

# Package ‘do’

July 27, 2020

**Type** Package

**Title** Data Operator

**Version** 1.4.0.0

**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

**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** 2020-07-27 16:10:07 UTC

## R topics documented:

<code>character.nms</code>	2
<code>chinese_utf8</code>	3
<code>col_split</code>	3
<code>common</code>	4
<code>delete_left</code>	5
<code>delete_up</code>	5
<code>dump.it</code>	6

dup.connect . . . . .	6
equal_length . . . . .	7
expand . . . . .	8
factor.nms . . . . .	9
get_names . . . . .	9
Grepl . . . . .	10
inner_Add_Symbol . . . . .	11
join . . . . .	11
left . . . . .	12
mid . . . . .	13
model.data . . . . .	14
NA.col.prob . . . . .	14
NA.col.sums . . . . .	15
NA.row.prob . . . . .	15
NA.row.sums . . . . .	16
NA.whole.prob . . . . .	16
NA.whole.sums . . . . .	17
name.n . . . . .	17
Nchar . . . . .	18
numeric.nms . . . . .	18
paste0_columns . . . . .	19
Replace . . . . .	19
Replace0 . . . . .	20
Replace_ex . . . . .	21
reshape_toLong . . . . .	22
reshape_towide . . . . .	23
reverse . . . . .	24
right . . . . .	25
row.freq . . . . .	26
split_expand . . . . .	26
take_out . . . . .	27
Trim . . . . .	27
unique_no.NA . . . . .	28
%==% . . . . .	28
%s=% . . . . .	29

## Index 30

---

character.nms	<i>Return character names in matrix or dataframe</i>
---------------	--

---

### Description

Return character names in matrix or dataframe

### Usage

character.nms(df)

**Arguments**

df                    dataframe or matrix

**Value**

character names vectors

---

chinese_utf8	<i>UTF8 Code for Chinese</i>
--------------	------------------------------

---

**Description**

UTF8 Code for Chinese

**Usage**

chinese\_utf8(x)

**Arguments**

x                    chinese characters

**Value**

an expression with UTF8 code.

---

col_split	<i>Split A Vector into Columns</i>
-----------	------------------------------------

---

**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)
```

---

delete_left	<i>Delete and Move Left the rest Values</i>
-------------	---

---

**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	<i>Delete and Move Up the Rest Values</i>
-----------	---

---

**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)

```

---

dump.it	<i>Create dump matrix for a vector</i>
---------	--

---

**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	<i>Connect Duplicated Values</i>
-------------	----------------------------------

---

**Description**

Connect Duplicated Values

**Usage**

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

**Arguments**

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

**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	<i>Equal Length</i>
--------------	---------------------

---

**Description**

Equal Length

**Usage**

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

**Arguments**

x	can be number, strings, vectors, dataframe or matrix.
suffix	suffix
nchar	maximum 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	<i>Return factor names in matrix or dataframe</i>
------------	---

---

**Description**

Return factor names in matrix or dataframe

**Usage**

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

**Arguments**

df                      dataframe or matrix

**Value**

factor names vectors

---

get_names	<i>Get Names of Object</i>
-----------	----------------------------

---

**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)
```

---

**Grep1***Judge for Included Character*

---

**Description**

Judge for Included Character

**Usage**

```
Grep1(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')  
  
# Grep1 for one vector  
pat1='b'  
Grep1(pat1, a)  
  
# Grep1 for two vectors  
pat2=c('c', 'd')  
Grep1(pat2, a)  
  
# use %or% in pattern  
pat3=c('a%or%c', 'd')  
Grep1(pat3, a)  
  
# use %and% in pattern  
pat4=c('a%and%c', 'd')  
Grep1(pat4, a)
```

---

inner_Add_Symbol	<i>Concatenate Strings</i>
------------------	----------------------------

---

**Description**

Concatenate vectors by adding a symbol.

**Usage**

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

**Arguments**

x	vectors
symbol	default 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	<i>Join two dataframes together</i>
------	-------------------------------------

---

**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 functions to join 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')
```

---

left

*Truncate Characters from the Left*


---

**Description**

Truncate Characters from the Left

**Usage**

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

**Arguments**

x	can be number, strings, vectors, 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","hjk")
)
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, vectors, 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","hjk")
)
mid(df,2,1)
mid(df,2,-2)
```

model.data                    *Extract data of model*

---

**Description**

Extract data of model

**Usage**

```
model.data(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))
```

---

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	<i>Sum of missing value by column</i>
-------------	---------------------------------------

---

**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	<i>Proportion of missing value by row</i>
-------------	---

---

**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	<i>Sum of missing value by row</i>
-------------	------------------------------------

---

**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	<i>Proportion of missing value in the whole dataframe</i>
---------------	---

---

**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	<i>Sum of missing value in the whole dataframe</i>
---------------	--

---

**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)
```

---

name.n	<i>Names with different letters</i>
--------	-------------------------------------

---

**Description**

Names with different letters

**Usage**

```
name.n(df, least = NULL, most = NULL)
```

**Arguments**

df            dataframe or matrix  
least        names with at least different letters, which means  $\geq$   
most        names with at most different letters, which means  $\leq$

**Value**

names

Nchar *Number of Characters*

---

**Description**

Number of Characters

**Usage**

Nchar(x)

**Arguments**

x can be number, strings, vectors, 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	<i>Paste Columns Together</i>
----------------	-------------------------------

---

**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	<i>Replace</i>
---------	----------------

---

**Description**

There are two methods in this function. You can use replace many objects to one by from 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, vectors, dataframe or matrix.
from	replaced strings
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, vectors, dataframe or matrix.
from	replaced strings

**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'))

```

---

reshape_toLong	<i>Convert Wide Data to Long</i>
----------------	----------------------------------

---

### 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, suffix, var.names)
```

### 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, id, col_change, prefix, suffix, sep = "_")
```

## Arguments

data	long data
id	column names for id, which can be one or more
col_change	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))

reshape_towide(data = df,
              id = 'id',
              col_change = '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))

reshape_towide(data = df,
              id = 'id',
              col_change = c('time','n'),
              prefix = 'w',sep = '_')

reshape_towide(data = df,
              id = 'id',
              col_change = 'time',
              prefix = c('w','h'))

reshape_towide(data = df,
              id = c('id','n'),
              col_change = 'time',
              prefix = c('w','h'))

df = data.frame(id = c(1,1,2,2,3,3,4,4),
               time = c('a','a','a','b','b','b','c','c'),
               n=c(5,5,6,6,7,7,8,8))
reshape_towide(data = df,id = 'id',col_change = 'time')
reshape_towide(data = df,id = c('id','time'),col_change = 'n')
reshape_towide(data = df,id = 'id',col_change = c('time','n'))

```

---

reverse

*Reverse String Order*


---

**Description**

Reverse String Order

**Usage**

reverse(x)



**Arguments**

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

**Value**

reversed string

**Examples**

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

df = data.frame(
  a = c(123, 456),
  b = c("abc", "def")
)
reverse(df)
```

---

right

*Truncate Characters from the Right*

---

**Description**

Truncate Characters from the Right

**Usage**

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

**Arguments**

x can be number, strings, vectors, 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", "hjk")
)
right(df,2)
```

---

row.freq	<i>Row Frequency</i>
----------	----------------------

---

**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	<i>Split One Column and Expand</i>
--------------	------------------------------------

---

**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	<i>Extract Some String</i>
----------	----------------------------

---

**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	<i>Trim</i>
------	-------------

---

**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 trimmed string

---

unique_no.NA	<i>Unique Without NA</i>
--------------	--------------------------

---

**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)
```

---

%==%	<i>Locate Accurately</i>
------	--------------------------

---

**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

`%==%`, 28  
`%s=%`, 29

`character.nms`, 2  
`chinese_utf8`, 3  
`col_split`, 3  
`common`, 4

`delete_left`, 5  
`delete_up`, 5  
`dump.it`, 6  
`dup.connect`, 6

`equal_length`, 7  
`expand`, 8

`factor.nms`, 9

`get_names`, 9  
`Grepl`, 10

`inner_Add_Symbol`, 11

`join`, 11  
`join_full (join)`, 11  
`join_inner (join)`, 11  
`join_left (join)`, 11  
`join_out (join)`, 11  
`join_right (join)`, 11

`left`, 12

`mid`, 13  
`model.data`, 14

`NA.col.prob`, 14  
`NA.col.sums`, 15  
`NA.row.prob`, 15  
`NA.row.sums`, 16  
`NA.whole.prob`, 16  
`NA.whole.sums`, 17

`name.n`, 17  
`Nchar`, 18  
`numeric.nms`, 18

`paste0_columns`, 19

`Replace`, 19  
`Replace0`, 20  
`Replace_ex`, 21  
`reshape_toLong`, 22  
`reshape_towide`, 23  
`reverse`, 24  
`right`, 25  
`row.freq`, 26

`split_expand`, 26

`take_out`, 27  
`Trim`, 27  
`trim-left (Trim)`, 27  
`trim-right (Trim)`, 27  
`Trim_left (Trim)`, 27  
`Trim_right (Trim)`, 27

`unique_no.NA`, 28