Package ‘rapportools’

February 20, 2015

Maintainer  Gergely Daróczi <daroczig@rapporter.net>
Title  Miscellaneous (stats) helper functions with sane defaults for reporting
Type  Package
Encoding  UTF-8
Description  Helper functions that act as wrappers to more advanced statistical methods with the advantage of having sane defaults for quick reporting.
Author  Aleksandar Blagotić <alex@rapporter.net> and Gergely Daróczi <daroczig@rapporter.net>
Version  1.0
Date  2014-01-07
BugReports  https://github.com/rapporter/rapportools/issues
License  AGPL-3
Depends  reshape
Imports  plyr, pander
NeedsCompilation  no
Repository  CRAN
Date/Publication  2014-01-07 07:28:36

R topics documented:

adj.rle ................................................................. 2
alike.integer ....................................................... 3
capitalise ............................................................. 4
catn ................................................................. 4
fml ................................................................. 5
htest ................................................................. 5
htest.short ...................................................... 6
iqr ................................................................. 7
is.boolean .......................................................... 7
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>is.empty</td>
<td>8</td>
</tr>
<tr>
<td>is.number</td>
<td>8</td>
</tr>
<tr>
<td>is.string</td>
<td>9</td>
</tr>
<tr>
<td>is.tabular</td>
<td>10</td>
</tr>
<tr>
<td>is.variable</td>
<td>10</td>
</tr>
<tr>
<td>kurtosis</td>
<td>11</td>
</tr>
<tr>
<td>label</td>
<td>12</td>
</tr>
<tr>
<td>label&lt;-</td>
<td>12</td>
</tr>
<tr>
<td>lambda.test</td>
<td>13</td>
</tr>
<tr>
<td>max</td>
<td>14</td>
</tr>
<tr>
<td>mean</td>
<td>15</td>
</tr>
<tr>
<td>median</td>
<td>15</td>
</tr>
<tr>
<td>messagef</td>
<td>16</td>
</tr>
<tr>
<td>min</td>
<td>16</td>
</tr>
<tr>
<td>n</td>
<td>17</td>
</tr>
<tr>
<td>name</td>
<td>17</td>
</tr>
<tr>
<td>nmissing</td>
<td>18</td>
</tr>
<tr>
<td>nvalid</td>
<td>18</td>
</tr>
<tr>
<td>pct</td>
<td>19</td>
</tr>
<tr>
<td>percent</td>
<td>19</td>
</tr>
<tr>
<td>range</td>
<td>20</td>
</tr>
<tr>
<td>rp.desc</td>
<td>21</td>
</tr>
<tr>
<td>rp.freq</td>
<td>22</td>
</tr>
<tr>
<td>rp.outlier</td>
<td>23</td>
</tr>
<tr>
<td>sd</td>
<td>24</td>
</tr>
<tr>
<td>se.mean</td>
<td>24</td>
</tr>
<tr>
<td>skewness</td>
<td>25</td>
</tr>
<tr>
<td>stopf</td>
<td>25</td>
</tr>
<tr>
<td>sum</td>
<td>26</td>
</tr>
<tr>
<td>tocamel</td>
<td>26</td>
</tr>
<tr>
<td>trim.space</td>
<td>27</td>
</tr>
<tr>
<td>univar</td>
<td>28</td>
</tr>
<tr>
<td>var</td>
<td>28</td>
</tr>
<tr>
<td>vgsub</td>
<td>29</td>
</tr>
<tr>
<td>warningf</td>
<td>29</td>
</tr>
</tbody>
</table>

**Index**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>adj.rle</td>
<td>31</td>
</tr>
</tbody>
</table>

**Description**

Similar to `rle` function, this function detects "runs" of adjacent integers, and displays vector of run lengths and list of corresponding integer sequences.
Usage

adj.rle(x)

Arguments

x a numeric vector with

Value

a list with two elements: vector of run lengths, and another list of values corresponding to generated sequences' lengths.

Author(s)

Gabor Grothendieck <ggrothendieck@gmail.com>

References

See original thread for more details http://stackoverflow.com/a/8467446/457898. Special thanks to Gabor Grothendieck for this one!

Description

This function tests if given variable "appears" to be an integer. To qualify as such, two conditions need to be satisfied: it should be stored as numeric object, and it should pass regular expression test if it consists only of digits.

Usage

alike.integer(x)

Arguments

x a numeric variable that is to be tested

Value

a logical value that indicates that tested variable "looks like" integer
capitalise  

**Description**

Capitalises strings in provided character vector

**Usage**

```r
capitalise(x)
```

**Arguments**

- `x`: a character vector to capitalise

**Value**

character vector with capitalised string elements

**Examples**

```r
capitalise(c("foo", "bar")) # [1] "Foo" "Bar"
```

---

**catn**

**Concatenate with newline**

**Description**

A simple wrapper for `cat` function that appends newline to output.

**Usage**

```r
catn(...)```

**Arguments**

- `...`: arguments to be passed to `cat` function

**Value**

None (invisible `NULL`).
Create Formula from Strings

**Description**

Takes multiple character arguments as left and right-hand side arguments of a formula, and concatenates them in a single string.

**Usage**

```r
fml(left, right, join.left = " + ", join.right = " + ")
```

**Arguments**

- `left`: a string with left-hand side formula argument
- `right`: a character vector with right-hand side formula arguments
- `join.left`: concatenation string for elements of character vector specified in `left`
- `join.right`: concatenation string for elements of character vector specified in `right`

**Examples**

```r
fml("hp", c("am", "cyl")) # "hp ~ am + cyl"
```

Hypothesis Tests

**Description**

This function uses `htest.short`, to extract statistic and p-value from `htest`-classed object. Main advantage of using `htest` is that it's vectorised, and can accept multiple methods.

**Usage**

```r
htest(x, ..., use.labels =getOption("rapport.use.labels"),
      use.method.names = TRUE, colnames = c("Method", "Statistic", "p-value"))
```

**Arguments**

- `x`: arguments to be passed to function specified in `test`
- `...`: additional arguments for function specified in `test`
- `use.labels`: a logical value indicating whether variable labels should be placed in row names. If set to `FALSE`, output of `deparse(substitute(x))` will be used.
- `use.method.names`: use the string provided in `method` attribute of `htest` object
- `colnames`: a character string containing column names
Details

Default parameters are read from options:

• ‘rapport.use.labels’.

Value

a data.frame with applied tests in rows, and their results (statistic and p-value) in columns

Examples

```r
library(nortest)
htest(rnorm(100), shapiro.test)
htest(rnorm(100), lillie.test, ad.test, shapiro.test)
htest(mtcars, lillie.test)
htest(mtcars, lillie.test, ad.test, shapiro.test)
```

## Not run:

```r
htest.short(shapiro.test(rnorm(100)))
```

## End(Not run)

---

**htest.short**

*Extract Values from htest Objects*

**Description**

Extract value of statistic and its p-value from htest object.

**Usage**

```r
htest.short(x)
```

**Arguments**

- `x` htest-class object

**Value**

named numeric vector with the value of statistic and its p-value

**Examples**

```r
## Not run:
htest.short(shapiro.test(rnorm(100)))
```

## End(Not run)
### iqr

**Interquartile Range**

**Description**
Calculates interquartile range of given variable. See `univar` for details.

**Usage**

`iqr(...)`

**Arguments**

... parameters to be passed to `univar` function

**Value**

a numeric value with variable’s interquartile range

### is.boolean

**Boolean**

**Description**
Checks if provided object is a boolean i.e. a length-one logical vector.

**Usage**

`is.boolean(x)`

**Arguments**

`x` an object to check

**Value**

a logical value indicating whether provided object is a boolean

**Examples**

```r
## Not run:
is.boolean(TRUE) # [1] TRUE
# the following will work on most systems, unless you have tweaked global Rprofile
is.boolean(T) # [1] TRUE
is.boolean(1) # [1] FALSE
is.string(c("foo", "bar")) # [1] FALSE

## End(Not run)
```
is.empty

Empty Value

Description

Rails-inspired helper that checks if vector values are "empty", i.e. if it's: NULL, zero-length, NA, NaN, FALSE, an empty string or 0. Note that unlike its native R is.<something> sibling functions, is.empty is vectorised (hence the "values").

Usage

is.empty(x, trim = TRUE, ...)

Arguments

x
an object to check its emptiness
trim
trim whitespace? (TRUE by default)
...additional arguments for sapply

Examples

## Not run:
is.empty(NULL) # [1] TRUE
is.empty(c()) # [1] TRUE
is.empty(NA) # [1] TRUE
is.empty(NaN) # [1] TRUE
is.empty("") # [1] TRUE
is.empty(0) # [1] TRUE
is.empty(0.00) # [1] TRUE
is.empty(" ") # [1] TRUE
is.empty("foobar") # [1] FALSE
is.empty(" ", trim = FALSE) # [1] FALSE
# is.empty is vectorised!
all(is.empty(rep("", 10))) # [1] TRUE
all(is.empty(matrix(NA, 10, 10))) # [1] TRUE

## End(Not run)

is.number

Numbers

Description

Checks if provided object is a number, i.e. a length-one numeric vector.
is.number

Usage

is.number(x, integer = FALSE)

Arguments

x

an object to check

integer

logical: check if number is integer

Value

a logical value indicating whether provided object is a number

Examples

is.number(3) # [1] TRUE
is.number(3:4) # [1] FALSE
is.number("3") # [1] FALSE
is.number(NaN) # [1] TRUE
is.number(NA_integer_) # [1] TRUE

is.string

Strings

Description

Checks if provided object is a string i.e. a length-one character vector.

Usage

is.string(x)

Arguments

x

an object to check

Value

a logical value indicating whether provided object is a string

Examples

is.string("fooobar") # [1] TRUE
is.string(1) # [1] FALSE
is.string(c("foo", "bar")) # [1] FALSE
is.tabular  

Tabular Structure

Description
Checks if object has "tabular" structure (not to confuse with table) - in this particular case, that means matrix and data.frame objects only.

Usage
is.tabular(x)

Arguments
x an object to be checked for "tabular" format

Value
a logical value indicating that provided object has tabular structure

Examples
is.tabular(HairEyeColor[, , 1]) # [1] TRUE
is.tabular(mtcars) # [1] TRUE
is.tabular(table(mtcars$cyl)) # [1] FALSE
is.tabular(rnorm(100)) # [1] FALSE
is.tabular(LETTERS) # [1] FALSE
is.tabular(pi) # [1] FALSE

is.variable  

Variables

Description
From rapport's point of view, a variable is a non-NULL atomic vector that has no dimension attribute (see dim for details). This approach bypasses factor issues with is.vector, and also eliminates multidimensional vectors, such as matrices and arrays.

Usage
is.variable(x)

Arguments
x an object to be checked for "variable" format
kurtosis

Value

a logical value indicating that provided object is a "variable"

Examples

is.variable(rnorm(100))  # [1] TRUE
is.variable(LETTERS)     # [1] TRUE
is.variable(NULL)        # [1] FALSE
is.variable(mtcars)      # [1] FALSE
is.variable(HairEyeColor[, , 1]) # [1] FALSE
is.variable(list())      # [1] FALSE

Description

Calculates kurtosis coefficient for given variable (see is.variable), matrix or a data.frame.

Usage

kurtosis(x, na.rm = TRUE)

Arguments

x          a variable, matrix or a data.frame
na.rm      should NAs be removed before computation?

References


Examples

set.seed(0)
x <- rnorm(100)
kurtosis(x)
kurtosis(matrix(x, 10))
kurtosis(mtcars)
rm(x)
label<-  

Get Variable Label

Description
This function returns character value previously stored in variable’s label attribute. If none found, and fallback argument is set to TRUE (default), the function returns object’s name (retrieved by \texttt{deparse(substitute(x)))}, otherwise NA is returned with a warning notice.

Usage
\f[\texttt{label(x, fallback = TRUE, simplify = TRUE)}\]

Arguments
\f[\begin{align*}
x & \quad \text{an R object to extract labels from} \\
fallback & \quad \text{a logical value indicating if labels should fallback to object name(s)} \\
simplify & \quad \text{coerce results to a vector (TRUE by default), otherwise, a list is returned}
\end{align*}\]

Value
a character vector with variable’s label(s)

Examples
\f[\begin{align*}
\text{## Not run:} \\
x & \leftarrow \text{rnorm(100)} \\
\text{label(x)} & \quad \text{# returns “x”} \\
\text{label(x, FALSE)} & \quad \text{# returns NA and issues a warning} \\
\text{label(mtcars$hp)} & \leftarrow \text{“Horsepower”} \\
\text{label(mtcars)} & \quad \text{# returns “Horsepower” instead of “hp”} \\
\text{label(mtcars, FALSE)} & \quad \text{# returns NA where no labels are found} \\
\text{label(sleep, FALSE)} & \quad \text{# returns NA for each variable and issues a warning}
\end{align*}\]

\f[\text{## End(Not run)}\]

label<-  

Set Variable Label

Description
This function sets a label to a variable, by storing a character string to its label attribute.

Usage
\f[\texttt{label(var) <- value}\]
Arguments

var a variable (see is.variable for details)
value a character value that is to be set as variable label

See Also

label

Examples

## Not run:
label(mtcars$mpg) <- "fuel consumption"
x <- rnorm(100)
(label(x) <- "pseudo-random normal variable")

## End(Not run)

lambda.test  

*Goodman and Kruskal’s lambda*

Description

Computes Goodman and Kruskal’s lambda for given table.

Usage

lambda.test(table, direction = 0)

Arguments

table a table of two variables or a data.frame representation of the cross-table of the two variables without marginals
direction numeric value of c(0, 1, 2) where 1 means the lambda value computed for row, 2 for columns and 0 for both

Value

computed lambda value(s) for row/col of given table

References

Examples

```r
## Not run:
## quick example
x <- data.frame(x = c(5, 4, 3), y = c(9, 8, 7), z = c(7, 11, 22), zz = c(1, 15, 8))
lambda.test(x) # 0.1 and 0.18333
lambda.test(t(x)) # 0.18333 and 0.1

## historical data (see the references above: p. 744)
men.hair.color <- data.frame(
  b1 = c(1768, 946, 115),
  b2 = c(807, 1387, 438),
  b3 = c(189, 746, 288),
  b4 = c(47, 53, 16)
)
row.names(men.hair.color) <- paste0("'a', 1:3)
lambda.test(men.hair.color)
lambda.test(t(men.hair.color))

## some examples on mtcars
lambda.test(table(mtcars$am, mtcars$gear))
lambda.test(table(mtcars$gear, mtcars$am))
lambda.test(table(mtcars$am, mtcars$gear), 1)
lambda.test(table(mtcars$am, mtcars$gear), 2)

## End(Not run)
```

---

### max

<table>
<thead>
<tr>
<th><strong>Maximum</strong></th>
</tr>
</thead>
</table>

**Description**

Returns the maximum of all values in a vector by passing `max` as `fn` argument to `univar` function.

**Usage**

```r
max(...)  
```

**Arguments**

... parameters to be passed to `univar` function

**Value**

a numeric value with maximum value
<table>
<thead>
<tr>
<th><strong>mean</strong></th>
<th><strong>Mean</strong></th>
</tr>
</thead>
</table>

**Description**

Calculates mean of given variable by passing `sum` as fn argument to `univar` function.

**Usage**

```
mean(...)  
```

**Arguments**

```
...  
```

parameters to be passed to `univar` function

**Value**

a numeric value with variable’s mean

<table>
<thead>
<tr>
<th><strong>median</strong></th>
<th><strong>Median</strong></th>
</tr>
</thead>
</table>

**Description**

Calculates median of given variable. See `univar` for details.

**Usage**

```
median(...)  
```

**Arguments**

```
...  
```

parameters to be passed to `univar` function

**Value**

a numeric value with variable’s median
### messagef

**Send Message with String Interpolated Messages**

**Description**

Combines `warning` with `sprintf` thus allowing string interpolated diagnostic messages.

**Usage**

```r
messagef(s, ...)
```

**Arguments**

- `s`: a character vector of format strings
- `...`: values to be interpolated

**Examples**

```r
# Not run:
messagef("%.3f is not larger than %d and/or smaller than %d", pi, 10, 40)

# End(Not run)
```

---

### min

**Minimum**

**Description**

Returns the minimum of all values in a vector by passing `min` as fn argument to `univar` function.

**Usage**

```r
min(...)
```

**Arguments**

- `...`: parameters to be passed to `univar` function

**Value**

a numeric value with minimum value
### Number of Cases

**Description**

Returns the number of valid (non-NA) values in a variable. This is a wrapper around `univar` function with `length` function passed in `fn` argument, but with missing values previously removed. However, it’s not possible to cancel NA omission with this function (doing so will yield error).

**Usage**

```r
n(...)```

**Arguments**

- `...` parameters to be passed to `univar` function

**Value**

- a numeric value with number of valid (non-NA) vector elements

---

### Variable Name

**Description**

This function returns character value previously stored in variable’s `name` attribute. If none found, the function defaults to object’s name.

**Usage**

```r
name(x)```

**Arguments**

- `x` an R (atomic or data.frame/list) object to extract names from

**Value**

- a character value with variable’s label

**Examples**

```r
## Not run:
name(mtcars$am)
x <- 1:10
name(x)

## End(Not run)```
nmissing  
*Number of Missing Cases*

**Description**

Returns a number of missing (NA) values in a variable. This is a wrapper around `univar` function with anonymous function passed to count number of NA elements in a variable.

**Usage**

`nmissing(...)`

**Arguments**

... parameters to be passed to `univar` function

**Value**

a numeric value with number of missing vector elements

---

nvalid  
*Number of Valid Cases*

**Description**

Returns the number of valid (non-NA) values in a variable. This is a wrapper around `univar` function with `length` function passed in `fn` argument, but with missing values previously removed. However, it’s not possible to cancel NA omission with this function (doing so will yield error).

**Usage**

`nvalid(...)`

**Arguments**

... parameters to be passed to `univar` function

**Value**

a numeric value with number of valid (non-NA) vector elements
### pct

**Description**

Appends a percent sign to provided numerical value. Rounding is carried out according to value passed in decimals formal argument (defaults to value specified in panderOptions('digits')).

**Usage**

```r
pct(x, digits = panderOptions("digits"), type = c("percent", ",\%", "proportion"), check.value = TRUE)
```

**Arguments**

- `x`: a numeric value that is to be rendered to percent
- `digits`: an integer value indicating number of decimal places
- `type`: a character value indicating whether percent or proportion value was provided (partial match is allowed)
- `check.value`: perform a sanity check to see if provided numeric value is correct (defaults to TRUE)

**Value**

a character value with formatted percent

---

### percent

**Description**

Calculates percentage of cases for provided variable and criteria specified in subset argument. Function accepts numeric, factor and logical variables for `x` parameter. If numeric and/or factor is provided, subsetting can be achieved via subset argument. Depending on value of `na.rm` argument, either valid (`na.rm = TRUE`) or all cases (`na.rm = FALSE`) are taken into account. By passing logical variable to `x`, a sum of (TRUE) elements is calculated instead, and valid percents are used (NA are excluded).

**Usage**

```r
percent(x, subset = NULL, na.rm = TRUE, pct = FALSE, ...)
```
Arguments

- `x`: a numeric variable to be summarised
- `subset`: an expression that evaluates to logical vector (defaults to NULL)
- `na.rm`: should missing values be
- `pct`: print percent string too?
- `...`: additional arguments for `pct` function

Value

a numeric or string depending on the value of `pct`

Examples

```r
## Not run:
set.seed(0)
x <- sample(5, 100, replace = TRUE)
percent(x > 2)

## End(Not run)
```

range

Description

Calculates difference between the largest and the smallest value in a vector. See `univar` for details.

Usage

```
range(...)  
```

Arguments

- `...`: parameters to be passed to `univar` function

Value

a numeric value with calculated range
rp.desc

Descriptive Statistics

Description

Aggregate table of descriptives according to functions provided in fn argument. This function follows melt/cast approach used in reshape package. Variable names specified in measure.vars argument are treated as measure.vars, while the ones in id.vars are treated as id.vars (see melt.data.frame for details). Other its formal arguments match with corresponding arguments for cast function. Some post-processing is done after reshaping, in order to get pretty row and column labels.

Usage

rp.desc(measure.vars, id.vars = NULL, fn = NULL, data = NULL, na.rm = TRUE, margins = NULL, subset = TRUE, fill = NA, add.missing = FALSE, total.name = "Total", varcol.name = "Variable", use.labels =getOption("rapport.use.labels"), remove.duplicate = TRUE)

Arguments

measure.vars either a character vector with variable names from data, a numeric vector, or a data.frame
id.vars same rules apply as in measure.vars, but defaults to NULL
fn a list with functions or a character vector with function names
data a data.frame holding variables specified in id.vars and measure.vars
na.rm a logical value indicating whether NA values should be removed
margins should margins be included? (see documentation for eponymous argument in melt.data.frame)
subset a logical vector to subset the data before aggregating
fill value to replace missing level combinations (see documentation for eponymous argument in melt.data.frame)
add.missing show missing level combinations
total.name a character string with name for "grand" margin (defaults to "Total")
varcol.name character string for column that contains summarised variables (defaults to "Variable")
use.labels use labels instead of variable names in table header (handle with care, especially if you have lengthy labels). Defaults to value specified in rapport.use.labels option.
remove.duplicate should name/label of the variable provided in measure.vars be removed from each column if only one measure.var is provided (defaults to TRUE)
Value

a data.frame with aggregated data

Examples

```r
rp.desc("cyl", "am", c(mean, sd), mtcars, margins = TRUE)
## c
rp.desc("hp", c("am", "gear"), c("Average" = mean, "Deviation" = sd),
         mtcars, remove.duplicate = FALSE)
```

**rp.freq**

**Frequency Table**

Description

Display frequency table with counts, percentage, and cumulatives.

Usage

```r
rp.freq(f.vars, data, na.rm = TRUE, include.na = FALSE,
         drop.unused.levels = FALSE, count = TRUE, pct = TRUE,
         cumul.count = TRUE, cumul.pct = TRUE, total.name = "Total",
         reorder = FALSE)
```

Arguments

- `f.vars` a character vector with variable names
- `data` a data.frame
- `na.rm` should missing values be removed?
- `include.na` should missing values be included in frequency table?
- `drop.unused.levels` should empty level combinations be left out
- `count` show frequencies?
- `pct` show percentage?
- `cumul.count` show cumulative frequencies?
- `cumul.pct` show cumulative percentage?
- `total.name` a sting containing footer label (defaults to "Total")
- `reorder` reorder the table based on frequencies?

Value

a data.frame with a frequency table
Examples

```r
## Not run:
rp.freq(c("am", "cyl", "vs"), mtcars)
## End(Not run)
```

---

rp.outlier \hspace{1cm} Outlier test

Description

A simple test for outliers. This function returns all extreme values (if any) found in the specified vector.

Usage

```r
rp.outlier(x)
```

Arguments

- `x` a numeric vector of values

Value

vector of outlier values

References

Credit goes to PaulHurleyuk: [http://stackoverflow.com/a/1444548/564164](http://stackoverflow.com/a/1444548/564164)


Examples

```r
## Not run:
rp.outlier(mtcars$hp)
rp.outlier(c(rep(1,100), 200))
rp.outlier(c(rep(1,100), 200,201))
## End(Not run)
```
**sd**  
*Standard Deviation*

**Description**  
Calculates standard deviation of given variable. See `univar` for details.

**Usage**  
`sd(...)`

**Arguments**  
`...` parameters to be passed to `univar` function

**Value**  
a numeric value with variable’s standard deviation

---

**se.mean**  
*Standard Error of Mean*

**Description**  
Calculates standard error of mean for given variable. See `univar` for details.

**Usage**  
`se.mean(...)`

**Arguments**  
`...` parameters to be passed to `univar` function

**Value**  
a numeric value with standard error of mean
### skewness

**Description**
Calculates skewness coefficient for given variable (see `is.variable`), matrix or a data.frame.

**Usage**

```r
skewness(x, na.rm = TRUE)
```

**Arguments**

- `x`: a variable, matrix or a data.frame
- `na.rm`: should NAs be removed before computation?

**References**


**Examples**

```r
set.seed(0)
x <- rnorm(100)
skewness(x)
skewness(matrix(x, 10))
skewness(mtcars)
rm(x)
```

---

### stopf

**Description**
This helper combines `stop` function with `sprintf` thus allowing string interpolated messages when execution is halted.

**Usage**

```r
stopf(s, ...)
```

**Arguments**

- `s`: a character vector of format strings
- `...`: values to be interpolated
Value
a string containing message that follows execution termination

Examples

## Not run:
stopf("%3f is not larger than %d and/or smaller than %d", pi, 10, 40)

## End(Not run)

sum $ Sum

Description
Returns the sum of variable’s elements, by passing sum as fn argument to univar function.

Usage
sum(…)

Arguments
… parameters to be passed to univar function

Value
a numeric value with sum of vector elements

tocamel $ CamelCase

Description
Convert character vector to camelcase - capitalise first letter of each word.

Usage
tocamel(x, delim = "[^[:alnum:]]", upper = FALSE, sep = "", …)

Arguments
x a character vector to be converted to camelcase
delim a string containing regular expression word delimiter
upper a logical value indicating if the first letter of the first word should be capitalised (defaults to FALSE)
sep a string to separate words
… additional arguments to be passed to strsplit
Value

a character vector with strings put in camelcase

Examples

tocamel("foo.bar")
  ## [1] "fooBar"

tocamel("foo.bar", upper = TRUE)
  ## [1] "FooBar"

tocamel(c("foobar", "foo.bar", "camel_case", "a.b.c.d"))
  ## [1] "foobar" "foobar" "camelCase" "aBCD"

trim.space

Trim Spaces

Description

Removes leading and/or trailing space(s) from a character vector. By default, it removes both leading and trailing spaces.

Usage

trim.space(x, what = c("both", "leading", "trailing", "none"),
         space.regex = "[[:space:]]", ...)

Arguments

  x         a character vector which values need whitespace trimming
  what    which part of the string should be trimmed. Defaults to both which removes trailing and leading spaces. If none, no trimming will be performed.
  space.regex    a character value containing a regex that defines a space character
  ...     additional arguments for gsub function

Value

a character vector with (hopefully) trimmed spaces
univar

Descriptive Statistics

Description
This function operates only on vectors or their subsets, by calculating a descriptive statistic specified in fn argument.

Usage
univar(x, subset = NULL, fn, na.rm = TRUE, ...)

Arguments
- x: a numeric variable to be summarised
- subset: an expression that evaluates to logical vector (defaults to NULL, in which case the function specified in fn is applied on a vector)
- fn: a function or a function name to be applied on a variable or its subset
- na.rm: a logical value indicating whether NA’s should be removed (defaults to TRUE)
- ...: additional arguments for function specified in fn

Value
a numeric

var

Variance

Description
Calculates variance of given variable. See univar for details.

Usage
var(...)
vgsub

Vectorised String Replacement

Description

A simple wrapper for `gsub` that replaces all patterns from `pattern` argument with ones in `replacement` over vector provided in argument `x`.

Usage

```r
vgsub(pattern, replacement, x, ...)
```

Arguments

- `pattern` see eponymous argument for `gsub` function
- `replacement` see eponymous argument for `gsub` function
- `x` see eponymous argument for `gsub` function
- `...` additional arguments for `gsub` function

Value

a character vector with string replacements

References

See original thread for more details [http://stackoverflow.com/a/6954308/457898](http://stackoverflow.com/a/6954308/457898). Special thanks to user Jean-Robert for this one!

warningf

Send Warning with String Interpolated Messages

Description

Combines `warning` with `sprintf` thus allowing string interpolated warnings.

Usage

```r
warningf(s, ...)
```

Arguments

- `s` a character vector of format strings
- `...` values to be interpolated
Examples

## Not run:

```r
warningf("%3f is not larger than %d and/or smaller than %d", pi, 10, 40)
```

## End(Not run)
Index

adj.rle, 2
alike.integer, 3
capitalise, 4
cast, 21
cat, 4
catn, 4
data.frame, 10
fml, 5
gsub, 27, 29
h.test, 5
h.test.short, 5, 6
IQR (iqr), 7
iqr, 7
is.boolean, 7
is.empty, 8
is.number, 8
is.string, 9
is.tabular, 10
is.variable, 10, 11, 13, 25
is.vector, 10
kurtosis, 11
labeled, 12, 13
label<-, 12
lambda.test, 13
length, 17, 18
matrix, 10
max, 14, 14
mean, 15
median, 15
melt.data.frame, 21
message.frame, 16
min, 16, 16
n, 17
name, 17
nmissing, 18
numeric, 3
nvalid, 18
pct, 19, 20
percent, 19
range, 20
rle, 2
rp.desc, 21
rp.freq, 22
rp.iqr (iqr), 7
rp.label (label), 12
rp.label< (label<), 12
rp.max (max), 14
rp.mean (mean), 15
rp.median (median), 15
rp.min (min), 16
rp.missing (nmissing), 18
rp.n (n), 17
rp.name (name), 17
rp.outlier, 23
rp.percent (percent), 19
rp.range (range), 20
rp.sd (sd), 24
rp.se.mean (se.mean), 24
rp.sum (sum), 26
rp.valid (nvalid), 18
rp.var (var), 28
sapply, 8
sd, 24
se.mean, 24
skewness, 25
stopf, 25
sum, 15, 26, 26
table, 10
tocamel, 26
trim.space, 27
univar, 7, 14–18, 20, 24, 26, 28, 28
var, 28
vgsub, 29
warningf, 29