Package 'yasp'

October 14, 2022

Type Package

Title String Functions for Compact R Code

Version 0.2.0

Description A collection of string functions designed for writing compact and expressive R code. 'yasp' (Yet Another String Package) is simple, fast, dependency-free, and written in pure R. The package provides: a coherent set of abbreviations for paste() from package 'base' with a variety of defaults, such as p() for ``paste'' and pcc() for ``paste and collapse with commas''; wrap(), bracket(), and others for wrapping a string in flanking characters; unwrap() for removing pairs of characters (at any position in a string); and sentence() for cleaning whitespace around punctuation and capitalization appropriate for prose sentences.

License MIT + file LICENSE

URL https://github.com/t-kalinowski/yasp

BugReports https://github.com/t-kalinowski/yasp/issues

RoxygenNote 6.0.1

NeedsCompilation no

Author Tomasz Kalinowski [aut, cre]

Maintainer Tomasz Kalinowski <tkalinow@asu.edu>

Repository CRAN

Date/Publication 2018-05-29 22:46:45 UTC

R topics documented:

p *paste variants*

Description

Wrappers around `base::paste` with a variety of defaults:

	mnemonic	collapse=	sep=
p(), p0()	paste, paste0	NULL	""
ps(), pss()	paste (sep) space	NULL	" "
psh()	paste sep hyphen	NULL	"-"
psu()	paste sep underscore	NULL	"_"
psnl()	paste sep newline	NULL	"\n"
pc()	paste collapse	""	""
pcs()	paste collapse space	" "	""
pcc()	paste collapse comma	", "	""
pcsc()	paste collapse semicolon	"; "	""
pcnl()	paste collapse newline	"\n"	""
pc_and()	paste collapse and	*varies*	""
pc_or()	paste collapse or	*varies*	""

Usage

```
p(..., sep = "")

ps(...)

pss(...)

psu(...)

psh(...)

psnl(...)

p0(...)

pc(..., sep = "")

pcs(..., sep = "")

pcc(..., sep = "")

pcnl(..., sep = "")

pcsc(..., sep = "")
```

```
pc_and(..., sep = "")

pc_or(..., sep = "")
```

Arguments

`..., sep`	passed on to `base::paste`

See Also

`wrap sentence`

Examples

```
x <- head(letters, 3)
y <- tail(letters, 3)
# paste
p(x, y)
p0(x, y)
# paste + collapse
pc(x)
pc(x, y)
pcs(x)
pcc(x)
pcc(x, y)
pcsc(x)
pcnl(x)
pc_and(x[1:2])
pc_and(x[1:3])
pc_or(x[1:2])
pc_or(x[1:3])
pc_and(x, y)
pc_and(x, y, sep = "-")
pc_and(x[1])
pc_and(x[0])
```

---

sentence *sentence*

Description

A wrapper around `paste` that does some simple cleaning appropriate for prose sentences. It

1. trims leading and trailing whitespace
2. collapses runs of whitespace into a single space
3. appends a period (`.`) if there is no terminal punctuation mark (`.`, ?, or `!`)
4. removes spaces preceding punctuation characters: `.?!,;:`

5. collapses sequences of punctuation marks (`.?!,;:`) (possibly separated by spaces), into a single punctuation mark. The first punctuation mark of the sequence is used, with priority given to terminal punctuation marks `.?!` if present
6. makes sure a space or end-of-string follows every one of `.?!,;:`, with an exception for the special case of `.,:` followed by a digit, indicating the punctuation is decimal period, number separator, or time delimiter
7. capitalizes the first letter of each sentence (start-of-string or following a `.?!`)

Usage

```
sentence(...)
```

Arguments

`...`	passed on to `paste`

Examples

```
compare <- function(x) cat(sprintf(' in: "%s"\nout: "%s"\n', x, sentence(x)))
compare("capitilized and period added")
compare("whitespace:added ,or removed ; like this.and this")
compare("periods and commas in numbers like 1,234.567 are fine !")
compare("colons can be punctuation or time : 12:00 !")
compare("only one punctuation mark at a time!.?,;")
compare("The first mark ,; is kept;,,with priority for terminal marks ;,.")

# vectorized like paste()
sentence(
 "The", c("first", "second", "third"), "letter is", letters[1:3],
 parens("uppercase:", sngl_quote(LETTERS[1:3])), ".")
```

`unwrap` *unwrap*

Description

Remove pair(s) of characters from a string. The pair(s) to be removed can be at any position within the string.

Usage

```
unwrap(x, left, right = left, n_pairs = Inf)

unparens(x, n_pairs = Inf)
```

Arguments

x	character vector
left	left character to remove
right	right character to remove. Only removed if position is after left
n_pairs	number of character pairs to remove

Value

character vector with pairs removed

See Also

wrap

Examples

```
# by default, removes all matching pairs of left and right
x <- c("a", "(a)", "((a))", "(a) b", "a (b)", "(a) (b)" )
data.frame( x, unparens(x), check.names = FALSE )

# specify n_pairs to remove a specific number of pairs
x <- c("(a)", "((a))", "(((a)))", "(a) (b)", "(a) (b) (c)", "(a) (b) (c) (d)")
data.frame( x,
            "n_pairs=1"   = unparens(x, n_pairs = 1),
            "n_pairs=2"   = unparens(x, n_pairs = 2),
            "n_pairs=3"   = unparens(x, n_pairs = 3),
            "n_pairs=Inf" = unparens(x), # the default
            check.names = FALSE )

# use unwrap() to specify any pair of characters for left and right
x <- "A string with some \\emph{latex tags}."
unwrap(x, "\\emph{", "}")

# by default, only pairs are removed. Set a character to "" to override.
x <- c("a)", "a))", "(a", "((a" )
data.frame(x, unparens(x),
  'left=""' = unwrap(x, left = "", right = ")"),
  check.names = FALSE)

# make your own functions like this
# markdown bold
unbold <- function(x) unwrap(x, "**")
bold <- function(...) wrap(paste(...), "**")
(x <- (p("make a word", bold("bold"))))
unbold(x)
```

wrap *Wrap strings*

Description

Wrap strings with flanking characters

Usage

```
wrap(x, left, right = left)

dbl_quote(..., sep = "")

sngl_quote(..., sep = "")

bracket(..., sep = "")

brace(..., sep = "")

parens(..., sep = "")
```

Arguments

x	character to wrap
left, right	character pair to wrap with
sep, ...	passed to base::paste before wrapping

See Also

unwrap p0 sentence

Examples

```
wrap("abc", "__")  #  __abc__
parens("abc")      #   (abc)
sngl_quote("abc")  #   'abc'
dbl_quote("abc")   #   "abc"
bracket("abc")     #   [abc]
brace("abc")       #   {abc}

label <- p("name", parens("attribute"))

label             # "name (attribute)"
unparens(label)   # "name attribute"

# make your own function like this:
# markdown bold
bold <- function(...) wrap(paste(...), "**")
```

```
p("make a word", bold("bold"))
# see unbold example in ?unwrap
```

Index