Package 'RCLabels'

March 5, 2022

Title Manipulate Matrix Row and Column Labels with Ease

Version 0.1.1

Date 2022-03-05

Description Functions to assist manipulation of matrix row and column labels for all types of matrix mathematics where row and column labels are to be respected.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.2

Imports assertthat, Hmisc, magrittr, purrr

Suggests dplyr, knitr, rmarkdown, spelling, testthat (>= 3.0.0), tibble

Config/testthat/edition 3

Depends R (>= 2.10)

LazyData true

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Matthew Heun [aut, cre] (<https://orcid.org/0000-0002-7438-214X>)

Maintainer Matthew Heun <matthew.heun@me.com>

Repository CRAN

Date/Publication 2022-03-05 16:00:02 UTC

R topics documented:

`arrow_notation` *Arrow notation*

Description

A description of arrow notation.

Usage

```
arrow_notation
```

Format

A vector of notational symbols that provides an arrow separator ("a -> b") between prefix and suffix.

Examples

```
arrow_notation
```

bracket_arrow_notation *Bracket arrow notation*

Description

A description of bracket arrow notation.

Usage

```
bracket_arrow_notation
```

Format

A vector of notational symbols that provides bracket arrow ("a [-> b]") notation.

Examples

```
bracket_arrow_notation
```

bracket_notation *Bracket notation*

Description

A description of bracket notation.

Usage

```
bracket_notation
```

Format

A vector of notational symbols that provides bracket ("a [b]") notation.

Examples

```
bracket_notation
```

first_dot_notation *First dot notation*

Description

A description of first dot notation. Note that "a.b.c" splits into prefix ("a") and suffix ("b.c").

Usage

```
first_dot_notation
```

Format

A vector of notational symbols that provides first dot ("a.b") notation.

Examples

```
first_dot_notation
```

from_notation *From notation*

Description

A description of from notation.

Usage

```
from_notation
```

Format

A vector of notational symbols that provides from ("a [from b]") notation.

Examples

```
from_notation
```

get_nouns *Extract nouns from labels*

Description

Nouns are the first part of a row-column label, "a" in "a [b]". Internally, this function calls `get_pref_suff()` and asks for the prefix.

Usage

```
get_nouns(labels, notation = RCLabels::bracket_notation)
```

Arguments

`labels`	A list or vector of labels from which nouns are to be extracted.
`notation`	The notation type to be used when extracting nouns. Default is `RCLabels::bracket_notation`.

Value

A list of nouns from row and column labels.

Examples

```
get_nouns("a [b]", bracket_notation)
# Also works with vectors and lists.
get_nouns(c("a [b]", "c [d]"))
get_nouns(list("a [b]", "c [d]"))
```

get_objects *Extract objects of prepositional phrases in row and column labels*

Description

This function extracts the objects of prepositional phrases from row and column labels. The format of the output is a list of named items, one name for each preposition encountered in labels. Objects are `NA` if there is no prepositional phrase starting with that preposition.

Usage

```
get_objects(
  labels,
  notation = RCLabels::bracket_notation,
  prepositions = RCLabels::prepositions
)
```

Arguments

labels	The row and column labels from which prepositional phrases are to be extracted.
notation	The notation object that describes the labels. Default is RCLabels::bracket_notation.
prepositions	A vector of strings to be treated as prepositions. Note that a space is appended to each word internally, so, e.g., "to" becomes "to ". Default is RCLabels::prepositions.

Value

A list of objects of prepositional phrases, with names being prepositions, and values being objects.

Examples

```
get_objects(c("a [of b into c]", "d [of Coal from e -> f]"))
```

---

get_piece *Get a piece of a label*

Description

This is a wrapper function for get_pref_suff(), get_nouns(), and get_objects(). It returns a piece of a row or column label.

Usage

```
get_piece(
  labels,
  piece = "all",
  notation = RCLabels::bracket_notation,
  prepositions = RCLabels::prepositions
)
```

Arguments

labels	The row and column labels from which prepositional phrases are to be extracted.
piece	The name of the item to return.
notation	The notation object that describes the labels. Default is RCLabels::bracket_notation.
prepositions	A vector of strings to be treated as prepositions. Note that a space is appended to each word internally, so, e.g., "to" becomes "to ". Default is RCLabels::prepositions.

Details

piece is typically one of

- "all" (which returns labels directly),
- "pref" (for the prefixes),
- "suff" (for the suffixes),
- "noun" (returns the noun),
- "pps" (prepositional phrases, returns prepositional phrases in full),
- "prepositions" (returns a list of prepositions),
- "objects" (returns a list of objects with prepositions as names), or
- a preposition in prepositions (as a string), which will return the object of that preposition named by the preposition itself.

piece must be a character vector of length 1.

If a piece is missing in a label, "" (empty string) is returned.

Value

A piece of labels.

Examples

```
labs <- c("a [from b in c]", "d [of e in f]", "Export [of Coal from USA to MEX]")
get_piece(labs, "pref")
get_piece(labs, "suff")
get_piece(labs, piece = "noun")
get_piece(labs, piece = "pps")
get_piece(labs, piece = "prepositions")
get_piece(labs, piece = "objects")
get_piece(labs, piece = "from")
get_piece(labs, piece = "in")
get_piece(labs, piece = "of")
get_piece(labs, piece = "to")
```

---

get_pps *Extract prepositional phrases of row and column labels*

Description

This function extracts the suffix of a row or column label as a single string.

Usage

```
get_pps(
  labels,
  notation = RCLabels::bracket_notation,
  prepositions = RCLabels::prepositions
)
```

Arguments

labels	A list or vector of labels from which nouns are to be extracted.
notation	The notation type to be used when extracting nouns. Default is RCLabels::bracket_notation.
prepositions	A list of prepositions, used to detect prepositional phrases. Default is RCLabels::prepositions.

Value

All prepositional phrases in a suffix.

Examples

```
get_pps(c("a [in b]", "c [of d]"))
get_pps(c("a [of b in c]", "d [-> e of f]"))
```

---

get_prepositions *Extract prepositions from row and column labels*

Description

This function extracts prepositions from a list of row and column labels. The list has outer structure of the number of labels and an inner structure of each prepositional phrase in the specific label.

Usage

```
get_prepositions(
  labels,
  notation = RCLabels::bracket_notation,
  prepositions = RCLabels::prepositions
)
```

Arguments

labels	The row and column labels from which prepositional phrases are to be extracted.
notation	The notation object that describes the labels. Default is RCLabels::bracket_notation.
prepositions	A vector of strings to be treated as prepositions. Note that a space is appended to each word internally, so, e.g., "to" becomes "to ". Default is RCLabels::prepositions.

Value

A list of prepositions.

Examples

```
get_prepositions(c("a [of b into c]", "d [-> e of f]"))
```

make_or_pattern *Create "or" regex patterns*

Description

This function makes "or" regex patterns from vectors or lists of strings. This function can be used with the `matsbyname::select_rows_byname()` and `matsbyname::select_cols_byname` functions. `make_or_pattern()` correctly escapes special characters in `strings`, such as `(` and `)`, as needed. Thus, it is highly recommended that `make_or_pattern` be used when constructing patterns for row and column selections with `matsbyname::select_rows_byname()` and `matsbyname::select_cols_byname()`.

Usage

```
make_or_pattern(
  strings,
  pattern_type = c("exact", "leading", "trailing", "anywhere", "literal")
)
```

Arguments

`strings`	A vector of row and column names.
`pattern_type`	One of "exact", "leading", "trailing", "anywhere", or "literal". Default is "exact".

Details

`pattern_type` controls the type of pattern created:

- `exact` produces a regex pattern that selects row or column names by exact match.
- `leading` produces a regex pattern that selects row or column names if the item in `strings` matches the beginnings of row or column names.
- `trailing` produces a regex pattern that selects row or column names if the item in `strings` matches the ends of row or column names.
- `anywhere` produces a regex pattern that selects row or column names if the item in `strings` matches any substring of row or column names.
- `literal` returns `strings` unmodified, and it is up to the caller to formulate a correct regex.

Value

An "or" regex pattern suitable for selecting row and column names. Amenable for use with `matsbyname::select_rows_byna` or `matsbyname::select_cols_byname`.

Examples

```
make_or_pattern(strings = c("a", "b"), pattern_type = "exact")
```

`modify_label_pieces` *Modify pieces of row and column labels*

Description

This function modifies pieces of row and column labels according to `label_map` that defines "one or many to one" relationships. This function is useful for aggregations. For example, replacing nouns can be done by modify_label_pieces(<<labels>>, piece = "noun", label_map = list(new_noun = c("a", "b", "c")). The string "new_noun" will replace any of "a", "b", or "c" when they appear as nouns in a row or column label. See examples for details.

Usage

```
modify_label_pieces(
  labels,
  piece,
  mod_map,
  prepositions = RCLabels::prepositions,
  notation = RCLabels::bracket_notation
)
```

Arguments

`labels`	The row and column labels in which pieces will be modified.
`piece`	The piece (or pieces) of the row or column label that will be modified.
`mod_map`	A modification map. See details.
`prepositions`	A list of prepositions, used to detect prepositional phrases. Default is `RCLabels::prepositions`.
`notation`	The notation used in `labels`. Default is `RCLabels::bracket_notation`.

Details

Typical `pieces` include "noun" or a preposition, such as "in" or "from". See `RCLabels::prepositions` for additional examples. This argument may be a single string or a character vector.

The `mod_map` argument should consist of a named list of character vectors in which names indicate strings to be inserted and values indicate values that should be replaced. The sense is `new = old` or `new = olds`, where "new" is the new name (the replacement) and "old" and "olds" is/are a string/vector of strings, any one of which will be replaced by "new".

Value

`labels` with replacements according to `piece` and `mod_map`.

Examples

```
# Simple case
modify_label_pieces("a [of b in c]",
                    piece = "noun",
                    mod_map = list(new_noun = c("a", "b")))
# Works with a vector or list of labels
modify_label_pieces(c("a [of b in c]", "d [-> e in f]"),
                    piece = "noun",
                    mod_map = list(new_noun = c("d", "e")))
# Works with multiple items in the mod_map
modify_label_pieces(c("a [of b in c]", "d [-> e in f]"),
                    piece = "noun",
                    mod_map = list(new_noun1 = c("a", "b", "c"),
                                   new_noun2 = c("d", "e", "f")))
# Works with multiple pieces to be modified
modify_label_pieces(c("a [of b in c]", "d [-> e in f]"),
                    piece = c("noun", "in"),
                    mod_map = list(new_noun = c("a", "b", "c"),
                                   new_in   = c("c", "f")))
```

`modify_nouns` *Modify nouns in labels*

Description

This function modifies the nouns of row and column labels. The length of `new_nouns` must be the same as the length of `labels`.

Usage

```
modify_nouns(labels, new_nouns, notation = RCLabels::bracket_notation)
```

Arguments

`labels`	The row and column labels in which the nouns will be modified.
`new_nouns`	The new nouns to be set in `labels`. Must be same length as `labels`.
`notation`	The notation used in `labels`. Default is `RCLabels::bracket_notation`.

Value

A character vector of same length as labels with nouns modified to be `new_nouns`.

Examples

```
labels <- c("a [of b in c]", "d [of e in USA]")
modify_nouns(labels, c("a_plus", "g"))
```

of_notation *Of notation*

Description

A description of of notation.

Usage

```
of_notation
```

Format

A vector of notational symbols that provides of ("a [of b]") notation.

Examples

```
of_notation
```

paren_notation *Parenthetical notation*

Description

A description of parenthetical notation.

Usage

```
paren_notation
```

Format

A vector of notational symbols that provides a parenthetical ("a (b)") notation.

Examples

```
paren_notation
```

paste_pieces *Recombine row and column labels*

Description

This function recombines (unsplits) row or column labels that have been separated by `split_labels()`.

Usage

```
paste_pieces(splt_labels, notation = RCLabels::bracket_notation)
```

Arguments

`splt_labels`	A vector of split row or column labels, probably created by `split_labels()`.
`notation`	The notation object that describes the labels. Default is `RCLabels::bracket_notation`.

Value

Recombined row and column labels.

Examples

```
labs <- c("a [of b in c]", "d [from Coal mines in USA]")
labs
split <- split_labels(labs)
split
paste_pieces(split)
# Also works in a data frame
df <- tibble::tibble(labels = c("a [in b]", "c [of d into USA]",
                                "e [of f in g]", "h [-> i in j]"))
recombined <- df %>%
  dplyr::mutate(
    splits = split_labels(labels),
    recombined = paste_pieces(splits)
  )
all(recombined$labels == recombined$recombined)
```

prepositions *Prepositions*

Description

Prepositions used in row and column labels.

Usage

```
prepositions
```

Format

A vector of prepositions used in row and column labels.

Examples

```
prepositions
```

`regex_funcs` *Find or replace row or column labels that match a regular expression*

Description

`match_by_pattern()` tells whether row or column labels match a regular expression. Internally, `grepl()` decides whether a match occurs. `replace_by_pattern()` replaces portions of row of column labels when a regular expression is matched. Internally, `gsub()` performs the replacements.

Usage

```
match_by_pattern(
  labels,
  regex_pattern,
  pieces = "all",
  prepositions = RCLabels::prepositions,
  notation = RCLabels::bracket_notation,
  ...
)

replace_by_pattern(
  labels,
  regex_pattern,
  replacement,
  pieces = "all",
  prepositions = RCLabels::prepositions,
  notation = RCLabels::bracket_notation,
  ...
)
```

Arguments

`labels`	The row and column labels to be modified.
`regex_pattern`	The regular expression pattern to determine matches and replacements. Consider using `Hmisc::escapeRegex()` to escape `regex_pattern` before calling this function.
`pieces`	The pieces of row or column labels to be checked for matches or replacements. See details.

prepositions	A vector of strings that count as prepositions. Default is RCLabels::prepositions. Used to detect prepositional phrases if pieces are to be interpreted as prepositions.
notation	The notation used in labels. Default is RCLabels::bracket_notation.
...	Other arguments passed to grepl() or gsub(), such as ignore.case, perl, fixed, or useBytes. See examples.
replacement	For replace_by_pattern(), the string that replaces all matches to regex_pattern.

Details

By default (pieces = "all"), complete labels (as strings) are checked for matches and replacements. If pieces == "pref" or pieces == "suff", only the prefix or the suffix is checked for matches and replacements. Alternatively, pieces = "noun" or pieces = <<preposition>> indicate that only specific pieces of labels are to be checked for matches and replacements. When pieces = <<preposition>>, only the object of <<preposition>> is checked for matches and replacement.

pieces can be a vector, indicating multiple pieces to be checked for matches and replacements. But if any of the pieces are "all", all pieces are checked and replaced. If pieces is "pref" or "suff", only one can be specified.

Value

A logical vector of same length as labels, where TRUE indicates a match was found and FALSE indicates otherwise.

Examples

```
labels <- c("Production [of b in c]", "d [of Coal in f]", "g [of h in USA]")
# With default `pieces` argument, matching is done for whole labels.
match_by_pattern(labels, regex_pattern = "Production")
match_by_pattern(labels, regex_pattern = "Coal")
match_by_pattern(labels, regex_pattern = "USA")
# Check beginnings of labels
match_by_pattern(labels, regex_pattern = "^Production")
# Check at ends of labels: no match.
match_by_pattern(labels, regex_pattern = "Production$")
# Can match on nouns or prepositions.
match_by_pattern(labels, regex_pattern = "Production", pieces = "noun")
# Gives FALSE, because "Production" is a noun.
match_by_pattern(labels, regex_pattern = "Production", pieces = "in")
```

---

remove_label_pieces *Remove a prepositional phrase in a row or column label*

Description

This function removes pieces from row and column labels.

Usage

```
remove_label_pieces(
  labels,
  pieces_to_remove,
  prepositions = RCLabels::prepositions,
  notation = RCLabels::bracket_notation
)
```

Arguments

labels	The row and column labels from which prepositional phrases will be removed.
pieces_to_remove	The names of pieces of the label to be removed, typically "noun" or a preposition such as "of" or "in" See RCLabels::prepositions for a list of known prepositions.
prepositions	A list of prepositions, used to detect prepositional phrases. Default is RCLabels::prepositions.
notation	The notation used in labels. Default is RCLabels::bracket_notation.

Value

labels with pieces removed.

Examples

```
labs <- c("a [of b in c]", "d [-> e in f]")
remove_label_pieces(labs, pieces_to_remove = "of")
remove_label_pieces(labs, pieces_to_remove = c("of", "->"))
remove_label_pieces(labs, pieces_to_remove = c("in", "into"))
remove_label_pieces(labs, pieces_to_remove = c("of", "in"))
```

---

row-col-notation *Row and column notation*

Description

It is often convenient to represent row and column names with notation that includes a prefix and a suffix, with corresponding separators or start-end string sequences. There are several functions that call notation_vec() to generate specialized versions or otherwise manipulate row and column names on their own or as row or column names.

- notation_vec() Builds a vector of notation symbols in a standard format that is used by matsbyname in several places. By default, it builds a list of notation symbols that provides an arrow separator (" -> ") between prefix and suffix.
- preposition_notation() Builds a list of notation symbols that provides (by default) square brackets around the suffix with a preposition ("prefix [preposition suffix]").
- paste_pref_suff() paste0's prefixes and suffixes, the inverse of split_pref_suff().

- `flip_pref_suff()` Switches the location of prefix and suffix, such that the prefix becomes the suffix, and the suffix becomes the prefix. E.g., "a -> b" becomes "b -> a" or "a [b]" becomes "b [a]".
- `get_pref_suff()` Selects only prefix or suffix, discarding notational elements and the rejected part. Internally, calls `split_pref_suff()` and selects only the `suff` portions.
- `switch_notation()` Switches from one type of notation to another based on the `from` and `to` arguments. Optionally, prefix and suffix can be `flip`ped.
- `split_pref_suff()` Splits prefixes from suffixes, returning each in a list with names `pref` and `suff`. If no prefix or suffix delimiters are found, `x` is returned in the `pref` item, unmodified, and the `suff` item is returned as `""` (an empty string). If there is no prefix, and empty string is returned for the `pref` item. If there is no suffix, and empty string is returned for the `suff` item.

If `sep` only is specified (default is " -> "), `pref_start`, `pref_end`, `suff_start`, and `suff_end` are set appropriately.

None of the strings in a notation vector are considered part of the prefix or suffix. E.g., "a -> b" in arrow notation means that "a" is the prefix and "b" is the suffix.

Usage

```
notation_vec(
  sep = " -> ",
  pref_start = "",
  pref_end = "",
  suff_start = "",
  suff_end = ""
)

preposition_notation(preposition, suff_start = " [", suff_end = "]")

split_pref_suff(x, notation = RCLabels::arrow_notation, transpose = FALSE)

paste_pref_suff(
  ps = list(pref = pref, suff = suff),
  pref = NULL,
  suff = NULL,
  notation = RCLabels::arrow_notation
)

flip_pref_suff(x, notation = RCLabels::arrow_notation)

get_pref_suff(
  x,
  which = c("pref", "suff"),
  notation = RCLabels::arrow_notation
)

switch_notation(x, from, to, flip = FALSE)
```

Arguments

sep	A string separator between prefix and suffix. Default is " -> ".
pref_start	A string indicating the start of a prefix. Default is NULL.
pref_end	A string indicating the end of a prefix. Default is the value of sep.
suff_start	A string indicating the start of a suffix. Default is the value of sep.
suff_end	A string indicating the end of a suffix. Default is NULL.
preposition	A string used to indicate position for energy flows, typically "from" or "to" in different notations.
x	A string or vector of strings to be operated upon.
notation	A notation vector generated by one of the *_notation() functions, such as notation_vec(), arrow_notation, or bracket_notation. Default is arrow_notation.
transpose	A boolean that tells whether to purr::transpose() the result. Set transpose = TRUE when using split_pref_suff() in a dplyr::mutate() call in the context of a data frame. Default is FALSE.
ps	A list of prefixes and suffixes in which each item of the list is itself a list with two items named pref and suff.
pref	A string or list of strings that are prefixes. Default is NULL.
suff	A string of list of strings that are suffixes. Default is NULL.
which	Tells which to keep, the prefix ("pref") or the suffix ("suff").
from	The notation to switch *away from*.
to	The notation to switch *to*.
flip	A boolean that tells whether to also flip the notation. Default is FALSE.

Value

For notation_vec(), arrow_notation, and bracket_notation, a string vector with named items pref_start, pref_end, suff_start, and suff_end; For split_pref_suff(), a string list with named items pref and suff. For paste_pref_suff(), split_pref_suff(), and switch_notation(), a string list in notation format specified by various notation arguments, including from, and to. For keep_pref_suff, one of the prefix or suffix or a list of prefixes or suffixes.

Examples

```
notation_vec()
arrow_notation
bracket_notation
split_pref_suff("a -> b", notation = arrow_notation)
split_pref_suff(c("a -> b", "c -> d", "e -> f"), notation = arrow_notation)
split_pref_suff(c("a -> b", "c -> d", "e -> f"), notation = arrow_notation,
                transpose = TRUE)
flip_pref_suff("a [b]", notation = bracket_notation)
get_pref_suff("a -> b", which = "suff", notation = arrow_notation)
switch_notation("a -> b", from = arrow_notation, to = bracket_notation)
switch_notation("a -> b", from = arrow_notation, to = bracket_notation,
                flip = TRUE)
# Also works for vectors
switch_notation(c("a -> b", "c -> d"), from = arrow_notation, to = bracket_notation)
```

split_labels *Split row and column labels into nouns and prepositional phrases*

Description

This function is similar to `split_pref_suff()` in that it returns a list. However, this function's list is more detailed than `split_pref_suff()`. The return value from this function is a list with the first named item being the prefix (with the name `noun`) followed by objects of prepositional phrases (with names being prepositions that precede the objects).

Usage

```
split_labels(
  labels,
  notation = RCLabels::bracket_notation,
  prepositions = RCLabels::prepositions
)
```

Arguments

`labels`	The row and column labels from which prepositional phrases are to be extracted.
`notation`	The notation object that describes the labels. Default is `RCLabels::bracket_notation`.
`prepositions`	A vector of strings to be treated as prepositions. Note that a space is appended to each word internally, so, e.g., "to" becomes "to ". Default is `RCLabels::prepositions`.

Details

Unlike `split_pref_suff()`, it does not make sense to have a `transpose` argument on `split_labels()`. Labels may not have the same structure, e.g., they may have different prepositions.

Value

A list of lists with items named `noun` and `pp`.

Examples

```
split_labels(c("a [of b in c]", "d [of e into f]"),
             notation = bracket_notation)
```

`to_notation` *To notation*

Description

A description of to notation.

Usage

```
to_notation
```

Format

A vector of notational symbols that provides to ("a [to b]") notation.

Examples

```
from_notation
```

Index