Package 'textTools'

July 22, 2025

Type Package

Title Functions for Text Cleansing and Text Analysis

Version 0.1.0

Author Timothy Conwell

Maintainer Timothy Conwell <timconwell@gmail.com>

Description A framework for text cleansing and analysis. Conveniently prepare and process large amounts of text for analysis.
Includes various metrics for word counts/frequencies that scale efficiently. Quickly analyze large amounts of text data using a text.table (a data.table created with one word (or unit of text analysis) per row, similar to the tidytext format).
Offers flexibility to efficiently work with text data stored in vectors as well as text data formatted as a text.table.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0), data.table

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-02-05 09:00:05 UTC

Contents

`as.text.table` *Convert a data.table column of character vectors into a column with one row per word grouped by a grouping column. Optionally will split a column of strings into vectors of constituents.*

Description

Convert a data.table column of character vectors into a column with one row per word grouped by a grouping column. Optionally will split a column of strings into vectors of constituents.

Usage

```
as.text.table(x, text, split = NULL, group_by = NULL)
```

Arguments

`x`	A data.table.
`text`	A string, the name of the column in x containing text to un-nest.
`split`	A string with a pattern to split the text in text column into constituent parts.
`group_by`	A vector of column names to group by. Doesn't work if the group by column is a list column.

Value

A data.table, text column un-nested to one row per word.

Examples

```
as.text.table(
  x = as.data.table(
    list(
      col1 = c(
        "a",
        "b"
      ),
      col2 = c(
        tolower("The dog is nice because it picked up the newspaper."),
        tolower("The dog is extremely nice because it does the dishes.")
      )
    )
  ),
  text = "col2",
  split = " "
)
```

flag_words *Flag rows in a text.table with specific words*

Description

Flag rows in a text.table with specific words

Usage

```
flag_words(x, text, flag = "flag", words)
```

Arguments

x	A text.table created by as.text.table().
text	A string, the name of the column in x to check for words to flag.
flag	A string, the name of the column created with the flag indicator.
words	A vector of words to flag x.

Value

A text.table, with rows marked with a 1 if the words in those rows are in the vector of words to delete, otherwise 0.

Examples

```
flag_words(
as.text.table(
  x = as.data.table(
    list(
      col1 = c(
        "a",
        "b"
      ),
      col2 = c(
        tolower("The dog is nice because it picked up the newspaper."),
        tolower("The dog is extremely nice because it does the dishes.")
      )
    )
  ),
  text = "col2",
  split = " "
),
text = "col2",
flag = "is_stopword",
words = stopwords
)
```

label_parts_of_speech *Add a column with the parts of speech for each word in a text.table*

Description

Add a column with the parts of speech for each word in a text.table

Usage

```
label_parts_of_speech(x, text)
```

Arguments

x	A text.table created by as.text.table().
text	A string, the name of the column in x to label the parts of speech.

Value

A text.table, with columns added for the matching part of speech and for flagging if the part of speech is for a multi-word phrase.

Examples

```
label_parts_of_speech(
as.text.table(
  x = as.data.table(
    list(
      col1 = c(
        "a",
        "b"
      ),
      col2 = c(
        tolower("The dog is nice because it picked up the newspaper."),
        tolower("The dog is extremely nice because it does the dishes.")
      )
    )
  ),
  text = "col2",
  split = " "
),
text = "col2"
)
```

l_pos *Parts of speech for English words from the Moby Project.*

Description

Parts of speech for English words/phrases from the Moby Project by Grady Ward. Words with non-ASCII characters have been removed. One row per word.

Usage

```
l_pos
```

Format

Data.table with 227519 rows and 3 variables #'

word Lowercase English word or phrase

pos Lowercase English part of speech, grouped by word into a vector if a word has multiple parts of speech.

multi_word TRUE if the word record has a space (contains multiple words), else FALSE.

Source

https://archive.org/details/mobypartofspeech03203gut

ngrams *Create n-grams*

Description

Create n-grams

Usage

```
ngrams(
  x,
  text,
  group_by = c(),
  count_col_name = "count",
  n,
  ngram_prefix = NULL
)
```

Arguments

x	A text.table created by as.text.table().
text	A string, the name of the column in x to build n-grams with.
group_by	A vector of column names to group by. Doesn't work if the group by column is a list column.
count_col_name	A string, the name of the output column containing the number of times each base record appears in the group.
n	A integer, the number of grams to make.
ngram_prefix	A string, a prefix to add to the output n-gram columns.

Value

A text.table, with columns added for n-grams (the word, the count, and percent of the time the gram follows the word).

Examples

```
ngrams(
as.text.table(
  x = as.data.table(
    list(
      col1 = c(
        "a",
        "b"
      ),
      col2 = c(
        tolower("The dog is nice because it picked up the newspaper."),
        tolower("The dog is extremely nice because it does the dishes.")
      )
    )
  ),
  text = "col2",
  split = " "
),
text = "col2",
group_by = "col1",
n = 2
)
```

---

pos *Parts of speech for English words from the Moby Project.*

Description

Parts of speech for English words/phrases from the Moby Project by Grady Ward. Words with non-ASCII characters have been removed. One row per word + part of speech

Usage

```
pos
```

Format

Data.table with 246690 rows and 3 variables #'

word Lowercase English word or phrase

pos Lowercase English part of speech, one per row

multi_word TRUE if the word record has a space (contains multiple words), else FALSE.

Source

https://archive.org/details/mobypartofspeech03203gut

---

regex_paragraph — *Regular expression that might be used to split strings of text into component paragraphs.*

Description

"\n", A regular expression to split strings when encountering a new line.

Usage

```
regex_paragraph
```

Format

A string

---

regex_sentence — *Regular expression that might be used to split strings of text into component sentences.*

Description

"[.?!]\s", A regular expression to split strings when encountering a common end of sentence punctuation pattern.

Usage

```
regex_sentence
```

Format

A string

regex_word — *Regular expression that might be used to split strings of text into component words.*

Description

" ", A regular expression to split strings when encountering a space.

Usage

```
regex_word
```

Format

A string

rm_frequent_words — *Delete rows in a text.table where the number of identical records within a group is more than a certain threshold*

Description

Delete rows in a text.table where the number of identical records within a group is more than a certain threshold

Usage

```
rm_frequent_words(
  x,
  text,
  count_col_name = NULL,
  group_by = c(),
  max_count,
  max_count_is_ratio = FALSE,
  total_count_col = NULL
)
```

Arguments

x	A text.table created by as.text.table().
text	A string, the name of the column in x used to determine deletion of rows based on the term frequency.
count_col_name	A string, the name to assign to the new column containing the count of each word. If NULL, does not return the counts.

group_by A vector of column names to group by. Doesn't work if the group by column is a list column.

max_count A number, the maximum number of times a word can occur to keep.

max_count_is_ratio TRUE/FALSE, if TRUE, implies the value passed to max_count should be considered a ratio.

total_count_col Name of the column containing the denominator (likely total count of records within a group) to use to calculate the ratio of a word count vs total if max_count_is_ratio is TRUE.

Value

A text.table, with rows having a duplicate count over a certain threshold deleted.

Examples

```
rm_frequent_words(
as.text.table(
  x = as.data.table(
    list(
      col1 = c(
        "a",
        "b"
      ),
      col2 = c(
        tolower("The dog is nice because it picked up the newspaper."),
        tolower("The dog is extremely nice because it does the dishes.")
      )
    )
  ),
  text = "col2",
  split = " "
),
text = "col2",
count_col_name = "count",
max_count = 1
)
```

---

rm_infrequent_words *Delete rows in a text.table where the number of identical records within a group is less than a certain threshold*

Description

Delete rows in a text.table where the number of identical records within a group is less than a certain threshold

Usage

```
rm_infrequent_words(
  x,
  text,
  count_col_name = NULL,
  group_by = c(),
  min_count,
  min_count_is_ratio = FALSE,
  total_count_col = NULL
)
```

Arguments

x	A text.table created by as.text.table().
text	A string, the name of the column in x used to determine deletion of rows based on the term frequency.
count_col_name	A string, the name to assign to the new column containing the count of each word. If NULL, does not return the counts.
group_by	A vector of column names to group by. Doesn't work if the group by column is a list column.
min_count	A number, the minimum number of times a word must occur to keep.
min_count_is_ratio	TRUE/FALSE, if TRUE, implies the value passed to min_count should be considered a ratio.
total_count_col	Name of the column containing the denominator (likely total count of records within a group) to use to calculate the ratio of a word count vs total if min_count_is_ratio is TRUE.

Value

A text.table, with rows having a duplicate count of less than a certain threshold deleted.

Examples

```
rm_infrequent_words(
as.text.table(
  x = as.data.table(
    list(
      col1 = c(
        "a",
        "b"
      ),
      col2 = c(
        tolower("The dog is nice because it picked up the newspaper."),
        tolower("The dog is extremely nice because it does the dishes.")
      )
    )
```

```
  ),
  text = "col2",
  split = " "
),
text = "col2",
count_col_name = "count",
min_count = 4
)

rm_infrequent_words(
as.text.table(
  x = as.data.table(
    list(
      col1 = c(
        "a",
        "b"
      ),
      col2 = c(
        tolower("The dog is nice because it picked up the
        newspaper and it is the nice kind of dog."),
        tolower("The dog is extremely nice because it does the dishes
        and it is cool.")
      )
    )
  ),
  text = "col2",
  split = " "
),
text = "col2",
count_col_name = "count",
group_by = "col1",
min_count = 2
)
```

`rm_long_words` *Delete rows in a text.table where the word has more than a minimum number of characters*

Description

Delete rows in a text.table where the word has more than a minimum number of characters

Usage

```
rm_long_words(x, text, max_char_length)
```

Arguments

x	A text.table created by as.text.table().

`text` A string, the name of the column in x used to determine deletion of rows based on the number of characters.

`max_char_length` A number, the maximum number of characters allowed to not delete a row.

Value

A text.table, with rows having more than a certain number of characters deleted.

Examples

```
rm_long_words(
as.text.table(
  x = as.data.table(
    list(
      col1 = c(
        "a",
        "b"
      ),
      col2 = c(
        tolower("The dog is nice because it picked up the newspaper."),
        tolower("The dog is extremely nice because it does the dishes.")
      )
    )
  ),
  text = "col2",
  split = " "
),
text = "col2",
max_char_length = 4
)
```

---

rm_no_overlap *Delete rows in a text.table where the records within a group are not also found in other groups (overlapping records)*

Description

Delete rows in a text.table where the records within a group are not also found in other groups (overlapping records)

Usage

```
rm_no_overlap(x, text, group_by = c())
```

Arguments

x	A text.table created by as.text.table().
text	A string, the name of the column in x to determine deletion of rows based on the lack of presence of overlapping records.
group_by	A vector of column names to group by. Doesn't work if the group by column is a list column.

Value

A text.table, with rows not having records found in multiple groups (overlapping records) deleted.

Examples

```
rm_no_overlap(
as.text.table(
  x = as.data.table(
    list(
      col1 = c(
        "a",
        "b"
      ),
      col2 = c(
        tolower("The dog is nice because it picked up the newspaper."),
        tolower("The dog is extremely nice because it does the dishes.")
      )
    )
  ),
  text = "col2",
  split = " "
),
text = "col2",
group_by = "col1"
)
```

---

rm_overlap *Delete rows in a text.table where the records within a group are also found in other groups (overlapping records)*

Description

Delete rows in a text.table where the records within a group are also found in other groups (overlapping records)

Usage

```
rm_overlap(x, text, group_by = c())
```

Arguments

x	A text.table created by as.text.table().
text	A string, the name of the column in x to determine deletion of rows based on the presence of overlapping records.
group_by	A vector of column names to group by. Doesn't work if the group by column is a list column.

Value

A text.table, with rows having records found in multiple groups (overlapping records) deleted.

Examples

```
rm_overlap(
as.text.table(
  x = as.data.table(
    list(
      col1 = c(
        "a",
        "b"
      ),
      col2 = c(
        tolower("The dog is nice because it picked up the newspaper."),
        tolower("The dog is extremely nice because it does the dishes.")
      )
    )
  ),
  text = "col2",
  split = " "
),
text = "col2",
group_by = "col1"
)
```

---

rm_parts_of_speech *Delete rows in a text.table where the word has a certain part of speech*

Description

Delete rows in a text.table where the word has a certain part of speech

Usage

```
rm_parts_of_speech(
  x,
  text,
  pos_delete = c("adjective", "adverb", "conjunction", "definite article",
    "interjection", "noun", "noun phrase", "plural", "preposition", "pronoun",
    "verb (intransitive)", "verb (transitive)", "verb (usu participle)")
)
```

Arguments

x	A text.table created by as.text.table().
text	A string, the name of the column in x used to determine deletion of rows based on the part of speech.
pos_delete	A vector of parts of speech to delete. At least one of the following: 'adjective', 'adverb', 'conjunction', 'definite article', 'interjection', 'noun', 'noun phrase', 'plural', 'preposition', 'pronoun', 'verb (intransitive)', 'verb (transitive)', 'verb (usu participle)'

Value

A text.table, with rows matching a part of speech deleted.

Examples

```
rm_parts_of_speech(
as.text.table(
  x = as.data.table(
    list(
      col1 = c(
        "a",
        "b"
      ),
      col2 = c(
        tolower("The dog is nice because it picked up the newspaper."),
        tolower("The dog is extremely nice because it does the dishes.")
      )
    )
  ),
  text = "col2",
  split = " "
),
text = "col2",
pos_delete = "conjunction"
)
```

---

rm_regexp_match *Delete rows in a text.table where the record has a certain pattern indicated by a regular expression*

Description

Delete rows in a text.table where the record has a certain pattern indicated by a regular expression

Usage

```
rm_regexp_match(x, text, pattern)
```

Arguments

x	A text.table created by as.text.table().
text	A string, the name of the column in x used to determine deletion of rows based on the regular expression.
pattern	A regular expression, gets passed to grepl().

Value

A text.table, with rows having a certain pattern indicated by a regular expression deleted.

Examples

```
rm_regexp_match(
as.text.table(
  x = as.data.table(
    list(
      col1 = c(
        "a",
        "b"
      ),
      col2 = c(
        tolower("The dog is nice because it picked up the newspaper."),
        tolower("The dog is extremely nice because it does the dishes.")
      )
    )
  ),
  text = "col2",
  split = " "
),
text = "col2",
pattern = "do"
)
```

---

rm_short_words *Delete rows in a text.table where the word has less than a minimum number of characters*

Description

Delete rows in a text.table where the word has less than a minimum number of characters

Usage

```
rm_short_words(x, text, min_char_length)
```

Arguments

x	A text.table created by as.text.table().
text	A string, the name of the column in x used to determine deletion of rows based on the number of characters.
min_char_length	A number, the minimum number of characters required to not delete a row.

Value

A text.table, with rows having less than a certain number of characters deleted.

Examples

```
rm_short_words(
as.text.table(
  x = as.data.table(
    list(
      col1 = c(
        "a",
        "b"
      ),
      col2 = c(
        tolower("The dog is nice because it picked up the newspaper."),
        tolower("The dog is extremely nice because it does the dishes.")
      )
    )
  ),
  text = "col2",
  split = " "
),
text = "col2",
min_char_length = 4
)
```

---

rm_words *Remove rows from a text.table with specific words*

Description

Remove rows from a text.table with specific words

Usage

```
rm_words(x, text, words = stopwords)
```

Arguments

x	A text.table created by as.text.table().
text	A string, the name of the column in x to check for words to delete.
words	A vector of words to delete from x.

Value

A text.table, with rows deleted if the words in those rows are in the vector of words to delete.

Examples

```
rm_words(
as.text.table(
  x = as.data.table(
    list(
      col1 = c(
        "a",
        "b"
      ),
      col2 = c(
        tolower("The dog is nice because it picked up the newspaper."),
        tolower("The dog is extremely nice because it does the dishes.")
      )
    )
  ),
  text = "col2",
  split = " "
),
text = "col2"
)
```

---

sampleStr *Generates (pseudo)random strings of the specified char length*

Description

Generates (pseudo)random strings of the specified char length

Usage

```
sampleStr(char)
```

Arguments

char	A integer, the number of chars to include in the output string.

Value

A string.

Examples

```
sampleStr(10)
```

stopwords — *Vector of lowercase English stop words.*

Description

Unique lowercase English stop words from 3 lexicons combined into one vector. Combines snowball, onix, and SMART lists of stopwords.

Usage

```
stopwords
```

Format

A vector of 728 unique English stop words in lowercase

Source

http://snowball.tartarus.org/algorithms/english/stop.txt

http://www.lextek.com/manuals/onix/stopwords1.html

http://www.lextek.com/manuals/onix/stopwords2.html

str_any_match — *Detect if there are any words in a vector also found in another vector.*

Description

Detect if there are any words in a vector also found in another vector.

Usage

```
str_any_match(x, y)
```

Arguments

x	A vector of words.
y	A vector of words to test against.

Value

TRUE/FALSE, TRUE if any words in x are also in y

Examples

```
str_any_match(
x = c("a", "dog", "went", "to", "the", "store"),
y = c("the")
)
str_any_match(
x = c("a", "dog", "went", "to", "the", "store"),
y = c("apple")
)
```

`str_counts` *Create a list of a vector of unique words found in x and a vector of the counts of each word in x.*

Description

Create a list of a vector of unique words found in x and a vector of the counts of each word in x.

Usage

```
str_counts(x)
```

Arguments

x	A vector of words.

Value

A list, position one is a vector of unique and sorted words from x, position two is a vector of the counts for each word.

Examples

```
str_counts(
x = c("a", "dog", "went", "to", "the", "store", "and", "a", "dog", "went", "to", "another", "store")
)
```

str_count_intersect *Count the intersecting words in a vector that are found in another vector (only counts unique words).*

Description

Count the intersecting words in a vector that are found in another vector (only counts unique words).

Usage

```
str_count_intersect(x, y)
```

Arguments

x	A vector of words.
y	A vector of words to test against.

Value

A number, the count of unique words in x also in y

Examples

```
str_count_intersect(
x = c("a", "dog", "went", "to", "the", "store"),
y = c("dog", "to", "store")
)
```

str_count_jaccard_similarity *Calculates the intersect divided by union of two vectors of words.*

Description

Calculates the intersect divided by union of two vectors of words.

Usage

```
str_count_jaccard_similarity(x, y)
```

Arguments

x	A vector of words.
y	A vector of words to test against.

Value

A number, the intersect divided by union of two vectors of words.

Examples

```
str_count_jaccard_similarity(
x = c("a", "dog", "went", "to", "the", "store"),
y = c("this", "dog", "went", "to", "the", "store")
)
```

`str_count_match` *Count the words in a vector that are found in another vector.*

Description

Count the words in a vector that are found in another vector.

Usage

```
str_count_match(x, y, ratio = FALSE)
```

Arguments

`x`	A vector of words.
`y`	A vector of words to test against.
`ratio`	TRUE/FALSE, if TRUE, returns the number of words in x with a match in y divided by the number of words in x.

Value

A number, the count of words in x also in y

Examples

```
str_count_match(
x = c("a", "dog", "went", "to", "the", "store"),
y = c("dog", "to", "store")
)
str_count_match(
x = c("a", "dog", "went", "to", "the", "store"),
y = c("dog", "to", "store"),
ratio = TRUE
)
```

str_count_nomatch *Count the words in a vector that are not found in another vector.*

Description

Count the words in a vector that are not found in another vector.

Usage

```
str_count_nomatch(x, y, ratio = FALSE)
```

Arguments

x	A vector of words.
y	A vector of words to test against.
ratio	TRUE/FALSE, if TRUE, returns the number of words in x without a match in y divided by the number of words in x.

Value

A number, the count of words in x and not in y

Examples

```
str_count_nomatch(
x = c("a", "dog", "went", "to", "the", "store"),
y = c("dog", "to", "store")
)
str_count_nomatch(
x = c("a", "dog", "went", "to", "the", "store"),
y = c("dog", "store"),
ratio = TRUE
)
```

str_count_positional_match *Count words from a vector that are found in the same position in another vector.*

Description

Count words from a vector that are found in the same position in another vector.

Usage

```
str_count_positional_match(x, y, ratio = FALSE)
```

Arguments

x	A vector of words.
y	A vector of words to test against.
ratio	TRUE/FALSE, if TRUE, returns the number of words in x with a positional match in y divided by the number of words in x.

Value

A count of the words in x with matches in the same position in y.

Examples

```
str_count_positional_match(
x = c("a", "dog", "went", "to", "the", "store"),
y = c("this", "dog", "ran", "from", "the", "store")
)
```

---

str_count_positional_nomatch

Count words from a vector that are not found in the same position in another vector.

Description

Count words from a vector that are not found in the same position in another vector.

Usage

```
str_count_positional_nomatch(x, y, ratio = FALSE)
```

Arguments

x	A vector of words.
y	A vector of words to test against.
ratio	TRUE/FALSE, if TRUE, returns the number of words in x without a positional match in y divided by the number of words in x.

Value

A count of the words in x without matches in the same position in y.

Examples

```
str_count_positional_nomatch(
x = c("a", "cool", "dog", "went", "to", "the", "store"),
y = c("a", "dog", "ran", "from", "the", "store")
)
```

str_count_setdiff *Count the words in a vector that don't intersect with another vector (only counts unique words).*

Description

Count the words in a vector that don't intersect with another vector (only counts unique words).

Usage

```
str_count_setdiff(x, y)
```

Arguments

x	A vector of words.
y	A vector of words to test against.

Value

A number, the count of unique words in x not also in y

Examples

```
str_count_setdiff(
x = c("a", "dog", "dog", "went", "to", "the", "store"),
y = c("dog", "to", "store")
)
```

str_dt_col_combine *Combine columns of a data.table into a list in a new column, wraps list(unlist(c(...)))*

Description

Combine columns of a data.table into a list in a new column, wraps list(unlist(c(...)))

Usage

```
str_dt_col_combine(...)
```

Arguments

...	Unquoted column names of a data.table.

Value

A list, with columns combined into a vector if grouped properly

Examples

```
as.data.table(
list(
  col1 = c(
    "a",
    "b"
  ),
  col2 = c(
    tolower("The dog is nice because it picked up the newspaper."),
    tolower("The dog is extremely nice because it does the dishes.")
  ),
  col3 = c(
    "test 1",
    "test 2"
  )
)
)[, col4 := .(str_dt_col_combine(col2, col3)), col1]
```

---

str_extract_match *Extract words from a vector that are found in another vector.*

Description

Extract words from a vector that are found in another vector.

Usage

```
str_extract_match(x, y)
```

Arguments

x	A vector of words.
y	A vector of words to test against.

Value

x, with the words not found in y removed.

Examples

```
str_extract_match(
x = c("a", "dog", "went", "to", "the", "store"),
y = c("dog", "to", "store")
)
```

str_extract_nomatch *Extract words from a vector that are not found in another vector.*

Description

Extract words from a vector that are not found in another vector.

Usage

```
str_extract_nomatch(x, y)
```

Arguments

x	A vector of words.
y	A vector of words to test against.

Value

x, with the words found in y removed.

Examples

```
str_extract_nomatch(
x = c("a", "dog", "went", "to", "the", "store"),
y = c("dog", "to", "store")
)
```

str_extract_positional_match *Extract words from a vector that are found in the same position in another vector.*

Description

Extract words from a vector that are found in the same position in another vector.

Usage

```
str_extract_positional_match(x, y)
```

Arguments

x	A vector of words.
y	A vector of words to test against.

Value

x, with the words without matches in the same position in y removed.

Examples

```
str_extract_positional_match(
x = c("a", "dog", "went", "to", "the", "store"),
y = c("this", "dog", "ran", "from", "the", "store")
)
```

`str_extract_positional_nomatch` *Extract words from a vector that are not found in the same position in another vector.*

Description

Extract words from a vector that are not found in the same position in another vector.

Usage

```
str_extract_positional_nomatch(x, y)
```

Arguments

x	A vector of words.
y	A vector of words to test against.

Value

x, with the words with matches found in the same position in y removed.

Examples

```
str_extract_positional_nomatch(
x = c("a", "dog", "went", "to", "the", "store"),
y = c("a", "crazy", "dog", "ran", "from", "the", "store")
)
```

str_rm_blank_space *Remove and replace excess white space from strings.*

Description

Remove and replace excess white space from strings.

Usage

```
str_rm_blank_space(x, replacement = " ")
```

Arguments

x	A vector or string.
replacement	A string to replace the blank space with, defaults to " ", which replaces excess space with a single space.

Value

x, with extra white space removed/replaced.

Examples

```
str_rm_blank_space(c("this     is   a test.  ", "    will it    work? "))
```

str_rm_long_words *Remove words from a vector that have more than a maximum number of characters.*

Description

Remove words from a vector that have more than a maximum number of characters.

Usage

```
str_rm_long_words(x, max_char_length)
```

Arguments

x	A vector of words.
max_char_length	An integer, the maximum number of characters a word can have to not be removed.

Value

x, with the words not having a character count less than or equal to the max_char_length removed.

Examples

```
str_rm_long_words(
x = c("a", "dog", "went", "to", "the", "store"),
max_char_length = 2
)
```

`str_rm_non_alphanumeric` *Remove and replace non-alphanumeric characters from strings.*

Description

Remove and replace non-alphanumeric characters from strings.

Usage

```
str_rm_non_alphanumeric(x, replacement = " ")
```

Arguments

`x`	A vector or string.
`replacement`	A string to replace the numbers with, defaults to " ".

Value

x, with non-alphanumeric (A-z, 0-9) characters removed/replaced.

Examples

```
str_rm_non_alphanumeric(c("test 67890 * % $ "))
```

`str_rm_non_printable` *Remove and replace non-printable characters from strings.*

Description

Remove and replace non-printable characters from strings.

Usage

```
str_rm_non_printable(x, replacement = " ")
```

Arguments

`x`	A vector or string.
`replacement`	A string to replace the numbers with, defaults to " ".

Value

x, with non-printable characters removed/replaced.

Examples

```
str_rm_non_printable(c("test \n\n\n67890 * % $ "))
```

---

str_rm_numbers *Remove and replace numbers from strings.*

Description

Remove and replace numbers from strings.

Usage

```
str_rm_numbers(x, replacement = "")
```

Arguments

x	A vector or string.
replacement	A string to replace the numbers with, defaults to "".

Value

x, with numbers 0-9 removed/replaced.

Examples

```
str_rm_numbers(c("test 1a234b5", "test 67890"))
```

---

str_rm_punctuation *Remove and replace punctuation from strings.*

Description

Remove and replace punctuation from strings.

Usage

```
str_rm_punctuation(x, replacement = "")
```

Arguments

x	A vector or string.
replacement	A string to replace the punctuation with, defaults to "".

Value

x, with punctuation removed/replaced.

Examples

```
str_rm_punctuation(c("wait, is this is a test?", "Tests: . ! ?"))
```

---

str_rm_regexp_match *Remove words from a vector that match a regular expression.*

Description

Remove words from a vector that match a regular expression.

Usage

```
str_rm_regexp_match(x, pattern)
```

Arguments

x	A vector of words.
pattern	A regular expression.

Value

x, with the words matching the regular expression removed.

Examples

```
str_rm_regexp_match(
x = c("a", "dog", "went", "to", "the", "store"),
pattern = "to"
)
```

---

str_rm_short_words *Remove words from a vector that don't have a minimum number of characters.*

Description

Remove words from a vector that don't have a minimum number of characters.

Usage

```
str_rm_short_words(x, min_char_length)
```

Arguments

x	A vector of words.
min_char_length	An integer, the minimum number of characters a word can have to not be removed.

Value

x, with the words not having a character count greater than or equal to the min_char_length removed.

Examples

```
str_rm_short_words(
x = c("a", "dog", "went", "to", "the", "store"),
min_char_length = 3
)
```

---

str_rm_words *Remove words from a vector of words found in another vector of words.*

Description

Remove words from a vector of words found in another vector of words.

Usage

```
str_rm_words(x, y = stopwords)
```

Arguments

x	A vector of words.
y	A vector of words to delete from x, defaults to English stop words.

Value

x, with the words found in y removed.

Examples

```
str_rm_words(
x = c("a", "dog", "went", "to", "the", "store"),
y = stopwords
)

str_rm_words(
x = c("a", "dog", "went", "to", "the", "store"),
y = c("dog", "store")
)
```

str_rm_words_by_length

Remove words from a vector based on the number of characters in each word.

Description

Remove words from a vector based on the number of characters in each word.

Usage

```
str_rm_words_by_length(x, min_char_length = 0, max_char_length = Inf)
```

Arguments

x	A vector of words.
min_char_length	An integer, the minimum number of characters a word can have to not be removed.
max_char_length	An integer, the maximum number of characters a word can have to not be removed.

Value

x, with the words not having a character count of at least the min_char_length and at most the max_char_length removed.

Examples

```
str_rm_words_by_length(
x = c("a", "dog", "went", "to", "the", "store"),
min_char_length = 3
)
```

str_stopwords_by_part_of_speech

Create a vector of English words associated with particular parts of speech.

Description

Create a vector of English words associated with particular parts of speech.

Usage

```
str_stopwords_by_part_of_speech(
 parts = c("adjective", "adverb", "conjunction", "definite article", "interjection",
  "noun", "noun phrase", "plural", "preposition", "pronoun", "verb (intransitive)",
   "verb (transitive)", "verb (usu participle)"),
 include_multi_word = FALSE
)
```

Arguments

parts	A vector, at least one of the following: 'adjective', 'adverb', 'conjunction', 'definite article', 'interjection', 'noun', 'noun phrase', 'plural', 'preposition', 'pronoun', 'verb (intransitive)', 'verb (transitive)', 'verb (usu participle)'
include_multi_word	TRUE/FALSE, if TRUE, includes records from the pos data.table where multi_word == TRUE, otherwise excludes these records.

Value

A vector of words matching the part of speech shown in the data.table pos.

Examples

```
str_stopwords_by_part_of_speech('adjective')
```

---

str_tolower

Calls base::tolower(), which converts letters to lowercase. Only included to point out that base::tolower exists and should be used directly.

Description

Calls base::tolower(), which converts letters to lowercase. Only included to point out that base::tolower exists and should be used directly.

Usage

```
str_tolower(x)
```

Arguments

x	A vector or string.

Value

x, converted to lowercase.

Examples

```
str_tolower(c("ALLCAPS", "Some capS"))
```

`str_weighted_count_match` *Weighted count of the words in a vector that are found in another vector.*

Description

Weighted count of the words in a vector that are found in another vector.

Usage

```
str_weighted_count_match(x, y)
```

Arguments

x	A list of words and counts created by str_counts(x).
y	A list of words and counts created by str_counts(y).

Value

A number, the count of words in x also in y scaled by the number of times each word appears in x and y. If a word appears 3 times in x and 2 times in y, the result is 6, assuming no other words match.

Examples

```
str_weighted_count_match(
x = str_counts(c("a", "dog", "dog", "went", "to", "the", "store")),
y = str_counts(c("dog", "dog", "dog"))
)
```

Index