Package: metaplot (via r-universe)

September 16, 2024

Type Package

Title Data-Driven Plot Design

Version 0.8.4

Author Tim Bergsma

Maintainer Tim Bergsma <bergsmat@gmail.com>

Description Designs plots in terms of core structure. See 'example(metaplot)'. Primary arguments are (unquoted) column names; order and type (numeric or not) dictate the resulting plot. Specify any y variables, x variable, any groups variable, and any conditioning variables to metaplot() to generate density plots, boxplots, mosaic plots, scatterplots, scatterplot matrices, or conditioned plots. Use multiplot() to arrange plots in grids. Wherever present, scalar column attributes 'label' and 'guide' are honored, producing fully annotated plots with minimal effort. Attribute 'guide' is typically units, but may be encoded() to provide interpretations of categorical values (see '?encode'). Utility unpack() transforms scalar column attributes to row values and pack() does the reverse, supporting tool-neutral storage of metadata along with primary data. The package supports customizable aesthetics such as such as reference lines, unity lines, smooths, log transformation, and linear fits. The user may choose between trellis and ggplot output. Compact syntax and integrated metadata promote workflow scalability.

Imports encode (>= 0.3.6), lattice, magrittr, dplyr (>= 0.7.1), tidyr, rlang, grid, gridExtra, gtable, ggplot2, scales

Suggests csv, nlme

Depends R (>= 2.10)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Repository https://bergsmat.r-universe.dev

RemoteUrl https://github.com/bergsmat/metaplot

RemoteRef HEAD

RemoteSha e1ad2327207ef16d516c1805bb3494d8c7225e2f

Contents

boxplot.data.frame *Boxplot Method for Data Frame*

Description

Boxplot for data.frame. Parses arguments and generates the call: fun(x, yvar, xvar, facets, ...).

Usage

```
## S3 method for class 'data.frame'
boxplot(
  x,
  ...,
  fun = metOption("box", "boxplot_data_frame"),
  verbose = metOption("verbose_boxplot", FALSE)
)
```

Arguments

x	data.frame
...	passed to fun
fun	function that does the actual plotting
verbose	generate messages describing process

See Also

Other mixedvariate plots: boxplot_data_frame(), boxplot_panel()

Other boxplot: boxplot_data_frame()

Other methods: axislabel.data.frame(), categorical.data.frame(), corsplom.data.frame(), densplot.data.frame(), metaplot.data.frame(), pack.data.frame(), plot.metaplot_gtable(), print.metaplot_gtable(), scatter.data.frame(), unpack.data.frame()

Examples

```
library(dplyr)
library(magrittr)
Theoph %<>% mutate(site = ifelse(as.numeric(Subject) > 6, 'Site A','Site B'))
boxplot(Theoph,'Subject','conc')
boxplot(Theoph,Subject,conc)
boxplot(Theoph,Subject,conc, gg = T)
boxplot(Theoph,conc,Subject)
boxplot(Theoph,conc,Subject, gg = T)
boxplot(Theoph,conc,Subject,site)
boxplot(Theoph,conc,Subject,site, gg = T)
boxplot(Theoph,conc,Subject,site, gg = T, scales = 'free_x')
attr(Theoph,'title') <- 'Theophylline'
boxplot(Theoph, Subject, conc, main = function(x,...)attr(x,'title'))
```

```
boxplot(Theoph, Subject, conc, main = function(x,...)attr(x,'title'), gg = T)
boxplot(Theoph, Subject, conc, sub= function(x,...)attr(x,'title'))
boxplot(Theoph, Subject, conc, sub= function(x,...)attr(x,'title'), gg = T)
boxplot(Theoph %>% filter(conc > 0),Subject,conc, log = T)
boxplot(Theoph %>% filter(conc > 0),Subject,conc, log = T, gg = T)
```

`boxplot_data_frame` *Boxplot Function for Data Frame*

Description

Boxplot for data.frame. Creates a boxplot using `boxplot_panel` by default.

Usage

```
boxplot_data_frame(
  x,
  yvar,
  xvar,
  facets = NULL,
  log = metOption("log_boxplot", FALSE),
  crit = metOption("crit_boxplot", 1.3),
  horizontal = metOption("horizontal_boxplot", NULL),
  scales = metOption("scales_boxplot", NULL),
  panel = metOption("panel_boxplot", "boxplot_panel"),
  ref = metOption("ref_boxplot", "metaplot_ref"),
  ref.col = metOption("ref.col_boxplot", "grey"),
  ref.lty = metOption("ref.lty_boxplot", "solid"),
  ref.lwd = metOption("ref.lwd_boxplot", 1),
  ref.alpha = metOption("ref.alpha_boxplot", 1),
  nobs = metOption("nobs_boxplot", FALSE),
  na.rm = metOption("na.rm_boxplot", TRUE),
  xlab = NULL,
  ylab = NULL,
  numlab = metOption("numlab_boxplot", "axislabel"),
  catlab = metOption("catlab_boxplot", "axislabel"),
  aspect = metOption("aspect_boxplot", 1),
  as.table = metOption("as.table_boxplot", TRUE),
  main = metOption("main_boxplot", NULL),
  sub = metOption("sub_boxplot", NULL),
  settings = metOption("settings_boxplot", NULL),
  padding = metOption("padding_boxplot", 1),
  reverse = metOption("reverse_boxplot", TRUE),
  pch = metOption("pch_boxplot", "|"),
  notch = metOption("notch_boxplot", FALSE),
  gg = metOption("gg_boxplot", FALSE),
  verbose = metOption("verbose_boxplot", FALSE),
  ...
)
```

Arguments

x	data.frame
yvar	y variable
xvar	x variable
facets	optional conditioning variables
log	whether to log transform numeric variable (auto-selected if NA)
crit	if log is NA, log-transform if mean/median ratio for non-missing values is greater than this value
horizontal	whether box/whisker axis should be horizontal (numeric x, categorical y); defaults TRUE if var[[2]] is numeric
scales	passed to xyplot (should be function(x = x, horizontal, log,...)) or facet_grid or facet_wrap
panel	panel function
ref	optional reference line(s) on numeric axis; can be function(x = x, var = con, ...) or NULL to suppress
ref.col	color for reference line(s); can be length one integer to auto-select that many colors
ref.lty	line type for reference line(s)
ref.lwd	line size for reference line(s)
ref.alpha	transparency for reference line(s)
nobs	whether to include the number of observations under the category label
na.rm	whether to remove data points with one or more missing coordinates
xlab	x axis label
ylab	y axis label
numlab	numeric axis label; can be function(x = x, var = numvar, log = ylog, ...)
catlab	categorical axis label; can be function(x = x, var = catvar, ...)
aspect	passed to bwplot or ggplot; use 'fill', NA, or NULL to calculate automatically
as.table	passed to xyplot
main	character, or a function of x, yvar, xvar, facets, and log
sub	character, or a function of x, yvar, xvar, facets, and log
settings	default parameter settings: a list from which matching elements are passed to lattice (as par.settings) or to ggplot theme() and facet_wrap() or facet_grid(). ncol and nrow are used as layout indices for lattice (for homology with facet_wrap).
padding	numeric (will be recycled to length 4) giving plot margins in default units: top, right, bottom, left (in multiples of 5.5 points for ggplot)
reverse	if y is categorical, present levels in reverse order (first at top)
pch	special character for box median: passed to panel.bwplot
notch	whether to draw notched boxes: passed to panel.bwplot
gg	logical: whether to generate ggplot instead of trellis
verbose	generate messages describing process
...	passed arguments

See Also

Other mixedvariate plots: boxplot.data.frame(), boxplot_panel()

Other boxplot: boxplot.data.frame()

Other metaplot: categorical_data_frame(), corsplom_data_frame(), densplot_data_frame(), metaplot_key(), metaplot(), scatter_data_frame(), test_metaplot()

Examples

```
library(magrittr)
library(dplyr)
boxplot_data_frame(Theoph,'Subject','conc')
boxplot_data_frame(Theoph %>% filter(conc > 0),
'conc','Subject', log = TRUE, ref = c(2,5),horizontal = FALSE)
```

---

categorical *Categorical Plot*

Description

Categorical Plot. Generic, with method for 'data.frame'.

Usage

```
categorical(x, ...)
```

Arguments

x	object of dispatch
...	passed arugments

See Also

Other generic functions: axislabel(), corsplom(), densplot(), metaplot(), pack(), scatter(), test_metaplot(), unpack()

Other categorical: categorical.data.frame(), categorical_data_frame(), categorical_panel(), panel_tile()

categorical.data.frame

Categorical Method for Data Frame

Description

Categorical method for 'data.frame'.

Usage

```
## S3 method for class 'data.frame'
categorical(
  x,
  ...,
  fun = metOption("categorical", "categorical_data_frame"),
  verbose = metOption("verbose_categorical_data_frame", FALSE)
)
```

Arguments

x	data.frame
...	other arguments
fun	function to draw the plot
verbose	generate messages describing process

Value

character

See Also

Other categorical: categorical_data_frame(), categorical_panel(), categorical(), panel_tile()

Other methods: axislabel.data.frame(), boxplot.data.frame(), corsplom.data.frame(), densplot.data.frame(), metaplot.data.frame(), pack.data.frame(), plot.metaplot_gtable(), print.metaplot_gtable(), scatter.data.frame(), unpack.data.frame()

categorical_data_frame *Categorical Function for Data Frame*

Description

Categorical function for class 'data.frame'. Implements a simple mosaic plot.

Usage

```
categorical_data_frame(
  x,
  yvar = NULL,
  xvar,
  groups = NULL,
  facets = NULL,
  ylab = metOption("xlab_categorical", "axislabel"),
  xlab = metOption("ylab_categorical", "axislabel"),
  na.rm = metOption("na.rm_categorical", TRUE),
  aspect = metOption("aspect_categorical", 1),
  space = metOption("space_categorical", "right"),
  key = metOption("key_categorical", "metaplot_key"),
  as.table = metOption("as.table_categorical", TRUE),
 prepanel = metOption("prepanel_categorical", function(...) list(xlim = 0:1, ylim =
    0:1)),
  scales = metOption("scales_categorical", NULL),
  panel = metOption("panel_categorical", "categorical_panel"),
  colors = metOption("colors_categorical", NULL),
  fill = metOption("fill_categorical", 0.5),
  lines = metOption("lines_categorical", TRUE),
  main = metOption("main_categorical", NULL),
  sub = metOption("sub_categorical", NULL),
  tex = metOption("tex_categorical", 0.9),
  rot = metOption("rot_categorical", c(90, 0)),
  subscripts = metOption("subscripts_categorical", TRUE),
  settings = metOption("settings_categorical", NULL),
  padding = metOption("padding_categorical", 1),
  loc = metOption("loc_categorical", 5),
  msg = metOption("msg_categorical", "tilestats"),
  cex = metOption("cex_categorical", 1),
  gg = metOption("gg_categorical", FALSE),
  verbose = metOption("verbose_categorical", FALSE),
  ...
)
```

Arguments

x	data.frame
yvar	character: y variable (optional)
xvar	character: x variable
groups	optional grouping variable (can be missing)
facets	optional conditioning variables
ylab	y axis label; can be function(x = x, var = yvar, ..)
xlab	x axis label; can be function(x = x, var = xvar, ..)
na.rm	whether to remove data points with one or more missing coordinates
aspect	passed to bwplot or ggplot; use 'fill', NA, or NULL to calculate automatically
space	location of key (right, left, top, bottom)
key	list: passed to xyplot as auto.key or to theme; can be a function groups name, groups levels, fill, lines, space, gg, type ('categorical'), and .... See metaplot_key.
as.table	passed to xyplot
prepanel	passed to xyplot (guessed if NULL)
scales	passed to xyplot or facet_grid or facet_wrap (guessed if NULL)
panel	name or definition of panel function for lattice
colors	replacements for default colors in group order; can be length one integer to auto-select that many colors
fill	whether to fill rectangles for each group: logical, or alpha values between 0 and 1
lines	whether to plot borders for each group: logical, or alpha values between 0 and 1
main	character, or a function of x, yvar, xvar, groups, facets
sub	character, or a function of x, yvar, xvar, groups, facets
tex	tile expansion: scale factor for reducing each tile size relative to full size (<= 1)
rot	rotation for axis labels; can be length 2 for y and x axes, respectively
subscripts	passed to xyplot
settings	default parameter settings: a list from which matching elements are passed to lattice (as par.settings) or to ggplot theme() and facet_wrap() or facet_grid(). ncol and nrow are used as layout indices for lattice (for homology with facet_wrap).
padding	numeric (will be recycled to length 4) giving plot margins in default units: top, right, bottom, left (in multiples of 5.5 points for ggplot)
loc	where to print statistics in a tile
msg	a function of x and y to print text in a tile
cex	expansion for msg text
gg	logical: whether to generate ggplot instead of trellis
verbose	generate messages describing process
...	passed to region

See Also

categorical_panel

Other categorical: categorical.data.frame(), categorical_panel(), categorical(), panel_tile()

Other metaplot: boxplot_data_frame(), corsplom_data_frame(), densplot_data_frame(), metaplot_key(), metaplot(), scatter_data_frame(), test_metaplot()

Examples

```
library(magrittr)
library(dplyr)
library(csv)
x <- as.csv(system.file(package = 'metaplot', 'extdata/theoph.csv'))
x %<>% pack
x %>% metaplot(site)
x %>% metaplot(site, gg = T)
x %>% metaplot(arm, site)
x %>% metaplot(arm, site, gg = T)
x %>% metaplot(arm, site, cohort)
x %>% metaplot(arm, site, cohort, gg = T)
x %>% metaplot(arm, site, cohort, space = 'top')
x %>% metaplot(arm, site, , cohort)
x %>% metaplot(arm, site, , cohort, gg = T)

x %>% metaplot(arm, site, , cohort, rot = c(0,90))
x %>% metaplot(arm, site, , cohort, rot = c(0,90), gg = T)
x %>% metaplot(arm, site, , cohort, rot = c(45, 45))
x %>% metaplot(subject,cohort,arm, site, lines = F, rot = c(45,45))
x %>% metaplot(subject,cohort,arm, site, lines = F, rot = c(45,45), gg=T)
# panel-specific axis not well-supported for gg version
x %>% metaplot(subject,cohort,,arm, site)
x %>% metaplot(subject,cohort,,arm, site, gg=T)
```

---

categorical_panel *Panel Function for Metaplot Categorical Plot*

Description

Default panel function for categorical_data_frame. Implements a simple mosaic plot. Global options are supported but typically are supplied by the calling function and may therefore be unreachable.

Usage

```
categorical_panel(
  x,
  y,
  groups,
```

```
  bivariate = TRUE,
  loc = metOption("loc_categorical_panel", 5),
  msg = metOption("msg_categorical_panel", "tilestats"),
  tex = metOption("tex_categorical_panel", 0.9),
  cex = metOption("cex_categorical_panel", 1),
  rot = metOption("rot_categorical_panel", c(90, 0)),
  subscripts,
  verbose = metOption("verbose_categorical_panel", FALSE),
  ...
)
```

Arguments

x	x values
y	y values
groups	optional grouping item
bivariate	whether to create y axis
loc	where to print statistics in a tile
msg	a function of x and y to print text in a tile
tex	tile expansion: scale factor for reducing each tile size relative to full size (<= 1)
cex	expansion for msg text
rot	rotation for axis labels; can be length 2 for y and x axes, respectively
subscripts	subscripts of the original data for this panel
verbose	generate messages describing process
...	passed to panel.superpose

See Also

tilestats

categorical.data.frame

Other panel functions: boxplot_panel(), corsplom_gg_correlation(), corsplom_gg_diagonal(), corsplom_gg_scatter(), corsplom_panel_correlation(), corsplom_panel_diagonal(), corsplom_panel_scatter(), dens_panel(), diag_label(), diag_pin(), iso_prepanel(), metaplot_key(), metaplot_ref(), panel.meta_densityplot(), panel_tile(), scatter_panel_ref(), scatter_panel()

Other categorical: categorical.data.frame(), categorical_data_frame(), categorical(), panel_tile()

cax *Calculate Categorical Axis Labels and Positions*

Description

Calculates axis labels and positions for categorical values.

Usage

```
cax(x, ...)
```

Arguments

x	x values
...	other arguments

Value

data.frame

See Also

categorical_panel

Other categorical family: tilestats(), tiles()

corsplom *Correlated Splom*

Description

Scatterplot matrix with correlations.

Usage

```
corsplom(x, ...)
```

Arguments

x	object
...	passed arguments

See Also

Other generic functions: axislabel(), categorical(), densplot(), metaplot(), pack(), scatter(), test_metaplot(), unpack()

Other corsplom: corsplom.data.frame(), corsplom_data_frame(), corsplom_gg_correlation(), corsplom_gg_diagonal(), corsplom_gg_scatter(), corsplom_panel_correlation(), corsplom_panel_scatter(), plot.metaplot_gtable(), print.metaplot_gtable()

---

corsplom.data.frame *Correlated Scatterplot Matrix Method for Data Frame*

Description

Creates a scatterplot matrix. Parses arguments and generates the call: fun(x, xvar, ...).

Usage

```
## S3 method for class 'data.frame'
corsplom(
  x,
  ...,
  fun = metOption("corsplom", "corsplom_data_frame"),
  verbose = metOption("verbose_corsplom_data_frame", FALSE)
)
```

Arguments

x	data.frame
...	passed to fun
fun	function to do the actual plotting
verbose	generate messages describing process

See Also

Other multivariate plots: corsplom_data_frame(), metaplot.data.frame()

Other corsplom: corsplom_data_frame(), corsplom_gg_correlation(), corsplom_gg_diagonal(), corsplom_gg_scatter(), corsplom_panel_correlation(), corsplom_panel_scatter(), corsplom(), plot.metaplot_gtable(), print.metaplot_gtable()

Other methods: axislabel.data.frame(), boxplot.data.frame(), categorical.data.frame(), densplot.data.frame(), metaplot.data.frame(), pack.data.frame(), plot.metaplot_gtable(), print.metaplot_gtable(), scatter.data.frame(), unpack.data.frame()

corsplom_data_frame *Correlated Scatterplot Matrix Function for Data Frame*

Description

Creates a scatterplot matrix with correlations in lower panel, by default.

Usage

```
corsplom_data_frame(
  x,
  xvar = names(x),
 upper.panel = metOption("upper.panel_corsplom", if (gg) "corsplom_gg_scatter" else
    "corsplom_panel_scatter"),
 lower.panel = metOption("lower.panel_corsplom", if (gg) "corsplom_gg_correlation" else
    "corsplom_panel_correlation"),
 diag.panel = metOption("diag.panel_corsplom", if (gg) "corsplom_gg_diagonal" else
    "corsplom_panel_diagonal"),
  pscales = metOption("pscales_corsplom", 0),
  xlab = metOption("xlab_corsplom", NULL),
  varname.cex = metOption("varname.cex_corsplom", 1),
  main = metOption("main_corsplom", NULL),
  sub = metOption("sub_corsplom", NULL),
  col = metOption("col_corsplom", "blue"),
  smooth.col = metOption("smooth.col_corsplom", NULL),
  smooth.lty = metOption("smooth.lty_corsplom", "solid"),
  smooth.lwd = metOption("smooth.lwd_corsplom", 1),
  smooth.alpha = metOption("smooth.alpha_corsplom", 1),
  density = metOption("density_corsplom", TRUE),
  diag.label = metOption("diag.label_corsplom", "diag_label"),
  pin = metOption("pin_corsplom", "diag_pin"),
  pin.col = metOption("pin.col_corsplom", "darkgrey"),
  pin.alpha = metOption("pin.alpha_corsplom", 1),
  dens.col = metOption("dens.col_corsplom", "grey"),
  dens.scale = metOption("dens.scale_corsplom", 0.2),
  dens.alpha = metOption("dens.alpha_corsplom", 0.5),
  settings = metOption("settings.corsplom", NULL),
  padding = metOption("padding_corsplom", 1),
  as.table = metOption("as.table_corsplom", FALSE),
  dens.up = metOption("dens.up_corsplom", TRUE),
  gg = metOption("gg_corsplom", FALSE),
  verbose = metOption("verbose_corsplom", FALSE),
  ...
)
```

Arguments

x	data.frame

xvar	variables to plot
upper.panel	passed to splom or ggplot
lower.panel	passed to splom or ggplot
diag.panel	passed to splom or ggplot
pscales	passed to splom
xlab	can be function(x = x, var = xvar, ...)
varname.cex	text size multiplier
main	character, or a function of x, xvar
sub	character, or a function of x, xvar
col	point color
smooth.col	smooth color, defaults to col
smooth.lty	smooth line type
smooth.lwd	smooth line size
smooth.alpha	smooth alpha
density	whether to plot density polygons
diag.label	label for the diagonal: a function of x, varname, .data
pin	location for a pin (reference line) in the density region; can be function(x, varname, .data) or NULL to suppress
pin.col	color of pin, if any
pin.alpha	alpha transparency of pin
dens.col	color for density region
dens.scale	inflation factor for height of density smooth
dens.alpha	alpha transparency for density region
settings	default parameter settings: a list from which matching elements are passed to lattice (as par.settings) or to ggplot theme()
padding	numeric (will be recycled to length 4) giving plot margins in default units: top, right, bottom, left (in multiples of 5.5 points for ggplot)
as.table	diagonal arranged top-left to bottom-right
dens.up	whether density plots in diagonal should face the upper triangle vs. lower
gg	logical: whether to generate ggplot instead of trellis
verbose	generate messages describing process
...	extra arguments passed to splom and ggplot

Value

trellis or grob

See Also

Other multivariate plots: corsplom.data.frame(), metaplot.data.frame()

Other corsplom: corsplom.data.frame(), corsplom_gg_correlation(), corsplom_gg_diagonal(), corsplom_gg_scatter(), corsplom_panel_correlation(), corsplom_panel_scatter(), corsplom(), plot.metaplot_gtable(), print.metaplot_gtable()

Other metaplot: boxplot_data_frame(), categorical_data_frame(), densplot_data_frame(), metaplot_key(), metaplot(), scatter_data_frame(), test_metaplot()

Examples

```
library(magrittr)
library(dplyr)
library(csv)
x <- as.csv(system.file(package = 'metaplot', 'extdata/theoph.csv'))
x %<>% pack
# setOption(gg = TRUE)
x %>% metaplot(lKe, lKa, lCl)
x %>% metaplot(
  lKe, lKa, lCl,
  col = 'black',smooth.col = 'red', pin.col = 'red',
  dens.col='blue',dens.alpha = 0.1
)
```

---

densplot *Density Plot*

Description

Creates a density plot.

Usage

```
densplot(x, ...)
```

Arguments

x	object
...	passed arguments

See Also

Other generic functions: axislabel(), categorical(), corsplom(), metaplot(), pack(), scatter(), test_metaplot(), unpack()

Other univariate plots: dens_panel(), densplot.data.frame(), densplot_data_frame(), metaplot.data.frame(), panel.meta_densityplot()

Other densplot: densplot.data.frame(), densplot_data_frame()

densplot.data.frame *Densplot Method for Data Frame*

Description

Plot density for object of class 'data.frame'. Parses arguments and generates the call: fun(x, xvar, groups, facets,...).

Usage

```
## S3 method for class 'data.frame'
densplot(
  x,
  ...,
  fun = metOption("densplot", "densplot_data_frame"),
  verbose = metOption("verbose_densplot_data_frame", FALSE)
)
```

Arguments

x	data.frame
...	passed to fun
fun	plotting function
verbose	generate messages describing process

See Also

Other univariate plots: dens_panel(), densplot_data_frame(), densplot(), metaplot.data.frame(), panel.meta_densityplot()

Other densplot: densplot_data_frame(), densplot()

Other methods: axislabel.data.frame(), boxplot.data.frame(), categorical.data.frame(), corsplom.data.frame(), metaplot.data.frame(), pack.data.frame(), plot.metaplot_gtable(), print.metaplot_gtable(), scatter.data.frame(), unpack.data.frame()

Examples

```
densplot(Theoph, conc, grid = TRUE )
densplot(Theoph, conc, grid = TRUE, gg = TRUE )
densplot(Theoph, conc, Subject )
densplot(Theoph, conc, , Subject )
densplot(Theoph, conc, , Subject, gg = TRUE, scales = 'free_y' )
attr(Theoph,'title') <- 'Theophylline'
densplot(Theoph, conc, main= function(x,...)attr(x,'title'))
densplot(Theoph, conc, sub= function(x,...)attr(x,'title'))
```

densplot_data_frame *Density Function for Data Frame*

Description

Plot density for object of class 'data.frame' using dens_panel by default.

Usage

```
densplot_data_frame(
  x,
  xvar,
  groups = NULL,
  facets = NULL,
  xlab = metOption("xlab_dens", "axislabel"),
  ref = metOption("ref_x_dens", "metaplot_ref"),
  ref.col = metOption("ref_col_dens", "grey"),
  ref.lty = metOption("ref_lty_dens", "solid"),
  ref.lwd = metOption("ref_lwd_dens", 1),
  ref.alpha = metOption("ref_alpha_dens", 1),
  log = metOption("log_dens", FALSE),
  crit = metOption("crit_dens", 1.3),
  aspect = metOption("aspect_dens", 1),
  scales = metOption("scales_dens", NULL),
  panel = metOption("panel_dens", "dens_panel"),
  points = metOption("points_dens", TRUE),
  colors = metOption("colors_dens", NULL),
  symbols = metOption("symbols_dens", NULL),
  sizes = metOption("sizes_dens", 1),
  lines = metOption("lines_dens", TRUE),
  types = metOption("types_dens", "solid"),
  widths = metOption("widths_dens", 1),
  fill = metOption("fill_dens", FALSE),
  space = metOption("space_dens", "right"),
  key = metOption("key_dens", "metaplot_key"),
  as.table = metOption("as.table_dens", TRUE),
  main = metOption("main_dens", NULL),
  sub = metOption("sub_dens", NULL),
  settings = metOption("settings_dens", NULL),
  padding = metOption("padding_dens", 1),
  gg = metOption("gg_dens", FALSE),
  verbose = metOption("verbose_dens", FALSE),
  ...
)
```

Arguments

x	data.frame

xvar	variable to plot
groups	optional grouping variable
facets	optional conditioning variables
xlab	x axis label; can be function(x = x, var = xvar, log = log, ...)
ref	reference line; can be function(x = x, var = xvar, ...) or NULL to suppress
ref.col	color for reference line(s); can be length one integer to auto-select that many colors
ref.lty	type for reference line(s)
ref.lwd	size for reference line(s)
ref.alpha	transparency for reference line(s)
log	whether to log-transform x axis (auto-selected if NA)
crit	if log is NA, log-transform if mean/median ratio for non-missing x is greater than this value (and no negative values)
aspect	passed to bwplot or ggplot; use 'fill', NA, or NULL to calculate automatically
scales	passed to xyplot or facet_grid or facet_wrap (guessed if NULL)
panel	passed to densityplot
points	whether to plot points: logical or alpha, same length as groups
colors	replacements for default colors in group order; can be length one integer to auto-select that many colors
symbols	replacements for default symbols in group order
sizes	replacements for default symbol sizes in group order
lines	whether to plot lines: logical or alpha, same length as groups
types	replacements for default line types in group order
widths	replacements for default line widths in group order
fill	whether to fill curves: logical or alpha, same length as groups (symbol fill color is same as point color)
space	location of key (right, left, top, bottom)
key	list: passed to xyplot as auto.key or to theme; can be a function groups name, groups levels, points, lines, space, gg, and .... See metaplot_key.
as.table	passed to xyplot
main	character, or a function of x, xvar, groups, facets, and log
sub	character, or a function of x, xvar, groups, facets, and log
settings	default parameter settings: a list from which matching elements are passed to lattice (as par.settings) or to ggplot theme() and facet_wrap() or facet_grid(). ncol and nrow are used as layout indices for lattice (for homology with facet_wrap).
padding	numeric (will be recycled to length 4) giving plot margins in default units: top, right, bottom, left (in multiples of 5.5 points for ggplot)
gg	logical: whether to generate ggplot instead of trellis
verbose	generate messages describing process
...	passed to densityplot

See Also

Other univariate plots: dens_panel(), densplot.data.frame(), densplot(), metaplot.data.frame(), panel.meta_densityplot()

Other densplot: densplot.data.frame(), densplot()

Other metaplot: boxplot_data_frame(), categorical_data_frame(), corsplom_data_frame(), metaplot_key(), metaplot(), scatter_data_frame(), test_metaplot()

Examples

```
densplot_data_frame(Theoph, 'conc', grid = TRUE)
densplot_data_frame(Theoph, 'conc', 'Subject')
densplot_data_frame(Theoph, 'conc', 'Subject',
space = 'top', columns = 4, legend.direction = 'horizontal')
densplot_data_frame(Theoph, 'conc', 'Subject',
space = 'top', columns = 4, legend.direction = 'horizontal', gg = TRUE)
densplot_data_frame(Theoph, 'conc', , 'Subject')
```

---

diag_label *Format a Diagonal Label*

Description

Formats a diagonal label. Can return a simple column name, a column label (if attribute defined), a fractured column label (split on spaces), or a processed symbol (over-rides label).

Usage

```
diag_label(
  varname,
  .data,
  diag_label_simple = metOption("diag_label_simple", FALSE),
  diag_label_split = metOption("diag_label_split", TRUE),
  diag_symbol_format = metOption("diag_symbol_format", "wikisym2plotmath"),
  verbose = metOption("verbose_diag_label", FALSE),
  ...
)
```

Arguments

varname	character
.data	data.frame
diag_label_simple	logical: just return varname?
diag_label_split	whether to substitute line breaks for spaces

diag_symbol_format	function to process symbol attribute, if present
verbose	generate messages describing process
...	ignored

Value

character

See Also

Other panel functions: boxplot_panel(), categorical_panel(), corsplom_gg_correlation(), corsplom_gg_diagonal(), corsplom_gg_scatter(), corsplom_panel_correlation(), corsplom_panel_diagonal(), corsplom_panel_scatter(), dens_panel(), diag_pin(), iso_prepanel(), metaplot_key(), metaplot_ref(), panel.meta_densityplot(), panel_tile(), scatter_panel_ref(), scatter_panel()

Other formatters: wikisym2plotmath_(), wikisym2plotmath()

---

diag_pin *Calculate Pin Placement*

Description

Calculates pin placement in the density region, inside margin of diagonal panels.

Usage

```
diag_pin(x, varname, .data, ...)
```

Arguments

x	vector of data
varname	name of vector in .data
.data	original dataset, possibly with column attributes such as 'reference'
...	passed arguments

Value

numeric

See Also

Other panel functions: boxplot_panel(), categorical_panel(), corsplom_gg_correlation(), corsplom_gg_diagonal(), corsplom_gg_scatter(), corsplom_panel_correlation(), corsplom_panel_diagonal(), corsplom_panel_scatter(), dens_panel(), diag_label(), iso_prepanel(), metaplot_key(), metaplot_ref(), panel.meta_densityplot(), panel_tile(), scatter_panel_ref(), scatter_panel()

Other reference lines: metaplot_ref(), scatter_panel_ref()

`metaplot` *Metaplot*

Description

Metaplot creates univariate, bivariate, or multivariate plots depending on the number and types of variables represented by the anonymous arguments. Types are either numeric (NUM, e.g. real, integer) or categorical (CAT, e.g. factor, character). A variable stored as numeric that nonetheless has an encoded `guide` attribute will be treated as categorical. Mnemonic: `x %>% metaplot(yvars, xvar, groupvar, facets)` where arguments are unquoted column names, and only xvar is required. Column attributes `label`, `guide`, `reference`, and `symbol` modify the behavior of the default handlers.

Usage

```
metaplot(x, ...)
```

Arguments

`x`	object
`...`	passed arguments

Details

Design your plot by specifying y variables (optional), the x variable, the groups variable (optional) and the conditioning variables (i.e., facets, optional).

The single groups variable, if any, is the first categorical in the third position or later. An earlier categorical gives a "mixed" bivariate plot or mosaic plot, depending on the type of the remaining variable.

The x variable is the last variable before groups, if present.

The y variables are those before x. If none, the result is univariate. If one, the result is typically a boxplot or scatterplot, depending on x. Several numeric y followed by a numeric x are treated as multivariate (scatterplot matrix). But if all y have the same `guide` attribute and it is different from that for x, the result is bivariate (i.e, an `overlay` scatterplot).

A single categorical variable results in a simple mosaic plot (see `link[graphics]{mosaicplot}` and **vcd** for more sophisticated treatment). Mosaic plots support only a single y variable; thus, whenever the first two variables are categorical, a two-way mosaic plot results, with remaining variables understood as groups and facets.

Wherever a groups argument is meaningful, it may be missing. This allows specification of facets in the absence of groups, e.g., `(metaplot(y, x, , facet1, facet2))`. For multiple y (overlay), the sources of y are the implied groups: any trailing categorical arguments are treated as facets.

Template designs follow; substitute behaviors by setting global options (see argument list).

NUM: univariate (densityplot)

CAT: categorical (one-way mosaic plot)

CAT, CAT: categorical (two-way mosaic plot)

CAT, CAT, CAT: grouped mosaic

CAT, CAT, CAT, CAT: grouped mosaic with one facet

CAT, CAT, CAT,, CAT: non-grouped mosaic with one facet

NUM, CAT: mixedvariate (vertical boxplot)

CAT, NUM: mixedvariate (horizontal boxplot)

CAT, NUM, CAT: mixedvariate with one facet

NUM, NUM: bivariate (scatterplot)

NUM, NUM, CAT: grouped bivariate (grouped scatterplot)

NUM, NUM,, CAT: non-grouped bivariate with one facet

NUM, NUM, CAT, CAT: grouped bivariate with one facet

NUM, NUM, CAT, CAT, CAT: grouped bivariate with two facets

NUM, NUM, NUM: multivariate, or grouped bivariate for `overlay`

NUM, NUM, NUM, CAT multivariate, or faceted bivariate for `overlay`

NUM, NUM, NUM, CAT, CAT multivariate, or bivariate with two facets for `overlay`

Variable attributes may be supplied by conventional means; `pack` and `unpack` support storing and retrieving scalar column attributes. The following scalar attributes are currently supported.

label: A variable descriptor. If present, panel functions will use label to create informative axis labels. See `axislabel`.

guide: Units for a numeric variable, or an encoding (scalar string giving codes and possibly decodes) for a categorical item. If present, units will be used to inform the corresponding axis label (`axislabel`). If present, codes will be used to impose sort order on categorical variables. If present, decodes will be used as substitutes for stored values when presenting categorical labels, legends, and facet names. For more on encodings, see `encode`.

reference: Some variables have values to which they can be compared. For example, residual error is often expected to be centered at zero. Default panel functions plot corresponding reference lines if this attribute is present. See for example `dens_panel`.

symbol: Variable names are useful for programming, and variable labels are useful as axis labels. A symbol can be more formal than a variable name and more compact than a label. For example, `diag_label` will use variable names as labels for the diagonal panels of a scatterplot matrix; but it will prefer labels, if available; and will prefer symbols most of all. Markup rules for symbols are given in `wikisym2plotmath_`.

See Also

Other generic functions: `axislabel()`, `categorical()`, `corsplom()`, `densplot()`, `pack()`, `scatter()`, `test_metaplot()`, `unpack()`

Other metaplot: `boxplot_data_frame()`, `categorical_data_frame()`, `corsplom_data_frame()`, `densplot_data_frame()`, `metaplot_key()`, `scatter_data_frame()`, `test_metaplot()`

Examples

```
library(magrittr)
library(dplyr)
library(csv)
x <- as.csv(system.file(package = 'metaplot', 'extdata/theoph.csv'))
x %<>% pack
# setOption(gg = TRUE)
# setOption(verbose = TRUE)  # all messages; equiv. to metaplot(verbose = T,...)
# setOption(verbose_densplot = TRUE) # densplot messages
# sample plots
x %>% metaplot(sres)
x %>% metaplot(site)
x %>% metaplot(conc, arm)
x %>% densplot(conc, arm)
x %>% metaplot(arm, conc)
x %>% metaplot(conc, arm, site)
x %>% metaplot(conc, site, arm)
x %>% metaplot(conc, time)
x %>% metaplot(arm, site)
x %>% metaplot(arm, site, cohort)
x %>% metaplot(arm, site, cohort, space = 'top')
x %>% metaplot(arm, site, , cohort)
x %>% metaplot(conc, time, subject)
x %>% metaplot(conc, time, , subject)
x %>% metaplot(conc, time, subject, site)
x %>% metaplot(conc, time, subject, site, arm)
x %>% metaplot(lKe, lKa, lCl)


x %>% metaplot(
  lKe, lKa, lCl,
  col = 'black',smooth.col = 'red', pin.col = 'red',
  dens.col='blue',dens.alpha = 0.1
)
x %>% metaplot(conc, pred, ipred, time, space = 'top')
x %>% metaplot(conc, pred, ipred, time, subject, space = 'top')
x %>% metaplot(conc, pred, ipred, time, subject,
  colors = c('black','blue','orange'),
  points = c(0.9,0, 0.4),
  lines = c(F,T,T),
  types = c('blank','dashed','solid'),
  space = 'top'
)

x %>% metaplot(conc, ipred, time, site, arm, space = 'top')
x %>% metaplot(res, conc, yref = 0, ysmooth = T, conf = T, grid = T, loc = 1)
x %>% metaplot(res, conc, arm, ysmooth = T, conf = T )
x %>% metaplot(res, conc, arm, ysmooth = T, conf = T, global = T, ref.col = 'red')
x %>% metaplot(subject,conc)

# manage metadata
attr(x$arm, 'guide') # //1/Arm A//2/Arm B//
```

```
x %>% metaplot(conc, arm) # default

x %>% mutate(arm = arm %>%
  structure(guide = '//2/Arm B//1/Arm A//')) %>%
  metaplot(conc, arm) # different presentation order

x %>% mutate(arm = arm %>%
  structure(guide = '//1/Both Arms//2/Both Arms//')) %>%
  metaplot(conc, arm) # collapse cases
```

`metaplot.data.frame` *Create Metaplot for Data Frame.*

Description

Creates a metaplot for class 'data.frame'. Implements a rule to decided whether to make a density plot, a boxplot, a scatter plot, or a scatterplot matrix, given the supplied column names.

Usage

```
## S3 method for class 'data.frame'
metaplot(
  x,
  ...,
  univariate = metOption("univariate", "densplot"),
  mixedvariate = metOption("mixedvariate", "boxplot"),
  bivariate = metOption("bivariate", "scatter"),
  multivariate = metOption("multivariate", "corsplom"),
  categorical = metOption("categorical", "categorical"),
  verbose = metOption("verbose", FALSE)
)
```

Arguments

`x`	object
`...`	passed arguments
`univariate`	function for univariate arguments
`mixedvariate`	function for bivariate combinations of numeric and categoral arguments
`bivariate`	function for arguments that resolve to two numerics (see rules)
`multivariate`	function for more than two numeric arguments
`categorical`	function for categorical arguments
`verbose`	generate messages describing process; passed to called functions if explicitly supplied

See Also

Other methods: axislabel.data.frame(), boxplot.data.frame(), categorical.data.frame(), corsplom.data.frame(), densplot.data.frame(), pack.data.frame(), plot.metaplot_gtable(), print.metaplot_gtable(), scatter.data.frame(), unpack.data.frame()

Other univariate plots: dens_panel(), densplot.data.frame(), densplot_data_frame(), densplot(), panel.meta_densityplot()

Other bivariate plots: iso_prepanel(), scatter.data.frame(), scatter_data_frame(), scatter()

Other multivariate plots: corsplom.data.frame(), corsplom_data_frame()

Examples

```
## Not run:
library(magrittr)
library(dplyr)
library(csv)
library(nlme)
x <- Theoph

# mixed effects model
m1 <- nlme(
  conc ~ SSfol(Dose, Time, lKe, lKa, lCl),
  data = x,
  fixed = lKe + lKa + lCl ~ 1,
  random = lKe + lKa + lCl ~ 1
)

# some numeric and categorical properties
names(x) <- tolower(names(x))
x %<>% mutate(arm = ifelse(as.numeric(as.character(subject)) %% 2 == 0, 1, 2))
x %<>% mutate(site = ifelse(as.numeric(as.character(subject)) < 6, 1, 2))
x %<>% mutate(cohort = ifelse(as.numeric(as.character(subject)) %in% c(1:2,6:8), 1,2))
x %<>% mutate(pred = predict(m1,level = 0) %>% signif(4))
x %<>% mutate(ipred = predict(m1) %>% signif(4))
x %<>% mutate(res = residuals(m1) %>% signif(4))
x %<>% mutate(sres = residuals(m1, type = 'pearson') %>% signif(4))
r <- ranef(m1) %>% signif(4)
r$subject <- rownames(r)
x %<>% left_join(r)
# metadata
attr(x$subject,'label') <- 'subject identifier'
attr(x$wt,'label') <- 'subject weight'
attr(x$dose,'label') <- 'theophylline dose'
attr(x$time,'label') <- 'time since dose administration'
attr(x$conc,'label') <- 'theophylline concentration'
attr(x$arm,'label') <- 'trial arm'
attr(x$site,'label') <- 'investigational site'
attr(x$cohort,'label') <- 'recruitment cohort'
attr(x$pred,'label') <- 'population-predicted concentration'
attr(x$ipred,'label') <- 'individual-predicted concentration'
attr(x$res,'label') <- 'residuals'
attr(x$sres,'label') <- 'standardized residuals'
```

```
attr(x$lKe,'label') <- 'natural log of elimination rate constant'
attr(x$lKa,'label') <- 'natural log of absorption rate constant'
attr(x$lCl,'label') <- 'natural log of clearance'
attr(x$subject,'guide') <- '....'
attr(x$wt,'guide') <- 'kg'
attr(x$dose,'guide') <- 'mg/kg'
attr(x$time,'guide') <- 'h'
attr(x$conc,'guide') <- 'mg/L'
attr(x$arm,'guide') <- '//1/Arm A//2/Arm B//'
attr(x$site,'guide') <- '//1/Site 1//2/Site 2//'
attr(x$cohort,'guide') <- '//1/Cohort 1//2/Cohort 2//'
attr(x$pred,'guide') <- 'mg/L'
attr(x$ipred,'guide') <- 'mg/L'

attr(x$lKe,'reference') <- 0
attr(x$lKa,'reference') <- 0
attr(x$lCl,'reference') <- 0
attr(x$res,'reference') <- 0
attr(x$sres,'reference') <- '//-1.96//1.96//'

attr(x$subject,'symbol') <- 'ID_i'
attr(x$wt,'symbol') <- 'W_i'
attr(x$dose,'symbol') <- 'A_i'
attr(x$time,'symbol') <- 't_i,j'
attr(x$conc,'symbol') <- 'C_i,j'
attr(x$arm,'symbol') <- 'Arm_i'
attr(x$site,'symbol') <- 'Site_i'
attr(x$cohort,'symbol') <- 'Cohort_i'
attr(x$pred,'symbol') <- 'C_pred_p'
attr(x$ipred,'symbol') <- 'C_pred_i'
attr(x$res,'symbol') <- '\\epsilon'
attr(x$sres,'symbol') <- '\\epsilon_st'
attr(x$lKe,'symbol') <- 'ln(K_e.)'
attr(x$lKa,'symbol') <- 'ln(K_a.)'
attr(x$lCl,'symbol') <- 'ln(Cl_c./F)'


x %>% unpack %>% as.csv('theoph.csv')

## End(Not run)
```

`metaplot_key` *Default Key*

Description

Default key function for constructing scatterplot legends.

Usage

```
metaplot_key(
  groups,
  levels,
  points = rep(FALSE, length.out = length(levels)),
  lines = rep(FALSE, length.out = length(levels)),
  fill = rep(FALSE, length.out = length(levels)),
  space = "right",
  gg = FALSE,
  type = "scatter",
  verbose = FALSE,
  ...
)
```

Arguments

groups	name of the grouping variable
levels	the (unique) levels of the grouping variable
points	logical or alpha, same length as groups
lines	logical or alpha, same length as groups
fill	logical or alpha, same length as groups
space	character: left, right, top, or bottom
gg	logical: whether to to return a list of arguments for theme instead of for auto.key as in xyplot
type	typically one of 'categorical','density', or 'scatter'
verbose	generate messages describing process
...	ignored

Value

list, or possibly logical if gg is FALSE

See Also

Other metaplot: boxplot_data_frame(), categorical_data_frame(), corsplom_data_frame(), densplot_data_frame(), metaplot(), scatter_data_frame(), test_metaplot()

Other scatter: scatter.data.frame(), scatter_data_frame(), scatter_panel(), scatter()

Other panel functions: boxplot_panel(), categorical_panel(), corsplom_gg_correlation(), corsplom_gg_diagonal(), corsplom_gg_scatter(), corsplom_panel_correlation(), corsplom_panel_diagonal(), corsplom_panel_scatter(), dens_panel(), diag_label(), diag_pin(), iso_prepanel(), metaplot_ref(), panel.meta_densityplot(), panel_tile(), scatter_panel_ref(), scatter_panel()

metaplot_ref *Calculate Reference Values*

Description

Calculates reference values for x and y axes. Coerces column attribute 'reference' to numeric: a single value or an encoding giving multiple numeric values (decodes are ignored).

Usage

```
metaplot_ref(x, var, ...)
```

Arguments

x	data.frame
var	name of vector in x
...	ignored

Value

numeric

See Also

Other panel functions: boxplot_panel(), categorical_panel(), corsplom_gg_correlation(), corsplom_gg_diagonal(), corsplom_gg_scatter(), corsplom_panel_correlation(), corsplom_panel_diagonal(), corsplom_panel_scatter(), dens_panel(), diag_label(), diag_pin(), iso_prepanel(), metaplot_key(), panel.meta_densityplot(), panel_tile(), scatter_panel_ref(), scatter_panel()

Other reference lines: diag_pin(), scatter_panel_ref()

metastats *Format GLM Statistics*

Description

Formats GLM statistics. Uses a gaussian family by default, or binomial family if all y are 0 or 1, to fit a general linear model. Formats number of observations, p-value, and Pearson correlation coefficient into a string for printing.

Usage

```
metastats(
  x,
  y,
  family = if (all(y %in% 0:1, na.rm = TRUE)) "binomial" else "gaussian",
  ...
)
```

Arguments

x	x values
y	y values
family	regression family
...	other arguments

Value

character

See Also

scatter_panel

Other regression functions: model(), region()

---

metOption *Get Metaplot Option with Partial Matching*

Description

Gets a metaplot option value from the named list getOption('metaplot'). If an exact match is not found, trailing elements of x, separated by underscore, are removed one at a time in search of a partial match. Thus 'ref.col' will match for 'ref.col_dens' and 'ref.col_scatter' if neither of these is set (allowing selective override). However, global' will never match 'global.col'.

Usage

```
metOption(x, default = NULL)
```

Arguments

x	a character string holding an option name
default	the value returned if option is not set

Details

If x is missing a list of all metaplot options is returned.

See Also

getOption setOption

Examples

```
library(magrittr)
library(dplyr)
library(csv)
x <- as.csv(system.file(package = 'metaplot', 'extdata/theoph.csv'))
x %<>% pack

multiplot(
x %>% metaplot(conc, gg = F),
x %>% metaplot(conc, time, gg = F),
x %>% metaplot(conc, arm, gg = F),
x %>% metaplot(conc, arm,  gg = T)
)

# Add a reference line at 9 mg/L
x$conc %<>% structure(reference = 9)

# Make the reference line green universally.
setOption(ref_col = 'green')

# Make the reference line orange for density plots
setOption(ref_col_dens = 'orange')

multiplot(
x %>% metaplot(conc, gg = F),
x %>% metaplot(conc, time, gg = F),
x %>% metaplot(conc, arm, gg = F),
x %>% metaplot(conc, arm,  gg = T)
)

# Restore defaults
# setOption() # clears all metaplot options
setOption(ref_col = NULL)
setOption(ref_col_dens = NULL)
```

---

multiplot *Arrange Multiple Trellis or GG Plots in a Grid*

Description

Arranges mutiple trellis plots or ggplots in a grid, automatically choosing number of rows and columns. By default, number of rows is one less than or equal to the number of columns.

Usage

```
multiplot(..., nrow = NULL, ncol = NULL)
```

Arguments

...	trellis or ggplot objects
nrow	number of rows of plots
ncol	number of columns of plots

Value

metaplot_gtable

See Also

arrangeGrob

Examples

```
library(lattice)
a <- xyplot(
conc ~ Time,
xlab=NULL,
ylab = NULL,
Theoph,
aspect = 1,
scales=list(draw=FALSE)
)
multiplot(a,a,a,a,a,a)
multiplot(a,a,a,a,a,a,a)
multiplot(a,a,a,a,a,a,a,a)
multiplot(a,a,a,a,a,a,a,a,a)
multiplot(a,a,a,a,a,a,a,a,a,a)
multiplot(a,a,a,a,a,a,a,a,a, nrow = 2)
multiplot(a,a,a,a,a,a,a,a,a, ncol = 4)
multiplot(a,a,a,a,a,a,a,a,a, ncol = 2)
multiplot(a,a,a,a,a,a,a,a,a, ncol = 4, nrow = 3)
multiplot(multiplot(a,a), a)
```

---

pack *Pack Something*

Description

Pack Something. Generic, with method for data.frame.

Usage

```
pack(x, ...)
```

Arguments

x	object
...	other arguments

See Also

Other generic functions: axislabel(), categorical(), corsplom(), densplot(), metaplot(), scatter(), test_metaplot(), unpack()

Other pack: pack.data.frame(), unpack.data.frame(), unpack()

---

pack.data.frame *Capture Scalar Column Metadata as Column Attributes*

Description

Captures scalar column metadata (row values) as column attributes. Excises rows with non-missing values of meta, converting column values to column attributes. Afterward, column classes are re-optimized using default behavior of read.table. It is an error if meta is not in names(x).

Usage

```
## S3 method for class 'data.frame'
pack(
  x,
  meta = getOption("meta", "meta"),
  as.is = TRUE,
  attributes = TRUE,
  na.rm = TRUE,
  ...
)
```

Arguments

x	data.frame
meta	column in x giving names of attributes
as.is	passed to type.convert
attributes	preserve non-standard attributes (ignores names, row.names, class)
na.rm	if TRUE, NA values (presumably structural) will not be assigned as column attributes.
...	ignored arguments

Value

data.frame

See Also

Other pack: pack(), unpack.data.frame(), unpack()

Other methods: axislabel.data.frame(), boxplot.data.frame(), categorical.data.frame(), corsplom.data.frame(), densplot.data.frame(), metaplot.data.frame(), plot.metaplot_gtable(), print.metaplot_gtable(), scatter.data.frame(), unpack.data.frame()

Examples

```
foo <- data.frame(head(Theoph))
attr(foo$Subject, 'label') <-  'subject identifier'
attr(foo$Wt, 'label') <-  'weight'
attr(foo$Dose, 'label') <-  'dose'
attr(foo$Time, 'label') <-  'time'
attr(foo$conc, 'label') <-  'concentration'
attr(foo$Subject, 'guide') <-  '////'
attr(foo$Wt, 'guide') <-  'kg'
attr(foo$Dose, 'guide') <-  'mg/kg'
attr(foo$Time, 'guide') <-  'h'
attr(foo$conc, 'guide') <-  'mg/L'
unpack(foo, pos = 1)
unpack(foo, pos = 2)
unpack(foo, pos = 3)
unpack(foo, pos = 4)
bar <- unpack(foo)
pack(bar)
attributes(pack(bar)$Subject)
```

---

panel_tile *Draw a Tile*

Description

Draws a tile in a mosaic.

Usage

```
panel_tile(
  x,
  y,
  subscripts,
  group.number,
  group.value,
  col,
  alpha,
  border,
  loc,
  msg,
  .src,
```

```
  cex,
  verbose,
  ...
)
```

Arguments

x	x values
y	y values
subscripts	subscripts
group.number	group number
group.value	group value
col	fill color
alpha	alpha transparency for fill
border	border color
loc	location for output of msg
msg	ignored
.src	data source for which subscripts give x, y, msg, and tile limits
cex	expansion for msg text; passed to msg
verbose	generate messages describing process
...	passed arguments

See Also

Other panel functions: boxplot_panel(), categorical_panel(), corsplom_gg_correlation(), corsplom_gg_diagonal(), corsplom_gg_scatter(), corsplom_panel_correlation(), corsplom_panel_diagonal(), corsplom_panel_scatter(), dens_panel(), diag_label(), diag_pin(), iso_prepanel(), metaplot_key(), metaplot_ref(), panel.meta_densityplot(), scatter_panel_ref(), scatter_panel()

Other categorical: categorical.data.frame(), categorical_data_frame(), categorical_panel(), categorical()

---

scatter *Scatterplot*

Description

Scatterplot.

Usage

```
scatter(x, ...)
```

Arguments

x	object
...	passed arguments

See Also

Other generic functions: axislabel(), categorical(), corsplom(), densplot(), metaplot(), pack(), test_metaplot(), unpack()

Other scatter: metaplot_key(), scatter.data.frame(), scatter_data_frame(), scatter_panel()

Other bivariate plots: iso_prepanel(), metaplot.data.frame(), scatter.data.frame(), scatter_data_frame()

---

scatter.data.frame *Scatterplot Method for Data Frame*

Description

Scatterplot method for class 'data.frame'. Parses arguments and generates the call: fun(x, yvar, xvar, groups, facets, ...).

Usage

```
## S3 method for class 'data.frame'
scatter(
  x,
  ...,
  fun = metOption("scatter", "scatter_data_frame"),
  verbose = metOption("verbose_scatter", FALSE)
)
```

Arguments

x	data.frame
...	passed to fun
fun	function to draw the plot
verbose	generate messages describing process

See Also

scatter_data_frame

Other bivariate plots: iso_prepanel(), metaplot.data.frame(), scatter_data_frame(), scatter()

Other scatter: metaplot_key(), scatter_data_frame(), scatter_panel(), scatter()

Other methods: axislabel.data.frame(), boxplot.data.frame(), categorical.data.frame(), corsplom.data.frame(), densplot.data.frame(), metaplot.data.frame(), pack.data.frame(), plot.metaplot_gtable(), print.metaplot_gtable(), unpack.data.frame()

Examples

```
library(magrittr)
library(dplyr)
attr(Theoph$conc,'label') <- 'theophylline concentration'
attr(Theoph$conc,'guide') <- 'mg/L'
attr(Theoph$Time,'label') <- 'time'
attr(Theoph$Time,'guide') <- 'h'
attr(Theoph$Subject,'guide') <- '////'
# setOption(gg = T)
scatter(Theoph,conc, Time)
scatter(Theoph, conc, Time, Subject) # Subject as groups
scatter(Theoph, conc, Time, , Subject) # Subject as facet
scatter(Theoph, conc, Time, , Subject, gg = TRUE, scales = 'free_y' )
scatter(Theoph %>% filter(conc > 0), conc, Time, Subject, ylog = TRUE, yref = 5)
scatter(Theoph, conc, Time, Subject, ysmooth = TRUE)
scatter(Theoph, conc, Time, conf = TRUE, loc = 3, yref = 6)
scatter(Theoph, conc, Time, conf = TRUE, loc = 3, yref = 6, global = TRUE)
## Not run:
\dontshow{
attr(Theoph,'title') <- 'Theophylline'
scatter(Theoph, conc, Time, main = function(x,...)attr(x,'title'))
scatter(Theoph, conc, Time, sub= function(x,...)attr(x,'title'))
setOption(main = function(x,...)attr(x,'title'))
scatter(Theoph, conc, Time)
}

## End(Not run)
```

---

scatter_data_frame *Scatterplot Function for Data Frame*

Description

Scatterplot function for class 'data.frame'.

Usage

```
scatter_data_frame(
  x,
  yvar,
  xvar,
  groups = NULL,
  facets = NULL,
  log = metOption("log_scatter", FALSE),
  ylog = metOption("ylog_scatter", log),
  xlog = metOption("xlog_scatter", log),
  crit = metOption("crit_scatter", 1.3),
  yref = metOption("yref_scatter", "metaplot_ref"),
  xref = metOption("xref_scatter", "metaplot_ref"),
```

```
ylab = metOption("ylab_scatter", "axislabel"),
xlab = metOption("xlab_scatter", "axislabel"),
ysmooth = metOption("ysmooth_scatter", FALSE),
xsmooth = metOption("xsmooth_scatter", FALSE),
iso = metOption("iso_scatter", FALSE),
na.rm = metOption("na.rm_scatter", TRUE),
aspect = metOption("aspect_scatter", 1),
space = metOption("space_scatter", "right"),
key = metOption("key_scatter", "metaplot_key"),
as.table = metOption("as.table_scatter", TRUE),
prepanel = metOption("prepanel_scatter", NULL),
isoprepanel = metOption("isoprepanel_scatter", "iso_prepanel"),
scales = metOption("scales_scatter", NULL),
panel = metOption("panel_scatter", "scatter_panel"),
points = metOption("points_scatter", TRUE),
colors = metOption("colors_scatter", NULL),
fill = metOption("fill_scatter", NULL),
symbols = metOption("symbols_scatter", NULL),
sizes = metOption("sizes_scatter", 1),
types = metOption("types_scatter", "solid"),
widths = metOption("widths_scatter", 1),
lines = metOption("lines_scatter", FALSE),
main = metOption("main_scatter", NULL),
sub = metOption("sub_scatter", NULL),
subscripts = metOption("subscripts_scatter", TRUE),
settings = metOption("settings_scatter", NULL),
padding = metOption("padding_scatter", 1),
ref.col = metOption("ref.col_scatter", "grey"),
ref.lty = metOption("ref.lty_scatter", "solid"),
ref.lwd = metOption("ref.lwd_scatter", 1),
ref.alpha = metOption("ref.alpha_scatter", 1),
xref.col = metOption("xref.col_scatter", NULL),
xref.lty = metOption("xref.lty_scatter", NULL),
xref.lwd = metOption("xref.lwd_scatter", NULL),
xref.alpha = metOption("xref.alpha_scatter", NULL),
yref.col = metOption("yref.col_scatter", NULL),
yref.lty = metOption("yref.lty_scatter", NULL),
yref.lwd = metOption("yref.lwd_scatter", NULL),
yref.alpha = metOption("yref.alpha_scatter", NULL),
smooth.lty = metOption("smooth.lty_scatter", "dashed"),
smooth.lwd = metOption("smooth.lwd_scatter", 1),
smooth.alpha = metOption("smooth.alpha_scatter", 1),
fit = metOption("fit_scatter", conf),
fit.lty = metOption("fit.lty_scatter", "solid"),
fit.lwd = metOption("fit.lwd_scatter", 1),
fit.alpha = metOption("fit.alpha_scatter", 1),
conf = metOption("conf_scatter", FALSE),
conf.alpha = metOption("conf.alpha_scatter", 0.3),
```

```
  loc = metOption("loc_scatter", 0),
  global = metOption("global_scatter", FALSE),
  global.col = metOption("global.col_scatter", "grey"),
  global.fill = metOption("global.fill_scatter", "grey"),
  msg = metOption("msg_scatter", "metastats"),
  gg = metOption("gg_scatter", FALSE),
  verbose = metOption("verbose", FALSE),
  ...
)
```

Arguments

x	data.frame
yvar	character: y variable(s)
xvar	character: x variable
groups	optional grouping variable; ignored if more than one yvar
facets	optional conditioning variables
log	a default shared by ylog and xlog
ylog	log transform y axis (auto-selected if NA)
xlog	log transform x axis (auto-selected if NA)
crit	if ylog or xlog missing, log transform if mean/median ratio for non-missing values is greater than crit
yref	reference line from y axis; can be function(x = x, var = yvar, ...) or NULL to suppress
xref	reference line from x axis; can be function(x = x, var = xvar, ...) or NULL to suppress
ylab	y axis label; can be function(x = x, var = yvar, log = ylog, ..)
xlab	x axis label; can be function(x = x, var = xvar, log = xlog, ..)
ysmooth	supply loess smooth of y on x
xsmooth	supply loess smmoth of x on y
iso	logical: plot line of unity (auto-selected if NA); can be a (partial) list of aesthetics (col, lty, lwd, alpha)
na.rm	whether to remove data points with one or more missing coordinates
aspect	passed to bwplot or ggplot; use 'fill', NA, or NULL to calculate automatically
space	location of key (right, left, top, bottom)
key	list: passed to xyplot as auto.key or to theme; can be a function groups name, groups levels, points, lines, space, gg, and .... See metaplot_key.
as.table	passed to xyplot
prepanel	passed to xyplot (guessed if NULL)
isoprepanel	passed to xyplot if iso is TRUE
scales	passed to xyplot or facet_grid or facet_wrap (guessed if NULL)

panel	name or definition of panel function
points	whether to plot points and fill for each group: logical, or alpha values between 0 and 1
colors	replacements for default colors in group order; can be length one integer to auto-select that many colors
fill	replacements for default fill colors in group order (means something different for densplot_data_frame and categorical_data_frame). Used for confidence regions and for filling symbols (pch 21:25).
symbols	replacements for default symbols in group order (i.e. values of pch)
sizes	replacements for default symbol sizes in group order
types	replacements for default line types in group order
widths	replacements for default line widths in group order
lines	whether to plot lines for each group: logical, or alpha values between 0 and 1. Points are connected in the order in which they appear in the data.
main	character, or a function of x, yvar, xvar, groups, facets, and log
sub	character, or a function of x, yvar, xvar, groups, facets, and log
subscripts	passed to xyplot
settings	default parameter settings: a list from which matching elements are passed to lattice (as par.settings) or to ggplot theme() and facet_wrap() or facet_grid(). ncol and nrow are used as layout indices for lattice (for homology with facet_wrap). Also merged with ....
padding	numeric (will be recycled to length 4) giving plot margins in default units: top, right, bottom, left (in multiples of 5.5 points for ggplot)
ref.col	default shared by xref.col and yref.col; can be length one integer to auto-select that many colors
ref.lty	default shared by xref.lty and yref.lty
ref.lwd	default shared by xref.lwd and yref.lwd
ref.alpha	default shared by xref.alpha and yref.alpha
xref.col	x reference line color (recycled)
xref.lty	x reference line type (recycled)
xref.lwd	x reference line size (recycled)
xref.alpha	x reference line alpha (recycled)
yref.col	y reference line color (recycled)
yref.lty	y reference line type (recycled)
yref.lwd	y reference line size (recycled)
yref.alpha	y reference line alpha (recycled)
smooth.lty	smooth line type
smooth.lwd	smooth line size
smooth.alpha	smooth alpha
fit	draw a linear fit of y ~ x

fit.lty	fit line type
fit.lwd	fit line size
fit.alpha	fit alpha
conf	logical, or width for a confidence region around a linear fit; passed to region; TRUE defaults to 95 percent confidence interval; may not make sense if xlog is TRUE
conf.alpha	alpha transparency for confidence region
loc	where to print statistics on a panel; suppressed for grouped plots an facetted ggplots
global	if TRUE, xsmooth, ysmooth, fit, and conf are applied to all data rather than groupwise
global.col	color for global aesthetics
global.fill	fill color for global aesthetics
msg	a function to print text on a panel: called with x values, y values, and ....
gg	logical: whether to generate ggplot instead of trellis
verbose	generate messages describing process
...	passed to called functions e.g., region

See Also

scatter_panel

Other bivariate plots: iso_prepanel(), metaplot.data.frame(), scatter.data.frame(), scatter()

Other metaplot: boxplot_data_frame(), categorical_data_frame(), corsplom_data_frame(), densplot_data_frame(), metaplot_key(), metaplot(), test_metaplot()

Other scatter: metaplot_key(), scatter.data.frame(), scatter_panel(), scatter()

Examples

```
library(magrittr)
library(dplyr)
attr(Theoph$conc,'label') <- 'theophylline concentration'
attr(Theoph$conc,'guide') <- 'mg/L'
attr(Theoph$Time,'label') <- 'time'
attr(Theoph$Time,'guide') <- 'h'
attr(Theoph$Subject,'guide') <- '////'
scatter_data_frame(Theoph, 'conc', 'Time')
scatter_data_frame(Theoph, 'conc','Time', 'Subject')
scatter_data_frame(Theoph, 'conc','Time', facets = 'Subject')
scatter_data_frame(Theoph %>% filter(conc > 0), 'conc','Time', 'Subject',ylog = TRUE, yref = 5)
scatter_data_frame(Theoph, 'conc','Time', 'Subject',ylog = TRUE, yref = 5)
scatter_data_frame(Theoph, 'conc','Time', 'Subject',ysmooth = TRUE)
scatter_data_frame(Theoph, 'conc','Time', 'Subject',ysmooth = TRUE,global = TRUE)
scatter_data_frame(Theoph, 'conc','Time', conf = TRUE, loc = 3, yref = 6)
scatter_data_frame(Theoph, 'conc','Time', conf = TRUE, loc = 3, yref = 6)
```

scatter_panel *Panel Function for Metaplot Scatterplot*

Description

Default panel function for scatter_data_frame. Calls panel.xyplot and optionally plots linear fit, confidence region, reference lines, and statistics. Note that, although global options are supported, typically these are unreachable since the calling function supplies appropriate values.

Usage

```
scatter_panel(
  x,
  y,
  groups,
  xref = metOption("xref_scatter_panel", scatter_panel_ref),
  yref = metOption("yref_scatter_panel", scatter_panel_ref),
  ref.col = metOption("ref.col_scatter_panel", "grey"),
  ref.lty = metOption("ref.lty_scatter_panel", "solid"),
  ref.lwd = metOption("ref.lwd_scatter_panel", 1),
  ref.alpha = metOption("ref.alpha_scatter_panel", 1),
  xref.col = metOption("xref.col_scatter_panel", NULL),
  xref.lty = metOption("xref.lty_scatter_panel", NULL),
  xref.lwd = metOption("xref_lwd_scatter_panel", NULL),
  xref.alpha = metOption("xref_alpha_scatter_panel", NULL),
  yref.col = metOption("yref_col_scatter_panel", NULL),
  yref.lty = metOption("yref_lty_scatter_panel", NULL),
  yref.lwd = metOption("yref_lwd_scatter_panel", NULL),
  yref.alpha = metOption("yref_alpha_scatter_panel", NULL),
  ysmooth = metOption("ysmooth_scatter_panel", FALSE),
  xsmooth = metOption("xsmooth_scatter_panel", FALSE),
  smooth.lty = metOption("smooth.lty_scatter_panel", "dashed"),
  smooth.lwd = metOption("smooth.lwd_scatter_panel", 1),
  smooth.alpha = metOption("smooth.alpha_scatter_panel", 1),
  fit = metOption("fit_scatter_panel", NULL),
  fit.lty = metOption("fit.lty_scatter_panel", "solid"),
  fit.lwd = metOption("fit.lwd_scatter_panel", 1),
  fit.alpha = metOption("fit.alpha_scatter_panel", 1),
  conf = metOption("conf_scatter_panel", FALSE),
  conf.alpha = metOption("conf.alpha_scatter_panel", 0.3),
  loc = metOption("loc_scatter_panel", 0),
  iso = metOption("iso_scatter_panel", FALSE),
  global = metOption("global_scatter_panel", FALSE),
  global.col = metOption("global.col_scatter_panel", "grey"),
  global.fill = metOption("global.fill_scatter_panel", "grey"),
  msg = metOption("msg_scatter_panel", "metastats"),
  type,
```

```
  verbose = metOption("verbose_scatter_panel", FALSE),
  ...
)
```

Arguments

x	x values
y	y values
groups	optional grouping item
xref	reference line from x axis; can be function(x, y, ...)
yref	reference line from y axis; can be function(y, x, ...)
ref.col	default shared by xref.col and yref.col
ref.lty	default shared by xref.lty and yref.lty
ref.lwd	default shared by xref.lwd and yref.lwd
ref.alpha	default shared by xref.alpha and yref.alpha
xref.col	x reference line color (recycled)
xref.lty	x reference line type (recycled)
xref.lwd	x reference line size (recycled)
xref.alpha	x reference line alpha (recycled)
yref.col	y reference line color (recycled)
yref.lty	y reference line type (recycled)
yref.lwd	y reference line size (recycled)
yref.alpha	y reference line alpha (recycled)
ysmooth	supply loess smooth of y on x
xsmooth	supply loess smmoth of x on y
smooth.lty	smooth line type
smooth.lwd	smooth line size
smooth.alpha	smooth alpha
fit	draw a linear fit of y ~ x; defaults to as.logical(conf)
fit.lty	fit line type
fit.lwd	fit line size
fit.alpha	fit alpha
conf	logical, or width for a confidence region around a linear fit; passed to region; TRUE defaults to 95 percent confidence interval; may not make sense if xlog is TRUE
conf.alpha	alpha transparency for confidence region
loc	where to print statistics on a panel; suppressed for grouped plots
iso	logical: use isometric axes with line of unity (auto-selected if NA); can be a (partial) list of aesthetics (col, lty, lwd, alpha)

global	if TRUE, xsmooth, ysmooth, fit, and conf are applied to all data rather than groupwise
global.col	color for global aesthetics
global.fill	fill color for global aesthetics
msg	a function to print text on a panel: called with x values, y values, and ....
type	overridden by scatter_panel
verbose	generate messages describing process
...	passed to panel.superpose, panel.xyplot, panel.polygon, region, panel.text

See Also

metastats

scatter.data.frame

Other panel functions: boxplot_panel(), categorical_panel(), corsplom_gg_correlation(), corsplom_gg_diagonal(), corsplom_gg_scatter(), corsplom_panel_correlation(), corsplom_panel_diagonal(), corsplom_panel_scatter(), dens_panel(), diag_label(), diag_pin(), iso_prepanel(), metaplot_key(), metaplot_ref(), panel.meta_densityplot(), panel_tile(), scatter_panel_ref()

Other scatter: metaplot_key(), scatter.data.frame(), scatter_data_frame(), scatter()

---

scatter_panel_ref *Calculate Panel Reference Values*

Description

Calculates reference values for x and y axes at the panel level.

Usage

```
scatter_panel_ref(a, b, ...)
```

Arguments

a	vector of interest
b	vector for other axis
...	ignored

Value

numeric

See Also

Other panel functions: boxplot_panel(), categorical_panel(), corsplom_gg_correlation(), corsplom_gg_diagonal(), corsplom_gg_scatter(), corsplom_panel_correlation(), corsplom_panel_diagonal(), corsplom_panel_scatter(), dens_panel(), diag_label(), diag_pin(), iso_prepanel(), metaplot_key(), metaplot_ref(), panel.meta_densityplot(), panel_tile(), scatter_panel()

Other reference lines: diag_pin(), metaplot_ref()

`setOption` *Set or Reset Metaplot Options*

Description

Sets an option value in the list `getOption('metaplot')`. If invoked without named arguments, option 'metaplot' is set to NULL. Setting an existing option moves it to the end of the list (breaks ties in `metOption`).

Usage

```
setOption(...)
```

Arguments

`...`	any metaplot options can be defined, using `name = value`.

Value

(invisible) character vector of option names that were set or unset

See Also

`metOptionoptions`

Examples

```
example(metOption)
```

test_metaplot *Test Metaplot Variants*

Description

Tests metaplot variants by example. Returns null. Use `example(test_metaplot)`.

Usage

```
test_metaplot()
```

See Also

Other generic functions: `axislabel()`, `categorical()`, `corsplom()`, `densplot()`, `metaplot()`, `pack()`, `scatter()`, `unpack()`

Other metaplot: `boxplot_data_frame()`, `categorical_data_frame()`, `corsplom_data_frame()`, `densplot_data_frame()`, `metaplot_key()`, `metaplot()`, `scatter_data_frame()`

Examples

```
library(magrittr)
library(dplyr)
library(csv)
x <- as.csv(system.file(package = 'metaplot', 'extdata/theoph.csv'))
x %<>% pack

multiplot(
x %>% metaplot(sres, gg = F),
x %>% metaplot(sres, gg = T, padding = 3.5)
)
multiplot(
x %>% metaplot(site, gg = F),
x %>% metaplot(site, gg = T, padding = 3.5)
)
multiplot(
x %>% metaplot(conc, arm, gg = F),
x %>% metaplot(conc, arm, gg = T, padding = 4)
)
multiplot(
x %>% densplot(conc, arm, gg = F),
x %>% densplot(conc, arm, gg = T, padding = 8)
)
multiplot(
x %>% densplot(
  conc, arm, gg = F, space = 'top',
  columns = 2,
  legend.direction = 'horizontal' # ignored
),
x %>% densplot(conc, arm, gg = T, space = 'top',
```

```
  columns = 2, # ignored
  legend.direction = 'horizontal' , padding = 3
))
multiplot(
x %>% metaplot(arm, conc, gg = F),
x %>% metaplot(arm, conc, gg = T, padding = 3.5)
)
multiplot(
x %>% metaplot(conc, arm, site, gg = F),
x %>% metaplot(conc, arm, site, gg = T, padding = 5)
)
multiplot(
x %>% metaplot(conc, site, arm, gg = F),
x %>% metaplot(conc, site, arm, gg = T, padding = 5)
)
multiplot(
x %>% metaplot(conc, time, gg = F),
x %>% metaplot(conc, time, gg = T, padding = 5)
)
multiplot(
x %>% metaplot(arm, site, gg = F),
x %>% metaplot(arm, site, gg = T, padding = 3)
)
multiplot(
x %>% metaplot(arm, site, cohort, gg = F),
x %>% metaplot(arm, site, cohort, gg = T, padding = 5)
)
multiplot(
x %>% metaplot(arm, site, cohort, gg = F, space = 'top',
  columns = 2, padding = c(5,1,1,1)),
x %>% metaplot(arm, site, cohort, gg = T, space = 'top',
  legend.direction = 'horizontal', padding = 2)
)
multiplot(
x %>% metaplot(arm, site, , cohort, gg = F),
x %>% metaplot(arm, site, , cohort, gg = T, padding = 4)
)
multiplot(
x %>% metaplot(conc, time, subject, gg = F),
x %>% metaplot(conc, time, subject, gg = T, padding = 3)
)
multiplot(
x %>% metaplot(conc, time, , subject, gg = F),
x %>% metaplot(conc, time, , subject, gg = T, padding = 5)
)
multiplot( ncol = 2,
x %>% metaplot(conc, time, subject, site, gg = F),
x %>% metaplot(conc, time, subject, site, gg = T, padding = 4)
)
multiplot(
x %>% metaplot(conc, time, subject, site, arm, gg = F, padding = 2),
x %>% metaplot(conc, time, subject, site, arm, gg = T)
)
```

```
multiplot(
x %>% metaplot(lKe, lKa, lCl, gg = F),
x %>% metaplot(lKe, lKa, lCl, gg = T, padding = 2)
)
multiplot(
x %>% metaplot(
  lKe, lKa, lCl,
  col = 'black',smooth.col = 'red', pin.col = 'red',
  dens.col = 'blue', dens.alpha = 0.1, gg = F
),
x %>% metaplot(
  lKe, lKa, lCl,
  col = 'black',smooth.col = 'red', pin.col = 'red',
  dens.col='blue',dens.alpha = 0.1, gg = T, padding = 2)
)
multiplot(
x %>% metaplot(conc, pred, ipred, time, space = 'top', gg = F),
x %>% metaplot(conc, pred, ipred, time, space = 'top', gg = T, padding = 3)
)
multiplot(
x %>% metaplot(conc, pred, ipred, time, subject, space = 'top', gg = F),
x %>% metaplot(conc, pred, ipred, time, subject, space = 'top', gg = T, padding = 5)
)
multiplot(
x %>% metaplot(
  conc, pred, ipred, time, subject,
  colors = c('black','blue','orange'),
  points = c(0.9,0, 0.4),
  lines = c(F,T,T),
  space = 'top', gg = F
),
x %>% metaplot(
  conc, pred, ipred, time, subject,
  colors = c('black','blue','orange'),
  points = c(0.9,0, 0.4),
  lines = c(F,T,T),
  space = 'top', gg = T, padding = 4
))
multiplot(
x %>% metaplot(conc, ipred, time, site, arm, space = 'top', gg = F),
x %>% metaplot(conc, ipred, time, site, arm, space = 'top', gg = T)
)
multiplot(
x %>% metaplot(res, conc, yref = 0, ysmooth = T, conf = T, grid = T, loc = 1, gg = F),
x %>% metaplot(res, conc, yref = 0, ysmooth = T, conf = T, grid = T, loc = 1, gg = T, padding = 3.5)
)
multiplot(
x %>% metaplot(res, conc, arm, ysmooth = T, conf = T , gg = F),
x %>% metaplot(res, conc, arm, ysmooth = T, conf = T , gg = T, padding = 3.5)
)
# Fill color can differ from point color but is the same for points and regions.
# 'points' controls alpha of point and point fill independently of conf.fill.
multiplot(
```

```
x %>% metaplot(res, conc, arm, conf = T , gg = F, yref = NULL, points = 0.3,
  symbols = 21:22, colors = c('blue','black'), fill = c('green','red')
),
x %>% metaplot(res, conc, arm, conf = T , gg = T, yref = NULL, points = 0.3, padding = 3.5,
  symbols = 21:22, colors = c('blue','black'), fill = c('green','red')
))
multiplot(
x %>% metaplot(res, conc, arm, ysmooth = T, conf = T, global = T,
  ref.col = 'red', gg = F),
x %>% metaplot(res, conc, arm, ysmooth = T, conf = T, global = T,
  ref.col = 'red', gg = T, padding = 3.5)
)
multiplot(
x %>% metaplot(subject,conc, gg = F),
x %>% metaplot(subject,conc, gg = T, padding = 3.5)
)

# manage metadata
attr(x$arm, 'guide') # //1/Arm A//2/Arm B//
multiplot(
x %>% metaplot(conc, arm, gg = F),
x %>% metaplot(conc, arm, gg = T, padding = 4)
) # default

multiplot(
x %>% mutate(arm = arm %>%
  structure(guide = '//2/Arm B//1/Arm A//')) %>%
  metaplot(conc, arm, gg = F),
x %>% mutate(arm = arm %>%
  structure(guide = '//2/Arm B//1/Arm A//')) %>%
  metaplot(conc, arm, gg = T, padding = 4) # different presentation order
)

multiplot(
x %>% mutate(arm = arm %>%
  structure(guide = '//1/Both Arms//2/Both Arms//')) %>%
  metaplot(conc, arm, gg = F),
x %>% mutate(arm = arm %>%
  structure(guide = '//1/Both Arms//2/Both Arms//')) %>%
  metaplot(conc, arm, gg = T, padding = 4) # collapse cases
)

x %>% densplot(
 main = 'Density Plot',
 sub = 'using lattice',
 gg = F,
 sres, subject,
 ref.col = 'red', ref.alpha = 0.5,
 ref.lty = 'dashed', ref.lwd = 2,
 log = F,
 aspect = NULL,
 colors = c('red','blue','darkgreen'),
 symbols = c(21, 22, 23),
```

```
 points = 0.3,
 lines = .5,
 fill = 0.1,
 space = 'left',
 padding = c(1,2,3,4),
 other = 'none'
)
x %>% densplot(
 main = 'Density Plot',
 sub = 'using ggplot',
 gg = T,
 sres, subject,
 ref.col = 'red', ref.alpha = 0.5,
 ref.lty = 'dashed', ref.lwd = 2,
 log = F,
 aspect = NULL,
 colors = c('red','blue','darkgreen'),
 symbols = c(21, 22, 23),
 points = 0.3,
 lines = 0.5,
 fill = 0.1,
 space = 'left',
 padding = 1:4,
 other = 'none'
)
x %>% filter(conc > 0) %>% metaplot(
 main = 'Box Plot',
 sub = 'using lattice',
 gg = F,
 arm, conc,
 log = T,
 ref = 4, ref.col = 'red',
 ref.lty = 'dashed', ref.lwd = 2,
 nobs = T,
 padding = 1:4,
 reverse = FALSE,
 pch = 20,
 notch = TRUE,
 aspect = NA,
 other = 'none'
)

x %>% filter(conc > 0) %>% metaplot(
 main = 'Box Plot',
 sub = 'using ggplot',
 gg = T,
 arm, conc,
 log = T,
 ref = 4, ref.col = 'red',
 ref.lty = 'dashed', ref.lwd = 2,
 nobs = T,
 padding = 1:4,
 reverse = FALSE,
```

```
 pch = 20,
 notch = TRUE,
 aspect = NA,
 other = 'none'
)
x %>% metaplot(
 main = 'Categorical Plot',
 sub = 'using lattice',
 gg = F,
 arm, site, cohort,
 aspect = 'fill', space = 'top',
 as.table = FALSE,
 colors = c('red','blue','green'),
 fill = c(0.3, 0.5, 0.7),
 lines = c(0.7, 0.5, 0.3),
 tex = 0.8, rot = 45,
 padding = 1:4, loc = 1,
 cex = .5,
 other = 'none'
)

x %>% metaplot(
 main = 'Categorical Plot',
 sub = 'using ggplot2',
 gg = T,
 arm, site, cohort,
 aspect = 'fill', space = 'top',
 as.table = FALSE,
 colors = c('red','blue','green'),
 fill = c(0.3, 0.5, 0.7),
 lines = c(0.7, 0.5, 0.3),
 tex = 0.8, rot = 45,
 padding = 1:4, loc = 1,
 cex = .5,
 other = 'none'
)
x %>% metaplot(
  main = 'Correlation Splom',
  sub = 'using lattice',
  gg = F,
  lKe, lKa, lCl,
  varname.cex = 2,
  col = 'purple',
  smooth.col = 'orange', smooth.alpha = 0.9,
  smooth.lty = 'dashed', smooth.lwd = 2,
  pin.col = 'orange', pin.alpha = 0.9,
  dens.col = 'purple',dens.alpha = 0.2, dens.scale = 0.1,
  padding = 1:4,
  other = 'none',
  xlab = 'parameters'
)
x %>% metaplot(
  main = 'Correlation Splom',
```

```
  sub = 'using ggplot',
  gg = T,
  lKe, lKa, lCl,
  varname.cex = 2,
  col = 'purple',
  smooth.col = 'orange', smooth.alpha = 0.9,
  smooth.lty = 'dashed', smooth.lwd = 2,
  pin.col = 'orange', pin.alpha = 0.9,
  dens.col = 'purple',dens.alpha = 0.2, dens.scale = 0.1,
  padding = 1:4,
  other = 'none',
  xlab = 'parameters'
)
x %>% metaplot(
  main = 'Scatterplot',
  sub = 'using lattice',
  gg = F,
  res, conc,
  yref = 0, ysmooth = T,
  smooth.lty = 'dotted', smooth.lwd = 2,
  smooth.alpha = 1,
  aspect = 0.8,
  space = 'bottom',
  colors = c('purple','darkgreen','peach'),
  symbols = 21:23,
  points = c(0.3, 0.5, 0.7),
  lines = F,
  padding = 1:4,
  ref.col = 'blue',
  ref.lty = 'dashed', ref.lwd = 2,
  ref.alpha = 0.5,
  conf = .99999,
  fit.lty = 'dashed', fit.lwd = 2,
  fit.alpha = 0.5,
  conf.alpha = 0.2,
  global = T,
  global.col = 'darkgreen',
  grid = T, loc = 1,
  other = 'none'
 )
x %>% metaplot(
  main = 'Scatterplot',
  sub = 'using ggplot',
  gg = T,
  res, conc,
  yref = 0, ysmooth = T,
  smooth.lty = 'dotted', smooth.lwd = 2,
  smooth.alpha = 1,
  aspect = 0.8,
  space = 'bottom',
  colors = c('purple','darkgreen','peach'),
  symbols = 21:23,
  points = c(0.3, 0.5, 0.7),
```

```
  lines = F,
  padding = 1:4,
  ref.col = 'blue',
  ref.lty = 'dashed', ref.lwd = 2,
  ref.alpha = 0.5,
  conf = .99999,
  fit.lty = 'dashed', fit.lwd = 2,
  fit.alpha = 0.5,
  conf.alpha = 0.2,
  global = T,
  global.col = 'darkgreen',
  grid = T, loc = 1,
  other = 'none'
 )

 # vectorized reference aesthetics
multiplot(
  x %>% metaplot(
   sres, gg = F,
   ref.col = c('blue','red'),
   ref.lty = c('dashed','dotted')
  ),
  x %>% metaplot(
   sres, gg = T,
   ref.col = c('blue','red'),
   ref.lty = c('dashed','dotted'),
   padding = 3.5
  )
)
multiplot(
  x %>% densplot(
   sres, arm, gg = F,
   ref.col = c('blue','red'),
   ref.lty = c('dashed','dotted')
  ),
  x %>% densplot(
   sres, arm, gg = T,
   ref.col = c('blue','red'),
   ref.lty = c('dashed','dotted'),
   padding = 3.5
  )
)
multiplot(
  x %>% densplot(
   sres,, arm, gg = F,
   ref.col = c('blue','red'),
   ref.lty = c('dashed','dotted')
  ),
  x %>% densplot(
   sres,, arm, gg = T,
   ref.col = c('blue','red'),
   ref.lty = c('dashed','dotted'),
   padding = 3.5
```

```
 )
)
multiplot(
  x %>% metaplot(
   sres, time,, arm, gg = F,
   yref = c(-4,0,4),
   xref = c(5, 10, 15),
   yref.col = c('blue','red'),
   yref.lty = c('dashed','dotted'),
   xref.col = c('green','orange')
  ),
  x %>% metaplot(
   sres, time,, arm, gg = T,
   yref = c(-4,0,4),
   xref = c(5, 10, 15),
   yref.col = c('blue','red'),
   yref.lty = c('dashed','dotted'),
   xref.col = c('green','orange'),
   padding = 3.5
 )
)
# use of settings
multiplot(
 x %>% metaplot(conc, ,subject, settings = list(ncol = 4, nrow = 3), gg = F),
 x %>% metaplot(conc, ,subject, settings = list(ncol = 4), padding = 4, gg = T)
)
multiplot(
 x %>% metaplot(conc, time,, subject, settings = list(ncol = 4, nrow = 3), gg = F),
 x %>% metaplot(conc, time,, subject, settings = list(ncol = 4), padding = 4,  gg = T)
)
multiplot(
 x %>% metaplot(conc, arm, site, settings = list(ncol = 1, nrow = 2), gg = F),
 x %>% metaplot(conc, arm, site, settings = list(ncol = 1), padding = 4,  gg = T)
)

#iso aesthetics
multiplot(
 x %>% metaplot(conc, ipred, iso = NA, gg = F),
 x %>% metaplot(conc, ipred, iso = NA, gg = T, padding = 4)
)
multiplot(
 x %>% metaplot(conc, ipred, iso = list(lty = 'dashed'), gg = F),
 x %>% metaplot(conc, ipred, iso = list(lty = 'dashed'), gg = T, padding = 4)
)
```

---

tiles *Calculate Tile Limits*

Description

Calculates limits for mosaic tiles

Usage

```
tiles(x, ..., tex = 0.9, msg = "tilestats", verbose = FALSE)
```

Arguments

x	a data.frame with at least columns x, y, and g, possibly f1 and f2 (facets)
...	other arguments
tex	tile shrinkage <= 1
msg	a function of x and y to create a tile message
verbose	generate messages describing process

Value

data.frame

See Also

categorical_panel

Other categorical family: cax(), tilestats()

---

tilestats *Format Tile Statistics*

Description

Formats statistics for a mosaic tile.

Usage

```
tilestats(x, y, ...)
```

Arguments

x	x values
y	y values
...	other arguments

Value

character

See Also

categorical_panel

Other categorical family: cax(), tiles()

---

unpack *Unpack Something*

Description

Unpack Something. Generic, with method for data.frame.

Usage

```
unpack(x, ...)
```

Arguments

x	object
...	other arguments

See Also

Other pack: pack.data.frame(), pack(), unpack.data.frame()

Other generic functions: axislabel(), categorical(), corsplom(), densplot(), metaplot(), pack(), scatter(), test_metaplot()

---

unpack.data.frame *Express Scalar Column Attributes as Column Metadata*

Description

Expresses scalar column attributes as column metadata (row values). Column with name meta is created to hold names of attributes, if any. A transposed table (sorted by attribute name) of scalar column attribute values (coerced to character) is bound to the existing data.frame (the attributes themselves are removed from columns). Bind position is controlled by position such that the intersection of new rows and column occurs in the corresponding corner, numbered clockwise from top-left. Resulting column classes are character. It is an error if meta is already in names(x).

Usage

```
## S3 method for class 'data.frame'
unpack(
  x,
  meta = getOption("meta", "meta"),
  position = 1L,
  ignore = c("class", "levels"),
  ...
)
```

Arguments

x	data.frame
meta	column in result giving names of attributes
position	1 (top-left), 2 (top-right), 3 (bottom-right), or 4 (bottom-left)
ignore	character: attributes to ignore
...	ignored arguments

Value

data.frame

data.frame with all columns of class character

See Also

Other pack: pack.data.frame(), pack(), unpack()

Other methods: axislabel.data.frame(), boxplot.data.frame(), categorical.data.frame(), corsplom.data.frame(), densplot.data.frame(), metaplot.data.frame(), pack.data.frame(), plot.metaplot_gtable(), print.metaplot_gtable(), scatter.data.frame()

---

wikisym2plotmath *Convert Wiki Symbol to Plotmath*

Description

Converts wiki symbol to plotmath. Vectorized version of wikisym2plotmath_.

Usage

```
wikisym2plotmath(x, ...)
```

Arguments

x	character
...	ignored

Value

expression

See Also

Other formatters: `diag_label()`, `wikisym2plotmath_()`

`wikisym2plotmath_` *Convert One Wiki Symbol to Plotmath*

Description

Converts one wiki symbol to plotmath. A Wiki symbol is simple text with arbitrarily nested subscript (`_`) and superscript (`^`) groupings. Use dot (`.`) to explicitly terminate a grouping, and use backslash-dot (`\.`) for a literal dot. Examples: `V_c./F`. Trailing dots need not be supplied. Leading/trailing whitespace is removed. Tab character not allowed.

Usage

```
wikisym2plotmath_(x, ...)
```

Arguments

`x`	character
`...`	ignored

Value

expression

See Also

Other formatters: `diag_label()`, `wikisym2plotmath()`

Examples

```
wikisym2plotmath_('V_c./F')
wikisym2plotmath_('AUC_ss')
wikisym2plotmath_('C_max_ss')
wikisym2plotmath_('var^eta_j')
```

Index