Package 'ore'

November 2, 2019

Version 1.6.3

Date 2019-10-31

Title An R Interface to the Onigmo Regular Expression Library

Author Jon Clayden, based on Onigmo by K. Kosako and K. Takata

Maintainer Jon Clayden <code@clayden.org>

Suggests crayon, rex, testthat

Description Provides an alternative to R's built-in functionality for handling regular expressions, based on the Onigmo library. Offers first-class compiled regex objects, partial matching and function-based substitutions, amongst other features.

Encoding UTF-8

Biarch true

License BSD_3_clause + file LICENCE

Collate workspace.R file.R dict.R ore.R match.R es.R zzz.R

URL https://github.com/jonclayden/ore

BugReports https://github.com/jonclayden/ore/issues

RoxygenNote 5.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-11-02 18:50:02 UTC

R topics documented:

`es` *Expression substitution*

Description

Evaluate R expressions and substitute their values into one or more strings.

Usage

```
es(text, round = NULL, signif = NULL, envir = parent.frame())
```

Arguments

`text`	A vector of strings to substitute into.
`round`	`NULL` or a single integer, giving the number of decimal digits for rounding numeric expressions. This argument takes priority over `signif`.
`signif`	`NULL` or a single integer, giving the number of significant decimal digits to use for numeric expressions. The `round` argument takes priority over this one, and will be used if not `NULL`.
`envir`	The environment to evaluate expressions in.

Details

Each part of the string surrounded by `"#{}"` is extracted, evaluated as R code in the specified environment, and then its value is substituted back into the string. The literal string `"#{}"` can be obtained by escaping the hash character, viz. `"\#{}"`. The block may contain multiple R expressions, separated by semicolons, but may not contain additional braces.

Value

The final strings, with expression values substituted into them.

See Also

`ore.subst`

Examples

```
es("pi is #{pi}")
es("pi is \#{pi}")
es("The square-root of pi is approximately #{sqrt(pi)}", signif=4)
es("1/(1+x) for x=3 is #{x <- 3; 1/(1+x)}")
```

matches *Extract matching substrings*

Description

These functions extract entire matches, or just subgroup matches, from objects of class "orematch". They can also be applied to lists of these objects, as returned by ore.search when more than one string is searched. For other objects they return NA.

Usage

```
matches(object, ...)

## S3 method for class 'orematches'
matches(object, simplify = TRUE, ...)

## S3 method for class 'orematch'
matches(object, ...)

## Default S3 method:
matches(object, ...)

groups(object, ...)

## S3 method for class 'orematches'
groups(object, simplify = TRUE, ...)

## S3 method for class 'orematch'
groups(object, ...)

## S3 method for class 'orearg'
groups(object, ...)

## Default S3 method:
groups(object, ...)
```

Arguments

object	An R object. Methods are provided for generic lists and "orematch" objects. If no object is provided (i.e. the method is called with no arguments), the value of ore.lastmatch will be used as a default.
...	Further arguments to methods.
simplify	For the list methods, should nonmatching elements be removed from the result?

Value

A vector, matrix, array, or list of the same, containing full matches or subgroups. If `simplify` is `TRUE`, the result may have a `dropped` attribute, giving the indices of nonmatching elements.

See Also

`ore.search`

`ore` *Oniguruma regular expressions*

Description

Create, test for, and print objects of class `"ore"`, which represent Oniguruma regular expressions. These are length-1 character vectors with additional attributes, including a pointer to the compiled version.

Usage

```
ore(..., options = "", encoding = "auto", syntax = c("ruby", "fixed"))

is.ore(x)

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

Arguments

`...`	One or more strings or dictionary labels, constituting a valid regular expression after being concatenated together. Elements drawn from the dictionary will be surrounded by parentheses, turning them into groups. Note that backslashes should be doubled, to avoid them being interpreted as character escapes by R. The `...` argument is ignored by the `print` method.
`options`	A string composed of characters indicating variations on the usual interpretation of the regex. These may currently include `"i"` for case-insensitive matching, and `"m"` for multiline matching (in which case `"."` matches the newline character).
`encoding`	The encoding that matching will take place in, a string string naming one of the encoding types discussed in `Encoding`, or `"auto"`. In the latter case, the encoding of `pattern` will be used.
`syntax`	The regular expression syntax being used. The default is `"ruby"`, which reflects the syntax of the Ruby language, which is very similar to that of Perl. An alternative is `"fixed"`, for literal matching without special treatment of characters.
`x`	An R object.

Value

The ore function returns the final pattern, with class "ore" and the following attributes:

.compiled	A low-level pointer to the compiled version of the regular expresion.
options	Options, copied from the argument of the same name.
encoding	The specified or detected encoding.
syntax	The specified syntax type.
nGroups	The number of groups in the pattern.
groupNames	Group names, if applicable.

The is.ore function returns a logical vector indicating whether its argument represents an "ore" object.

See Also

For full details of supported syntax, please see https://raw.githubusercontent.com/k-takata/Onigmo/master/doc/RE. The regex page is also useful as a quick reference, since PCRE (used by base R) and Oniguruma (used by ore) have similar features. See ore.dict for details of the pattern dictionary.

Examples

```
# This matches a positive or negative integer
ore("-?\\d+")

# This matches words of exactly four characters
ore("\\b\\w{4}\\b")
```

---

ore.dict *Get or set entries in the pattern dictionary*

Description

This function allows the user to get or set entries in the pattern dictionary, a library of regular expressions whose elements can be referred to by name in ore, and therefore easily reused.

Usage

```
ore.dict(..., enclos = parent.frame())
```

Arguments

...	One or more strings or dictionary keys. Unnamed, literal strings will be returned unmodified, named strings will be added to the dictionary, and unquoted names will be resolved using the dictionary.
enclos	Enclosure for resolving names not present in the dictionary. Passed to eval.

Value

If no arguments are provided, the whole dictionary is returned. Otherwise the return value is a (possibly named) character vector of resolved strings.

See Also

`ore`, which passes its arguments through this function

Examples

```
# Literal strings are returned as-is
ore.dict("protocol")

# Named arguments are added to the dictionary
ore.dict(protocol="\\w+://")

# ... and can be retrieved by name
ore.dict(protocol)
```

`ore.escape` *Escape regular expression special characters*

Description

Escape characters that would usually be interpreted specially in a regular expression, returning a modified version of the argument. This can be useful when incorporating a general-purpose string into a larger regex.

Usage

```
ore.escape(text)
```

Arguments

`text`	A character vector.

Value

A modified version of the argument, with special characters escaped by prefixing them with a backslash.

See Also

`ore`

ore.file *Use a file as a text source*

Description

Identify a file path to be used as a text source for a subsequent call to ore.search.

Usage

```
ore.file(path, encoding = getOption("ore.encoding"), binary = FALSE)
```

Arguments

path	A character string giving the file path.
encoding	A character string giving the encoding of the file. This should match the encoding of the regular expression used in a call to ore.search.
binary	A logical value: if TRUE, the file will be search bytewise, and encoding will be fixed to be "ASCII".

See Also

ore.search for actually searching through the file.

ore.ismatch *Does text match a regex?*

Description

These functions test whether the elements of a character vector match a Oniguruma regular expression. The actual match can be retrieved using ore.lastmatch.

Usage

```
ore.ismatch(regex, text, ...)

X %~% Y

X %~~% Y

X %~|% Y
```

Arguments

regex	A single character string or object of class "ore".
text	A character vector of strings to search.
...	Further arguments to ore.search.
X	A character vector or "ore" object. See Details.
Y	A character vector. See Details.

Details

The %~% infix shorthand corresponds to ore.ismatch(...,all=FALSE), while %~~% corresponds to ore.ismatch(...,all=TRUE). Either way, the first argument can be an "ore" object, in which case the second is the text to search, or a character vector, in which case the second argument is assumed to contain the regex. The %~|% shorthand returns just those elements of the text vector which match the regular expression.

Value

A logical vector, indicating whether elements of text match regex, or not.

See Also

ore.search

Examples

```
# Test for the presence of a vowel
ore.ismatch("[aeiou]", c("sky","lake"))  # => c(FALSE,TRUE)

# The same thing, in shorter form
c("sky","lake") %~% "[aeiou]"

# Same again: the first argument must be an "ore" object this way around
ore("[aeiou]") %~% c("sky","lake")
```

---

ore.lastmatch *Retrieve the last match*

Description

This function can be used to obtain the "orematch" object, or list, corresponding to the last call to ore.search. This can be useful after performing a search implicitly, for example with %~%.

Usage

```
ore.lastmatch(simplify = TRUE)
```

Arguments

simplify	If TRUE and the last match was against a single string, then the "orematch" object will be returned, instead of a list with one element.

Value

An "orematch" object or list. See ore.search for details.

---

ore.search *Search for matches to a regular expression*

Description

Search a character vector for one or more matches to an Oniguruma-compatible regular expression. The result is of class "orematches", for which printing and indexing methods are available.

Usage

```
ore.search(regex, text, all = FALSE, start = 1L, simplify = TRUE,
  incremental = !all)

is.orematch(x)

## S3 method for class 'orematch'
x[j, k, ...]

## S3 method for class 'orematches'
x[i, j, k, ...]

## S3 method for class 'orematch'
print(x, lines = NULL, context = NULL, width = NULL,
  ...)

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

Arguments

regex	A single character string or object of class "ore". In the former case, this will first be passed through ore.
text	A vector of strings to match against, or the result of a call to ore.file to search in a file. In the latter case, match offsets will be relative to the file's encoding.
all	If TRUE, then all matches within each element of text will be found. Otherwise, the search will stop at the first match.
start	An optional vector of offsets (in characters) at which to start searching. Will be recycled to the length of text.

simplify	If TRUE, an object of class "orematch" will be returned if text is of length 1. Otherwise, a list of such objects, with class "orematches", will always be returned.
incremental	If TRUE and the text argument points to a file, the file is read in increasingly large blocks. This can reduce search time in large files.
x	An R object.
j	For indexing, the match number.
k	For indexing, the group number.
...	Ignored.
i	For indexing into an "orematches" object only, the string number.
lines	The maximum number of lines to print. If NULL, this defaults to the value of the "ore.lines" option, or 0 if that is unset or invalid. Zero means no limit.
context	The number of characters of context to include either side of each match. If NULL, this defaults to the value of the "ore.context" option, or 30 if that is unset or invalid.
width	The number of characters in each line of printed output. If NULL, this defaults to the value of the standard "width" option.

Value

For ore.search, an "orematch" object, or a list of the same, each with elements

text	A copy of the text element for the current match.
nMatches	The number of matches found.
offsets	The offsets (in characters) of each match.
byteOffsets	The offsets (in bytes) of each match.
lengths	The lengths (in characters) of each match.
byteLengths	The lengths (in bytes) of each match.
matches	The matched substrings.
groups	Equivalent metadata for each parenthesised subgroup in regex, in a series of matrices. If named groups are present in the regex then dimnames will be set appropriately.

For is.orematch, a logical vector indicating whether the specified object has class "orematch". For extraction with one index, a vector of matched substrings. For extraction with two indices, a vector or matrix of substrings corresponding to captured groups.

Note

Only named *or* unnamed groups will currently be captured, not both. If there are named groups in the pattern, then unnamed groups will be ignored.

By default the print method uses the crayon package (if it is available) to determine whether or not the R terminal supports colour. Alternatively, colour printing may be forced or disabled by setting the "ore.colour" (or "ore.color") option to a logical value.

See Also

ore for creating regex objects; matches and groups for an alternative to indexing for extracting matching substrings.

Examples

```
# Pick out pairs of consecutive word characters
match <- ore.search("(\\w)(\\w)", "This is a test", all=TRUE)

# Find the second matched substring ("is", from "This")
match[2]

# Find the content of the second group in the second match ("s")
match[2,2]
```

---

ore.split *Split strings using a regex*

Description

This function breaks up the strings provided at regions matching a regular expression, removing those regions from the result. It is analogous to the strsplit function in base R.

Usage

```
ore.split(regex, text, start = 1L, simplify = TRUE)
```

Arguments

regex	A single character string or object of class "ore". In the former case, this will first be passed through ore.
text	A vector of strings to match against.
start	An optional vector of offsets (in characters) at which to start searching. Will be recycled to the length of text.
simplify	If TRUE, a character vector containing the pieces will be returned if text is of length 1. Otherwise, a list of such objects will always be returned.

Value

A character vector or list of substrings.

See Also

ore.search

Examples

```
ore.split("-?\\d+", "I have 2 dogs, 3 cats and 4 hamsters")
```

`ore.subst` *Replace matched substrings with new text*

Description

This function substitutes new text into strings in regions that match a regular expression. The substitutions may be simple text, may include references to matched subgroups, or may be created by an R function.

Usage

```
ore.subst(regex, replacement, text, ..., all = FALSE)
```

Arguments

`regex`	A single character string or object of class `"ore"`. In the former case, this will first be passed through `ore`.
`replacement`	A single character string, or a function to be applied to the matches.
`text`	A vector of strings to match against.
`...`	Further arguments to `replacement`, if it is a function.
`all`	If `TRUE`, then all matches within each element of `text` will be found. Otherwise, the search will stop at the first match.

Details

If `replacement` is a function, then it will be passed as its first argument an object of class `"orearg"`. This is a character vector containing as its elements the matched substrings, and with an attribute containing the matches for parenthesised subgroups, if there are any. A `groups` method is available for this class, so the groups attribute can be easily obtained that way. The substitution function will be called once per element of `text`.

Value

A version of `text` with the substitutions made.

See Also

`ore.search`

Examples

```
# Simple text substitution (produces "no dogs")
ore.subst("\\d+", "no", "2 dogs")

# Back-referenced substitution (produces "22 dogs")
ore.subst("(\\d+)", "\\1\\1", "2 dogs")
```

```
# Function-based substitution (produces "4 dogs")
ore.subst("\\d+", function(i) as.numeric(i)^2, "2 dogs")
```

Index