Package 'namedCapture'

February 26, 2019

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

Author Toby Dylan Hocking

Version 2019.02.25

License GPL-3

Title Named Capture Regular Expressions

Description User-friendly wrappers for
named capture regular expressions.

Depends R (>= 2.14)

Suggests testthat, data.table, re2r, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2019-02-26 20:50:11 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 or pattern incorrect type.

Usage

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

Arguments

subject.vec

pattern

Author(s)

Toby Dylan Hocking

`df_match_variable` *df match variable*

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

```
library(namedCapture)
(sacct.df <- data.frame(
  JobID = c(
    "13937810_25", "13937810_25.batch",
    "13937810_25.extern", "14022192_[1-3]", "14022204_[4]"),
  ExitCode = c("0:0", "0:0", "0:0", "0:0", "0:0"),
  State = c(
```

```
    "COMPLETED", "COMPLETED", "COMPLETED",
    "PENDING", "PENDING"),
  MaxRSS = c("", "394960K", "750K", "", ""),
  Elapsed = c(
    "07:04:42", "07:04:42", "07:04:49",
    "00:00:00", "00:00:00"),
  stringsAsFactors=FALSE))
range.pattern <- list(
  "[[]",
  task1="[0-9]+", as.integer,
  "(?:-",#begin optional end of range.
  taskN="[0-9]+", as.integer,
  ")?", #end is optional.
  "[]]")
task.pattern <- list(
  "(?:",#begin alternate
  task="[0-9]+", as.integer,
  "|",#either one task(above) or range(below)
  range.pattern,
  ")")#end alternate
(task.df <- df_match_variable(
  sacct.df,
  JobID=list(
    job="[0-9]+", as.integer,
    "_",
    task.pattern,
    "(?:[.]",
    type=".*",
    ")?"),
  Elapsed=list(
    hours="[0-9]+", as.integer,
    ":",
    minutes="[0-9]+", as.integer,
    ":",
    seconds="[0-9]+", as.integer)))
str(task.df)
```

`namedCapture.engine` *namedCapture engine*

Description

Get current regex engine used by str_match_named and str_match_all_named (if no arguments), or set the current engine (if an argument is given). The "namedCapture.engine" option is used, and can be set by the user, i.e. options(namedCapture.engine=e) is the same as namedCapture.engine(e). RE2 is used by default if the re2r package is available.

Usage

```
namedCapture.engine(e)
```

Arguments

e	regex engine, either "PCRE" or "RE2" – RE2 can only be used if the re2r package is available.

Author(s)

Toby Dylan Hocking

Examples

```
library(namedCapture)
old.engine <- namedCapture.engine()
namedCapture.engine("PCRE")
namedCapture.engine()
namedCapture.engine("RE2")
namedCapture.engine()
namedCapture.engine(old.engine)
namedCapture.engine()
```

---

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)
```

Arguments

vec.with.attrs	Output from g?regexpr.

Value

Character vector.

Author(s)

Toby Dylan Hocking

only_captures *only captures*

Description

extract capture group columns, stop if any are un-named, and assign optional groups to "".

Usage

```
only_captures(match.mat)
```

Arguments

```
match.mat
```

Author(s)

Toby Dylan Hocking

stop_for_names *stop for names*

Description

informative error message when named group(s) missing.

Usage

```
stop_for_names()
```

Author(s)

Toby Dylan Hocking

`str_match_all_named` *str match all named*

Description

Parse several occurances of pattern from each of several subject strings using named capturing regular expressions. Uses re2r::re2_match_all if namedCapture.engine()=="RE2" otherwise uses gregexpr(perl=TRUE).

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.

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

```
library(namedCapture)
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 <- 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>[^ ]+)")
str_match_all_named(name.value.vec, name.value.pattern)
```

`str_match_all_variable` *str match all variable*

Description

Extract each occurance of a named capture regex pattern from one subject string.

Usage

```
str_match_all_variable(...)
```

Arguments

`...`	subject, name1=pattern1, fun1, etc, which creates the regex (?<name1>pattern1) and uses fun1 for conversion. The first argument must be 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. The 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

```
library(namedCapture)
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.
(match.df <- str_match_all_variable(
  chr.pos.vec,
  chrom="chr.*?",
  ":",
  chromStart=".*?", keep.digits,
  "-",
  chromEnd="[0-9,]*", keep.digits))
str(match.df)
```

`str_match_named` *str match named*

Description

Parse the first occurance of pattern from each of several subject strings using a named capture regular expression. Uses re2r::re2_match if namedCapture.engine()=="RE2" otherwise uses regexpr(perl=TRUE).

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.

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 there is 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

```
library(namedCapture)
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 <- str_match_named(chr.pos.vec, chr.pos.pattern, conversion.list))
str(match.df)
```

---

str_match_variable *str match variable*

Description

Extract the first occurance of a named capture regex pattern from each of several subject strings.

Usage

```
str_match_variable(...)
```

Arguments

...	subject, name1=pattern1, fun1, etc, which creates the regex (?P<name1>pattern1) and uses fun1 for conversion. The first argument must be the subject character vector. The 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 subject, and one column for each named group, see ?str_match_named for details.

Author(s)

Toby Dylan Hocking

Examples

```
library(namedCapture)
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_variable finds the first match in each element of
## the subject character vector. Named arguments are used to create
## named capture groups, and conversion functions such as
## keep.digits are used to convert the previously named group.
(match.df <- str_match_variable(
  chr.pos.vec,
  chrom="chr.*?",
  ":",
  chromStart=".*?", keep.digits,
  "-",
  chromEnd="[0-9,]*", keep.digits))
str(match.df)
```

`variable_args_list` *variable args list*

Description

Parse the variable-length argument list.

Usage

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

Arguments

`...`	character vectors or functions (for converting extracted character vectors to other types). The first element must be the subject character vector, and the second element must be a pattern. 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

List with three named elements: subject.vec is the subject character vector, pattern is the regular expression string, and fun.list is a list of conversion functions.

Author(s)

Toby Dylan Hocking

Index