

# Package: yasp (via r-universe)

September 4, 2024

**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

**Repository** <https://t-kalinowski.r-universe.dev>

**RemoteUrl** <https://github.com/t-kalinowski/yasp>

**RemoteRef** HEAD

**RemoteSha** b25cc0cff4350fd78a96257b9da5ea1a06937918

## Contents

<code>p</code> . . . . .	2
<code>sentence</code> . . . . .	3
<code>unwrap</code> . . . . .	4
<code>wrap</code> . . . . .	6

<b>Index</b>	<b>8</b>
--------------	----------

---

p *paste variants*

---

### Description

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

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

### 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**

<code>x</code>	character vector
<code>left</code>	left character to remove
<code>right</code>	right character to remove. Only removed if position is after left
<code>n_pairs</code>	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

<code>x</code>	character to wrap
<code>left, right</code>	character pair to wrap with
<code>sep, ...</code>	passed to <code>base::paste</code> 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(...), "**")
```

*wrap*

7

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

# Index

base::paste, [2](#), [3](#), [6](#)

brace (wrap), [6](#)

bracket (wrap), [6](#)

dbl\_quote (wrap), [6](#)

p, [2](#)

p0, [6](#)

p0 (p), [2](#)

parens (wrap), [6](#)

pc (p), [2](#)

pc\_and (p), [2](#)

pc\_or (p), [2](#)

pcc (p), [2](#)

pcnl (p), [2](#)

pcs (p), [2](#)

pcsc (p), [2](#)

ps (p), [2](#)

psh (p), [2](#)

psnl (p), [2](#)

pss (p), [2](#)

psu (p), [2](#)

sentence, [3](#), [3](#), [6](#)

sngl\_quote (wrap), [6](#)

unparens (unwrap), [4](#)

unwrap, [4](#), [6](#)

wrap, [3](#), [5](#), [6](#)