Package ‘dostats’

October 13, 2022

Version 1.3.3
Date 2022-05-10
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)
Language en-US
Depends R (>= 2.12.0)
Imports methods, stats
Suggests plyr, testthat
Description A small package containing helper utilities for creating functions for computing statistics.
Collate 'T.R' 'capply.R' 'collect.R' 'compose.R' 'consecutive.R'
          'dostats.R' 'wargs.R' 'onarg.R' 'pval.R' 'utils.R'
RoxygenNote 7.1.2
NeedsCompilation no
Repository CRAN
Date/Publication 2022-05-10 19:30:02 UTC

R topics documented:

  .T ......................................................... 2
capply ....................................................... 2
class.stats .................................................. 3
collect ....................................................... 4
compose ...................................................... 4
dostats ....................................................... 5
fill_v ......................................................... 6
first ......................................................... 7
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
x data to apply fun to.
fun to apply
... other arguments to fun
class.stats  

Filter by class

Description
Filter by class

Usage

```
class.stats(.class)
numeric.stats(x, ...)
factor.stats(x, ...)
integer.stats(x, ...)
```

Arguments

- `.class` string for class to filter by
- `x` vector of any class
- `...` passed to `dostats`

Value

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

Functions

- `numeric.stats`: Numeric class statistics
- `factor.stats`: Factor class statistics
- `integer.stats`: Integer class statistics @export

See Also

dostats
collect  

**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 \( \text{fun}(v[1], v[2], ...) \) The result is x. Then \( f(x, v[3]) \) is called and so forth, until all elements has been exhausted.

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

**Examples**

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

---

compose  

**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, i.e. 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

Functions

• %.%: infix compose operator

Author(s)

Andrew Redd

Examples

compose(any, is.na)(c(NA,1:3))
(sum%.%is.na)(c(1,NA)) #correct
## Not run:
sum%.%is.an(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
...
.stats the vector 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 ...

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 Fill vector to length with a specified value

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


**Description**

Shortcuts for head(x,1) and tail(x,1)

**Usage**

first(x, ..., n = 1)

last(x, ..., n = 1)

**Arguments**

- **x**: vector object
- **...**: passed on to head or tail
- **n**: the new number to take of only one.

---

**listrows**  

*List rows of a data frame in a list.*

**Description**

List rows of a data frame in a list.

**Usage**

listrows(d)

**Arguments**

- **d**: a data.frame
make_call  
*Make a call with extra arguments incorporated into call.*

**Description**

Useful for using with **plyr** functions

**Usage**

```r
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**

```r
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 argument 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**

```r
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 from a test result.

Description

Extract a p-value from a test result.

Usage

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

Arguments

x a testing result object
extended should an extended result be given or a single p-value.
... extra arguments passed to methods.

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 includes extra information such as

redirf

Create a function that redirects to the named function.

Description

This is useful 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  
compute an indicator to group consecutive values

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.

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

contains(table,y)

Arguments

<table>
<thead>
<tr>
<th>table</th>
<th>a table of values</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>a value</td>
</tr>
</tbody>
</table>

Details

Literally %in% in reverse order, just for convenience.

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

* misc
  compose, 4
dostats, 5
wargs, 11
* utilities
  compose, 4
dostats, 5
wargs, 11
.T, 2
%.%.% (compose), 4
%contains%, 12

apply, 9
capply, 2
class.stats, 3
collect, 4
compose, 4
composition (compose), 4
contains (%contains%), 12

data.frame, 10
do.call, 8
dostats, 3, 5, 9

factor.stats (class.stats), 3
fill_v, 6
first, 7

integer.stats (class.stats), 3
last (first), 7
ldply, 6
listrows, 7

make_call, 8
make_new_id, 8
match, 12
me, 9

nest (compose), 4

numeric.stats (class.stats), 3
onarg, 9
pval, 10
redirf, 10
seq_consecutive, 11
sys.function, 9
wargs, 9, 11