Package ‘namedCapture’

October 13, 2022

Maintainer Toby Dylan Hocking <toby.hocking@r-project.org>

Author Toby Dylan Hocking

Version 2020.4.1

License GPL-3

Title Named Capture Regular Expressions

Description User-friendly wrappers for
named capture regular expressions.
Introduction and comparison in research paper
by Hocking (2019), R Journal.
<doi:10.32614/RJ-2019-050>
RE2 engine ('re2r' package)
<https://github.com/qinwf/re2r>
was removed from CRAN in Mar 2020
so must be installed from github.

Depends R (>= 2.14)

Suggests testthat, data.table, re2r, knitr, rmarkdown, rex, dplyr,
tidyr, rematch2

VignetteBuilder knitr

URL https://github.com/tdhock/namedCapture

BugReports https://github.com/tdhock/namedCapture/issues

Additional_repositories https://tdhock.github.io/drat

NeedsCompilation no

Repository CRAN

Date/Publication 2020-04-01 17:30:06 UTC

R topics documented:

`apply_type_funs` *apply type funs*

Description

Convert columns of `match.mat` using corresponding functions from `type.list`.

Usage

```
apply_type_funs(match.mat,
    type.list)
```

Arguments

`match.mat`	character matrix (matches X groups).
`type.list`	named list of functions to apply to captured groups.

Value

If `type.list` is a list of functions, then return a data.frame whose columns are defined by calling the functions in `type.list` on the corresponding column of `match.mat`. Otherwise just return a character matrix. If `match.mat` does not already have rownames, and it has a column named "name", then that column will be used for the rownames, and that column will not be returned.

Author(s)

Toby Dylan Hocking

`check_subject_pattern` *check subject pattern*

Description

Error if `subject.vec` or `pattern` incorrect type.

Usage

```
check_subject_pattern(subject.vec,
    pattern)
```

Arguments

`subject.vec`

`pattern`

Author(s)

Toby Dylan Hocking

`df_match_variable` *First match from every row, variable argument syntax*

Description

Extract text from several columns of a data.frame, using a different named capture regular expression for each column. Uses `str_match_variable` on each column/pattern indicated in `...` – argument names are interpreted as column names of subject; argument values are passed as the pattern to `str_match_variable`.

Usage

```
df_match_variable(...)
```

Arguments

`...`	subject.df, colName1=list(groupName1=pattern1, fun1, etc), colName2=list(etc), etc. First (un-named) argument should be a data.frame with character columns of subjects for matching. The other arguments need to be named (and the names e.g. colName1 and colName2 need to be column names of the subject data.frame). The other argument values specify the regular expression, and must be character/function/list. All patterns must be character vectors of length 1. If the pattern is a named argument in R, we will add a named capture group (?<groupName1>pattern1) in the regex. All patterns are pasted together to obtain the final pattern used for matching. Each named pattern may be followed by at most one function (e.g. fun1) which is used to convert the previous named pattern. Lists are parsed recursively for convenience.

Value

data.frame with same number of rows as subject, with an additional column for each named capture group specified in `...` (actually the value is created via `cbind` so if subject is something else like a `data.table` then the value is too).

Author(s)

Toby Dylan Hocking

Examples

```
## The JobID column can be match with a complicated regular
## expression, that we will build up from small sub-pattern list
## variables that are easy to understand independently.
(sacct.df <- data.frame(
  JobID = c(
    "13937810_25", "13937810_25.batch",
    "13937810_25.extern", "14022192_[1-3]", "14022204_[4]"),
  Elapsed = c(
    "07:04:42", "07:04:42", "07:04:49",
    "00:00:00", "00:00:00"),
  stringsAsFactors=FALSE))

## Just match the end of the range.
int.pattern <- list("[0-9]+", as.integer)
end.pattern <- list(
  "-",
  task_end=int.pattern)
namedCapture::df_match_variable(sacct.df, JobID=end.pattern)

## Match the whole range inside square brackets.
range.pattern <- list(
  "[[]",
  task_start=int.pattern,
  end.pattern, "?", #end is optional.
  "[]]")
namedCapture::df_match_variable(sacct.df, JobID=range.pattern)

## Match either a single task ID or a range, after an underscore.
task.pattern <- list(
  "_",
  list(
    task_id=int.pattern,
    "|",#either one task(above) or range(below)
    range.pattern))
namedCapture::df_match_variable(sacct.df, JobID=task.pattern)

## Match type suffix alone.
type.pattern <- list(
  "[.]",
  type=".*")
```

```
namedCapture::df_match_variable(sacct.df, JobID=type.pattern)

## Match task and optional type suffix.
task.type.pattern <- list(
  task.pattern,
  type.pattern, "?")
namedCapture::df_match_variable(sacct.df, JobID=task.type.pattern)

## Match full JobID and Elapsed columns.
(task.df <- namedCapture::df_match_variable(
  sacct.df,
  JobID=list(
    job=int.pattern,
    task.type.pattern),
  Elapsed=list(
    hours=int.pattern,
    ":",
    minutes=int.pattern,
    ":",
    seconds=int.pattern)))
str(task.df)
```

`engine` *engine*

Description

Get current regex engine used by `str_match_named` and `str_match_all_named`. RE2 is used by default if the re2r package is available; otherwise, PCRE is used by default. The user can set `options(namedCapture.engine="PCRE")` to use PCRE even when RE2 is available.

Usage

```
engine()
```

Author(s)

Toby Dylan Hocking

Examples

```
namedCapture::engine()
old.opt <- options(namedCapture.engine="PCRE")
namedCapture::engine()
options(old.opt)
```

`names_or_error` *names or error*

Description

Extract capture group names. Stop with an error if there are no capture groups, or if there are any capture groups without names.

Usage

```
names_or_error(vec.with.attrs,
    pattern)
```

Arguments

`vec.with.attrs`	Output from g?regexpr.
`pattern`	

Value

Character vector.

Author(s)

Toby Dylan Hocking

`only_captures` *only captures*

Description

Extract capture group columns from `match.mat`, stop if any are un-named, and assign optional groups to "".

Usage

```
only_captures(match.mat,
    pattern)
```

Arguments

`match.mat`	
`pattern`	

Author(s)

Toby Dylan Hocking

`stop_for_names` *stop for names*

Description

Informative error message when named group(s) missing.

Usage

```
stop_for_names(pattern)
```

Arguments

`pattern`

Author(s)

Toby Dylan Hocking

`str_match_all_named` *All matches from multiple subjects, three argument syntax*

Description

Extract all matches of `pattern` from each element of `subject.vec` using named capturing regular expressions. For the common case of extracting all matches of a regex from a multi-line text file, please use `str_match_all_variable` instead. Result depends on `engine` (either PCRE or RE2) which can be specified via the namedCapture.engine option.

Usage

```
str_match_all_named(subject.vec,
    pattern, type.list = NULL)
```

Arguments

`subject.vec`	character vector of subjects.
`pattern`	named capture regular expression (character vector of length 1).
`type.list`	named list of functions to apply to captured groups, in order to create non-character (typically numeric) columns in the result.

Value

A list of data.frames with one row for each subject and one column for each capture group if `type.list` is a list of functions. Otherwise a list of character matrices. If `pattern` contains a group named "name" then it will not be returned as a column, and will instead be used for the rownames of the data.frames or matrices. If `subject.vec` has names, they will be used as the names of the returned list.

Author(s)

Toby Dylan Hocking

Examples

```
chr.pos.vec <- c(
  "chr10:213,054,000-213,055,000",
  "chrM:111,000-222,000",
  "this will not match",
  NA, # neither will this.
  "chr1:110-111 chr2:220-222") # two possible matches.
chr.pos.pattern <- paste0(
  "(?P<chrom>chr.*?)",
  ":",
  "(?P<chromStart>.*?)",
  "-",
  "(?P<chromEnd>[0-9,]*)")
## Specifying a list of conversion functions means that str_match_*
## should convert the matched groups from character to whatever is
## returned by those functions.
keep.digits <- function(x)as.integer(gsub("[^0-9]", "", x))
conversion.list <- list(chromStart=keep.digits, chromEnd=keep.digits)
## Use str_match_all_named to get ALL matches in each subject (not
## just the first match).
(match.df.list <- namedCapture::str_match_all_named(
  chr.pos.vec, chr.pos.pattern, conversion.list))
str(match.df.list)
## If there is a capture group named "name" then it will be used for
## the rownames of the result.
name.value.vec <- c(
  H3K27me3="  sampleType=monocyte   assayType=H3K27me3    cost=5",
  H3K27ac="sampleType=monocyte assayType=H3K27ac",
  H3K4me3=" sampleType=Myeloidcell cost=30.5  assayType=H3K4me3")
name.value.pattern <- paste0(
  "(?P<name>[^ ]+?)",
  "=",
  "(?P<value>[^ ]+)")
namedCapture::str_match_all_named(name.value.vec, name.value.pattern)
```

str_match_all_variable

All matches from one subject, variable argument syntax

Description

Extract all matches of a named capture regex pattern from one subject string. It is for the common case of extracting all matches of a regex from a single multi-line text file subject; for other subjects, `str_match_all_named` can be used to find all matches. This function uses `variable_args_list` to analyze the arguments and `str_match_all_named` to perform the matching.

Usage

```
str_match_all_variable(subject.vec,
    ...)
```

Arguments

`subject.vec`	The subject character vector. We treat elements of subject as separate lines; i.e. we do the regex matching on the single subject string formed by pasting together the subject character vector using newlines as the separator.
`...`	name1=pattern1, fun1, etc, which creates the regex (?<name1>pattern1) and uses fun1 for conversion. These other arguments specify the regular expression pattern and must be character/function/list. All patterns must be character vectors of length 1. If the pattern is a named argument in R, we will add a named capture group (?P<name>pattern) in the regex. All patterns are pasted together to obtain the final pattern used for matching. Each named pattern may be followed by at most one function which is used to convert the previous named pattern. Lists are parsed recursively for convenience.

Value

matrix or data.frame with one row for each match, and one column for each named group, see `str_match_all_named` for details.

Author(s)

Toby Dylan Hocking

Examples

```
chr.pos.vec <- c(
  "chr10:213,054,000-213,055,000",
  "chrM:111,000-222,000",
  "this will not match",
  NA, # neither will this.
  "chr1:110-111 chr2:220-222") # two possible matches.
```

```
keep.digits <- function(x)as.integer(gsub("[^0-9]", "", x))
## str_match_all_variable treats elements of subject as separate
## lines (and ignores NA elements). Named arguments are used to
## create named capture groups, and conversion functions such as
## keep.digits are used to convert the previously named group.
int.pattern <- list("[0-9,]+", keep.digits)
(match.df <- namedCapture::str_match_all_variable(
  chr.pos.vec,
  name="chr.*?",
  ":",
  chromStart=int.pattern,
  "-",
  chromEnd=int.pattern))
str(match.df)
match.df["chr1", "chromEnd"]
```

`str_match_named` *First match from multiple subjects, three argument syntax*

Description

Extract the first match of `pattern` from each element of `subject.vec` using a named capture regular expression. This function is mostly for internal use; most users should use `str_match_variable` instead. Result depends on `engine` (either PCRE or RE2) which can be specified via the namedCapture.engine option.

Usage

```
str_match_named(subject.vec,
    pattern, type.list = NULL)
```

Arguments

`subject.vec`	character vector of subjects.
`pattern`	named capture regular expression (character vector of length 1).
`type.list`	named list of functions to apply to captured groups, in order to create non-character (typically numeric) columns in the result.

Value

A data.frame with one row for each subject and one column for each capture group if `type.list` is a list of functions. Otherwise a character matrix. If `subject.vec` has names then they will be used for the rownames of the returned data.frame or character matrix. Otherwise if `pattern` has a group named "name" then it will not be returned as a column, and will instead be used for the rownames.

Author(s)

Toby Dylan Hocking

Examples

```
chr.pos.vec <- c(
  "chr10:213,054,000-213,055,000",
  "chrM:111,000-222,000",
  "this will not match",
  NA, # neither will this.
  "chr1:110-111 chr2:220-222") # two possible matches.
chr.pos.pattern <- paste0(
  "(?P<chrom>chr.*?)",
  ":",
  "(?P<chromStart>.*?)",
  "-",
  "(?P<chromEnd>[0-9,]*)")
## Specifying a list of conversion functions means that str_match_*
## should convert the matched groups from character to whatever is
## returned by those functions.
keep.digits <- function(x)as.integer(gsub("[^0-9]", "", x))
conversion.list <- list(chromStart=keep.digits, chromEnd=keep.digits)
(match.df <- namedCapture::str_match_named(chr.pos.vec, chr.pos.pattern, conversion.list))
str(match.df)
```

`str_match_variable` *First match from multiple subjects, variable argument syntax*

Description

Extract the first match of a named capture regex pattern from each of several subject strings. This function uses `variable_args_list` to analyze the arguments and `str_match_named` to perform the matching. For the first match in every row of a data.frame, using a different regex for each column, use `df_match_variable`. For all matches in one character subject use `str_match_all_variable`; for all matches in several character subjects use `str_match_all_named`.

Usage

```
str_match_variable(subject.vec,
    ..., nomatch.error = FALSE)
```

Arguments

`subject.vec`	The subject character vector.
`...`	name1=pattern1, fun1, etc, which creates the regex (?P<name1>pattern1) and uses fun1 for conversion. These other arguments specify the regular expression pattern and must be character/function/list. All patterns must be character vectors of length 1. If the pattern is a named argument in R, we will add a named capture group (?P<name>pattern) in the regex. All patterns are pasted together

	to obtain the final pattern used for matching. Each named pattern may be followed by at most one function which is used to convert the previous named pattern. Lists are parsed recursively for convenience.
`nomatch.error`	if TRUE, stop with an error if any subject does not match; otherwise (default), subjects that do not match are reported as missing/NA rows of the result.

Value

matrix or data.frame with one row for each subject, and one column for each named group, see `str_match_named` for details.

Author(s)

Toby Dylan Hocking

Examples

```
named.subject.vec <- c(
  ten="chr10:213,054,000-213,055,000",
  M="chrM:111,000",
  one="chr1:110-111 chr2:220-222") # two possible matches.
## str_match_variable finds the first match in each element of the
## subject character vector. Named arguments are used to create
## named capture groups, which become column names in the
## result. Since the subject is named, those names are used for the
## rownames of the result.
(mat.subject.names <- namedCapture::str_match_variable(
  named.subject.vec,
  chrom="chr.*?",
  ":",
  chromStart="[0-9,]+",
  list( # un-named list becomes non-capturing group.
    "-",
    chromEnd="[0-9,]+"
  ), "?")) # chromEnd is optional.

## When no type conversion functions are specified, the result is a
## character matrix.
str(mat.subject.names)

## Conversion functions are used to convert the previously named
## group, and patterns may be saved in lists for re-use.
keep.digits <- function(x)as.integer(gsub("[^0-9]", "", x))
int.pattern <- list("[0-9,]+", keep.digits)
range.pattern <- list(
  name="chr.*?", # will be used for rownames when subject is un-named.
  ":",
  chromStart=int.pattern,
  list(
    "-",
    chromEnd=int.pattern
  ), "?")

## Rownames taken from subject if it has names.
(df.subject.names <- namedCapture::str_match_variable(
  named.subject.vec, range.pattern))

## Conversion functions used to create non-char columns.
str(df.subject.names)

## Rownames taken from name group if subject is un-named.
namedCapture::str_match_variable(
  unname(named.subject.vec), range.pattern)

## NA used to indicate no match or missing subject.
na.vec <- c(
  nomatch="this will not match",
  missing=NA, # neither will this.
  named.subject.vec)
namedCapture::str_match_variable(
  na.vec, range.pattern)
```

`variable_args_list` *variable args list*

Description

Parse the variable-length argument list used in `str_match_variable`, `str_match_all_variable`, and `df_match_variable`. This function is mostly intended for internal use, but is useful if you want to see the regex pattern generated by the variable argument syntax.

Usage

```
variable_args_list(...)
```

Arguments

`...`	character vectors (for regex patterns) or functions (which specify how to convert extracted character vectors to other types). All patterns must be character vectors of length 1. If the pattern is a named argument in R, we will add a name tag in the regex pattern. All patterns are pasted together to obtain the final pattern used for matching. Each named pattern may be followed by at most one function which is used to convert the previous named pattern. Patterns may also be lists, which are parsed recursively for convenience.

Value

a list with two named elements

`fun.list`	list of conversion functions or NULL
`pattern`	regular expression string

Author(s)

Toby Dylan Hocking

Examples

```
pos.pattern <- list("[0-9]+", as.integer)
namedCapture::variable_args_list(
  "some subject",
  chrom="chr.*?",
  ":",
  chromStart=pos.pattern,
  list(
    "-",
    chromEnd=pos.pattern
  ), "?")
```

Index