Package ‘comprehenr’

October 12, 2022

Type Package

Title List Comprehensions

Version 0.6.10

Maintainer Gregory Demin <gdemin@gmail.com>

Description Provides 'Python'-style list comprehensions.
List comprehension expressions use usual loops (for(), while() and repeat()) and usual if() as list producers. In many cases it gives more concise notation than standard ``*apply + filter'' strategy.

URL https://github.com/gdemin/comprehenr

BugReports https://github.com/gdemin/comprehenr/issues

Depends R (>= 3.3.0),

Suggests knitr, tinytest, rmarkdown

VignetteBuilder knitr

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Gregory Demin [aut, cre]

Repository CRAN

Date/Publication 2021-01-31 05:40:05 UTC

R topics documented:

`numerate`	*Auxiliary functions for working with lists*

Description

- `numerate` returns list of lists. Each list consists of two elements: sequential number of element and element. Reverse operation - `unumerate`.
- `mark` returns list of lists. Each list consists of two elements: name of element and element. Reverse operation - `unmark`.
- `zip_lists` combines lists side-by-sidy. Reverse operation - `unzip_list`.
- `unzip_list` is similair to matrix transposition but for list of lists.
- `lag_list` converts argument to list of arguments with previous values: `x -> list(x[i-1], x[i])`.

Usage

```
numerate(x)

enumerate(x)

unnumerate(x, item = 2)

mark(x)

unmark(x, item = 2)

unzip_list(x)

zip_lists(...)

lag_list(x)
```

Arguments

`x`	list, vector or list of lists
`item`	numeric number of list in which stored values
`...`	lists which will be zipped

Value

list or list of lists

Examples

```
cities = c('Chicago', 'Detroit', 'Atlanta')
airports = c('ORD', 'DTW', 'ATL')
pairs = zip_lists(cities, airports)

str(pairs)
str(unzip_list(pairs))

str(numerate(cities))

named_list = c('Chicago' = 'ORD', 'Detroit' = 'DTW', 'Atlanta' = 'ATL')
str(mark(named_list))

set.seed(123)
rand_sequence = runif(20)
# gives only locally increasing values
to_vec(for(`i, j` in lag_list(rand_sequence)) if(j>i) j)
```

`to_list` *List comprehensions for R*

Description

- `to_list` converts usual R loops expressions to list producers. Expression should be started with `for`, `while` or `repeat`. You can iterate over multiple lists if you provide several loop variables in backticks. See examples.
- `to_vec` is the same as 'to_list' but return vector. See examples.
- `to_df` is the same as 'to_list' but return data.frame. All elements of resulted list will be converted to data.frame and combined via `rbind`.
- `alter` returns the same type as its argument but with modified elements. It is useful for altering existing data.frames or lists. See examples.
- `exclude` is an auxiliary function for dropping elements in `alter`. There are no arguments for this function.

Usage

```
to_list(expr)

to_vec(expr, recursive = TRUE, use.names = FALSE)

alter(expr, data = NULL)

to_df(expr, fill = TRUE)

exclude()
```

Arguments

`expr`	expression which starts with `for`, `while` or `repeat`.
`recursive`	logical. Should unlisting be applied to list components of result? See unlist for details.
`use.names`	logical. Should names be preserved? See unlist for details.
`data`	data.frame/list/vector which we want to alter
`fill`	logical. TRUE by default. Should we combine data.frames with different names in the `to_df`?

Value

list for `to_list` and vector for `to_vec`

Examples

```
# rather useless expression - squares of even numbers
to_list(for(i in 1:10) if(i %% 2==0) i*i)

# Pythagorean triples
to_list(for (x in 1:30) for (y in x:30) for (z in y:30) if (x^2 + y^2 == z^2) c(x, y, z))

colours = c("red", "green", "yellow", "blue")
things = c("house", "car", "tree")
to_vec(for(x in colours) for(y in things) paste(x, y))

# prime numbers
noprimes = to_vec(for (i in 2:7) for (j in seq(i*2, 99, i)) j)
primes = to_vec(for (x in 2:99) if(!x %in% noprimes) x)
primes

# iteration over multiple lists
to_vec(for(`i, j` in numerate(letters)) if(i %% 2==0) paste(i, j))

set.seed(123)
rand_sequence = runif(20)
# gives only locally increasing values
to_vec(for(`i, j` in lag_list(rand_sequence)) if(j>i) j)

# to_df
to_df(for(`name, x` in mark(mtcars)) list(mean = mean(x), sd = sd(x), var = name))

# 'alter' examples
data(iris)
# scale numeric variables
res = alter(for(i in iris) if(is.numeric(i)) scale(i))
str(res)

# convert factors to characters
res = alter(for(i in iris) if(is.factor(i)) as.character(i))
str(res)
```

```
# exclude factors from data.frame
res = alter(for(i in iris) if(is.factor(i)) exclude())
str(res)

# 'data' argument example
# specify which columns to map with a numeric vector of positions:
res = alter(
    for(`i, value` in numerate(mtcars)) if(i %in% c(1, 4, 5)) as.character(value),
    data = mtcars
)
str(res)

# or with a vector of names:
res = alter(
    for(`name, value` in mark(mtcars)) if(name %in% c("cyl", "am")) as.character(value),
    data = mtcars
)
str(res)
```

Index