Package 'do'

December 3, 2020

Type Package

Title Data Operator

Version 1.6.0.0

Description Flexibly convert data between long and wide format using just two functions: reshape_toLong() and reshape_toWide().

Author Jing Zhang, Zhi Jin

Maintainer Jing Zhang<zj391120@163.com>

License GPL-3

Encoding UTF-8

LazyData true

Imports data.table, plyr, tmcn, methods, tidyselect, reshape2, tidyr

RoxygenNote 7.1.1

URL https://github.com/yikeshu0611/do

BugReports https://github.com/yikeshu0611/do/issues

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-12-03 16:30:02 UTC

R topics documented:

Apriori.Basket *Convert vector to sparse matrix*

Description

Convert vector or dataframe to sparse matrix.

Usage

```
Apriori.Basket(x, sep = ";", dup.delete = FALSE)
```

Arguments

x	a vector
sep	one separator
dup.delete	whether to delete duplicated values in the same row, default is FALSE

Value

a sparse matrix

Examples

```
# convert a vector to sparse matrix
g=c('a,b,a,,','a,b,c,d','d,c,f,g,h')
Apriori.Basket(x=g,sep = ',')

# convert a dataframe to sparse matrix
library(data.table)
df=fread(text = '
t1 t2 t3
a NA d
g a j')
Apriori.Basket(x=df,sep = ',')
```

as.data.frame.rules *Transform rules to dataframe*

Description

Transform rules to dataframe

Usage

```
## S3 method for class 'rules'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	data with rules class for package 'arules'
row.names	ignore
optional	ignore
...	ignore

Value

a dataframe

---

as.transactions *Transform to transactions*

Description

Transform to transactions

Usage

```
as.transactions(x)
```

Arguments

x	dataframe or matrix

Value

a transaction data

---

character.nms *Return character names in matrix or dataframe*

Description

Return character names in matrix or dataframe

Usage

```
character.nms(df)
```

Arguments

df	dataframe or matrix

Value

character names vectors

chinese_utf8 *UTF8 Code for Chinese*

Description

UTF8 Code for Chinese

Usage

```
chinese_utf8(x)
```

Arguments

x	chinese characters

Value

an expression with UTF8 code.

columntrans *Change data type*

Description

Change data type

Usage

```
factor.it(x) <- value

factor.it(x, value)

numeric.it(x, value)

numeric.it(x) <- value
```

Arguments

x	dataframe
value	column names

Value

factor or numeric columns in a dataframe

Examples

```
str(mtcars)
factor.it(mtcars,c("cyl", "vs", "am", "gear"))
factor.it(mtcars)=c("cyl", "vs", "am", "gear")
str(mtcars)

numeric.it(mtcars,c("cyl", "vs", "am", "gear"))
numeric.it(mtcars)=c("cyl", "vs", "am", "gear")
str(mtcars)
```

`col_split` *Split A Vector into Columns*

Description

Split A Vector into Columns

Usage

```
col_split(x, split, reg_expr, colnames)
```

Arguments

`x`	a vector
`split`	one or more characters. Split exactly
`reg_expr`	character. Split by regular expressions
`colnames`	optional. Column names for outcome

Value

A dataframe with several columns.

Examples

```
x=c('1a2','3a4','4a4')
col_split(x,split='a')
col_split(x = x,reg_expr = '[a-z]')

#two splits
df=data.frame(result=c('A, B-C',
                       'A, C-D',
                       'E, F-G'))
col_split(x = df[,1],split = c(',','-'))
```

common *Find Common Objects from Vectors*

Description

Find Common Objects from Vectors

Usage

```
common(...)
```

Arguments

...	must be several vectors

Value

common objects

Examples

```
x1=c('a','e','d')
x2=c('a','c','e')
x3=c('a','e','j','d')
common(x1,x2,x3)
```

complete.data *Complete data*

Description

Removing rows with NA in dataframe or matrix. Removing NA atomic.

Usage

```
complete.data(x)
```

Arguments

x	dataframe or matrix or atomic

Value

complete data

Examples

```
x=c(1,NA,2)
complete.data(x)

x=data.frame(a=c(1,NA))
complete.data(x)
```

`delete_left` *Delete and Move Left the rest Values*

Description

Delete and Move Left the rest Values

Usage

```
delete_left(x, delete)
```

Arguments

`x`	dataframe or matrix
`delete`	one delete object

Value

dataframe or matrix

Examples

```
a=c(1,NA,7,NA)
b=c(NA,2,2,7)
d=c(1,NA,40,7)
df=data.frame(a,b,d)
delete_left(x=df,NA)
```

`delete_up` *Delete and Move Up the Rest Values*

Description

Delete and Move Up the Rest Values

Usage

```
delete_up(x, delete)
```

Arguments

x	dataframe or matrix
delete	one delete object

Value

dataframe or matrix

Examples

```
a=c(1,NA,7,NA)
b=c(NA,2,2,7)
d=c(1,NA,40,7)
df=data.frame(a,b,d)

delete_up(x = df,delete = NA)
```

---

detach2 *Detach package*

Description

Detach package

Usage

```
detach2(x)
```

Arguments

x	one package name, if missing, detach all packages

Value

detach one package

dump.it *Create dump matrix for a vector*

Description

Create dump matrix for a vector

Usage

```
dump.it(..., include.name = TRUE)
```

Arguments

...	one vector
include.name	logical, default is TRUE, wether to include name of variable

Value

a dump matix contains 0 and 1

Examples

```
x=c('a','b','c','a','a')
dump.it(x)
dump.it(mtcars$am)
dump.it(mtcars[,'am'])
```

dup.connect *Connect Duplicated Values*

Description

Connect Duplicated Values

Usage

```
dup.connect(data, id, dup.var)
```

Arguments

data	dataframe or matrix
id	id column names or indexs
dup.var	duplicated column names or indexs

Value

dataframe contains id and duplicated values

Examples

```
dup.connect(data = mtcars,id = 'am',dup.var = 'cyl')
dup.connect(data = mtcars,
             id = c('am','gear'),
             dup.var = c('cyl','qsec'))
```

---

equal_length *Equal Length*

Description

Equal Length

Usage

```
equal_length(x, suffix = " ", nchar, colname = FALSE, rowname = FALSE)
```

Arguments

x	can be number, strings, verctors, dataframe or matrix.
suffix	suffix
nchar	maximun length
colname	a logistic value, default is FALSE
rowname	a logistic value, default is FALSE

Value

equal length results

Examples

```
a=c(123,1,24,5,1.22554)
equal_length(a,0)

df = data.frame(
    a=c(12,1,1.23),
    b=c('a','abcd','d')
)
equal_length(x = df,suffix = 'x')

equal_length(x = df,suffix = 0,nchar =5)
```

expand *Expand Data by Weight*

Description

Expand Data by Weight

Usage

```
expand(x, weight)
```

Arguments

x	dataframe or matrix
weight	weight column names or index

Value

expanded data

Examples

```
df=data.frame(v=c(1,2,3),
              x=c(7,8,9),
              n=c(2,3,4))
expand(x = df,weight = 3)
expand(x = df,weight = 'n')
```

factor.nms *Return factor names in matrix or dataframe*

Description

Return factor names in matrix or dataframe

Usage

```
factor.nms(df)
```

Arguments

df	dataframe or matrix

Value

factor names vectors

get_names *Get Names of Object*

Description

Return the names of input. For example: if you input a, you will get 'a'.

Usage

```
get_names(...)
```

Arguments

...	any type of data object

Value

names of object

Examples

```
a=c(1,2,3)
get_names(a,mtcars)
```

Grepl *Judge for Included Character*

Description

Judge for Included Character

Usage

```
Grepl(pattern, x)
```

Arguments

pattern	one or more vectors
x	one or more vectors

Details

,

Value

a matrix with logical words

Examples

```
a=c('abcd','agj','abcu')

# Grepl for one vector
pat1='b'
Grepl(pat1,a)

# Grepl for two vectors
pat2=c('c','d')
Grepl(pat2,a)

# use %or% in pattern
pat3=c('a%or%c','d')
Grepl(pat3,a)

# use %and% in pattern
pat4=c('a%and%c','d')
Grepl(pat4,a)
```

---

inner_Add_Symbol *Concatenate Strings*

Description

Concatenate vectors by adding a symbol.

Usage

```
inner_Add_Symbol(x, symbol = "+")
```

Arguments

x	vectors
symbol	defulat is '+'

Value

a concatenated string

Examples

```
inner_Add_Symbol(c('a','b'))
inner_Add_Symbol(c('a','b'),"$")
inner_Add_Symbol(c('a','b'),"")
```

join *Join two dataframes together*

Description

Join two dataframes by the same id column.

Usage

```
join_inner(x, y, by = NULL)

join_full(x, y, by = NULL)

join_left(x, y, by = NULL)

join_right(x, y, by = NULL)

join_out(x, y, by = NULL)
```

Arguments

x	one dataframe
y	the other dataframe
by	the id name in x and y dataframe

Details

join_inner(), join_full(), join_left(), join_right() and join_out() are five functons to joint two dataframes together. They are based on package 'data.table', so they are more efficient and fast.

Value

one joined dataframe.

Examples

```
df1=data.frame(x=rep(c('b','a','c'),each=3),
               y=c(1,3,6),
               v=1:9)

df2=data.frame(x=c('c','b','e'),
               v=8:6,
               foo=c(4,2,1))
join_inner(df1,df2,'x')
join_full(df1,df2,'x')
join_left(df1,df2,'x')
join_right(df1,df2,'x')
join_out(df1,df2,'x')
```

knife *Knife characters*

Description

Knife characters

Usage

```
knife_left(x, n)

knife_right(x, n)
```

Arguments

x	one character
n	number

Examples

```
knife_left(123,2)
knife_right(123,2)
```

left *Truncate Characters from the Left*

Description

Truncate Characters from the Left

Usage

```
left(x, n)
```

Arguments

x	can be number, strings, verctors, dataframe or matrix.
n	length

Value

substring

Examples

```
left("abcd",3)
left(c("abc","gjh"),2)
df = data.frame(
  a = c(123,234,456),
  b = c("abc","bcd","hjg")
)
left(df,2)
```

`mid` *Truncate Characters from the Inside*

Description

Truncate Characters from the Inside

Usage

```
mid(x, start, n)
```

Arguments

`x`	can be number, strings, verctors, dataframe or matrix.
`start`	starting position
`n`	length, n can be less than zero

Value

substring

Examples

```
mid("abcd",3,1)
mid(c("abc","gjh"),2,2)
df = data.frame(
  a = c(123,234,456),
  b = c("abc","bcd","hjg")
)
mid(df,2,1)
mid(df,2,-2)
```

model.data *Extract data of model*

Description

Extract data of model

Usage

```
model.data(fit)

model.y(fit)

model.x(fit)
```

Arguments

fit	fitted results

Value

dataframe in the model

Examples

```
fit <- lm(mpg~vs+am+poly(qsec,2),data=mtcars)
head(model.data(fit))
model.y(fit)
model.x(fit)
```

NA.col.prob *Proportion of missing value by column*

Description

NA is treated as missing value.

Usage

```
NA.col.prob(data)
```

Arguments

data	must be dataframe or matrix

Value

proportion of missing value by column

Examples

```
df = data.frame(x=rep(c(1,NA,2,NA,6,NA),10),
                y=rep(c(1,NA,2),20))
NA.col.prob(df)
```

`NA.col.sums` *Sum of missing value by column*

Description

NA is treated as missing value.

Usage

```
NA.col.sums(data)
```

Arguments

`data`	must be dataframe or matrix

Value

sum of missing value by column

Examples

```
df = data.frame(x=rep(c(1,NA,2,NA,6,NA),10),
                y=rep(c(1,NA,2),20))
NA.col.sums(df)
```

`NA.row.prob` *Proportion of missing value by row*

Description

NA is treated as missing value.

Usage

```
NA.row.prob(data)
```

Arguments

data	must be dataframe or matrix

Value

proportion of missing value by row

Examples

```
df = data.frame(x=rep(c(1,NA,2,NA,6,NA),10),
                y=rep(c(1,NA,2),20))
NA.row.prob(df)
```

NA.row.sums *Sum of missing value by row*

Description

NA is treated as missing value.

Usage

```
NA.row.sums(data)
```

Arguments

data	must be dataframe or matrix

Value

sum of missing value by row

Examples

```
df = data.frame(x=rep(c(1,NA,2,NA,6,NA),10),
                y=rep(c(1,NA,2),20))
NA.row.sums(df)
```

NA.whole.prob *Proportion of missing value in the whole dataframe*

Description

NA is treated as missing value.

Usage

```
NA.whole.prob(data)
```

Arguments

data	must be dataframe or matrix

Value

proportion of missing value in the whole dataframe

Examples

```
df = data.frame(x=rep(c(1,NA,2,NA,6,NA),10),
                y=rep(c(1,NA,2),20))
NA.whole.prob(df)
```

NA.whole.sums *Sum of missing value in the whole dataframe*

Description

NA is treated as missing value.

Usage

```
NA.whole.sums(data)
```

Arguments

data	must be dataframe or matrix

Value

sum of missing value in the whole dataframe

Examples

```
df = data.frame(x=rep(c(1,NA,2,NA,6,NA),10),
                y=rep(c(1,NA,2),20))
NA.whole.sums(df)
```

names_n *Names with different letters*

Description

Names with different letters

Usage

```
names_n(df, most = NULL, least = NULL)
```

Arguments

df	datafame or matrix
most	names with at most different letters, which means <=
least	names with at least different letters, which means >=

Value

names

Nchar *Number of Characters*

Description

Number of Characters

Usage

```
Nchar(x)
```

Arguments

x	can be number, strings, verctors, dataframe or matrix.

Value

number of characters in each location

Examples

```
Nchar("abcd")
Nchar(c("abc","gjh"))
df = data.frame(
  a = c(1,12,12.3),
  b = c("a","ab","abc")
)
Nchar(df)
```

---

numeric.nms *Return numeric names in matrix or dataframe*

Description

Return numeric names in matrix or dataframe

Usage

```
numeric.nms(df)
```

Arguments

df	dataframe or matrix

Value

numeric names vectors

---

paste0_columns *Paste Columns Together*

Description

Paste each column in a dataframe together.

Usage

```
paste0_columns(df, collapse = ",")
```

Arguments

df	a dataframe
collapse	collapse, default is comma

Value

a character

Examples

```
df=data.frame(a=c(1,2,30),
              b=c('x','y','z'))
paste0_columns(df)

df=data.frame(a=c(1,2,30),b=c('x','y','z'),c=c(1,7,8))
paste0_columns(df)
```

`Replace` *Replace*

Description

There are two methods in this function. You can use repalce many objects to one by form and to. pattern can be used to one object replaced by the other one.

Usage

```
Replace(data, from, to, pattern)
```

Arguments

`data`	can be number, strings, verctors, dataframe or matrix.
`from`	replaced stings
`to`	replacements
`pattern`	like from:to

Value

replaced data

Examples

```
Replace(data = 232,from = 2,to = 1)
Replace(data = c(232,'a4b'),
        from = c(2,'.*4'),to = 1,
        pattern = c('a:e','b:h'))
df = data.frame(
  a = c(232, 452),
  b = c("nba", "cba")
)
Replace(data = df,
        from = 2,to = 1,
        pattern = c('a:e','b:h'))
```

Replace0 *Replaced by Empty*

Description

Replaced by Empty

Usage

```
Replace0(data, from)
```

Arguments

data	can be number, strings, verctors, dataframe or matrix.
from	replaced stings

Value

replaced data

Examples

```
Replace0(data = 232,from = 2)
Replace0(data = c(232,'a4b'),from = c(2,'.*4'))

df = data.frame(
  a = c(232, 452),
  b = c("nba", "cba")
)
Replace0(data = df, from = c(2,'a'))
```

Replace_ex *Replace Exactly*

Description

Replace Exactly

Usage

```
Replace_ex(x, from, to, pattern)
```

Arguments

x	vector, dataframe or matrix
from	replaced stings
to	replacements
pattern	a special pattern, see examples for detail

Value

replaced data

Examples

```
a=c(1,2,3,1,4)
Replace_ex(x = a,from = c(1,2),to=5)
Replace_ex(x=a,pattern = c('1:5','2:5'))
Replace_ex(x=a,pattern = '[12]:5')


a=data.frame(v=c(1,2,3,2,4),
             b=c(7,8,9,4,6))
Replace_ex(x = a,from = c(1,2),to=5)
Replace_ex(x=a,pattern = c('1:5','2:5'))
```

---

replicate *Replicate Each Elements of Vectors*

Description

Replicate Each Elements of Vectors

Usage

```
rep_n(x, each)

rep_character(x, each)
```

Arguments

x	vectors
each	one or more numbers for replication

Value

replicated vectors

Examples

```
rep_n(c('ab','cde','k','op'),5)
rep_n(c('ab','cde','k','op'),c(4,6))
rep_n(c('ab','cde','k','op'),c(1,2,3,4))

rep_character(c('ab','cde','k','op'),5)
rep_character(c('ab','cde','k','op'),c(4,6))
rep_character(c('ab','cde','k','op'),c(1,2,3,4))
```

---

reshape_toLong *Convert Wide Data to Long*

Description

It is easy to convert wide data to long in this function. Be careful, id must be unique. prefix, suffix and var.names can be used together.

Usage

```
reshape_toLong(data, prefix = NULL, suffix = NULL, var.names = NULL)
```

Arguments

data	wide data
prefix	prefix of value variables
suffix	suffix of value variables
var.names	names of value variables, do.value will be created as the name of value column

Value

long data

Examples

```
df = data.frame(
  id = c(101,102,103,104),
  w1 = c(1,2,3,4),
  w2 = c(6,7,8,9),
  h1 = c(5,6,7,8),
  names = c('s1','s2','s3','s4')
)
reshape_toLong(data = df,prefix = c('w','h'))


# using suffix
df = data.frame(
  id = c(102,103,104,105),
```

```
  t1w = c(1,2,3,4),
  t2w = c(6,7,8,9),
  t1h = c(5,6,7,8),
  t2h = c(1,3,5,7),
  sex=c('female','male','male','female')
 )
 reshape_toLong(data = df,suffix = c('w','h'))
#'
#' # using prefix and suffix together
 df = data.frame(
  id = c(102,103,104,105),
  wt1 = c(1,2,3,4),
  wt2 = c(6,7,8,9),
  t1h = c(5,6,7,8),
  t2h = c(1,3,5,7),
  sex=c('female','male','male','female')
 )
 reshape_toLong(data = df,prefix = 'w',suffix = 'h')

 # using var.names
 df = data.frame(
  id = c(102,103,104,105),
  w = c(1,2,3,4),
  h = c(1,3,5,7),
  sex=c('female','male','male','female')
 )
 reshape_toLong(data = df,var.names = c('w','h'))
```

---

reshape_toWide *Reshape to Wide Format*

Description

Reshape to Wide Format

Usage

```
reshape_toWide(
  data,
  key = NULL,
  value = NULL,
  prefix = NULL,
  suffix = NULL,
  sep = "_"
)
```

Arguments

data	long data

key	column names for key, which can be one or more
value	column names for exchange, which can be one or more
prefix	column names for prefix, which can be one or more
suffix	column names for suffix, which can be one or more
sep	seperation

Value

A wide data.

Examples

```
df = data.frame(id = c(1,1,2,2,3,3,4,4),
             time = c(1,2,1,2,1,2,1,2),
             w = c(1,6,2,7,3,8,4,9))
df
reshape_toWide(data = df,
            key = 'time',
            prefix = 'w')
df = data.frame(id = c(1,1,2,2,3,3,4,4),
      time = c(1,2,1,2,1,2,1,2),
      w = c(1,6,2,7,3,8,4,9),
      h = c(5,1,6,3,7,5,8,7),
      n = c(2,2,3,3,4,4,5,5))
df
reshape_toWide(data = df,
            key = 'time',
            prefix  = c('w','h','n'))
```

---

reverse *Reverse String Order*

Description

Reverse String Order

Usage

```
reverse(x)
```

Arguments

x	can be number, strings, verctors

Value

reversed string

Examples

```
reverse(123)
reverse(c(123,'abc'))
```

`right` *Truncate Characters from the Right*

Description

Truncate Characters from the Right

Usage

```
right(x, n)
```

Arguments

x	can be number, strings, verctors, dataframe or matrix.
n	length

Value

substring

Examples

```
right("abcd",3)
right(c("abc","gjh"),2)
df = data.frame(
  a = c(123,234,456),
  b = c("abc","bcd","hjg")
)
right(df,2)
```

`rm_all` *Remove all objects*

Description

Remove all objects

Usage

```
rm_all()
```

Value

empty object

row.freq *Row Frequency*

Description

Row Frequency

Usage

```
row.freq(x)
```

Arguments

x	dataframe or matrix

Value

data with frequency column

Examples

```
row.freq(x=mtcars[,8:11])
```

split_expand *Split One Column and Expand*

Description

Split One Column and Expand

Usage

```
split_expand(data, variable, sep)
```

Arguments

data	dataframe or matrix
variable	one column name with connected values
sep	seperated symbol, which can be one or more

Value

expanded dataframe or matrix

Examples

```
df=data.frame(a=c(1,0),
              b=c('a','n'),
           cyl=c('6;6;4;4;4',
                 '6;8;'))
split_expand(data=df,variable='cyl',sep=';')
```

`take_out` *Extract Some String*

Description

Extract Some String

Usage

```
take_out(x, ..., type = "c")
```

Arguments

`x`	string
`...`	patterns of c('begin','after')
`type`	any left characters of character or list

Value

characters

Examples

```
x='abdghtyu'
take_out(x,c('a','d'),c('h','u'))
```

`Trim` *Trim*

Description

Trim

Usage

```
Trim(x, pattern = " ")

Trim_left(x, pattern = " ")

Trim_right(x, pattern = " ")
```

Arguments

x	can be vector or dataframe or matrix
pattern	one or more pattern pattern

Value

a trimed string

---

unique_no.NA *Unique Without NA*

Description

Unique Without NA

Usage

```
unique_no.NA(x)
```

Arguments

x	vector

Value

unique values with no NA

Examples

```
x=c(1,2,3,1,NA)
unique(x)
unique_no.NA(x)
```

%==% *Locate Accurately*

Description

Locate Accurately

Usage

```
a %==% b
```

Arguments

a	vector for matching
b	vector for searching

Value

If length of a is one, a vector will be return. If length of a is more than one, a list for each element will be return.

Examples

```
a=c(1,2,3,4)
b=c(1,2,3,1,4,1,5,6,1,4,1)
a %==% b
```

%s=% *Locate Similarly by grep()*

Description

Locate Similarly by grep()

Usage

```
a %s=% b
```

Arguments

a	vector for matching
b	vector for searching

Value

A list contains location information.

Examples

```
1 %s=% c(1,12,3)
c(1,2) %s=% c(1,12,3)
```

Index