Package 'lettercase'

August 29, 2016

Title Utilities for Formatting Strings with Consistent Capitalization, Word Breaks and White Space

Version 0.13.1

Date 2016-03-03

Description Utilities for formatting strings and character vectors to for capitalization, word break and white space. Supported formats are: snake_case, spine-case, camelCase, PascalCase, Title Case, UPPERCASE, lowercase, Sentence case or combinations thereof. 'lettercase' strives to provide a simple, consistent, intuitive and high performing interface.

Depends R (>= 3.1.0), stringr (>= 1.0.0)

Suggests testthat, knitr, magrittr (>= 1.5)

License GPL-2 | file LICENSE

LazyData true

Collate 'lettercase-package.R' 'make_names.R' 'make_str_replace.R' 'patterns.R' 'str_cap_words.R' 'str_collapse_whitespace.R' 'str_delete.R' 'str_functions.R' 'str_is.R' 'str_lowercase.R' 'str_sentence_case.R' 'str_snake_case.R' 'str_spine_case.R' 'str_title_case.R' 'str_transform.R' 'str_ucfirst.R' 'str_uppercase.R' 'utils.R' 'zzz.R'

VignetteBuilder knitr

NeedsCompilation no

Author Christopher Brown [aut, cre], Decision Patterns [cph]

Maintainer Christopher Brown <chris.brown@decisionpatterns.com>

Repository CRAN

Date/Publication 2016-03-03 23:54:35

R topics documented:

`lettercase` *lettercase*

Description

Utilities for formatting strings according to common cases: upper case, lower case, snake case, camel case, spine case, title case, sentence case etc.

Details

The **lettercase** package provides utilities that help manage and maintain string formats related to:

- capitalization
- whitespace / word separating characters
- special characters
- acronyms

Most often, the common convenience functions will be used. These are:

- str_upper_case, str_uppercase, str_all_caps
- str_lower_case, str_lowercase
- str_cap_wordsloa

References

http://en.wikipedia.org/wiki/Letter_case http://en.wikipedia.org/wiki/Capitalization
http://perldoc.perl.org/perlrecharclass.html

make_names *make_names*

Description

Make syntactically valid names out from character vectors replacing . (dot) with _ underscore. Hhis is useful when you wish to use snake_case naming convention or are using SQL.

Usage

```
make_names(names, unique = FALSE, leading_ = "")
```

Arguments

names	character; vector to be coerced to syntactically valid names. This is coerced to character if necessary.
unique	logical; if TRUE, the resulting elements are unique. This may be desired for, e.g., column names.
leading_	What to replace leading '_' and '.' with. Can be only: A-Z, a-z, ., or "" (Defaults) Calls nake.names and then replaces . by _ See make.names for details. Multiple consecutive underscores are replaced by a single underscore. Names the end up with leading underscores are replaced with leading_ which can be a string of any length beginning with a letter or '.'. The default is to drop leading underscores. This function is idempotent – multiple application of the function do not change the results.

Value

a character vector containing

Author(s)

Christopher Brown

References

https://en.wikipedia.org/wiki/Snake_case
http://titlecase.com

See Also

make.names, make.unique

Examples

```
make_names(c("foo and bar", "foo-and-bar"), unique = TRUE)
  # "foo_and_bar"   "foo_and_bar_1"

  make_names(c("foo and bar", "foo.and_bar"), unique = FALSE)
  # "foo.and.bar"  "foo_and_bar"

  make_names(c("foo and bar", "foo.and_bar"), unique = TRUE)
  # "foo_and_bar"   "foo_and_bar_1"

  make_names( c(".foo", "_bar") )  # "foo" "bar"

  make_names( c(".foo", "_bar"), leading="." )  # ".foo" ".bar"
```

`make_str_replace` *Make replace, delete and "is" functions for strings*

Description

Functions for building string functions for replacement, deletion and testing

Usage

```
make_str_replace(pattern, replacement)

make_str_delete(pattern)
```

Arguments

`pattern`	pattern to look for, as defined by a POSIX regular expression. See the "Extended Regular Expressions" section of regex for details. See fixed, ignore.case and perl for how to use other types of matching: fixed, case insensitive and perl-compatible expressions.
`replacement`	string. References of the form `\1`, `\2` will be replaced with the contents of the respective matched group (created by ()) within the pattern.
	These functions build functions that take a single string argument and return a vector as a result.
	`make_str_replace` builds a functions that replaces the strings according to the `pattern` and `replacement` arguments.
	`make_str_delete` builds a functions that deletes the `pattern` from the string.
	`make_str_is` builds a function that detects is the string is has a certain type of formatting.

Examples

```
# -tk
```

pattern_whitespace *Common regular expression patterns used by lettercase*

Description

These are patterns that are used in the lettercase package.

Usage

```
pattern_whitespace

pattern_whitespace_like

pattern_separators

pattern_ucfirst

pattern_word

pattern_nonword

pattern_uppercase

pattern_lowercase
```

Format

```
chr "\\s"
```

Details

`lettercase` provides a number of regular expression patterns that are used by the case conversion functions.

These are not exported; to use them prefix them with `lettercase:::*`

str_cap_words *str_cap_words*

Description

Function used to convert character vectors to CapWord format.

Usage

```
str_cap_words(string)
```

Arguments

string	character vector to turn into a CapWords CapWords is distinguished by: 1. All words with upper case first character 2. No non-word characters allowed (letters/numbers only allowed) The recipe for chaging into capwords is: 1. replace whitespace and other word separators with space 2. str_ucfirst 3. strstdelete whitespace

Examples

```
# CAP WORDS
    str_cap_words( "One Flew Over The Cuckoo's Nest" )
    str_cap_words( "Catch-22" )  # CATCH
```

---

str_collapse_whitespace

Collpases multiple adjacent whitespace characters into one

Description

Collapses adjacent whitespace into a single character

Usage

```
str_collapse_whitespace(string, pattern = getOption("lettercase.whitespace",
  c(pattern_whitespace, pattern_whitespace_like)),
  replacement = getOption("lettercase.whitespace.replacement", "\\1"))

str_collapse_ws(string, pattern = getOption("lettercase.whitespace",
  c(pattern_whitespace, pattern_whitespace_like)),
  replacement = getOption("lettercase.whitespace.replacement", "\\1"))
```

Arguments

string	object to turn into a title case
pattern	character; one or more regular expression patterns for matching whitespace characters. Defaults to the `lettercase.whitespace` option or `\s, -, _` if the option has not been set.
replacement	character; the whitespace character to use for replacing. The default is "\1" – the character that matched.

Details

collapse_whitespace replaces repeated whitespace characters with a whitespace replacement. By default, it will find and replace any of multiple adjacent pattern characters with the replacement.

pattern can be a named patterns (see ?patterns) or character strings that are treated as regular expressions by default.

To collapse mixed whitespace, provide a single patterns in the form of a regular expression class, e.g. "[\s-_]". This can lead to confusing results if a back-reference is use as a replacement value. See note below.

Note

It is inadvisable to have pattern be a multiple-character regex class, e.g. "[\s_]" while also using a back reference replacement value, e.g. "\1". This leads to confusing results.

```
> str_collapse_whitespace( "A _B_ C", '[\s-_]', "\1" )      [1] "A_B C"
```

The solution is to either provide a character vector for a {pattern} or not use back-references for the replacement depending on the desired outcome.

```
>   str_collapse_whitespace( "A _B_ C", c("\s", "_") )      [1] "A _B_ C"
>   str_collapse_whitespace( "A _B_ C", '[\s-_]', " " )      [1] "A B C"
```

See Also

?patterns
gsub which is used to implement this function.

Examples

```
str_collapse_whitespace( "A  B" )
  str_collapse_whitespace( "A  B  C" )
  str_collapse_whitespace( "A__B__C" )
  str_collapse_whitespace( "A  B__C" )
  str_collapse_whitespace( "A _B_ C" )  # No transformation, no matches

  # See note above:
  str_collapse_whitespace( "A _B_ C", '[\\s-_]' ) # possibly ill-defined
  str_collapse_whitespace( "A _B_ C", c("\\s", "_") )
  str_collapse_whitespace( "A _B_ C", '[\\s-_]', " " )
```

---

str_delete *str_delete*

Description

Delete or remove characters from a string based on one or more patterns

Usage

```
str_delete(string, ...)
```

Arguments

string	atomic character vector
...	stringr-style matching modifiers

Details

Deletes all occurences of the patterns from the string using str_replace_all

References

modifiers
str_replace_all

Examples

```
str_delete( "ABC & 123", stringr::regex("\\W") )  # ABC123
```

---

str_delete_whitespace *Common string transformations*

Description

Perform common transformations on strings

Usage

```
str_delete_whitespace(string)

str_delete_separators(string)

str_delete_nonword(string)

str_expand_capwords(string)

str_delete_leading_nonword(string)

str_delete_space(string)
```

Arguments

string	character vector

These function take a single character vector argument and return a character vector that has had one or more functions applied to it. They are the building blocks for building up case transformations.

Value

character vector

Examples

```
# TRANSFORMATIONS
  str_ucfirst( "abc def" )              # Abc Def
  str_expand_capwords( "AbcDef")        # Abc Def

# DELETION
  str_delete_whitespace( "ABC 123" )    # ABC123
  str_delete_separators( "A_B-C.123" )  # ABC123
  str_delete_nonword( "ABC & 123" )     # ABC123
```

`str_is` *Test whether strings are of the specified type*

Description

Family of functions for testing whether strings are of the specified type.

Usage

```
str_is(string, type, autostart = 30L)

str_is_all(...)

make_str_is(type)

make_str_are(type)

str_are_uppercase(string)

str_are_upper_case(string)

str_are_uppercase(string)

str_are_upper_case(string)
```

Arguments

`string`	vector. This must be an atomic vector, and will be coerced to a character vector
`type`	function that transforms the string to type
`autostart`	integer; number of elements to test before concluding that `string` is of the suggested `type`. (Default: 30)
`...`	arguments passed to subsequent functions

Details

`str_is` determines if the string belongs to one of the supported lettercase types. Comparisons are made by comparing the original string to a transformed version of the string. If the two are identical, then the functions are equivalent.

`autostart` determines the maximum number of values

`make_str_is` and `make_string_are` are metafunctions that return functions that test for the given lettercase types.

Value

For `str_is` a logical vector indicating which entries of `string` are of the specified `type`,

For `str_is_all` and `str_are` a logical vector of length one indicating whether all of `string` is of the specified type(s)

For `make_str_is` and `make_str_are` functions that return functions that accept a single argument and return whether the string is of the specified types.

See Also

`str_transform`
`make_str_replace`
`make_str_delete`

Examples

```
string = c( "catch-22", "finnegans wake" )
  str_is( string, str_lower_case )

  str_transform( string, str_capitalize, str_delete_nonword )
  str_delete_nonword( str_capitalize( string ) )      # SAME

  # magrittr:
  ## Not run:
    string %>% str_capitalize %>% str_delete_nonword   # SAME

## End(Not run)
```

`str_lowercase` *str_uppercase, str_lowercases*

Description

convert string to upper or lower case

Usage

```
str_lowercase(string)

str_lower_case(string)

str_lower(string)

str_decapitalize(string)

is_lowercase(string)

is_lower_case(string)

is_lower(string)

str_uppercase(string)

str_upper_case(string)

str_upper(string)

str_all_caps(string)

str_allcaps(string)

str_capitalize(string)

is_uppercase(string)

is_upper_case(string)

is_upper(string)

is_all_caps(string)

is_allcaps(string)
```

Arguments

`string`	character; argument to be converted

Upper Case and All Caps

uppercase, **upper_case**, **allcaps**, **all_caps** are all synonyms.

upper case or `all caps` contains all capital letters.

Lower Case

`str_all_caps` is a synonym for `str_uppercase`. `is_all_caps` is a synonym for `is_upper_case`.

See Also

`str_is`

Examples

```
str_decapitalize( "ABC" )     # abc
# is_lower_case
  is_lowercase( 'ABC123' )      # FALSE
  is_lowercase( 'abc123' )      # TRUE
  is_lowercase( 'aB'  )         # FALSE
  is_lowercase( '123' )         # TRUE
# str_uppercase
  str_uppercase( "one flew over the cuckoo's nest" )
  str_uppercase( "catch-22" )

  str_capitalize( "abc" )               # ABC
  str_all_caps( "abc" )                 # ABC
# is_uppercase
  is_uppercase( 'ABC123' )      # TRUE
  is_uppercase( 'abc123' )      # FALSE
  is_uppercase( 'aB'  )         # FALSE
  is_uppercase( '123' )         # TRUE
```

`str_sentence_case` *str_sentence_case*

Description

Format a string in sentence case

Usage

```
str_sentence_case(string)
```

Arguments

`string`	character string to transform
	Sentance case is ...

str_snake_case *str_snake_case*

Description

Function used to convert character vectors to snake case format.

Usage

```
str_snake_case(string, whitespace = getOption("lettercase.whitespace",
  "[^\\w\\s-\\.]"))
```

Arguments

string	object to turn into a title case
whitespace	regular expression pattern to match for white-space * characters are all lower case * non- \w, \s and - are dropped * \w and - are converted to underscore * no support for acronyms * multiple adjacent underscores are replaced by single underscore * Underscores at beginning or end of names are dropped

Examples

```
str_snake_case( "One Flew Over The Cuckoo's Nest" )
  str_snake_case( "Catch-22" )  # catch_22
  str_snake_case( "Catch.22" )
  str_snake_case( "Catch_22" )
  str_snake_case( "Catch  22" )
  str_snake_case( " Catch 22 " )
```

str_spine_case *str_spine_case*

Description

Function used to convert character vectors to spine case format.

Usage

```
str_spine_case(string, whitespace = getOption("lettercase.whitespace",
  "[\\s-_]"), collapse.whitespace = getOption("collapse.whitespace", TRUE))

str_spinal_case(string, whitespace = getOption("lettercase.whitespace",
  "[\\s-_]"), collapse.whitespace = getOption("collapse.whitespace", TRUE))

str_hyphen_case(string, whitespace = getOption("lettercase.whitespace",
  "[\\s-_]"), collapse.whitespace = getOption("collapse.whitespace", TRUE))
```

Arguments

string	object to turn into a title case
whitespace	character; regular expression pattern for matching whitespace characters
collapse.whitespace	logical; whether adjacent whitespace is collapsed

* characters are all lower case * non- \w, \s and - are dropped * \w and - are converted to underscore * no support for acronyms

Examples

```
str_spine_case( "One Flew Over The Cuckoo's Nest" ) # One Flew Over The Cuckoo's Nest
  str_spine_case( "Catch 22" )  # catch-22
  str_spine_case( "Catch_ 22" )
```

---

str_title_case *str_title_case*

Description

internal function for converting character to title case. This is not exported and should not be called directly.

Usage

```
str_title_case(x)
```

Arguments

x	object to turn into a title case

Examples

```
str_title_case( "One Flew Over The Cuckoo's Nest" )
  str_title_case( "one_flew_over_the_cuckoo_'_s_nest" )
```

str_transform *str_transform*

Description

Convert a string by applying one or more str_* functions.

Usage

```
str_transform(string, ...)
```

Arguments

string	vector. This must be an atomic vector, and will be coerced to a character vector
...	one or more functions to apply to the string

Details

str_transform applies successive functions to its first argument, string.

Value

a character vector

See Also

make_str_replace make_str_delete

Examples

```
string = c( "catch-22", "finnegans wake" )
  str_transform( string, str_capitalize )

  str_transform( string, str_capitalize, str_delete_nonword )
  str_delete_nonword( str_capitalize( string ) )      # SAME

  ## Not run:
    # magrittr:
    string %>% str_capitalize %>% str_delete_nonword   # SAME

## End(Not run)
```

str_ucfirst *str_ucfirst*

Description

Convert first character of a string to uppercase

Usage

```
str_ucfirst(string)

is_ucfirst(string)
```

Arguments

string	character vector to be converted.

See Also

str_title_case

Examples

```
str_ucfirst( "one flew over the cuckoo's nest" )
 str_ucfirst( "catch-22" )
 str_ucfirst( "Portrait of the Artist as a Young Man" )
# is_ucfirst
  is_ucfirst( 'ABC123' )      # TRUE
  is_ucfirst( 'abc123' )      # FALSE
  is_ucfirst( 'aBC'  )        # FALSE
  is_ucfirst( 'Abc' )         # TRUE
  is_ucfirst( 'Abc dEF' )     # FALSE
  is_ucfirst( '123' )         # TRUE
```

Index