Package: rutabaga (via r-universe)

August 21, 2024

Type Package

Title Simple R Tools for Analysis and Visualizations

Version 0.1.7

Description Provides functions (R, C++) to speed up array calculations. Includes various tools for prettier visualizations via R base plots.

License GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/dipterix/rutabaga

BugReports https://github.com/dipterix/rutabaga/issues

Depends R (>= 3.5)

Imports utils, stats, grDevices, graphics, methods, magrittr, stringr (>= 1.3.1), digest, dipsaus

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Repository https://rave-ieeg.r-universe.dev

RemoteUrl https://github.com/dipterix/rutabaga

RemoteRef HEAD

RemoteSha e47423870b34ca17a1736ce8d4fe146b2697de19

Contents

add_points — *Same As Points, But Can Be Jittered (maturing)*

Description

Same As Points, But Can Be Jittered

(maturing)

Usage

```
add_points(x, y, jitr_x = 0, pch = 19, ...)
```

Arguments

`x, y`	plot data
`jitr_x`	jitter shift size for x
`pch`	data point type, default is 19, See `points`
`...`	other params to `jitr`

as_pdf — *pdf wrapper that evaluates an arbitrary expression.*

Description

pdf wrapper that evaluates an arbitrary expression.

Usage

```
as_pdf(fname, w, h, expr, TEST = FALSE, bg = "white")
```

Arguments

`fname`	file path
`w`	the requested width of the PDF
`h`	the requested height of the PDF
`expr`	the expression to evaluate to produce the plot
`TEST`	if FALSE (the default) print to PDF, otherwise print to the current graphics device
`bg`	set the background color of the PDF, defaults to 'white'

Value

the output of the resulting expression (the plot is likely produced by side effect but may additionally 'return' a value)

as_title *Make nice plot titles*

Description

Make nice plot titles

Usage

```
as_title(x, ...)

## S3 method for class 'formula'
as_title(x, ...)

## Default S3 method:
as_title(
  x,
  capitalize_all = TRUE,
  excluded = c("is", "are", "vs", "v.s.", "from", "of", "be", "for", "over"),
  ...
)

## S3 method for class 'fres'
as_title(x, ...)

## S3 method for class 'tres'
as_title(x, ...)
```

Arguments

x	title content
...	pass to other methods
capitalize_all	make the first letter cap
excluded	exclude making first letter cap, numeric vector

Examples

```
## Not run:
# Display normal title
plot_clean(1,1, main = as_title('this is title'))

# Only capitalize the first character
plot_clean(1,1, main = as_title('this is title', capitalize_all = F))

# What if title is a formula
plot_clean(1,1, main = as_title(p[value](beta[1]) < ~.(0.01) ~ ' is significant'))
```

```
# Display t-statistics
plot_clean(1,1)
t_stat = get_t(rnorm(10))
title(main=as_title(t_stat))

## End(Not run)
```

cbind_list *Make cbind magrittr compatible (stable)*

Description

Make cbind magrittr compatible (stable)

Usage

```
cbind_list(ll)
```

Arguments

ll	the list to bind, each element will be a column

clip_x *Ensure Data Are Within Some Bounds (stable)*

Description

Ensure Data Are Within Some Bounds

(stable)

Usage

```
clip_x(x, lim)
```

Arguments

x	data
lim	clip range, length of 1 or more. If length(lim) is 1, then the clip is symmetric

`create_frames` *add decoration (frames) to plots based on their layout location*

Description

add decoration (frames) to plots based on their layout location

Usage

```
create_frames(
  layout,
  common = NULL,
  bottom = NULL,
  left = NULL,
  top = NULL,
  right = NULL,
  env = parent.frame(),
  exclude = NULL
)
```

`dir_signature` *Create directory signature*

Description

Create directory signature

Usage

```
dir_signature(
  path = ".",
  file.info = FALSE,
  md5sum = TRUE,
  recursive = TRUE,
  ...
)
```

Arguments

`path` directory path to generate signatures

`file.info`, `md5sum`, `recursive`, `...`
passed to `fileSnapshot`

do_if *Clauses With Side Effects (Plotting Etc) (questioning)*

Description

Clauses With Side Effects (Plotting Etc)

(questioning)

Usage

```
do_if(boolean_expression, if_clause, else_clause = NULL)
```

Arguments

`boolean_expression`	expression that returns true or false
`if_clause`	if true, do if_clause
`else_clause`	if false, do else_clause

do_poly *Polygon plot*

Description

Polygon plot

Usage

```
do_poly(x, y, col, alpha = 50, border = NA, ...)
```

Arguments

`x, y`	x and y
`col`	color
`alpha`	0-255 transparency
`border`	border of polygon
`...`	passed to polygon

ebars *Draw Symmetric Error Bars*

Description

Draw Symmetric Error Bars

Usage

```
ebars(
  x,
  y = NULL,
  sem = NULL,
  length = 0.05,
  type = "n",
  col = "black",
  pt.col = col,
  code = 2,
  ...
)
```

Arguments

x, y	data to plot
sem	error range (half)
length	line length
type	type of xy data see points
col	default color for points and error areas
pt.col	point colors
code	arrow code
...	other params passed to ebars.y

Examples

```
## Not run:
plot_clean(0:10, -1:5, xlab = 'X')
ruta_axis(side = 2, at = -1:8)
ebars(x = c(2, 4), y = c(0, 3), sem = c(1, 0.5), col = c(2, 3))

## End(Not run)
```

ebar_polygon *Draw Symmetric Error Bars (stable)*

Description

Draw Symmetric Error Bars

(stable)

Usage

```
ebar_polygon(
  x,
  y,
  sem,
  alpha = 100,
  col = "black",
  fill = col,
  stroke = col,
  border = NA,
  add_line = TRUE,
  lwd = 1,
  ...
)
```

Arguments

`x, y`	plot data
`sem`	error bar half width
`alpha`	transparency
`col`	color
`fill`	error bar color
`stroke`	if add lines, line color
`border`	errorbar border width
`add_line`	add line(x,y) or not
`lwd`	line weight
`...`	passed to `lines`

Examples

```
## Not run:
plot_clean(0:10, -1:5, xlab = 'X')
ruta_axis(side = 2, at = -1:8)
ebar_polygon(1:10, (1:10)/2, rnorm(10))

## End(Not run)
```

fix_pdf_name *Ensure that the file names ends in ".pdf"*

Description

Ensure that the file names ends in ".pdf"

Usage

```
fix_pdf_name(fname)
```

Arguments

`fname`	potential file name

format_stat *helper function to build value labels*

Description

(questioning)

Usage

```
format_stat(nm, stats = c("b", "t", "p"))
```

Arguments

`nm`	statistics
`stats`	names of statistics

F_NOOP *Apply function to input but returns input itself*

Description

Execute the function (usually for its side effect) then return (invisibly) the input to the function

Usage

```
F_NOOP(x, FUN, ...)
```

Arguments

`x`	input
`FUN`	function to apply on `x`
`...`	additional parameters passing to `FUN`

getAlphaRGB *Get Hex Color With Transparency*

Description

Get Hex Color With Transparency

Usage

```
getAlphaRGB(colname, alpha)
```

Arguments

colname	name or number of color
alpha	transparency

Examples

```
## Not run:
getAlphaRGB('red', 0.5)

## End(Not run)
```

get_data_range *Get Data Range From A Collection Of Named Lists (questioning)*

Description

Get Data Range From A Collection Of Named Lists

(questioning)

Usage

```
get_data_range(ll, name = "range", ...)
```

Arguments

ll	list
name	element name
...	additional params for get_list_elements

get_f *Get statistics from linear model*

Description

(questioning)

Usage

```
get_f(formula, ...)

format_f(lm.mod, test_name = "All")
```

Arguments

`formula, ...`	passed to lm
`lm.mod`	linear model returned by lm()
`test_name`	name

get_filename *Get the file name from a full file path*

Description

Get the file name from a full file path

Usage

```
get_filename(full_path, keep_extension = FALSE)
```

Arguments

`full_path`	path to file
`keep_extension`	whether the file extension should be retained (defaults to FALSE)

get_list_elements *Get Elements/Slot/Attributes From List (stable)*

Description

Get Elements/Slot/Attributes From List

(stable)

Usage

```
get_list_elements(
  ll,
  name,
  drop_nulls = TRUE,
  is_attr = FALSE,
  use_sapply = TRUE,
  ...
)
```

Arguments

`ll`	list of elements
`name`	attribute/name/slot to extract
`drop_nulls`	drop NULL results
`is_attr`	are we extracting attributes?
`use_sapply`	try to return a matrix/vector if possible?
`...`	if is_attr, additional params to `attr`

get_t *helper function for t-tests that returns the values wanted by format_stat*

Description

(questioning)

Usage

```
get_t(...)
```

Arguments

`...`	passed to t.test

`is_within` *Check if a is within the range of b (stable)*

Description

Check if a is within the range of b

(stable)

Usage

```
is_within(a, b)

a %within% b
```

Arguments

`a`	element to check (numeric)
`b`	vector of numbers

`jitr` *Return Jittered X (experimental)*

Description

Return Jittered X

(experimental)

Usage

```
jitr(x, len = length(x), r)
```

Arguments

`x`	data
`len`	length of x
`r`	jitter size

`m_sd` *Function To Return Mean And Standard Deviation (Na Ignored by default) (stable)*

Description

Function To Return Mean And Standard Deviation (Na Ignored by default)

(stable)

Usage

```
m_sd(x, na.rm = TRUE)
```

Arguments

`x`	data
`na.rm`	remove NAs?

`m_se` *Function To Return Mean And Standard Error (stable)*

Description

Function To Return Mean And Standard Error

(stable)

Usage

```
m_se(x)

mat_m_se(m, DIM = 2)
```

Arguments

`x`	data
`m`	matrix data
`DIM`	margin, 1 for row, 2 for column. See apply

NOOP *Apply expression but returns something else*

Description

Evaluate an expression, but then return the input

Usage

```
NOOP(x, expr = NULL)
```

Arguments

`x`	object to return
`expr`	expression to run

not_NA *Return True If Not Na (stable)*

Description

Return True If Not Na

(stable)

make it easier to say not is.na in a pipe'd context

Usage

```
not_NA(x)
```

```
not_NA(x)
```

Arguments

`x`	data

not_null *Return True If Not Null (stable)*

Description

Return True If Not Null

(stable)

Usage

```
not_null(x)
```

Arguments

x	data

plot_clean *Create A Blank Plot With Given X And Y Range (stable)*

Description

Create A Blank Plot With Given X And Y Range

(stable)

Usage

```
plot_clean(
  xlim,
  ylim,
  x = 1,
  y = 1,
  type = "n",
  xlab = "",
  ylab = "",
  cex.main = rave_cex.main,
  cex.axis = rave_cex.axis,
  cex.lab = rave_cex.lab,
  ...
)
```

Arguments

`xlim`	numeric vector
`ylim`	numeric vector
`x`	x
`y`	y
`type`	default is 'n', i.e. plot nothing. See `plot.default`
`xlab`	x label
`ylab`	y label
`cex.main`	title font size
`cex.axis`	axis ticks font size
`cex.lab`	axis label font size
`...`	other params passed to `plot`

Examples

```
## Not run:
# create a blank plot with x from 0 to 10 and y from 1 to 5
plot_clean(0:10, 1:5, xlab = 'X')

## End(Not run)
```

---

plot_msg *Show A Blank Plot With Messages (maturing)*

Description

Show A Blank Plot With Messages

(maturing)

Usage

```
plot_msg(main = "No Conditions Specified")
```

Arguments

`main`	the title/msg to show

Examples

```
## Not run:
plot_msg("Let's Say Something")

## End(Not run)
```

`plus_minus` *Easy Way To Get +/- From A Long Vector*

Description

Easy Way To Get +/- From A Long Vector

Usage

```
plus_minus(x, d)
```

Arguments

`x`	data
`d`	plus minus value(s)

`pm` *Easy Way To Get +/- From A Long Vector (deprecated)*

Description

Easy Way To Get +/- From A Long Vector

(deprecated)

Usage

```
pm(x, d)
```

Arguments

`x`	data
`d`	plus minus value(s)

`print_summary` *Print summary and return original input*

Description

output of a function, but return its input to allow chaining

Usage

```
print_summary(x, FUN = summary, ...)
```

Arguments

`x`	object to print summary
`FUN`	function to print on
`...`	additional parameters passing to `FUN`

`pscl` *Enforce Sum To 1, Ignoring Na In The Sum, But Keeping Them In The Output (questioning)*

Description

Enforce Sum To 1, Ignoring Na In The Sum, But Keeping Them In The Output

(questioning)

Usage

```
pscl(x)
```

Arguments

`x`	data

rave_barplot *Barplot Function That Uses All The Rave Sizes And Colors (stable)*

Description

Barplot Function That Uses All The Rave Sizes And Colors

(stable)

Usage

```
rave_barplot(
  height,
  cex.axis = rave_cex.axis,
  cex.lab = rave_cex.lab,
  cex.names = rave_cex.lab,
  ...
)
```

Arguments

`height`, `cex.axis`, `cex.lab`, `cex.names`, `...`	passed to barplot but the default values are changed

rbind_list *Make rbind magrittr compatible (stable)*

Description

Make rbind magrittr compatible (stable)

Usage

```
rbind_list(ll)
```

Arguments

`ll`	the list to bind, each element will be a row

`remove_tail` *Remove the last k elements from a vector (list) Returns x (with a warning) if k < 1*

Description

Remove the last k elements from a vector (list) Returns x (with a warning) if k < 1

Usage

```
remove_tail(x, k = 1)
```

Arguments

x	the vector
k	the number of items to remove

`round_range` *Get A Integer Interval That Contains X (maturing)*

Description

Get A Integer Interval That Contains X

(maturing)

Usage

```
round_range(x)
```

Arguments

x	vector/matrix, numeric

Examples

```
## Not run:
# 0 - 11
round_range(0.5:10.5)

## End(Not run)
```

`row_apply_ii` *Apply function along the first dimension*

Description

row applier with an additional index variable, ii

Usage

```
row_apply_ii(mat, FUN_, ...)
```

Arguments

`mat`	matrix or array
`FUN_`	function to apply to each row
`...`	additional parameters passing to `FUN_`

`row_scale` *helper to do row scaling*

Description

helper to do row scaling

Usage

```
row_scale(mat)
```

Arguments

`mat`	a matrix

ruta_axis *A Neat Way To Show Axis (stable)*

Description

A Neat Way To Show Axis

(stable)

Usage

```
ruta_axis(
  side,
  at,
  tcl = -0.3,
  labels = at,
  las = 1,
  cex.axis = rave_cex.axis,
  cex.lab = rave_cex.lab,
  mgpy = c(3, 0.6, 0),
  mgpx = c(3, 0.75, 0),
  ...
)
```

Arguments

side	1 to 4: bottom, left, up, right. See axis
at, tcl, labels, las, mgpy, mgpx, ...	passed to axis
cex.axis	tick size
cex.lab	label size

Examples

```
## Not run:
# create a blank plot with x from 0 to 10 and y from 1 to 5
plot_clean(0:10, 1:5, xlab = 'X')
ruta_axis(side = 1, at = 1:8)

## End(Not run)
```

sapply_ii *Apply each elements with index as inputs*

Description

sapply with an additional index variable, ii

Usage

```
sapply_ii(X, FUN_, simplify = TRUE, USE.NAMES = TRUE, ...)
```

Arguments

X	vector or list
FUN_	function to apply on X
simplify, USE.NAMES, ...	passed to sapply

scale_01 *0-1 Scale The Data So We Can Manage The Plot Ranges Easily (stable)*

Description

0-1 Scale The Data So We Can Manage The Plot Ranges Easily

(stable)

Usage

```
scale_01(x)
```

Arguments

x	data to be rescaled

`stretch` *Useful For Plotting When You Want To Go A Bit Beyond The Data (experimental)*

Description

Useful For Plotting When You Want To Go A Bit Beyond The Data

(experimental)

Usage

```
stretch(x, pct)
```

Arguments

`x`	data
`pct`	stretch percentage

`str_collapse` *Convert vector into comma-separated string*

Description

Convert vector into comma-separated string

Usage

```
str_collapse(x, by = ", ", ...)
```

Arguments

`x`	the vector to collapse
`by`	the separating token (default is ', ')
`...`	further arguments for paste0

Value

a character vector as produced by paste0

`to_pdf` *Function for repeatedly writing plots to PDFs*

Description

Function for repeatedly writing plots to PDFs

Usage

```
to_pdf(PLOT, w, h, mar = rep(1, 4))
```

Arguments

`PLOT`	a function that produces a plot
`w`	the requested width of the PDF
`h`	the requested height of the PDF
`mar`	the margins of the PDF, set by a call to par(mar=mar). Defaults to c(1,1,1,1)

Value

A function that takens optional arguments to the PLOT function and fname, the name of the PDF. One-off changes to w, h, and mar, are specified with width, height, and margin, respectively

`trim` *Trim Data By Standard Error (stable)*

Description

Trim Data By Standard Error

(stable)

Usage

```
trim(x, cutoff = 6)
```

Arguments

`x`	data to be trimmed
`cutoff`	default is 6, then x is clipped +-6 times sd

`trimmed.mean` *Mean Of Data After Trimmed (questioning)*

Description

Mean Of Data After Trimmed
(questioning)

Usage

```
trimmed.mean(x, cutoff = 4)
```

Arguments

`x, cutoff`	passed to trim

`trimmed.mse` *Sd Of Data After Trimmed (questioning)*

Description

Sd Of Data After Trimmed
(questioning)

Usage

```
trimmed.mse(x, cutoff = 4)
```

Arguments

`x, cutoff`	passed to trim

`which.equal` *like which.min, but for equality useful when an expression for x or y is long*

Description

like which.min, but for equality useful when an expression for x or y is long

Usage

```
which.equal(x, y)
```

Arguments

`x, y`	vectors to compare

`%&%` *Pipe Function To Paste Two Characters*

Description

Pipe Function To Paste Two Characters

Usage

```
x %&% y
```

Arguments

`x`	character
`y`	character

Value

paste0(x,y)

Index