Package 'stringstatic'

July 13, 2023

Title Dependency-Free String Operations

Version 0.1.2

Description Provides drop-in replacements for functions from the 'stringr' package, with the same user interface. These functions have no external dependencies and can be copied directly into your package code using the 'staticimports' package.

License CC0

URL https://github.com/rossellhayes/stringstatic

BugReports https://github.com/rossellhayes/stringstatic/issues

Suggests testthat (>= 3.0.0)

Config/Needs/roxygen wch/staticimports

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Alexander Rossell Hayes [aut, cre, cph] (<https://orcid.org/0000-0001-9412-0457>), Eli Pousson [ctb] (<https://orcid.org/0000-0001-8280-1706>, str_pad() and str_split() functions), Hadley Wickham [ctb, cph] (stringr package), RStudio [cph] (stringr package)

Maintainer Alexander Rossell Hayes <alexander@rossellhayes.com>

Repository CRAN

Date/Publication 2023-07-12 22:30:03 UTC

R topics documented:

`fixed` *Compare literal bytes in the string*

Description

Compare literal bytes in the string

Usage

```
fixed(pattern, ignore_case = FALSE)
```

Arguments

`pattern`	Pattern to modify behavior.
`ignore_case`	Should case differences be ignored in the match?

Value

An integer vector.

Source

Adapted from the stringr package.

Dependency-free drop-in alternative for `stringr::fixed()`. This is very fast, but not usually what you want for non-ASCII character sets.

`regex` *Control regex matching behavior*

Description

Dependency-free drop-in alternative for `stringr::regex()`.

Usage

```
regex(
  pattern,
  ignore_case = FALSE,
  multiline = FALSE,
  comments = FALSE,
  dotall = FALSE
)
```

Arguments

`pattern`	Pattern to modify behavior.
`ignore_case`	Should case differences be ignored in the match?
`multiline`	If `TRUE`, `$` and `^` match the beginning and end of each line. If `FALSE`, the default, only match the start and end of the input.
`comments`	If `TRUE`, white space and comments beginning with # are ignored. Escape literal spaces with `\\`.
`dotall`	If `TRUE`, `.` will also match line terminators.

Value

An integer vector.

Source

Adapted from the stringr package.

str_c *Join multiple strings into a single string*

Description

Dependency-free drop-in alternative for `stringr::str_c()`.

Usage

```
str_c(..., sep = "", collapse = NULL)
```

Arguments

`...`	One or more character vectors. Zero length arguments are removed. Short arguments are recycled to the length of the longest. Like most other R functions, missing values are "infectious": whenever a missing value is combined with another string the result will always be missing. Use `str_replace_na()` to convert `NA` to "NA"
`sep`	String to insert between input vectors.
`collapse`	Optional string used to combine input vectors into single string.

Value

If `collapse = NULL` (the default) a character vector with length equal to the longest input string. If collapse is non-`NULL`, a character vector of length 1.

Source

Adapted from the stringr package.

str_count *Count the number of matches in a string*

Description

Dependency-free drop-in alternative for `stringr::str_count()`.

Usage

```
str_count(string, pattern = "")
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.

Value

An integer vector.

Source

Adapted from the stringr package.

`str_detect` *Detect the presence or absence of a pattern in a string*

Description

Dependency-free drop-in alternative for `stringr::str_detect()`.

Usage

```
str_detect(string, pattern, negate = FALSE)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.
`negate`	If `TRUE`, return non-matching elements.

Value

A logical vector.

Source

Adapted from the stringr package.

str_dup *Duplicate and concatenate strings within a character vector*

Description

Dependency-free drop-in alternative for `stringr::str_dup()`.

Usage

```
str_dup(string, times)
```

Arguments

`string`	Input character vector.
`times`	Number of times to duplicate each string.

Value

A character vector.

Source

Adapted from the stringr package.

str_ends *Detect the presence or absence of a pattern at the end of a string*

Description

Dependency-free drop-in alternative for `stringr::str_ends()`.

Usage

```
str_ends(string, pattern, negate = FALSE)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.
`negate`	If `TRUE`, return non-matching elements.

Value

A logical vector.

Source

Adapted from the stringr package.

`str_extract` *Extract matching patterns from a string*

Description

Dependency-free drop-in alternative for `stringr::str_extract()`.

Usage

```
str_extract(string, pattern)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.

Value

A character matrix. The first column is the complete match, followed by one column for each capture group.

Source

Adapted from the stringr package.

str_extract_all *Extract matching patterns from a string*

Description

Dependency-free drop-in alternative for `stringr::str_extract_all()`.

Usage

```
str_extract_all(string, pattern, simplify = FALSE)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.
`simplify`	If `FALSE`, the default, returns a list of character vectors. If `TRUE` returns a character matrix.

Value

A list of character vectors if `simplify = FALSE`, or a character matrix if `simplify = TRUE`.

Source

Adapted from the stringr package.

str_length *Compute the length of a string*

Description

Dependency-free drop-in alternative for `stringr::str_length()`.

Usage

```
str_length(string)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.

Value

A numeric vector the same length as string.

Source

Adapted from the stringr package.

`str_match` *Extract matched groups from a string*

Description

Dependency-free drop-in alternative for `stringr::str_match()`.

Usage

```
str_match(string, pattern)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.

Value

A character matrix. The first column is the complete match, followed by one column for each capture group.

Source

Adapted from the stringr package.

`str_pad` *Duplicate and concatenate strings within a character vector*

Description

Dependency-free drop-in alternative for `stringr::str_pad()`.

Usage

```
str_pad(
  string,
  width,
  side = c("left", "right", "both"),
  pad = " ",
  use_width = TRUE
)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`width`	Minimum width of padded strings.
`side`	Side on which padding character is added (left, right or both).
`pad`	Single padding character (default is a space).
`use_width`	If `FALSE`, use the length of the string instead of the width; see `str_width()`/`str_length()` for the difference.

Value

A character vector.

Author(s)

Eli Pousson <eli.pousson@gmail.com> (ORCID)

Alexander Rossell Hayes <alexander@rossellhayes.com> (ORCID)

Source

Adapted from the stringr package.

str_remove *Remove matched patterns in a string*

Description

Dependency-free drop-in alternative for `stringr::str_remove()`.

Usage

```
str_remove(string, pattern)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.

Value

A character vector.

Source

Adapted from the stringr package.

str_remove_all *Remove matched patterns in a string*

Description

Dependency-free drop-in alternative for `stringr::str_remove_all()`.

Usage

```
str_remove_all(string, pattern)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.

Value

A character vector.

Source

Adapted from the stringr package.

`str_replace` *Replace matched patterns in a string*

Description

Dependency-free drop-in alternative for `stringr::str_replace()`.

Usage

```
str_replace(string, pattern, replacement)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.
`replacement`	A character vector of replacements. Should be either length one, or the same length as `string` or `pattern`. References of the form `\1`, `\2`, etc. will be replaced with the contents of the respective matched group (created by `()`). To replace the complete string with `NA`, use `replacement = NA_character_`. Using a function for `replacement` is not yet supported.

Value

A character vector.

Source

Adapted from the stringr package.

str_replace_all *Replace matched patterns in a string*

Description

Dependency-free drop-in alternative for `stringr::str_replace_all()`.

Usage

```
str_replace_all(string, pattern, replacement)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.
`replacement`	A character vector of replacements. Should be either length one, or the same length as `string` or `pattern`. References of the form `\1`, `\2`, etc. will be replaced with the contents of the respective matched group (created by `()`). To perform multiple replacements in each element of `string`, pass a named vector (`c(pattern1 = replacement1)`) to `str_replace_all()`. To replace the complete string with `NA`, use `replacement = NA_character_`. Using a function for `replacement` is not yet supported.

Value

A character vector.

Source

Adapted from the stringr package.

str_replace_na *Turn NA into "NA"*

Description

Dependency-free drop-in alternative for `stringr::str_replace_na()`.

Usage

```
str_replace_na(string, replacement = "NA")
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`replacement`	A single string.

Value

A character vector.

`str_split` *Split up a string into pieces*

Description

Dependency-free drop-in alternative for `stringr::str_split()`.

Usage

```
str_split(string, pattern, n = Inf, simplify = FALSE)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.
`n`	Maximum number of pieces to return. Default (`Inf`) uses all possible split positions. This determines the maximum length of each element of the output.
`simplify`	A boolean. • `FALSE` (the default): returns a list of character vectors. • `TRUE`: returns a character matrix.

Value

A list the same length as `string`/`pattern` containing character vectors, or if `simplify` = `FALSE`, a character matrix with n columns and the same number of rows as the length of `string`/`pattern`.

Author(s)

Eli Pousson <eli.pousson@gmail.com> (ORCID)

Alexander Rossell Hayes <alexander@rossellhayes.com> (ORCID)

Source

Adapted from the stringr package.

`str_split_fixed` *Split up a string into pieces*

Description

Dependency-free drop-in alternative for `stringr::str_split_fixed()`.

Usage

```
str_split_fixed(string, pattern, n)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.
`n`	Maximum number of pieces to return. This determines the number of columns in the output; if an input is too short, the result will be padded with `""`.

Value

A character matrix with `n` columns and the same number of rows as the length of `string`/`pattern`.

Author(s)

Eli Pousson <eli.pousson@gmail.com> (ORCID)

Alexander Rossell Hayes <alexander@rossellhayes.com> (ORCID)

Source

Adapted from the stringr package.

str_squish *Remove whitespace*

Description

Dependency-free drop-in alternative for `stringr::str_squish()`.

Usage

```
str_squish(string)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.

Value

A character vector the same length as `string`.

Source

Adapted from the stringr package.

str_starts *Detect the presence or absence of a pattern at the beginning of a string*

Description

Dependency-free drop-in alternative for `stringr::str_starts()`.

Usage

```
str_starts(string, pattern, negate = FALSE)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.
`negate`	If `TRUE`, return non-matching elements.

Value

A logical vector.

str_subset *Keep strings matching a pattern*

Description

Dependency-free drop-in alternative for `stringr::str_subset()`.

Usage

```
str_subset(string, pattern, negate = FALSE)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.
`negate`	If `TRUE`, return non-matching elements.

Value

A character vector.

Source

Adapted from the stringr package.

str_trim *Remove whitespace*

Description

Dependency-free drop-in alternative for `stringr::str_trim()`.

Usage

```
str_trim(string, side = c("both", "left", "right"))
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`side`	Side on which to remove whitespace: `"left"`, `"right"`, or `"both"`, the default.

Value

A character vector the same length as `string`.

Source

Adapted from the stringr package.

`str_which` *Find positions of strings matching a pattern*

Description

Dependency-free drop-in alternative for `stringr::str_which()`.

Usage

```
str_which(string, pattern, negate = FALSE)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.
`pattern`	Pattern to look for. The default interpretation is a regular expression, as described in base::regex. Control options with `regex()`. Match a fixed string (i.e. by comparing only bytes), using `fixed()`. This is fast, but approximate.
`negate`	If `TRUE`, return non-matching elements.

Value

An integer vector.

Source

Adapted from the stringr package.

`str_width` *Compute the width of a string*

Description

Dependency-free drop-in alternative for `stringr::str_width()`. Results for non-ASCII characters may be inaccurate in R < 4.0.

Usage

```
str_width(string)
```

Arguments

`string`	Input vector. Either a character vector, or something coercible to one.

Value

A numeric vector the same length as string.

Source

Adapted from the stringr package.

Index