Package 'unglue'

May 8, 2026

Title Extract Matched Substrings Using a Pattern

Version 0.1.0

Description Use syntax inspired by the package 'glue' to extract matched substrings in a more intuitive and compact way than by using standard regular expressions.

Depends R (>= 3.1.0)

License GPL-3

Encoding UTF-8

LazyData true

Suggests glue, testthat (>= 2.1.0), rlang, covr, knitr, rmarkdown, magrittr

RoxygenNote 7.1.0

NeedsCompilation no

Author Antoine Fabri [aut, cre]

Maintainer Antoine Fabri <antoine.fabri@gmail.com>

Repository CRAN

Date/Publication 2020-06-11 05:50:03 UTC

Contents

unglue *unglue*

Description

The functions `unglue_data()`, `unglue()`, `unglue_vec()` and `unglue_unnest()` extract matched substrings using a syntax inspired from `glue::glue()`. Simple cases don't require regex knowledge at all.

Usage

```
unglue(x, patterns, open = "{", close = "}", convert = FALSE, multiple = NULL)

unglue_data(
  x,
  patterns,
  open = "{",
  close = "}",
  convert = FALSE,
  multiple = NULL,
  na = NA_character_
)

unglue_vec(
  x,
  patterns,
  var = 1,
  open = "{",
  close = "}",
  convert = FALSE,
  multiple = NULL,
  na = NA_character_
)

unglue_unnest(
  data,
  col,
  patterns,
  open = "{",
  close = "}",
  remove = TRUE,
  convert = FALSE,
  multiple = NULL,
  na = NA_character_
)
```

Arguments

x	a character vector to unglue.
patterns	a character vector or a list of character vectors, if a list, items will be pasted using an empty separator ("").
open	The opening delimiter.
close	The closing delimiter.
convert	If TRUE, will convert columns of output using utils::type.convert() with parameter as.is = TRUE, alternatively, can be a converting function, such as readr::type_convert. Formula notation is supported if the package rlang is installed, so things like convert = ~type_convert(., numerals = "warn.loss") are possible.
multiple	The aggregation function to use if several subpatterns are named the same, by default no function is used and subpatterns named the same will match the same value. If a function is provided it will be fed the conflicting values as separate arguments. Formula notation is supported if the package rlang is installed.
na	string to use when there is no match
var	for unglue_vec(), the numeric index or the name of the subpattern to extract from
data	a data frame.
col	column containing the character vector to extract values from.
remove	whether to remove the column col once extraction is performed

Details

Depending on the task you might want:

- unglue_data() to return a data frame from a character vector, just as glue::glue_data() does in reverse
- unglue() to return a list of data frames containing the matches
- unglue_vec() to extract one value by element of x, chosen by index or by name.
- unglue_unnest() to extract value from a column of a data frame to new columns

To build the relevant regex pattern special characters will be escaped in the input pattern and the subpatterns will be replaced with (.*?) if in standard "{foo}" form. An alternate regular expression can be provided after = so that "{foo=\\d}" will be translated into "(\\d)".

Sometimes we might want to use regex to match a part of the text that won't be extracted, in these cases we just need to omit the name as in "{=\\d}".

unglue_unnest()'s name is a tribute to tidyr::unnest() because unglue_unnest(data, col, patterns) returns a similar output as dplyr::mutate(data, unglued = unglue(col, patterns)) %>% tidyr::unnest() (without requiring any extra package). It is also very close to tidyr::extract() and efforts were made to make the syntax consistent with the latter.

Value

For `unglue()`a list of one row data frames, for `unglue_data` a data frame, for `unglue_unnest` the data frame input with additional columns built from extracted values, for `unglue_vec` an atomic vector.

Examples

```
# using an awample from ?glue::glue
if(require(magrittr) && require(glue)) {
  glued_data <- mtcars %>% glue_data("{rownames(.)} has {hp} hp")
  unglue_data(glued_data, "{rownames(.)} has {hp} hp")
}

facts <- c("Antarctica is the largest desert in the world!",
"The largest country in Europe is Russia!",
"The smallest country in Europe is Vatican!",
"Disneyland is the most visited place in Europe! Disneyland is in Paris!",
"The largest island in the world is Green Land!")
facts_df <- data.frame(id = 1:5, facts)

patterns <- c("The {adjective} {place_type} in {bigger_place} is {place}!",
            "{place} is the {adjective} {place_type=[^ ]+} in {bigger_place}!{=.*}")
unglue_data(facts, patterns)

sentences <- c("666 is [a number]", "foo is [a word]",
              "42 is [the answer]", "Area 51 is [unmatched]")
patterns <- c("{number=\\d+} is [{what}]", "{word=\\D+} is [{what}]")
unglue_data(sentences, patterns)

unglue_unnest(facts_df, facts, patterns)
unglue_unnest(facts_df, facts, patterns, remove = FALSE)
```

---

unglue_detect *Detect if strings are matched by a set of unglue patterns*

Description

Returns a logical indicating whether input strings were matched by one or more patterns

Usage

```
unglue_detect(
  x,
  patterns,
  open = "{",
  close = "}",
  convert = FALSE,
  multiple = NULL
)
```

Arguments

x	a character vector to unglue.
patterns	a character vector or a list of character vectors, if a list, items will be pasted using an empty separator ("").
open	The opening delimiter.
close	The closing delimiter.
convert	If TRUE, will convert columns of output using utils::type.convert() with parameter as.is = TRUE, alternatively, can be a converting function, such as readr::type_convert. Formula notation is supported if the package rlang is installed, so things like convert = ~type_convert(., numerals = "warn.loss") are possible.
multiple	The aggregation function to use if several subpatterns are named the same, by default no function is used and subpatterns named the same will match the same value. If a function is provided it will be fed the conflicting values as separate arguments. Formula notation is supported if the package rlang is installed.

Value

a vector of logical.

Examples

```
sentences <- c("666 is [a number]", "foo is [a word]",
               "42 is [the answer]", "Area 51 is [unmatched]")
patterns <- c("{number=\\d+} is [{what}]", "{word=\\D+} is [{what}]")
unglue_detect(sentences, patterns)
```

---

unglue_regex *Converts unglue pattern to regular regex pattern*

Description

Transforms a vector of patterns given in the unglue format to a vector of proper regex (PCRE) patterns (so they can for instance be used with functions from other packages).

Usage

```
unglue_regex(
  patterns,
  open = "{",
  close = "}",
  use_multiple = FALSE,
  named_capture = FALSE,
  attributes = FALSE
)
```

Arguments

patterns	a character vector or a list of character vectors, if a list, items will be pasted using an empty separator ("").
open	The opening delimiter.
close	The closing delimiter.
use_multiple	whether we should consider that duplicate labels can match different substrings.
named_capture	whether to incorporate the names of the groups in the output regex
attributes	whether to give group attributes to the output

Value

a character vector.

Examples

```
patterns <- c("{number=\\d+} is [{what}]", "{word=\\D+} is [{what}]")
unglue_regex(patterns)
```

---

unglue_sub *unglue_sub*

Description

substitute substrings using strings or replacement functions

Usage

```
unglue_sub(x, patterns, repl, open = "{", close = "}")
```

Arguments

x	character vector
patterns	a character vector or a list of character vectors, if a list, items will be pasted using an empty separator ("").
repl	function to apply on matched substrings, formula (if package rlang is installed), substring, or named list of such.
open	The opening delimiter.
close	The closing delimiter.

Examples

```
unglue_sub(
  c("a and b", "foo or bar"),
  c("{x} and {y}", "{x} or {z}"),
  "XXX")

unglue_sub(
  c("a and b", "foo or bar"),
  c("{x} and {y}", "{x} or {z}"),
  toupper)

unglue_sub(
  c("a and b", "foo or BAR"),
  c("{x} and {y}", "{x} or {z}"),
  list(x= "XXX", y = toupper, z = tolower))
```

Index