Package ‘rematch2’

July 23, 2025

Title Tidy Output from Regular Expression Matching

Version 2.1.2

Description Wrappers on 'regexpr' and 'gregexpr' to return the match
results in tidy data frames.

License MIT + file LICENSE

LazyData true

URL `https://github.com/r-lib/rematch2#readme`

BugReports `https://github.com/r-lib/rematch2/issues`

RoxygenNote 7.1.0

Imports tibble

Suggests covr, testthat

Encoding UTF-8

NeedsCompilation no

Author Gábor Csárdi [aut, cre],
Matthew Lincoln [ctb]

Maintainer Gábor Csárdi <`csardi.gabor@gmail.com`>

Repository CRAN

Date/Publication 2020-05-01 06:50:02 UTC

Contents

rematch2-package *Match Regular Expressions with a Nicer 'API'*

Description

A small wrapper on 'regexpr' to extract the matches and captured groups from the match of a regular expression to a character vector. See `re_match`.

Author(s)

Maintainer: Gábor Csárdi <`csardi.gabor@gmail.com`>

Other contributors:

- Matthew Lincoln <`matthew.d.lincoln@gmail.com`> [contributor]

See Also

Useful links:

- `https://github.com/r-lib/rematch2#readme`
- Report bugs at `https://github.com/r-lib/rematch2/issues`

bind_re_match *Match results from a data frame column and attach results*

Description

Taking a data frame and a column name as input, this function will run `re_match` and bind the results as new columns to the original table., returning a `tibble`. This makes it friendly for pipe-oriented programming with magrittr.

Usage

```
bind_re_match(df, from, ..., keep_match = FALSE)

bind_re_match_(df, from, ..., keep_match = FALSE)
```

Arguments

`df`	A data frame.
`from`	Name of column to use as input for `re_match`. `bind_re_match` takes unquoted names, while `bind_re_match_` takes quoted names.
`...`	Arguments (including `pattern`) to pass to `re_match`.
`keep_match`	Should the column `.match` be included in the results? Defaults to `FALSE`, to avoid column name collisions in the case that `bind_re_match` is called multiple times in succession.

Functions

- `bind_re_match_`: Standard-evaluation version that takes a quoted column name.

Note

If named capture groups will result in multiple columns with the same column name, `repair_names` will be called on the resulting table.

See Also

Standard-evaluation version `bind_re_match_` that is suitable for programming.

Examples

```
match_cars <- tibble::rownames_to_column(mtcars)
bind_re_match(match_cars, rowname, "^(?<make>\\w+) ?(?<model>.+)?$")
```

`re_exec` *Extract Data From First Regular Expression Match Into a Data Frame*

Description

Match a regular expression to a string, and return matches, match positions, and capture groups. This function is like its `match` counterpart, except it returns match/capture group start and end positions in addition to the matched values.

Usage

```
re_exec(text, pattern, perl = TRUE, ...)

## S3 method for class 'rematch_records'
x$name

## S3 method for class 'rematch_allrecords'
x$name
```

Arguments

`text`	Character vector.
`pattern`	A regular expression. See `regex` for more about regular expressions.
`perl`	logical should perl compatible regular expressions be used? Defaults to TRUE, setting to FALSE will disable capture groups.
`...`	Additional arguments to pass to `gregexpr` (or `regexpr` if `text` is of length zero).
`x`	Object returned by `re_exec` or `re_exec_all`.
`name`	`match`, `start` or `end`.

Value

A tidy data frame (see Section "Tidy Data"). Match record entries are one length vectors that are set to NA if there is no match.

Tidy Data

The return value is a tidy data frame where each row corresponds to an element of the input character vector `text`. The values from `text` appear for reference in the `.text` character column. All other columns are list columns containing the match data. The `.match` column contains the match information for full regular expression matches while other columns correspond to capture groups if there are any, and PCRE matches are enabled with `perl = TRUE` (this is on by default). If capture groups are named the corresponding columns will bear those names.

Each match data column list contains match records, one for each element in `text`. A match record is a named list, with entries `match`, `start` and `end` that are respectively the matching (sub) string, the start, and the end positions (using one based indexing).

Extracting Match Data

To make it easier to extract matching substrings or positions, a special `$` operator is defined on match columns, both for the `.match` column and the columns corresponding to the capture groups. See examples below.

See Also

`regexpr`, which this function wraps

Other tidy regular expression matching: `re_exec_all()`, `re_match_all()`, `re_match()`

Examples

```
name_rex <- paste0(
  "(?<first>[[:upper:]][[:lower:]]+) ",
  "(?<last>[[:upper:]][[:lower:]]+)"
)
notables <- c(
  "  Ben Franklin and Jefferson Davis",
  "\tMillard Fillmore"
)
# Match first occurrence
pos <- re_exec(notables, name_rex)
pos

# Custom $ to extract matches and positions
pos$first$match
pos$first$start
pos$first$end
```

re_exec_all — *Extract Data From All Regular Expression Matches Into a Data Frame*

Description

Match a regular expression to a string, and return matches, match positions, and capture groups. This function is like its `match` counterpart, except it returns match/capture group start and end positions in addition to the matched values.

Usage

```
re_exec_all(text, pattern, perl = TRUE, ...)
```

Arguments

`text`	Character vector.
`pattern`	A regular expression. See `regex` for more about regular expressions.
`perl`	logical should perl compatible regular expressions be used? Defaults to TRUE, setting to FALSE will disable capture groups.
`...`	Additional arguments to pass to `gregexpr` (or `regexpr` if `text` is of length zero).

Value

A tidy data frame (see Section “Tidy Data”). The entries within the match records within the list columns will be one vectors as long as there are matches for the corresponding text element.

Tidy Data

The return value is a tidy data frame where each row corresponds to an element of the input character vector `text`. The values from `text` appear for reference in the `.text` character column. All other columns are list columns containing the match data. The `.match` column contains the match information for full regular expression matches while other columns correspond to capture groups if there are any, and PCRE matches are enabled with `perl = TRUE` (this is on by default). If capture groups are named the corresponding columns will bear those names.

Each match data column list contains match records, one for each element in `text`. A match record is a named list, with entries `match`, `start` and `end` that are respectively the matching (sub) string, the start, and the end positions (using one based indexing).

Extracting Match Data

To make it easier to extract matching substrings or positions, a special `$` operator is defined on match columns, both for the `.match` column and the columns corresponding to the capture groups. See examples below.

See Also

gregexpr, which this function wraps

Other tidy regular expression matching: re_exec(), re_match_all(), re_match()

Examples

```
name_rex <- paste0(
  "(?<first>[[:upper:]][[:lower:]]+) ",
  "(?<last>[[:upper:]][[:lower:]]+)"
)
notables <- c(
  "  Ben Franklin and Jefferson Davis",
  "\tMillard Fillmore"
)
# All occurrences
allpos <- re_exec_all(notables, name_rex)
allpos

# Custom $ to extract matches and positions
allpos$first$match
allpos$first$start
allpos$first$end
```

---

re_match *Extract Regular Expression Matches Into a Data Frame*

Description

re_match wraps regexpr and returns the match results in a convenient data frame. The data frame has one column for each capture group if perl=TRUE, and one final columns called .match for the matching (sub)string. The columns of the capture groups are named if the groups themselves are named.

Usage

```
re_match(text, pattern, perl = TRUE, ...)
```

Arguments

text	Character vector.
pattern	A regular expression. See regex for more about regular expressions.
perl	logical should perl compatible regular expressions be used? Defaults to TRUE, setting to FALSE will disable capture groups.
...	Additional arguments to pass to regexpr.

Value

A data frame of character vectors: one column per capture group, named if the group was named, and additional columns for the input text and the first matching (sub)string. Each row corresponds to an element in the `text` vector.

Note

`re_match` uses PCRE compatible regular expressions by default (i.e. `perl = TRUE` in `regexpr`). You can switch this off but if you do so capture groups will no longer be reported as they are only supported by PCRE.

See Also

Other tidy regular expression matching: `re_exec_all()`, `re_exec()`, `re_match_all()`

Examples

```
dates <- c("2016-04-20", "1977-08-08", "not a date", "2016",
  "76-03-02", "2012-06-30", "2015-01-21 19:58")
isodate <- "([0-9]{4})-([0-1][0-9])-([0-3][0-9])"
re_match(text = dates, pattern = isodate)

# The same with named groups
isodaten <- "(?<year>[0-9]{4})-(?<month>[0-1][0-9])-(?<day>[0-3][0-9])"
re_match(text = dates, pattern = isodaten)
```

`re_match_all` *Extract All Regular Expression Matches Into a Data Frame*

Description

This function is a thin wrapper on the `gregexpr` base R function, to extract the matching (sub)strings as a data frame. It extracts all matches, and potentially their capture groups as well.

Usage

```
re_match_all(text, pattern, perl = TRUE, ...)
```

Arguments

`text`	Character vector.
`pattern`	A regular expression. See `regex` for more about regular expressions.
`perl`	logical should perl compatible regular expressions be used? Defaults to TRUE, setting to FALSE will disable capture groups.
`...`	Additional arguments to pass to `gregexpr` (or `regexpr` if `text` is of length zero).

Value

A tidy data frame (see Section "Tidy Data"). The list columns contain character vectors with as many entries as there are matches for each input element.

Tidy Data

The return value is a tidy data frame where each row corresponds to an element of the input character vector `text`. The values from `text` appear for reference in the `.text` character column. All other columns are list columns containing the match data. The `.match` column contains the match information for full regular expression matches while other columns correspond to capture groups if there are any, and PCRE matches are enabled with `perl = TRUE` (this is on by default). If capture groups are named the corresponding columns will bear those names.

Each match data column list contains match records, one for each element in `text`. A match record is a named list, with entries `match`, `start` and `end` that are respectively the matching (sub) string, the start, and the end positions (using one based indexing).

Note

If the input text character vector has length zero, `regexpr` is called instead of `gregexpr`, because the latter cannot extract the number and names of the capture groups in this case.

See Also

Other tidy regular expression matching: `re_exec_all()`, `re_exec()`, `re_match()`

Examples

```
name_rex <- paste0(
  "(?<first>[[:upper:]][[:lower:]]+) ",
  "(?<last>[[:upper:]][[:lower:]]+)"
)
notables <- c(
  "  Ben Franklin and Jefferson Davis",
  "\tMillard Fillmore"
)
re_match_all(notables, name_rex)
```

Index