_escape on all elements of the list.
- default: coerce to character and character_class_escape.

counts

<table>
<thead>
<tr>
<th>Counts</th>
</tr>
</thead>
</table>

Description
Functions to restrict a regex to a specific number

Usage
n_times(x, n, type = c("greedy", "lazy", "possessive"))
between(x, low, high, type = c("greedy", "lazy", "possessive"))
at_least(x, n, type = c("greedy", "lazy", "possessive"))
at_most(x, n, type = c("greedy", "lazy", "possessive"))

Arguments
- x: A regex pattern.
- n: An integer number
- type: the type of match to perform.
  There are three match types
  1. greedy: match the longest string. This is the default matching type.
  2. lazy: match the shortest string. This matches the shortest string from the
     same anchor point, not necessarily the shortest global string.
  3. possessive: match and don’t allow backtracking
- low: An integer number for the lower limit.
- high: An integer number for the upper limit.

Functions
- n_times: x must occur exactly n times.
- between: x must occur between low and high times.
- at_least: x must occur at least n times.
- at_most: x must occur at most n times.
See Also

Other rex: `%|\|`. `capture()`, `character_class()`, `group()`, `lookarounds`, `not()`, `rex()`, `shortcuts`, `wildcards`
Create a grouped expression

**Description**

This is similar to *capture* except that it does not store the value of the group. Best used when you want to combine several parts together and do not reference or extract the grouped value later.

**Usage**

```plaintext
group(...) 
```

**Arguments**

- ...: shortcuts, R variables, text, or other *rex* functions.

**See Also**

*capture* for grouping with capturing. Perl 5 Extended Patterns [http://perldoc.perl.org/perlre.html#Extended-Patterns](http://perldoc.perl.org/perlre.html#Extended-Patterns)

Other rex: `%or%`, `capture()`, `character_class()`, `counts`, `lookarounds`, `not()`, `rex()`, `shortcuts`, `wildcards`
Details

These functions provide an interface to perl lookarounds. Special binary functions are used to infer an ordering, since often you might wish to match a word / set of characters conditional on the start and end of that word.

- `%if_next_is%`: TRUE if x follows y
- `%if_next_isnt%`: TRUE if x does not follow y
- `%if_prev_is%`: TRUE if y comes before x
- `%if_prev_isnt%`: TRUE if y does not come before x

See Also

Perl 5 Documentation [http://perldoc.perl.org/perlre.html#Extended-Patterns](http://perldoc.perl.org/perlre.html#Extended-Patterns)

Other rex: `%or%()`, `capture()`, `character_class()`, `counts`, `group()`, `not()`, `rex()`, `shortcuts`, `wildcards`

Examples

stopifnot(grepl(rex("crab" %if_next_is% "apple"), "crabapple", perl = TRUE))
stopifnot(grepl(rex("crab" %if_prev_is% "apple"), "applecrab", perl = TRUE))
stopifnot(grepl(rex(range("a", "e" %if_next_isnt% range("f", "g")), "ah", perl = TRUE))
stopifnot(grepl(rex(range("a", "e" %if_next_is% range("f", "i")), "ah", perl = TRUE))

<table>
<thead>
<tr>
<th>not</th>
<th>Do not match</th>
</tr>
</thead>
</table>

Description

Do not match

Usage

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

Arguments

- `...`: shortcuts, R variables, text, or other rex functions.
- `type`: the type of match to perform.

There are three match types

1. greedy: match the longest string. This is the default matching type.
2. lazy: match the shortest string. This matches the shortest string from the same anchor point, not necessarily the shortest global string.
3. possessive: match and don’t allow backtracking
register_shortcuts

See Also

Other rex: %or%, capture(), character_class(), counts, group(), lookarounds, rex(), shortcuts, wildcards

register_shortcuts  Register the Rex shortcuts

Description

If you are using rex in another package you need to call this function to register all of the rex shortcuts so that spurious NOTEs about global variables being generated during R CMD check.

Usage

register_shortcuts(pkg_name)

Arguments

pkg_name  the package to register the shortcuts in

rex  Generate a regular expression.

Description

Generate a regular expression.

Usage

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

Arguments

...  shortcuts, R variables, text, or other rex functions.
env  environment to evaluate the rex expression in.

See Also

Other rex: %or%, capture(), character_class(), counts, group(), lookarounds, not(), shortcuts, wildcards
**rex_mode**

*Toggles rex mode.*

**Description**

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

**Usage**

`rex_mode()`

---

**re_matches**

*Match function*

**Description**

Match function

**Usage**

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

**Arguments**

- **data**: character vector to match against
- **pattern**: regular expression to use for matching
- **global**: use global matching
- **options**: regular expression options
- **locations**: rather than returning the values of the matched (or captured) string, return a `data.frame` of the match locations in the string.
- **...**: options passed to regexpr or gregexpr

**Value**

If no captures, returns a logical vector the same length as the input character vector specifying if the relevant value matched or not. If there are captures in the regular expression, returns a `data.frame` with a column for each capture group. If `global` is `TRUE`, returns a list of `data.frames`. 
See Also

 regexp Section "Perl-like Regular Expressions" for a discussion of the supported options

Examples

string <- c("this is a", "test string")
re_matches(string, rex("test")) # FALSE FALSE

# named capture
re_matches(string, rex(capture(alphas, name = "first_word"), space,
capture(alphas, name = "second_word")))
# first_word second_word
# 1   this   is
# 2   test   string

# capture returns NA when it fails to match
re_matches(string, rex(capture("test")))
# 1
# 1  test
# 2 <NA>

---

re_substitutes  Substitute regular expressions in a string with another string.

Description

Substitute regular expressions in a string with another string.

Usage

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

Arguments

data  character vector to substitute
pattern  regular expression to match
replacement  replacement text to use
global  substitute all occurrences
options  option flags
...  options passed to sub or gsub

See Also

 regexp Section "Perl-like Regular Expressions" for a discussion of the supported options
Examples

```r
string <- c("this is a Test", "string")
re_substitutes(string, "test", "not a test", options = "insensitive")
re_substitutes(string, "i", "x", global = TRUE)
re_substitutes(string, "(test)", "not a \1", options = "insensitive")
```

<table>
<thead>
<tr>
<th>shortcuts</th>
<th>Shortcuts</th>
</tr>
</thead>
</table>

Description

Commonly used character classes and regular expressions. These shortcuts are substituted inside `rex` calls.

Usage

`shortcuts`

Format

An object of class `shortcut` of length 116.

Details

`names(shortcuts)` will give you the full list of available shortcuts.

See Also

Other `rex`: `%or%`, `capture()`, `character_class()`, `counts`, `group()`, `lookarounds`, `not()`, `rex()`, `wildcards`

<table>
<thead>
<tr>
<th>single_shortcuts</th>
<th>Single shortcuts</th>
</tr>
</thead>
</table>

Description

Each of these shortcuts has both a plural (-s) and inverse (non_) form.

Usage

`single_shortcuts`

Format

An object of class `shortcut` of length 18.
Description

Wildcards

Usage

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

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

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

Arguments

... shortcuts, R variables, text, or other rex functions.

type the type of match to perform.

There are three match types

1. greedy: match the longest string. This is the default matching type.
2. lazy: match the shortest string. This matches the shortest string from the same anchor point, not necessarily the shortest global string.
3. possessive: match and don’t allow backtracking

Functions

- zero_or_more: match ... zero or more times.
- one_or_more: match ... one or more times.
- maybe: match ... zero or one times.

See Also

Other rex: %or%(), capture(), character_class(), counts, group(), lookarounds, not(), rex(), shortcuts
%or%  

Description

The special binary function %or% can be used to specify a set of optional matches.

Usage

```r
x %or% y
```
```
or(...)
```

Arguments

- `x`: A string.
- `y`: A string.
- `...`: shortcuts, R variables, text, or other `rex` functions.

See Also

Other rex: `capture()`, `character_class()`, `counts`, `group()`, `lookarounds`, `not()`, `rex()`, `shortcuts`, `wildcards`
Index

+Topic datasets
  shortcuts, 14
  single_shortcuts, 14
. (capture), 3
: (character_class), 4
%if_next_is% (lookarounds), 9
%if_next_isnt% (lookarounds), 9
%if_prev_is% (lookarounds), 9
%if_prev_isnt% (lookarounds), 9
%or%, 4, 5, 8–11, 14, 15, 16
any_of (character_class), 4
as.character.regex, 2
as.regex, 2, 3
at_least (counts), 7
at_most (counts), 7
between (counts), 7
capture, 3, 5, 8–11, 14–16
capture_group, 3
capture_group (capture), 3
capture_class, 4, 4, 8–11, 14–16
capture_class_escape, 6
counts, 4, 5, 7, 9–11, 14–16
escape, 8
except (character_class), 4
except_any_of (character_class), 4
except_some_of (character_class), 4
exclude_range (character_class), 4
group, 4, 5, 8, 9, 10, 11, 14–16
lookarounds, 4, 5, 8, 9, 11, 14–16
m (re_matches), 12
matches (re_matches), 12
maybe (wildcards), 15
n (counts), 7
n_times (counts), 7
none_of (character_class), 4
not, 4, 5, 8–10, 10, 11, 14–16
one_of (character_class), 4
one_or_more (wildcards), 15
or (%or%), 16
print.regex (as.character.regex), 2
range (character_class), 4
re_matches, 12
re_substitutes, 13
regex, 3
regex (as.character.regex), 2
regexp, 13
register_shortcuts, 11
rex, 4, 5, 8–11, 11, 12, 14–16
rex_ (rex), 11
rex_mode, 12
s (re_substitutes), 13
shortcuts, 3–5, 8–11, 14, 15, 16
single_shortcuts, 14
some_of (character_class), 4
substitutes (re_substitutes), 13
wildcards, 4, 5, 8–11, 14, 15, 16
zero_or_more (wildcards), 15
zero_or_one (wildcards), 15