Package 'stringb'

November 1, 2016

Title Convenient Base R String Handling

Date 2016-11-01

Version 0.1.13

Description Base R already ships with string handling capabilities 'out-of-the-box' but lacks streamlined function names and workflow. The 'stringi' ('stringr') package on the other hand has well named functions, extensive Unicode support and allows for a streamlined workflow. On the other hand it adds dependencies and regular expression interpretation between base R functions and 'stringi' functions might differ. This packages aims at providing a solution to the use case of unwanted dependencies on the one hand but the need for streamlined text processing on the other. The packages' functions are solely based on wrapping base R functions into 'stringr'/'stringi' like function names. Along the way it adds one or two extra functions and last but not least provides all functions as generics, therefore allowing for adding methods for other text structures besides plain character vectors.

Depends R (>= 3.0.0)

License MIT + file LICENSE

LazyData TRUE

Imports graphics, tools, backports

Suggests testthat, knitr, rmarkdown

BugReports https://github.com/petermeissner/stringb/issues

URL https://github.com/petermeissner/stringb

RoxygenNote 5.0.1

NeedsCompilation no

Author Peter Meissner [aut, cre]

Maintainer Peter Meissner <retep.meissner@gmail.com>

Repository CRAN

Date/Publication 2016-11-01 15:18:21

R topics documented:

`invert_spans` *function to invert spans to those numbers not covered*

Description

function to invert spans to those numbers not covered

Usage

```
invert_spans(from, to = NULL, start = 1, end = Inf)
```

Arguments

`from`	vector of span starts
`to`	vector of span ends
`start`	minimum
`end`	maximum value

`plot.character` *function for plotting text*

Description

function for plotting text

Usage

```
## S3 method for class 'character'
plot(x, y = NULL, col = "grey", border = "grey",
  pattern = NULL, pattern_col = "#ED4C4C", ...)
```

Arguments

`x`	object of class rtext
`y`	either NULL or a data.frame with columns "start", "end", "line"
`col`	color for text
`border`	border color for text
`pattern`	regular expression to be searched in text and marked up in plot
`pattern_col`	color for text to be marked up via pattern or y option
`...`	further parameters passed through to text_locate

stringb_arrange *function to sort df by variables*

Description

function to sort df by variables

Usage

```
stringb_arrange(df, ...)
```

Arguments

df	data.frame to be sorted
...	column names to use for sorting

text_c *generic for concatenating strings*

Description

generic for concatenating strings

text_c default

Usage

```
text_c(..., sep = "", coll = NULL)

## Default S3 method:
text_c(..., sep = "", coll = NULL)
```

Arguments

...	one or more texts to be concatonated (see also paste)
sep	separator between concatonated elements (see also paste)
coll	if texts (not only there elements) are to be collapsed as well, how should the be separated (see also paste)

See Also

%..% and %.%

`text_collapse` *function for collapsing text vectors*

Description

function for collapsing text vectors

default method for text_collapse()

text_collapse() mehtod for list

text_collapse() method for data.frames

text_collapse() method for matrix

Usage

```
text_collapse(x, coll = "")

## Default S3 method:
text_collapse(x, coll = "")

## S3 method for class 'list'
text_collapse(x, coll = "")

## S3 method for class 'data.frame'
text_collapse(x, coll = "")

## S3 method for class 'matrix'
text_collapse(x, coll = "")
```

Arguments

`x`	object to be collapsed
`coll`	separator between collapsed text parts
`...`	additional parameter passed through to methods

`text_count` *generic for counting pattern occurences*

Description

generic for counting pattern occurences

text_count defaul method

Usage

```
text_count(string, pattern, sum = FALSE, vectorize = FALSE, ...)

## Default S3 method:
text_count(string, pattern, sum = FALSE,
  vectorize = FALSE, ...)
```

Arguments

string	text to search through
pattern	regex to search for
sum	if true all element-wise counts will be summed up
vectorize	should function be used in vectorized mode, i.e. should a pattern with length larger than 1 be allowed and if so, should it be matched to lines (with recycling if needed) instead of using on element on all lines
...	further arguments passed through to gregexpr

---

text_delete *deleting patterns in string*

Description

deleting patterns in string

deleting patterns in string

Usage

```
text_delete(string, pattern = NULL, ...)

## Default S3 method:
text_delete(string, pattern = NULL, ...)
```

Arguments

string	text to be replaced
pattern	regex to look for and delete
...	further parameter passed through to sub

text_detect *generic function to test if a regex can be found within a string*

Description

generic function to test if a regex can be found within a string

text_detect default method

generic function to test if a regex can be found within a string

Usage

```
text_detect(string, pattern, ...)

## Default S3 method:
text_detect(string, pattern, ...)

text_grepl(string, pattern, ...)
```

Arguments

string	text to be searched through
pattern	regex to look for
...	further arguments passed through to grepl

text_eval *wrapper function of eval() and parse() to evaluate character vector*

Description

wrapper function of eval() and parse() to evaluate character vector

Usage

```
text_eval(x, envir = parent.frame(), ...)
```

Arguments

x	character vector to be parsed and evaluated
envir	where to evaluate character vector
...	arguments passed through to eval()

text_extract *extract regex matches*

Description

wrapper function around regexec and regmatches

Usage

```
text_extract(x, pattern, ignore.case = FALSE, perl = FALSE, fixed = FALSE,
  useBytes = FALSE, invert = FALSE)
```

Arguments

x	text from which to extract
pattern	see grep
ignore.case	see grep
perl	see grep
fixed	see grep
useBytes	see grep
invert	if TRUE non-regex-matches are extracted instead

text_extract_all *extract regex matches*

Description

wrapper function around gregexec and regmatches

Usage

```
text_extract_all(x, pattern, ignore.case = FALSE, perl = FALSE,
  fixed = FALSE, useBytes = FALSE, invert = FALSE)
```

Arguments

x	text from which to extract
pattern	see grep
ignore.case	see grep
perl	see grep
fixed	see grep
useBytes	see grep
invert	if TRUE non-regex-matches are extracted instead

text_extract_group *generic for getting regex group matches*

Description

generic for getting regex group matches

text default

Usage

```
text_extract_group(string, pattern, group, invert = FALSE, ...)

## Default S3 method:
text_extract_group(string, pattern, group = NULL,
  invert = FALSE, ...)
```

Arguments

string	text from which to extract character sequence
pattern	regex to be searched for
group	integer vector to indicate those regex group matches to extract
invert	whether or no matches or non-matches should be extracted
...	further parameter passed through to regexec

text_extract_group_all *generic for getting all regex group matches*

Description

generic for getting all regex group matches

text default

Usage

```
text_extract_group_all(string, pattern, group = NULL, invert = FALSE, ...)

## Default S3 method:
text_extract_group_all(string, pattern, group = NULL,
  invert = FALSE, ...)
```

Arguments

string	text from which to extract character sequence
pattern	regex to be searched for
group	integer vector to indicate those regex group matches to extract
invert	whether or no matches or non-matches should be extracted
...	further parameter passed through to gregexpr

---

text_filter *generic for subsetting/filtering vectors*

Description

generic for subsetting/filtering vectors

Usage

```
text_filter(string, pattern, ...)
```

Arguments

string	text to be subsetted
pattern	regular expression to subset by
...	further arguments passed through to grep

---

text_length *wrapper around nchar to return text length*

Description

wrapper around nchar to return text length

Usage

```
text_length(x, type = "chars", allowNA = FALSE, keepNA = TRUE,
  na.rm = FALSE)
```

Arguments

x	see nchar
type	see nchar
allowNA	see nchar
keepNA	see nchar
na.rm	see nchar

text_locate *function to get start, end, length form pattern match*

Description

function to get start, end, length form pattern match

text_locate default

Usage

```
text_locate(string, pattern, vectorize = FALSE, ...)

## Default S3 method:
text_locate(string, pattern, vectorize = FALSE, ...)
```

Arguments

string	text to be searched through
pattern	regex to look for
vectorize	should function be used in vectorized mode, i.e. should a pattern with length larger than 1 be allowed and if so, should it be matched to lines (with recycling if needed) instead of using on element on all lines
...	further options passed through to regexpr

text_locate_all *function to get start, end, length form pattern match for all matches*

Description

function to get start, end, length form pattern match for all matches

text_locate_all default

Usage

```
text_locate_all(string, pattern, vectorize = FALSE, simplify = FALSE, ...)

## Default S3 method:
text_locate_all(string, pattern, vectorize = FALSE,
  simplify = FALSE, ...)
```

Arguments

string	text to search through
pattern	regex to search for
vectorize	should function be used in vectorized mode, i.e. should a pattern with length larger than 1 be allowed and if so, should it be matched to lines (with recycling if needed) instead of using on element on all lines
simplify	either getting back a list of results or all list elements merged into a data.frame with columns identifying original line (i) and pattern (p) number
...	further arguments passed through to gregexpr

---

text_locate_all_worker *helper function to get start, end, length form pattern match*

Description

helper function to get start, end, length form pattern match

Usage

```
text_locate_all_worker(string, pattern, ...)
```

Arguments

string	text to be searched through
pattern	regex to look for
...	further options passed through to regexpr

---

text_locate_group *generic for getting positions regex groups*

Description

generic for getting positions regex groups

text_locate_group default

Usage

```
text_locate_group(string, pattern, group, ...)

## Default S3 method:
text_locate_group(string, pattern, group, ...)
```

Arguments

string	text to be searched through
pattern	regex to look for
group	integer vector specifying groups to return
...	further options passed through to regexpr

---

text_locate_worker *helper function to get start, end, length form pattern match*

Description

helper function to get start, end, length form pattern match

Usage

```
text_locate_worker(string, pattern, ...)
```

Arguments

string	text to be searched through
pattern	regex to look for
...	further options passed through to regexpr

---

text_nchar *wrapper around nchar to return text length*

Description

wrapper around nchar to return text length

Usage

```
text_nchar(x, type = "chars", allowNA = FALSE, keepNA = TRUE)
```

Arguments

x	see nchar
type	see nchar
allowNA	see nchar
keepNA	see nchar

text_pad *padding text to specified width*

Description

padding text to specified width

text_wrap default

Usage

```
text_pad(string, width = max(nchar(string)), pad = " ", side = c("left",
  "right", "both", "l", "r", "b", 1, 2, 3))

## Default S3 method:
text_pad(string, width = max(nchar(string)), pad = " ",
  side = c("left", "right", "both", "l", "r", "b", 1, 2, 3))
```

Arguments

string	text to be wrapped
width	width text should have after padding; defaults to: max(nchar(string))
pad	the character or character sequence to use for padding
side	one of: c("left", "right", "both", "l", "r", "b", 1, 2, 3)

text_read *read in text*

Description

A wrapper to readLines() to make things more ordered and convenient. In comparison to the wrapped up readLines() function text_read() does some things differently: (1) If no encoding is given, it will always assume files are stored in UTF-8 instead of the system locale. (2) it will always converts text to UTF-8 instead of transforming it to the system locale. (3) in addition to loading, it offers to tokenize the text using a regular expression or NULL for no tokenization at all.

Usage

```
text_read(file, tokenize = "\n", encoding = "UTF-8", ...)
```

Arguments

file	name or path to the file to be read in or a connection object (see readLines)
tokenize	either NULL so that no splitting is done; a regular expression to use to split text into parts; or a function that does the splitting (or whatever other transformation)
encoding	character encoding of file passed throught to readLines
...	further arguments passed through to readLines like: n, ok, warn, skipNul

text_rep *generic repeating text*

Description

generic repeating text

text_rep defaul method

Usage

```
text_rep(string, times, vectorize = FALSE, ...)

text_dup(string, times, vectorize = FALSE, ...)

## Default S3 method:
text_rep(string, times, vectorize = FALSE, ...)
```

Arguments

string	text to be repeated
times	how many times shal string be repeated
vectorize	should function be used in vectorized mode, i.e. should a pattern with length larger than 1 be allowed and if so, should it be matched to lines (with recycling if needed) instead of using on element on all lines
...	further arguments passed through

text_replace *replacing patterns in string*

Description

replacing patterns in string

replacing patterns default

Usage

```
text_replace(string, pattern = NULL, replacement = NULL, ...)

## Default S3 method:
text_replace(string, pattern = NULL, replacement = NULL,
  recycle = FALSE, ...)
```

Arguments

string	text to be replaced
pattern	regex to look for
replacement	replacement for pattern found
...	further parameter passed through to sub
recycle	should arguments be recycled if lengths do not match?

---

text_replace_all *replacing patterns in string*

Description

replacing patterns in string

replacing patterns default

Usage

```
text_replace_all(string, pattern = NULL, replacement = NULL, ...)

## Default S3 method:
text_replace_all(string, pattern = NULL,
  replacement = NULL, recycle = FALSE, ...)
```

Arguments

string	text to be replaced
pattern	regex to look for
replacement	replacement for pattern found
...	further parameter passed through to gsub
recycle	should arguments be recycled if lengths do not match?

---

text_replace_group *function for replacing regex group matches generic for getting regex group matches*

Description

function for replacing regex group matches generic for getting regex group matches

text_replace_group default

Usage

```
text_replace_group(string, pattern, replacement,
  group = seq_along(replacement), invert = FALSE, ...)

## Default S3 method:
text_replace_group(string, pattern, replacement,
  group = TRUE, invert = FALSE, ...)
```

Arguments

string	text from which to extract character sequence
pattern	regex to be searched for
replacement	character vector of replacements of length 1 or length(group) to replace regex group matches (marked character spans provided by the found parameter)
group	vector of integers identifying thos regex groups to be replaced
invert	should character spans provided by found or their counterparts be replaced
...	further parameter passed through to regexec

---

text_replace_locates *text_replace_locates default*

Description

text_replace_locates default

text_replace_locates default

Usage

```
text_replace_locates(string, found, replacement, group, invert)

## Default S3 method:
text_replace_locates(string, found, replacement, group,
  invert)
```

Arguments

string	text for which to replace parts
found	result of an call to text_locate_group or text_locate - i.e. a list of data.frames with two columns named 'start' and 'end' that mark character spans to be replaced within the text elements
replacement	character vector of replacements of length 1 or length(group) to replace regex group matches (marked character spans provided by the found parameter)
group	vector of integers identifying thos regex groups to be replaced
invert	should character spans provided by found or their counterparts be replaced

text_show *showing text*

Description

shows text or portions of the text via cat and the usage of text_snippet()

text_show default

Usage

```
text_show(x, length = 500, from = NULL, to = NULL, coll = FALSE,
  wrap = FALSE, ...)

## Default S3 method:
text_show(x, length = 500, from = NULL, to = NULL,
  coll = FALSE, wrap = FALSE, ...)
```

Arguments

x	text to be shown
length	number of characters to be shown
from	show from ith character
to	show up to ith character
coll	should x be collapsed using newline character as binding?
wrap	should text be wrapped, or wrapped to certain width, or wrapped by certain function
...	further arguments passed through to cat

text_snippet *retrieving text snippet*

Description

function will give back snippets of text via using length, length and from, length and to, or from and to to specify the snippet

Usage

```
text_snippet(x, length = max(nchar(x)), from = NULL, to = NULL,
  coll = FALSE)
```

Arguments

x	character vector to be snipped
length	length of snippet
from	starting character
to	last character
coll	should a possible vector x with length > 1 collapsed with newline character as separator?

Functions

- text_snippet: retrieving text snippet

---

text_split *generic splitting strings*

Description

generic splitting strings

text_split defaul method

Usage

```
text_split(string, pattern, vectorize = FALSE, ...)

## Default S3 method:
text_split(string, pattern, vectorize = FALSE, ...)
```

Arguments

string	text to search through
pattern	regex to search for
vectorize	should function be used in vectorized mode, i.e. should a pattern with length larger than 1 be allowed and if so, should it be matched to lines (with recycling if needed) instead of using on element on all lines
...	further arguments passed through to gregexpr

text_split_n *generic splitting strings into pieces of length n*

Description

generic splitting strings into pieces of length n

text_split_n defaul method

Usage

```
text_split_n(string, n, vectorize = FALSE)

## Default S3 method:
text_split_n(string, n, vectorize = FALSE)
```

Arguments

string	text to search through
n	length of pieces
vectorize	should function be used in vectorized mode, i.e. should a pattern with length larger than 1 be allowed and if so, should it be matched to lines (with recycling if needed) instead of using on element on all lines

text_sub *generic for extracting characters sequences by position*

Description

generic for extracting characters sequences by position

text_sub default

Usage

```
text_sub(string, start = NULL, end = NULL)

## Default S3 method:
text_sub(string, start = NULL, end = NULL)
```

Arguments

string	text from which to extract character sequence
start	first character position
end	last character position

See Also

text_snippet

`text_subset` *generic for subsetting/filtering vectors*

Description

generic for subsetting/filtering vectors

Usage

```
text_subset(string, pattern, ...)
```

Arguments

`string`	text to be subsetted
`pattern`	regular expression to subset by
`...`	further arguments passed through to grep

`text_tokenize` *generic for gregexpr wrappers to tokenize text*

Description

generic for gregexpr wrappers to tokenize text

default method for text_tokenize generic

Usage

```
text_tokenize(string, regex = NULL, ignore.case = FALSE, fixed = FALSE,
  perl = FALSE, useBytes = FALSE, non_token = FALSE)

## Default S3 method:
text_tokenize(string, regex = NULL, ignore.case = FALSE,
  fixed = FALSE, perl = FALSE, useBytes = FALSE, non_token = FALSE)
```

Arguments

string	text to be tokenized
regex	regex expressing where to cut see (see gregexpr)
ignore.case	whether or not reges should be case sensitive (see gregexpr)
fixed	whether or not regex should be interpreted as is or as regular expression (see gregexpr)
perl	whether or not Perl compatible regex should be used (see gregexpr)
useBytes	byte-by-byte matching of regex or character-by-character (see gregexpr)
non_token	should information for non-token, i.e. those patterns by which the text was splitted, be returned as well

---

text_tokenize_lines *generic to tokenize text into lines*

Description

generic to tokenize text into lines

text_tokenize default

Usage

```
text_tokenize_lines(string, non_token = FALSE)

## Default S3 method:
text_tokenize_lines(string, non_token = FALSE)
```

Arguments

string	the text to be tokenized
non_token	whether or not token as well as non tokens shall be returned.

---

text_tokenize_sentences *generic to tokenize text into sentences*

Description

generic to tokenize text into sentences

text_tokenize default

Usage

```
text_tokenize_sentences(string, non_token = FALSE)

## Default S3 method:
text_tokenize_sentences(string, non_token = FALSE)
```

Arguments

string	the text to be tokenized
non_token	whether or not token as well as non tokens shall be returned.

---

text_tokenize_words *generic to tokenize text into words*

Description

A wrapper to text_tokenize that tokenizes text into words. Since using text_tokenize()'s option non_token might slow things down considerably this one purpose wrapper is a little more clever than the general implementation and hence much faster.

text_tokenize default

Usage

```
text_tokenize_words(string, non_token = FALSE)

## Default S3 method:
text_tokenize_words(string, non_token = FALSE)
```

Arguments

string	the text to be tokenized
non_token	whether or not token as well as non tokens shall be returned.

---

text_to_lower *function for make text lower case*

Description

function for make text lower case

default method for text_tolower()

Usage

```
text_to_lower(x)

## Default S3 method:
text_to_lower(x)
```

Arguments

x	text to be processed

---

text_to_title_case *function for make text lower case*

Description

function for make text lower case

default method for text_to_title_case.()

Usage

```
text_to_title_case(x)

## Default S3 method:
text_to_title_case(x)
```

Arguments

x	text to be processed

---

text_to_upper *function for make text lower case*

Description

function for make text lower case

default method for text_to_upper()

Usage

```
text_to_upper(x)

## Default S3 method:
text_to_upper(x)
```

Arguments

x	text to be processed

`text_trim` *trim spaces*

Description

trim spaces

trim spaces default

trim spaces list

trim spaces numeric

Usage

```
text_trim(string, side = c("both", "left", "right"), pattern = " ", ...)

## Default S3 method:
text_trim(string, side = c("both", "left", "right"),
  pattern = " ", ...)

## S3 method for class 'list'
text_trim(string, side = c("both", "left", "right"),
  pattern = " ", ...)

## S3 method for class 'numeric'
text_trim(string, side = c("both", "left", "right"),
  pattern = " ", ...)
```

Arguments

`string`	text to be trimmed
`side`	defaults to both might also be left, right, both or b, r, l to express where to trim pattern away
`pattern`	regex to look for
`...`	further arguments passed through to text_replace()

`text_which` *generic function to know in which elements a pattern can be found*

Description

generic function to know in which elements a pattern can be found

text_which default method

generic function to know in which elements a pattern can be found

Usage

```
text_which(string, pattern, ...)

## Default S3 method:
text_which(string, pattern, ...)

text_grep(string, pattern, ...)
```

Arguments

string	the text to be searched through
pattern	regex to look for
...	further arguments passed through to grepl

---

text_which_value *generic function to get whole elements in which pattern was found*

Description

generic function to get whole elements in which pattern was found

generic function to get whole elements in which pattern was found

text_which_value default method

Usage

```
text_which_value(string, pattern, ...)

text_grepv(string, pattern, ...)

## Default S3 method:
text_which_value(string, pattern, ...)
```

Arguments

string	the character vector to be searched through
pattern	regex to look for
...	further arguments passed through to grep

text_wrap *wraping text to specified width*

Description

wraping text to specified width

text_wrap default

Usage

```
text_wrap(string, ...)

## Default S3 method:
text_wrap(string, ...)
```

Arguments

string	text to be wrapped
...	further arguments passed through to strwrap

See Also

strwrap

text_write *write text to file*

Description

A generic function to write text to file (or a connection) and accompanying methods that wrap writeLines to do so. In contrast to vanilla writeLines() text_write() (1) is a generic so methods, handling something else than character vectors, can be implemented (2) in contrast to writeLines()' default to transform to write text in the system locale text_write() will default to UTF-8 no matter the locale (3) furthermore this encoding can be changed to any encoding supported by iconv (see also iconvlist)

text_write() default

Usage

```
text_write(string, file, sep = "\n", encoding = "UTF-8", ...)

## Default S3 method:
text_write(string, file, sep = "\n", encoding = "UTF-8",
  ...)
```

Arguments

string	text to be written
file	file name or file path or an connection object - passed through to writeLines()'s con argument
sep	character to separate lines (i.e. vector elements) from each other - passed through to writeLines()'s con argument
encoding	encoding in which to write text to disk
...	further arguments that might be passed to methods (not used at the moment)

%.% *concatenating strings operator*

Description

concatenating strings operator

Usage

```
a %.% b
```

Arguments

a	first text
b	second text

See Also

text_c (and paste)

%..% *concatenating strings*

Description

concatenating strings

Usage

```
a %..% b
```

Arguments

a	first text
b	first text

See Also

text_c (and paste)

Index