

# Package ‘dostats’

January 27, 2015

**Version** 1.3.1

**Date** 2014-05-28

**Title** Compute statistics helper functions

**Author** Andrew Redd <Andrew.Redd@hsc.utah.edu>

**Maintainer** Andrew Redd <Andrew.Redd@hsc.utah.edu>

**URL** <https://github.com/halpo/dostats>

**License** GPL (>= 3)

**Depends** R (>= 2.12.0)

**Imports** stats

**Suggests** plyr, testthat

**Description** A small package containing helper utilities for creating function for computing statistics.

**Collate** 'T.R' 'capply.R' 'collect.R' 'compose.R' 'consecutive.R' 'dostats.R' 'wargs.R' 'formula.R' 'hdf.R' 'onarg.R' 'pval.R' 'utils.R'

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2014-05-28 18:37:26

## R topics documented:

.T	2
capply	2
class.stats	3
collect	3
compose	4
dostats	5
fill_v	6
hdf	6
listrows	7
make_call	7

make_new_id . . . . .	8
me . . . . .	8
onarg . . . . .	8
pval . . . . .	9
redirf . . . . .	10
seq_consecutive . . . . .	10
Table1 . . . . .	11
take_names . . . . .	11
wargs . . . . .	12
%contains% . . . . .	13

## Index 14

---

`.T` *create a text vector*

---

### Description

create a text vector

### Usage

`.T(...)`

### Arguments

... names, quoted or not, no substitution made

### Examples

`.T(min, mean, 'median')`

---

`capply` *Conditional Apply*

---

### Description

A wrapper for `ifelse(test(x), fun(x, ...), x)`

### Usage

`capply(test, x, fun, ...)`

### Arguments

`test` a test that returns a logical  
`fun` to apply  
`x` data to apply fun to.  
... other arguments to fun

---

class.stats	<i>Filter by class</i>
-------------	------------------------

---

**Description**

Filter by class

**Usage**

```
class.stats(.class)
```

```
numeric.stats(x, ...)
```

```
factor.stats(x, ...)
```

```
integer.stats(x, ...)
```

**Arguments**

x	vector of any class
.class	string for class to filter by
...	passed to <a href="#">dostats</a>

**Value**

data frame of computed statistics if x is of class .class otherwise returns NULL.

**See Also**

[dostats](#)

---

collect	<i>collect results</i>
---------	------------------------

---

**Description**

collect results

**Usage**

```
collect(v, fun, ...)
```

**Arguments**

v	a vector, list, array, etc.
fun	a function to collect on
...	passed to f

**Details**

Collect results by recursively calling the elements of the vector v. The first two elements are called as fun(v[1], v[2], ...) The result is x. Then f(x, v[3]) is called and so forth, until all elements has been exhausted.

as such fun must take two arguments and return a single element, although there are no restrictions on what that single thing might be.

**Examples**

```
collect(v=letters, fun=function(x,y,...)paste(y,x, ...), sep='/')
```

---

compose

*Nest functions*

---

**Description**

Nest functions

**Usage**

```
compose(..., .list)
```

```
x %.% y
```

**Arguments**

...	functions to be nested together
.list	alternatively an explicit list of functions. If specified ... will be ignored.
x	a function
y	a function

**Details**

compose creates a functional composition of the listed functions. Functional composition of functions f and g is defined as f(g(.)). Order matters the right most function listed will be the innermost function in the composition, same with the operator version. To remember the order lists will be the order read out, ie. compose(f,g) = f(g(x))

When using the operator version it is good to remember that parentheses are recommended see the examples

**Value**

new function consisting of the functions nested

**Author(s)**

Andrew Redd

**Examples**

```
compose(any, is.na)(c(NA,1:3))
(sum%is.na)(c(1,NA)) #correct
## Not run:
sum%is.na(NA) #incorrect

## End(Not run)
```

---

dostats

*Convenient interface for computing statistics on a vector*

---

**Description**

Convenient interface for computing statistics on a vector

**Usage**

```
dostats(x, ..., .na.action = na.fail)
```

**Arguments**

x	the vector
...	statistics to compute, must take a vector and return a vector
.na.action	the action to take on NA values, for all statistics

**Value**

A one row data.frame with columns named as in ...

**Author(s)**

Andrew Redd

**See Also**

[ldply](#)

**Examples**

```
data(mtcars)
library(plyr)
dostats(1:10, mean, median, sd, quantile, IQR)
ldply(mtcars, dostats, median, mean, sd, quantile, IQR)
```

---

fill_v	<i>Fill vector to length with a specified value</i>
--------	---

---

**Description**

Fill vector to length with a specified value

**Usage**

```
fill_v(x, l = length(x), with = last(x), after = length(x))
```

**Arguments**

x	vector
l	length
with	What to fill with
after	where to insert

---

hdf	<i>Heirachical Data Frames</i>
-----	--------------------------------

---

**Description**

The primary purpose of the hdf is to represent data with grouped columns and grouped rows.

**Usage**

```
hdf(...)
```

**Arguments**

... vector, data.frame, or hdf objects to bind together.

---

listrows	<i>List rows of a data frame in a list.</i>
----------	---

---

**Description**

List rows of a data frame in a list.

**Usage**

```
listrows(d)
```

**Arguments**

d	a data.frame
---	--------------

---

make_call	<i>Make a call with extra arguments incorporated into call.</i>
-----------	---

---

**Description**

Usefull for using with plyr functions

**Usage**

```
make_call(args, ..., what, quote = F, envir = parent.frame())
```

**Arguments**

args	a list of arguments
...	extra arguments to be incorporated into args
what	the function to execute
quote	should the arguments be quoted
envir	the environment to call the function in

**See Also**

[do.call](#) which this function wraps.

make\_new\_id                    *Make a helper ID counter*

---

**Description**

Make a helper ID counter

**Usage**

```
make_new_id(startat = 0)
```

**Arguments**

startat                    where to start counting

---

me                            *Return the current function*

---

**Description**

Return the current function

**Usage**

```
me()
```

**See Also**

[sys.function](#)

---

onarg                        *change first argument of a function*

---

**Description**

change first argument of a function

**Usage**

```
onarg(f, arg)
```

**Arguments**

f                            the function  
arg                         the arg to be called as the first argument



**Value**

a function that calls `f` with `arg` as the first argument.

**See Also**

[wargs](#), [dostats](#), and [apply](#)

**Examples**

```
formals(runif)
onarg(runif, 'max')(1:10, 1)
onarg(runif, 'max')(1:10, 10)
#another version of contains
onarg(`%in%`, 'table')(letters, 'y')
```

---

pval

*Extract a p-value fomr a test result.*

---

**Description**

Extract a p-value fomr a test result.

**Usage**

```
pval(x, extended = F, ...)
```

**Arguments**

<code>x</code>	a testing result object
<code>extended</code>	should an extended result be given or a single p-value.
<code>...</code>	extra agruments

**Details**

This is a generic helper function for extracting p values from objects. The idea being to extract the overall p-value for the model that can be interpreted simply.

**Value**

either a single value (`extended=FALSE`) representing the p-value of the test or a single row. [data.frame](#) object that also incldues extra information such as

---

redirf	<i>Create a function that redirects to the named function.</i>
--------	--

---

**Description**

This is usefull for debugging to know what function has been called form within do.call or plyr functions.

**Usage**

```
redirf(f, envir = parent.frame())
```

**Arguments**

f	a function to wrap a call around
envir	environment to use for the function.

---

seq_consecutive	<i>compute an indicator to group consecutive values</i>
-----------------	---

---

**Description**

computes a vector that changes every time the element is different from the previous.

**Usage**

```
seq_consecutive(x, ...)
```

**Arguments**

x	a vector
...	ignored, might be used for forward compatibility.

**Value**

an integer vector.

---

Table1	<i>Create a table of descriptives of a dataset.</i>
--------	---

---

**Description**

Create a table of descriptives of a dataset.

**Usage**

```
Table1(formula, data)
```

**Arguments**

formula	a formula description of the table.
data	a data.frame or environment to extract the data from.

---

take_names	<i>Internal formula Commands</i>
------------	----------------------------------

---

**Description**

Internal formula Commands  
Internal formula manipulation and evaluation

**Usage**

```
take_names(x)
get_vars(x)
dseval_left(x, y, idf)
dseval_left_var(x, y, idf)
dseval_right(y, x, idf)
apply_by_two_fun(fun, x, y, idf)
dseval_fork(var, y, x, idf, gen_call)
```

**Arguments**

x	parsed left side of formula
y	parsed right side of formula
idf	idata.frame object
fun	function to apply
var	variable to extract call from
gen_call	generic call

---

wargs

*Call with arguments.*

---

**Description**

Call with arguments.

**Usage**

```
wargs(f, ..., args = pairlist(...), envir = parent.frame())
```

**Arguments**

f	a function
...	extra arguments
args	alternate way to provide arguments as a pairlist.
envir	environment to use for the function.

**Value**

a function that takes 1 argument and calls f with the single argument and the additional ... appended.

**Examples**

```
mean2 <- wargs(mean, na.rm=TRUE)
```

---

*%contains%*                      *Does a table contain a value*

---

**Description**

Does a table contain a value

**Usage**

table %contains% y

**Arguments**

table	a table of values
y	a value

**Details**

Literally

**Value**

a logical vector of the same length as y indicating if y is in table, i.e. the table contains y.

**See Also**

[match](#)

# Index

- \*Topic **misc**
  - compose, 4
  - dostats, 5
  - wargs, 12
- \*Topic **utilities**,
  - compose, 4
  - dostats, 5
  - wargs, 12
- .T, 2
- %% (compose), 4
- %contains%, 13
  
- apply, 9
- apply\_by\_two\_fun (take\_names), 11
  
- caply, 2
- class.stats, 3
- collect, 3
- compose, 4
- composition (compose), 4
- contains (%contains%), 13
  
- data.frame, 9
- do.call, 7
- dostats, 3, 5, 9
- dseval\_fork (take\_names), 11
- dseval\_left (take\_names), 11
- dseval\_left\_var (take\_names), 11
- dseval\_right (take\_names), 11
  
- factor.stats (class.stats), 3
- fill\_v, 6
  
- get\_vars (take\_names), 11
  
- hdf, 6
  
- integer.stats (class.stats), 3
  
- ldply, 5
- listrows, 7
  
- make\_call, 7
- make\_new\_id, 8
- match, 13
- me, 8
  
- nest (compose), 4
- numeric.stats (class.stats), 3
  
- onarg, 8
  
- pval, 9
  
- redirf, 10
  
- seq\_consecutive, 10
- sys.function, 8
  
- Table1, 11
- take\_names, 11
  
- wargs, 9, 12