Package 'GGally'

January 14, 2016

Version 1.0.1

Date 2016-01-13

License GPL (>= 2.0)

Title Extension to ggplot2

Type Package

LazyLoad yes

LazyData true

URL https://ggobi.github.io/ggally, https://github.com/ggobi/ggally

BugReports https://github.com/ggobi/ggally/issues

Description The R package '\{\}href{http://docs.ggplot2.org/current/}{ggplot2}' is a plotting system based on the grammar of graphics. '\{\}href{https://ggobi.github.io/ggally}{GGally}' extends 'ggplot2' by adding several functions to reduce the complexity of combining geoms with transformed data. Some of these functions include a pairwise plot matrix, a scatterplot plot matrix, a parallel coordinates plot, a survival plot, and several functions to plot networks.

Depends R (>= 2.14)

Imports ggplot2 (>= 2.0.0), grid (>= 3.0.0), gtable (>= 0.1.2), plyr (>= 1.8), reshape (>= 0.8.4)

Suggests arm (>= 1.7), geosphere (>= 1.3-11), ggmap (>= 2.3), igraph (>= 1.0.0), intergraph (>= 2.0-0), maps (>= 2.3-9), network (>= 1.7.2), RColorBrewer (>= 1.0-5), scagnostics (>= 0.2-4), scales (>= 0.2.3), sna (>= 2.3-1), survival (>= 2.37-4), tnet (>= 3.0), knitr (>= 1.12), rmarkdown (>= 0.9.2), roxygen2 (>= 5.0.0), testthat

RoxygenNote 5.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Barret Schloerke [aut, cre] (author for ggpairs and ggally_*. contributor for all functions.),
Jason Crowley [aut] (ggparcoord),

Di Cook [aut, ths] (ggscatmat, gglyph),
Heike Hofmann [ths],
Hadley Wickham [ths],
Francois Briatte [aut] (ggcorr, ggnet, ggnet2),
Moritz Marbach [aut] (ggnet, ggnet2),
Edwin Thoen [aut] (ggsurv),
Amos Elberg [aut] (ggnetworkmap)

Maintainer Barret Schloerke <schloerke@gmail.com>

Repository CRAN

Date/Publication 2016-01-14 11:24:09

R topics documented:

`+.gg` *Modify a ggmatrix object by adding an ggplot2 object to all plots*

Description

This operator allows you to add ggplot2 objects to a ggmatrix object.

Usage

```
## S3 method for class 'gg'
e1 + e2
```

Arguments

`e1`	An object of class `ggplot` or `theme`
`e2`	A component to add to `e1`

Details

If the first object is an object of class `ggmatrix`, you can add the following types of objects, and it will return a modified ggplot object.

- `theme`: update plot theme

The + operator completely replaces elements with elements from e2.

See Also

`+.gg` and `theme`

Examples

```
data(tips, package = "reshape")
pm <- ggpairs(tips[, 2:3])
## change to black and white theme
pm + ggplot2::theme_bw()
## change to linedraw theme
# pm + ggplot2::theme_linedraw()
## change to custom theme
# pm + ggplot2::theme(panel.background = ggplot2::element_rect(fill = "lightblue"))
```

`add_ref_boxes` *Add reference boxes around each cell of the glyphmap.*

Description

Add reference boxes around each cell of the glyphmap.

Usage

```
add_ref_boxes(data, var_fill = NULL, color = "white", size = 0.5,
  fill = NA, ...)
```

Arguments

`data`	A glyphmap structure.
`var_fill`	Variable name to use to set the fill color
`color`	Set the color to draw in, default is "white"
`size`	Set the line size, default is 0.5
`fill`	fill value used if `var_fill` is `NULL`
`...`	other arguments passed onto `geom_rect`

`add_ref_lines` *Add reference lines for each cell of the glyphmap.*

Description

Add reference lines for each cell of the glyphmap.

Usage

```
add_ref_lines(data, color = "white", size = 1.5, ...)
```

Arguments

data	A glyphmap structure.
color	Set the color to draw in, default is "white"
size	Set the line size, default is 1.5
...	other arguments passed onto geom_line

---

flea *Historical data used for classification examples.*

Description

This data contains physical measurements on three species of flea beetles.

Usage

```
data(flea)
```

Format

A data frame with 74 rows and 7 variables

Details

- species Ch. concinna, Ch. heptapotamica, Ch. heikertingeri
- tars1 width of the first joint of the first tarsus in microns
- tars2 width of the second joint of the first tarsus in microns
- head the maximal width of the head between the external edges of the eyes in 0.01 mm
- aede1 the maximal width of the aedeagus in the fore-part in microns
- aede2 the front angle of the aedeagus (1 unit = 7.5 degrees)
- aede3 the aedeagus width from the side in microns

References

Lubischew, A. A. (1962), On the Use of Discriminant Functions in Taxonomy, Biometrics 18:455-477.

getPlot *getPlot*

Description

Retrieves the ggplot object at the desired location.

Usage

```
getPlot(x, i, j)

## S3 method for class 'ggmatrix'
x[i, j, ...]
```

Arguments

x	ggpair object to select from
i	row from the top
j	column from the left
...	ignored

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
plotMatrix2 <- ggpairs(tips[, 3:2], upper = list(combo = "denstrip"))
plotMatrix2[1, 2]
```

ggally_barDiag *Plots the Bar Plots by Using Diagonal*

Description

Plots the bar plots by using Diagonal.

Usage

```
ggally_barDiag(data, mapping, ..., rescale = FALSE)
```

Arguments

data	data set using
mapping	aesthetics being used
...	other arguements are sent to geom_bar
rescale	boolean to decide whether or not to rescale the count output

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
ggally_barDiag(tips, mapping = ggplot2::aes(x = day))
ggally_barDiag(tips, mapping = ggplot2::aes(x = tip), binwidth = 0.25)
```

---

ggally_blank *Blank*

Description

Draws nothing.

Usage

```
ggally_blank(...)

ggally_blankDiag(...)
```

Arguments

...	other arguments ignored

Details

Makes a "blank" ggplot object that will only draw white space

Author(s)

Barret Schloerke <schloerke@gmail.com>

ggally_box *Plots the Box Plot*

Description

Make a box plot with a given data set

Usage

```
ggally_box(data, mapping, ...)
```

Arguments

data	data set using
mapping	aesthetics being used
...	other arguments being supplied to geom_boxplot

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
ggally_box(tips, mapping = ggplot2::aes(x = total_bill, y = sex))
ggally_box(tips, mapping = ggplot2::aes_string(x = "total_bill", y = "sex"))
ggally_box(
  tips,
  mapping        = ggplot2::aes_string(y = "total_bill", x = "sex", color = "sex"),
  outlier.colour = "red",
  outlier.shape  = 13,
  outlier.size   = 8
)
```

ggally_cor *Correlation from the Scatter Plot*

Description

Estimate correlation from the given data.

Usage

```
ggally_cor(data, mapping, alignPercent = 0.6, method = "pearson",
  use = "complete.obs", corAlignPercent = NULL, corMethod = NULL,
  corUse = NULL, ...)
```

Arguments

data	data set using
mapping	aesthetics being used
alignPercent	right align position of numbers. Default is 60 percent across the horizontal
method	method suppied to cor function
use	use supplied to cor function
corAlignPercent	deprecated. Use parameter alignPercent
corMethod	deprecated. Use parameter method
corUse	deprecated. Use parameter use
...	other arguments being supplied to geom_text

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
ggally_cor(tips, mapping = ggplot2::aes_string(x = "total_bill", y = "tip"))
ggally_cor(
  tips,
  mapping = ggplot2::aes(x = total_bill, y = tip),
  size = 15,
  colour = I("red")
)
ggally_cor(
  tips,
  mapping = ggplot2::aes_string(x = "total_bill", y = "tip", color = "sex"),
  size = 5
)
```

---

ggally_density *Plots the Scatter Density Plot*

Description

Make a scatter density plot from a given data.

Usage

```
ggally_density(data, mapping, ...)
```

Arguments

data	data set using
mapping	aesthetics being used
...	parameters sent to either stat_density2d or geom_density2d

Details

The aesthetic "fill" determines whether or not stat_density2d (filled) or geom_density2d (lines) is used.

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
ggally_density(tips, mapping = ggplot2::aes(x = total_bill, y = tip))
ggally_density(tips, mapping = ggplot2::aes_string(x = "total_bill", y = "tip"))
ggally_density(
  tips,
  mapping = ggplot2::aes_string(x = "total_bill", y = "tip", fill = "..level..")
)
ggally_density(
  tips,
  mapping = ggplot2::aes_string(x = "total_bill", y = "tip", fill = "..level..")
) + ggplot2::scale_fill_gradient(breaks = c(0.05, 0.1, 0.15, 0.2))
```

---

ggally_densityDiag *Plots the Density Plots by Using Diagonal*

Description

Plots the density plots by using Diagonal.

Usage

```
ggally_densityDiag(data, mapping, ..., rescale = FALSE)
```

Arguments

data	data set using
mapping	aesthetics being used.
...	other arguments sent to stat_density
rescale	boolean to decide whether or not to rescale the count output

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
ggally_densityDiag(tips, mapping = ggplot2::aes(x = total_bill))
ggally_densityDiag(tips, mapping = ggplot2::aes(x = total_bill, color = day))
```

---

ggally_denstrip *Plots a tile plot with facets*

Description

Make Tile Plot as densely as possible.

Usage

```
ggally_denstrip(data, mapping, ...)
```

Arguments

data	data set using
mapping	aesthetics being used
...	other arguments being sent to stat_bin

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
ggally_denstrip(tips, mapping = ggplot2::aes(x = total_bill, y = sex))
ggally_denstrip(tips, mapping = ggplot2::aes_string(x = "total_bill", y = "sex"))
ggally_denstrip(
  tips,
  mapping = ggplot2::aes_string(x = "sex", y = "tip", binwidth = "0.2")
) + ggplot2::scale_fill_gradient(low = "grey80", high = "black")
```

`ggally_diagAxis` *Internal Axis Labeling Plot for ggpairs*

Description

This function is used when `axisLabels == "internal"`.

Usage

```
ggally_diagAxis(data, mapping, label = mapping$x, labelSize = 5,
  labelXPercent = 0.5, labelYPercent = 0.55, labelHJust = 0.5,
  labelVJust = 0.5, gridLabelSize = 4, ...)
```

Arguments

`data`	dataset being plotted
`mapping`	aesthetics being used (x is the variable the plot will be made for)
`label`	title to be displayed in the middle. Defaults to `mapping$x`
`labelSize`	size of variable label
`labelXPercent`	percent of horizontal range
`labelYPercent`	percent of vertical range
`labelHJust`	hjust supplied to label
`labelVJust`	vjust supplied to label
`gridLabelSize`	size of grid labels
`...`	other arguments for geom_text

Author(s)

Jason Crowley <crowley.jason.s@gmail.com> and Barret Schloerke

Examples

```
data(tips, package = "reshape")
ggally_diagAxis(tips, ggplot2::aes(x=tip))
ggally_diagAxis(tips, ggplot2::aes(x=sex))
```

ggally_dot *Plots the Box Plot with Dot*

Description

Add jittering with the box plot

Usage

```
ggally_dot(data, mapping, ...)
```

Arguments

data	data set using
mapping	aesthetics being used
...	other arguments being supplied to geom_jitter

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
ggally_dot(tips, mapping = ggplot2::aes(x = total_bill, y = sex))
ggally_dot(tips, mapping = ggplot2::aes_string(x = "total_bill", y = "sex"))
ggally_dot(
  tips,
  mapping = ggplot2::aes_string(y = "total_bill", x = "sex", color = "sex")
)
ggally_dot(
  tips,
  mapping = ggplot2::aes_string(y = "total_bill", x = "sex", color = "sex", shape = "sex")
) + ggplot2::scale_shape(solid=FALSE)
```

ggally_dotAndBox *Plots either Box Plot or Dot Plots*

Description

Place box plots or dot plots on the graph

Usage

```
ggally_dotAndBox(data, mapping, ..., boxPlot = TRUE)
```

Arguments

data	data set using
mapping	aesthetics being used
...	parameters passed to either geom_jitter or geom_boxplot
boxPlot	boolean to decide to plot either box plots (TRUE) or dot plots (FALSE)

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
ggally_dotAndBox(
  tips,
  mapping = ggplot2::aes(x = total_bill, y = sex, color = sex),
  boxPlot = TRUE
)
ggally_dotAndBox(tips, mapping = ggplot2::aes(x = total_bill, y = sex, color = sex), boxPlot=FALSE)
```

---

ggally_facetbar *Plots the Bar Plots Faceted by Conditional Variable*

Description

X variables are plotted using geom_bar and faceted by the Y variable.

Usage

```
ggally_facetbar(data, mapping, ...)
```

Arguments

data	data set using
mapping	aesthetics being used
...	other arguements are sent to geom_bar

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
ggally_facetbar(tips, ggplot2::aes(x = sex, y = smoker, fill = time))
ggally_facetbar(tips, ggplot2::aes(x = smoker, y = sex, fill = time))
```

ggally_facetdensity *Plots the density plots by faceting*

Description

Make density plots by displaying subsets of the data in different panels.

Usage

```
ggally_facetdensity(data, mapping, ...)
```

Arguments

data	data set using
mapping	aesthetics being used
...	other arguments being sent to stat_density

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
ggally_facetdensity(tips, mapping = ggplot2::aes(x = total_bill, y = sex))
ggally_facetdensity(
  tips,
  mapping = ggplot2::aes_string(y = "total_bill", x = "sex", color = "sex")
)
```

ggally_facetdensitystrip *Plots a density plot with facets or a tile plot with facets*

Description

Make Tile Plot as densely as possible.

Usage

```
ggally_facetdensitystrip(data, mapping, ..., den_strip = FALSE)
```

Arguments

data	data set using
mapping	aesthetics being used
...	other arguments being sent to either geom_histogram or stat_density
den_strip	boolean to decide whether or not to plot a density strip(TRUE) or a facet density(FALSE) plot.

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
example(ggally_facetdensity)
example(ggally_denstrip)
```

---

ggally_facethist *Plots the Histograms by Faceting*

Description

Make histograms by displaying subsets of the data in different panels.

Usage

```
ggally_facethist(data, mapping, ...)
```

Arguments

data	data set using
mapping	aesthetics being used
...	parameters sent to stat_bin()

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
ggally_facethist(tips, mapping = ggplot2::aes(x = tip, y = sex))
ggally_facethist(tips, mapping = ggplot2::aes_string(x = "tip", y = "sex"), binwidth = 0.1)
```

ggally_na *NA plot*

Description

Draws a large NA in the middle of the plotting area. This plot is useful when all X or Y data is NA

Usage

```
ggally_na(data = NULL, mapping = NULL, size = 10, color = "grey20", ...)

ggally_naDiag(...)
```

Arguments

data	ignored
mapping	ignored
size	size of the geom_text 'NA'
color	color of the geom_text 'NA'
...	other arguments sent to geom_text

Author(s)

Barret Schloerke <schloerke@gmail.com>

ggally_points *Plots the Scatter Plot*

Description

Make a scatter plot with a given data set.

Usage

```
ggally_points(data, mapping, ...)
```

Arguments

data	data set using
mapping	aesthetics being used
...	other arguments are sent to geom_point

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(mtcars)
ggally_points(mtcars, mapping = ggplot2::aes(x = disp, y = hp))
ggally_points(mtcars, mapping = ggplot2::aes_string(x = "disp", y = "hp"))
ggally_points(
  mtcars,
  mapping = ggplot2::aes_string(
    x     = "disp",
    y     = "hp",
    color = "as.factor(cyl)",
    size  = "gear"
  )
)
```

`ggally_ratio` *Plots a mosaic plots*

Description

Plots the mosaic plot by using fluctuation.

Usage

```
ggally_ratio(data)
```

Arguments

`data`	data set using

Details

Must send only two discrete columns in the data set.

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
ggally_ratio(tips[, c("sex", "smoker")])
ggally_ratio(tips[, c("sex", "smoker")]) + ggplot2::coord_equal()
ggally_ratio(
  tips[, c("sex", "day")]
) + ggplot2::theme( aspect.ratio = 4/2)
```

ggally_smooth *Plots the Scatter Plot with Smoothing*

Description

Add a smoothed condition mean with a given scatter plot.

Usage

```
ggally_smooth(data, mapping, ...)
```

Arguments

data	data set using
mapping	aesthetics being used
...	other arguments to add to geom_point

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
ggally_smooth(tips, mapping = ggplot2::aes(x = total_bill, y = tip))
ggally_smooth(tips, mapping = ggplot2::aes_string(x = "total_bill", y = "tip"))
ggally_smooth(tips, mapping = ggplot2::aes_string(x = "total_bill", y = "tip", color = "sex"))
```

ggally_text *GGplot Text*

Description

Plot text for a plot.

Usage

```
ggally_text(label, mapping = ggplot2::aes(color = "black"), xP = 0.5,
  yP = 0.5, xrange = c(0, 1), yrange = c(0, 1), ...)
```

Arguments

label	text that you want to appear
mapping	aesthetics that don't relate to position (such as color)
xP	horizontal position percentage
yP	vertical position percentage
xrange	range of the data around it. Only nice to have if plotting in a matrix
yrange	range of the data around it. Only nice to have if plotting in a matrix
...	other arguments for geom_text

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
ggally_text("Example 1")
ggally_text("Example\nTwo", mapping = ggplot2::aes(size = 15), color = I("red"))
```

---

ggcorr *ggcorr - Plot a correlation matrix with ggplot2*

Description

Function for making a correlation matrix plot, using ggplot2. The function is directly inspired by Tian Zheng and Yu-Sung Su's corrplot function. Please visit http://github.com/briatte/ggcorr for the latest version of ggcorr, and see the vignette at https://briatte.github.io/ggcorr/ for many examples of how to use it.

Usage

```
ggcorr(data, method = c("pairwise", "pearson"), cor_matrix = NULL,
  nbreaks = NULL, digits = 2, name = "", low = "#3B9AB2",
  mid = "#EEEEEE", high = "#F21A00", midpoint = 0, palette = NULL,
  geom = "tile", min_size = 2, max_size = 6, label = FALSE,
  label_alpha = FALSE, label_color = "black", label_round = 1,
  label_size = 4, limits = TRUE, drop = !limits, layout.exp = 0,
  legend.position = "right", legend.size = 9, ...)
```

Arguments

data	a data frame or matrix containing numeric (continuous) data. If any of the columns contain non-numeric data, they will be dropped with a warning.

method	a vector of two character strings. The first value gives the method for computing covariances in the presence of missing values, and must be (an abbreviation of) one of "everything", "all.obs", "complete.obs", "na.or.complete" or "pairwise.complete.obs". The second value gives the type of correlation coefficient to compute, and must be one of "pearson", "kendall" or "spearman". See cor for details. Defaults to c("pairwise", "pearson").
cor_matrix	the named correlation matrix to use for calculations. Defaults to the correlation matrix of data when data is supplied.
nbreaks	the number of breaks to apply to the correlation coefficients, which results in a categorical color scale. See 'Note'. Defaults to NULL (no breaks, continuous scaling).
digits	the number of digits to show in the breaks of the correlation coefficients: see cut for details. Defaults to 2.
name	a character string for the legend that shows the colors of the correlation coefficients. Defaults to "" (no legend name).
low	the lower color of the gradient for continuous scaling of the correlation coefficients. Defaults to "#3B9AB2" (blue).
mid	the midpoint color of the gradient for continuous scaling of the correlation coefficients. Defaults to "#EEEEEE" (very light grey).
high	the upper color of the gradient for continuous scaling of the correlation coefficients. Defaults to "#F21A00" (red).
midpoint	the midpoint value for continuous scaling of the correlation coefficients. Defaults to 0.
palette	if nbreaks is used, a ColorBrewer palette to use instead of the colors specified by low, mid and high. Defaults to NULL.
geom	the geom object to use. Accepts either "tile", "circle", "text" or "blank".
min_size	when geom has been set to "circle", the minimum size of the circles. Defaults to 2.
max_size	when geom has been set to "circle", the maximum size of the circles. Defaults to 6.
label	whether to add correlation coefficients to the plot. Defaults to FALSE.
label_alpha	whether to make the correlation coefficients increasingly transparent as they come close to 0. Also accepts any numeric value between 0 and 1, in which case the level of transparency is set to that fixed value. Defaults to FALSE (no transparency).
label_color	the color of the correlation coefficients. Defaults to "grey75".
label_round	the decimal rounding of the correlation coefficients. Defaults to 1.
label_size	the size of the correlation coefficients. Defaults to 4.
limits	whether to bound the color scaling of the correlation coefficients between -1 and +1. Defaults to TRUE (recommended).
drop	if using nbreaks, whether to drop unused breaks from the color scale. Defaults to FALSE (recommended).

layout.exp	a multiplier to expand the horizontal axis to the left if variable names get clipped. Defaults to 0 (no expansion).
legend.position	where to put the legend of the correlation coefficients: see theme for details. Defaults to "bottom".
legend.size	the size of the legend title and labels, in points: see theme for details. Defaults to 9.
...	other arguments supplied to geom_text for the diagonal labels.

Note

Recommended values for the nbreaks argument are 3 to 11, as values above 11 are visually difficult to separate and are not supported by diverging ColorBrewer palettes.

Author(s)

Francois Briatte, with contributions from Amos B. Elberg and Barret Schloerke

See Also

cor and corrplot in the arm package.

Examples

```
# Basketball statistics provided by Nathan Yau at Flowing Data.
dt <- read.csv("http://datasets.flowingdata.com/ppg2008.csv")

# Default output.
ggcorr(dt[, -1])

# Labelled output, with coefficient transparency.
ggcorr(dt[, -1],
       label = TRUE,
       label_alpha = TRUE)

# Custom options.
ggcorr(
  dt[, -1],
  name = expression(rho),
  geom = "circle",
  max_size = 10,
  min_size = 2,
  size = 3,
  hjust = 0.75,
  nbreaks = 6,
  angle = -45,
  palette = "PuOr" # colorblind safe, photocopy-able
)

# Supply your own correlation matrix
ggcorr(
```

```
  data = NULL,
  cor_matrix = cor(dt[, -1], use = "pairwise")
)
```

---

ggfluctuation2 *Fluctuation plot*

Description

Create a fluctuation plot.

Usage

```
ggfluctuation2(table_data, floor = 0, ceiling = max(table_data$freq, na.rm =
  TRUE))
```

Arguments

table_data	a table of values, or a data frame with three columns, the last column being frequency
floor	don't display cells smaller than this value
ceiling	max value to compare to

Details

A fluctutation diagram is a graphical representation of a contingency table. This fuction currently only supports 2D contingency tables. The function was adopted from experiemntal functions within GGplot2 developed by Hadley Wickham.

Author(s)

Hadley Wickham <h.wickham@gmail.com>, Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
ggfluctuation2(table(tips$sex, tips$day))
ggfluctuation2(table(tips[, c("sex", "day")]))
```

ggmatrix *ggpairs - A ggplot2 Matrix*

Description

Make a generic matrix of ggplot2 plots

Usage

```
ggmatrix(plots, nrow, ncol, xAxisLabels = NULL, yAxisLabels = NULL,
  title = NULL, byrow = TRUE, showStrips = NULL,
  showAxisPlotLabels = TRUE, showXAxisPlotLabels = TRUE,
  showYAxisPlotLabels = TRUE, verbose = FALSE, data = NULL, gg = NULL,
  legends = FALSE)
```

Arguments

plots	list of plots to be put into matrix
nrow, ncol	number of rows and columns
xAxisLabels, yAxisLabels, title	labels for plot. Set the variable to NULL to not be displayed
byrow	boolean that determines whether the plots should be ordered by row or by column
showStrips	boolean to determine if each plot's strips should be displayed. NULL will default to the top and right side plots only. TRUE or FALSE will turn all strips on or off respectively.
showAxisPlotLabels, showXAxisPlotLabels, showYAxisPlotLabels	booleans that determine if the plots axis labels are printed on the X (bottom) or Y (left) part of the plot matrix. If showAxisPlotLabels is set, both showXAxisPlotLabels and showYAxisPlotLabels will be set to the given value.
verbose	boolean to determine the printing of "Plot #1, Plot #2...."
data	data set using. This is the data to be used in place of 'ggally_data' if the plot is a string to be evaluated at print time
gg	ggplot2 theme objects to be applied to every plot
legends	boolean to determine the printing of the legend in each plot. Not recommended.

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
plotList <- list()
for (i in 1:6) {
  plotList[[i]] <- ggally_text(paste("Plot #", i, sep = ""))
}
a <- ggmatrix(
  plotList,
  2, 3,
  c("A", "B", "C"),
  c("D", "E"),
  byrow = TRUE
)
#a

a <- ggmatrix(
  plotList,
  2, 3,
  xAxisLabels = c("A", "B", "C"),
  yAxisLabels = NULL,
  byrow = FALSE,
  showXAxisPlotLabels = FALSE
)
#a
```

---

ggnet *ggnet - Plot a network with ggplot2*

Description

Function for plotting network objects using ggplot2, now replaced by the ggnet2 function, which provides additional control over plotting parameters. Please visit http://github.com/briatte/ggnet for the latest version of ggnet2, and https://briatte.github.io/ggnet for a vignette that contains many examples and explanations.

Usage

```
ggnet(net, mode = "fruchtermanreingold", layout.par = NULL,
  layout.exp = 0, size = 9, alpha = 1, weight = "none",
  weight.legend = NA, weight.method = weight, weight.min = NA,
  weight.max = NA, weight.cut = FALSE, group = NULL, group.legend = NA,
  node.group = group, node.color = NULL, node.alpha = alpha,
  segment.alpha = alpha, segment.color = "grey50", segment.label = NULL,
  segment.size = 0.25, arrow.size = 0, arrow.gap = 0,
  arrow.type = "closed", label = FALSE, label.nodes = label,
  label.size = size/2, label.trim = FALSE, legend.size = 9,
  legend.position = "right", names = c("", ""), quantize.weights = FALSE,
  subset.threshold = 0, top8.nodes = FALSE, trim.labels = FALSE, ...)
```

Arguments

net	an object of class network, or any object that can be coerced to this class, such as an adjacency or incidence matrix, or an edge list: see edgeset.constructors and network for details. If the object is of class igraph and the intergraph package is installed, it will be used to convert the object: see asNetwork for details.
mode	a placement method from those provided in the sna package: see gplot.layout for details. Also accepts the names of two numeric vertex attributes of net, or a matrix of numeric coordinates, in which case the first two columns of the matrix are used. Defaults to the Fruchterman-Reingold force-directed algorithm.
layout.par	options to be passed to the placement method, as listed in gplot.layout. Defaults to NULL.
layout.exp	a multiplier to expand the horizontal axis if node labels get clipped: see expand_range for details. Defaults to 0 (no expansion).
size	size of the network nodes. If the nodes are weighted, their area is proportionally scaled up to the size set by size. Defaults to 9.
alpha	a level of transparency for nodes, vertices and arrows. Defaults to 1.
weight	the weighting method for the nodes, which might be a vertex attribute or a vector of size values. Also accepts "indegree", "outdegree", "degree" or "freeman" to size the nodes by their unweighted degree centrality ("degree" and "freeman" are equivalent): see degree for details. All node weights must be positive. Defaults to "none" (no weighting).
weight.legend	the name to assign to the legend created by weight. Defaults to NA (no name).
weight.method	see weight
weight.min	whether to subset the network to nodes with a minimum size, based on the values of weight. Defaults to NA (preserves all nodes).
weight.max	whether to subset the network to nodes with a maximum size, based on the values of weight. Defaults to NA (preserves all nodes).
weight.cut	whether to cut the size of the nodes into a certain number of quantiles. Accepts TRUE, which tries to cut the sizes into quartiles, or any positive numeric value, which tries to cut the sizes into that many quantiles. If the size of the nodes do not contain the specified number of distinct quantiles, the largest possible number is used. See quantile and cut for details. Defaults to FALSE (does nothing).
group	the groups of the nodes, either as a vector of values or as a vertex attribute. If set to mode on a bipartite network, the nodes will be grouped as "actor" if they belong to the primary mode and "event" if they belong to the secondary mode.
group.legend	the name to assign to the legend created by group.
node.group	see group
node.color	a vector of character strings to color the nodes with, holding as many colors as there are levels in node.group. Defaults to NULL, which will assign grayscale colors to each group.
node.alpha	transparency of the nodes. Inherits from alpha.

segment.alpha — the level of transparency of the edges. Defaults to alpha, which defaults to 1.

segment.color — the color of the edges, as a color value, a vector of color values, or as an edge attribute containing color values. Defaults to "grey50".

segment.label — the labels to plot at the middle of the edges, as a single value, a vector of values, or as an edge attribute. Defaults to NULL (no edge labels).

segment.size — the size of the edges, in points, as a single numeric value, a vector of values, or as an edge attribute. Defaults to 0.25.

arrow.size — the size of the arrows for directed network edges, in points. See arrow for details. Defaults to 0 (no arrows).

arrow.gap — a setting aimed at improving the display of edge arrows by plotting slightly shorter edges. Accepts any value between 0 and 1, where a value of 0.05 will generally achieve good results when the size of the nodes is reasonably small. Defaults to 0 (no shortening).

arrow.type — the type of the arrows for directed network edges. See arrow for details. Defaults to "closed".

label — whether to label the nodes. If set to TRUE, nodes are labeled with their vertex names. If set to a vector that contains as many elements as there are nodes in net, nodes are labeled with these. If set to any other vector of values, the nodes are labeled only when their vertex name matches one of these values. Defaults to FALSE (no labels).

label.nodes — see label

label.size — the size of the node labels, in points, as a numeric value, a vector of numeric values, or as a vertex attribute containing numeric values. Defaults to size / 2 (half the maximum node size), which defaults to 6.

label.trim — whether to apply some trimming to the node labels. Accepts any function that can process a character vector, or a strictly positive numeric value, in which case the labels are trimmed to a fixed-length substring of that length: see substr for details. Defaults to FALSE (does nothing).

legend.size — the size of the legend symbols and text, in points. Defaults to 9.

legend.position — the location of the plot legend(s). Accepts all legend.position values supported by theme. Defaults to "right".

names — deprecated: see group.legend and size.legend

quantize.weights — deprecated: see weight.cut

subset.threshold — deprecated: see weight.min

top8.nodes — deprecated: this functionality was experimental and has been removed entirely from ggnet

trim.labels — deprecated: see label.trim

... — other arguments passed to the geom_text object that sets the node labels: see geom_text for details.

Details

The degree centrality measures that can be produced through the `weight` argument will take the directedness of the network into account, but will be unweighted. To compute weighted network measures, see the `tnet` package by Tore Opsahl.

Author(s)

Moritz Marbach and Francois Briatte, with help from Heike Hoffmann, Pedro Jordano and Ming-Yu Liu

See Also

`ggnet2` in this package, `gplot` in the `sna` package, and `plot.network` in the `network` package

Examples

```
if (require(network)){

  # random adjacency matrix
  x           <- 10
  ndyads      <- x * (x - 1)
  density     <- x / ndyads
  m           <- matrix(0, nrow = x, ncol = x)
  dimnames(m) <- list(letters[ 1:x ], letters[ 1:x ])
  m[ row(m) != col(m) ] <- runif(ndyads) < density
  m

  # random undirected network
  n <- network::network(m, directed = FALSE)
  n

  ggnet(n, label = TRUE, alpha = 1, color = "white", segment.color = "black")

  # random groups
  g <- sample(letters[ 1:3 ], 10, replace = TRUE)

  # color palette
  p <- c("a" = "steelblue", "b" = "forestgreen", "c" = "tomato")

  ggnet(n, node.group = g, node.color = p, label = TRUE, color = "white")

  # edge arrows on a directed network
  ggnet(network(m, directed = TRUE), arrow.gap = 0.05, arrow.size = 10)

}
```

ggnet2 *ggnet2 - Plot a network with ggplot2*

Description

Function for plotting network objects using ggplot2, with additional control over graphical parameters that are not supported by the ggnet function. Please visit http://github.com/briatte/ggnet for the latest version of ggnet2, and https://briatte.github.io/ggnet for a vignette that contains many examples and explanations.

Usage

```
ggnet2(net, mode = "fruchtermanreingold", layout.par = NULL,
  layout.exp = 0, alpha = 1, color = "grey75", shape = 19, size = 9,
  max_size = 9, na.rm = NA, palette = NULL, alpha.palette = NULL,
  alpha.legend = NA, color.palette = palette, color.legend = NA,
  shape.palette = NULL, shape.legend = NA, size.palette = NULL,
  size.legend = NA, size.zero = FALSE, size.cut = FALSE, size.min = NA,
  size.max = NA, label = FALSE, label.alpha = 1, label.color = "black",
  label.size = max_size/2, label.trim = FALSE, node.alpha = alpha,
  node.color = color, node.label = label, node.shape = shape,
  node.size = size, edge.alpha = 1, edge.color = "grey50",
  edge.lty = "solid", edge.size = 0.25, edge.label = NULL,
  edge.label.alpha = 1, edge.label.color = label.color,
  edge.label.fill = "white", edge.label.size = max_size/2, arrow.size = 0,
  arrow.gap = 0, arrow.type = "closed", legend.size = 9,
  legend.position = "right", ...)
```

Arguments

net	an object of class network, or any object that can be coerced to this class, such as an adjacency or incidence matrix, or an edge list: see edgeset.constructors and network for details. If the object is of class igraph and the intergraph package is installed, it will be used to convert the object: see asNetwork for details.
mode	a placement method from those provided in the sna package: see gplot.layout for details. Also accepts the names of two numeric vertex attributes of net, or a matrix of numeric coordinates, in which case the first two columns of the matrix are used. Defaults to the Fruchterman-Reingold force-directed algorithm.
layout.par	options to be passed to the placement method, as listed in gplot.layout. Defaults to NULL.
layout.exp	a multiplier to expand the horizontal axis if node labels get clipped: see expand_range for details. Defaults to 0 (no expansion).
alpha	the level of transparency of the edges and nodes, which might be a single value, a vertex attribute, or a vector of values. Also accepts "mode" on bipartite networks (see 'Details'). Defaults to 1 (no transparency).

color	the color of the nodes, which might be a single value, a vertex attribute, or a vector of values. Also accepts "mode" on bipartite networks (see 'Details'). Defaults to grey75.
shape	the shape of the nodes, which might be a single value, a vertex attribute, or a vector of values. Also accepts "mode" on bipartite networks (see 'Details'). Defaults to 19 (solid circle).
size	the size of the nodes, in points, which might be a single value, a vertex attribute, or a vector of values. Also accepts "indegree", "outdegree", "degree" or "freeman" to size the nodes by their unweighted degree centrality ("degree" and "freeman" are equivalent): see degree for details. All node sizes must be strictly positive. Also accepts "mode" on bipartite networks (see 'Details'). Defaults to 9.
max_size	the *maximum* size of the node when size produces nodes of different sizes, in points. Defaults to 9.
na.rm	whether to subset the network to nodes that are *not* missing a given vertex attribute. If set to any vertex attribute of net, the nodes for which this attribute is NA will be removed. Defaults to NA (does nothing).
palette	the palette to color the nodes, when color is not a color value or a vector of color values. Accepts named vectors of color values, or if RColorBrewer is installed, any ColorBrewer palette name: see brewer.pal and http://colorbrewer2.org/ for details. Defaults to NULL, which will create an array of grayscale color values if color is not a color value or a vector of color values.
alpha.palette	the palette to control the transparency levels of the nodes set by alpha when the levels are not numeric values. Defaults to NULL, which will create an array of alpha transparency values if alpha is not a numeric value or a vector of numeric values.
alpha.legend	the name to assign to the legend created by alpha when its levels are not numeric values. Defaults to NA (no name).
color.palette	see palette
color.legend	the name to assign to the legend created by palette. Defaults to NA (no name).
shape.palette	the palette to control the shapes of the nodes set by shape when the shapes are not numeric values. Defaults to NULL, which will create an array of shape values if shape is not a numeric value or a vector of numeric values.
shape.legend	the name to assign to the legend created by shape when its levels are not numeric values. Defaults to NA (no name).
size.palette	the palette to control the sizes of the nodes set by size when the sizes are not numeric values.
size.legend	the name to assign to the legend created by size. Defaults to NA (no name).
size.zero	whether to accept zero-sized nodes based on the value(s) of size. Defaults to FALSE, which ensures that zero-sized nodes are still shown in the plot and its size legend.
size.cut	whether to cut the size of the nodes into a certain number of quantiles. Accepts TRUE, which tries to cut the sizes into quartiles, or any positive numeric value, which tries to cut the sizes into that many quantiles. If the size of the nodes

do not contain the specified number of distinct quantiles, the largest possible number is used. See quantile and cut for details. Defaults to FALSE (does nothing).

size.min whether to subset the network to nodes with a minimum size, based on the values of size. Defaults to NA (preserves all nodes).

size.max whether to subset the network to nodes with a maximum size, based on the values of size. Defaults to NA (preserves all nodes).

label whether to label the nodes. If set to TRUE, nodes are labeled with their vertex names. If set to a vector that contains as many elements as there are nodes in net, nodes are labeled with these. If set to any other vector of values, the nodes are labeled only when their vertex name matches one of these values. Defaults to FALSE (no labels).

label.alpha the level of transparency of the node labels, as a numeric value, a vector of numeric values, or as a vertex attribute containing numeric values. Defaults to 1 (no transparency).

label.color the color of the node labels, as a color value, a vector of color values, or as a vertex attribute containing color values. Defaults to "black".

label.size the size of the node labels, in points, as a numeric value, a vector of numeric values, or as a vertex attribute containing numeric values. Defaults to max_size / 2 (half the maximum node size), which defaults to 4.5.

label.trim whether to apply some trimming to the node labels. Accepts any function that can process a character vector, or a strictly positive numeric value, in which case the labels are trimmed to a fixed-length substring of that length: see substr for details. Defaults to FALSE (does nothing).

node.alpha see alpha

node.color see color

node.label see label

node.shape see shape

node.size see size

edge.alpha the level of transparency of the edges. Defaults to the value of alpha, which defaults to 1.

edge.color the color of the edges, as a color value, a vector of color values, or as an edge attribute containing color values. Defaults to "grey50".

edge.lty the linetype of the edges, as a linetype value, a vector of linetype values, or as an edge attribute containing linetype values. Defaults to "solid".

edge.size the size of the edges, in points, as a numeric value, a vector of numeric values, or as an edge attribute containing numeric values. All edge sizes must be strictly positive. Defaults to 0.25.

edge.label the labels to plot at the middle of the edges, as a single value, a vector of values, or as an edge attribute. Defaults to NULL (no edge labels).

edge.label.alpha
the level of transparency of the edge labels, as a numeric value, a vector of numeric values, or as an edge attribute containing numeric values. Defaults to 1 (no transparency).

edge.label.color
: the color of the edge labels, as a color value, a vector of color values, or as an edge attribute containing color values. Defaults to `label.color`, which defaults to `"black"`.

edge.label.fill
: the background color of the edge labels. Defaults to `"white"`.

edge.label.size
: the size of the edge labels, in points, as a numeric value, a vector of numeric values, or as an edge attribute containing numeric values. All edge label sizes must be strictly positive. Defaults to `max_size / 2` (half the maximum node size), which defaults to `4.5`.

arrow.size
: the size of the arrows for directed network edges, in points. See `arrow` for details. Defaults to `0` (no arrows).

arrow.gap
: a setting aimed at improving the display of edge arrows by plotting slightly shorter edges. Accepts any value between `0` and `1`, where a value of `0.05` will generally achieve good results when the size of the nodes is reasonably small. Defaults to `0` (no shortening).

arrow.type
: the type of the arrows for directed network edges. See `arrow` for details. Defaults to `"closed"`.

legend.size
: the size of the legend symbols and text, in points. Defaults to `9`.

legend.position
: the location of the plot legend(s). Accepts all `legend.position` values supported by `theme`. Defaults to `"right"`.

...
: other arguments passed to the `geom_text` object that sets the node labels: see `geom_text` for details.

Details

The degree centrality measures that can be produced through the `size` argument will take the directedness of the network into account, but will be unweighted. To compute weighted network measures, see the `tnet` package by Tore Opsahl.

The nodes of bipartite networks can be mapped to their mode by passing the `"mode"` argument to any of `alpha`, `color`, `shape` and `size`, in which case the nodes of the primary mode will be mapped as `"actor"`, and the nodes of the secondary mode will be mapped as `"event"`.

Author(s)

Moritz Marbach and Francois Briatte, with help from Heike Hoffmann, Pedro Jordano and Ming-Yu Liu

See Also

`ggnet` in this package, `gplot` in the `sna` package, and `plot.network` in the `network` package

Examples

```
if(require(network)) {

  # random adjacency matrix
  x           <- 10
  ndyads      <- x * (x - 1)
  density     <- x / ndyads
  m           <- matrix(0, nrow = x, ncol = x)
  dimnames(m) <- list(letters[ 1:x ], letters[ 1:x ])
  m[ row(m) != col(m) ] <- runif(ndyads) < density
  m

  # random undirected network
  n <- network::network(m, directed = FALSE)
  n

  ggnet2(n, label = TRUE)
  ggnet2(n, label = TRUE, shape = 15)
  ggnet2(n, label = TRUE, shape = 15, color = "black", label.color = "white")

  # add vertex attribute
  x = network.vertex.names(n)
  x = ifelse(x %in% c("a", "e", "i"), "vowel", "consonant")
  n %v% "phono" = x

  ggnet2(n, color = "phono")
  ggnet2(n, color = "phono", palette = c("vowel" = "gold", "consonant" = "grey"))
  ggnet2(n, shape = "phono", color = "phono")

  if (require(RColorBrewer)) {

    # random groups
    n %v% "group" <- sample(LETTERS[1:3], 10, replace = TRUE)

    ggnet2(n, color = "group", palette = "Set2")

  }

  # random weights
  n %e% "weight" <- sample(1:3, network.edgecount(n), replace = TRUE)
  ggnet2(n, edge.size = "weight", edge.label = "weight")

  # edge arrows on a directed network
  ggnet2(network(m, directed = TRUE), arrow.gap = 0.05, arrow.size = 10)

  # Padgett's Florentine wedding data
  data(flo, package = "network")
  flo

  ggnet2(flo, label = TRUE)
  ggnet2(flo, label = TRUE, label.trim = 4, vjust = -1, size = 3, color = 1)
  ggnet2(flo, label = TRUE, size = 12, color = "white")
```

```
}
```

---

ggnetworkmap *ggnetworkmap - Plot a network with ggplot2 suitable for overlay on a ggmap:: map ggplot, or other ggplot*

Description

This is a descendent of the original ggnet function. ggnet added the innovation of plotting the network geographically. However, ggnet needed to be the first object in the ggplot chain. ggnetworkmap does not. If passed a ggplot object as its first argument, such as output from ggmap, ggnetworkmap will plot on top of that chart, looking for vertex attributes lon and lat as coordinates. Otherwise, ggnetworkmap will generate coordinates using the Fruchterman-Reingold algorithm.

Usage

```
ggnetworkmap(gg, net, size = 3, alpha = 0.75, weight, node.group,
  node.color = NULL, node.alpha = NULL, ring.group, segment.alpha = NULL,
  segment.color = "grey", great.circles = FALSE, segment.size = 0.25,
  arrow.size = 0, label.nodes = FALSE, label.size = size/2, ...)
```

Arguments

gg	an object of class ggplot.
net	an object of class network, or any object that can be coerced to this class, such as an adjacency or incidence matrix, or an edge list: see edgeset.constructors and network for details. If the object is of class igraph and the intergraph package is installed, it will be used to convert the object: see asNetwork for details.
size	size of the network nodes. Defaults to 3. If the nodes are weighted, their area is proportionally scaled up to the size set by size.
alpha	a level of transparency for nodes, vertices and arrows. Defaults to 0.75.
weight	if present, the unquoted name of a vertex attribute in data. Otherwise nodes are unweighted.
node.group	NULL, the default, or the unquoted name of a vertex attribute that will be used to determine the color of each node.
node.color	If node.group is null, a character string specifying a color.
node.alpha	transparency of the nodes. Inherits from alpha.
ring.group	if not NULL, the default, the unquoted name of a vertex attribute that will be used to determine the color of each node border.
segment.alpha	transparency of the vertex links. Inherits from alpha
segment.color	color of the vertex links. Defaults to "grey".

great.circles	whether to draw edges as great circles using the geosphere package. Defaults to FALSE
segment.size	size of the vertex links, as a vector of values or as a single value. Defaults to 0.25.
arrow.size	size of the vertex arrows for directed network plotting, in centimeters. Defaults to 0.
label.nodes	label nodes with their vertex names attribute. If set to TRUE, all nodes are labelled. Also accepts a vector of character strings to match with vertex names.
label.size	size of the labels. Defaults to size / 2.
...	other arguments supplied to geom_text for the node labels. Arguments pertaining to the title or other items can be achieved through ggplot2 methods.

Details

This is a function for plotting graphs generated by network or igraph in a more flexible and elegant manner than permitted by ggnet. The function does not need to be the first plot in the ggplot chain, so the graph can be plotted on top of a map or other chart. Segments can be straight lines, or plotted as great circles. Note that the great circles feature can produce odd results with arrows and with vertices beyond the plot edges; this is a gglot2 limitation and cannot yet be fixed. Nodes can have two color schemes, which are then plotted as the center and ring around the node. The color schemes are selected by adding scale_fill_ or scale_color_ just like any other ggplot2 plot. If there are no rings, scale_color sets the color of the nodes. If there are rings, scale_color sets the color of the rings, and scale_fill sets the color of the centers. Note that additional arguments in the ... are passed to geom_text for plotting labels.

Author(s)

Amos Elberg <amos.elberg@gmail.com>. Original by Moritz Marbach <mmarbach@mail.uni-mannheim.de>, Francois Briatte <f.briatte@gmail.com>

Examples

```
if (require(ggplot2) && require(maps) && require(network) && require(sna)) {

  ## Example showing great circles on a simple map of the USA
  ## http://flowingdata.com/2011/05/11/how-to-map-connections-with-great-circles/

  airports <- read.csv("http://datasets.flowingdata.com/tuts/maparcs/airports.csv", header = TRUE)
  rownames(airports) <- airports$iata

  # select some random flights
  set.seed(1234)
  flights <- data.frame(
    origin = sample(airports[200:400, ]$iata, 200, replace = TRUE),
    destination = sample(airports[200:400, ]$iata, 200, replace = TRUE)
  )

  # convert to network
  flights <- network(flights, directed = TRUE)
```

```
  # add geographic coordinates
  flights %v% "lat" <- airports[ network.vertex.names(flights), "lat" ]
  flights %v% "lon" <- airports[ network.vertex.names(flights), "long" ]

  # drop isolated airports
  delete.vertices(flights, which(degree(flights) < 2))

  # compute degree centrality
  flights %v% "degree" <- degree(flights, gmode = "digraph")

  # add random groups
  flights %v% "mygroup" <- sample(letters[1:4], network.size(flights), replace = TRUE)

  # create a map of the USA
  usa <- ggplot(map_data("usa"), aes(x = long, y = lat)) +
    geom_polygon(aes(group = group), color = "grey65",
                 fill = "#f9f9f9", size = 0.2)

  # overlay network data to map
  ggnetworkmap(usa, flights, size = 4, great.circles = TRUE,
               node.group = mygroup, segment.color = "steelblue",
               ring.group = degree, weight = degree)

## Exploring a community of spambots found on Twitter
## Data by Amos Elberg: see ?twitter_spambots for details

data(twitter_spambots)

# create a world map
world <- fortify(map("world", plot = FALSE, fill = TRUE))
world <- ggplot(world, aes(x = long, y = lat)) +
  geom_polygon(aes(group = group), color = "grey65",
               fill = "#f9f9f9", size = 0.2)

# view global structure
ggnetworkmap(world, twitter_spambots)

# domestic distribution
ggnetworkmap(net = twitter_spambots) + xlim(-130,-60)

# topology
ggnetworkmap(net = twitter_spambots, arrow.size = 0.5)

# compute indegree and outdegree centrality
twitter_spambots %v% "indegree" <- degree(twitter_spambots, cmode = "indegree")
twitter_spambots %v% "outdegree" <- degree(twitter_spambots, cmode = "outdegree")

ggnetworkmap(net = twitter_spambots,
           arrow.size = 0.5,
           node.group = indegree,
           ring.group = outdegree, size = 4) +
  scale_fill_continuous("Indegree", high = "red", low = "yellow") +
```

```
  labs(color = "Outdegree")

# show some vertex attributes associated with each account
ggnetworkmap(net = twitter_spambots,
             arrow.size = 0.5,
             node.group = followers,
             ring.group = friends,
             size = 4,
             weight = indegree,
             label.nodes = TRUE, vjust = -1.5) +
  scale_fill_continuous("Followers", high = "red", low = "yellow") +
  labs(color = "Friends") +
  scale_color_continuous(low = "lightgreen", high = "darkgreen")
}
```

---

ggpairs *ggpairs - A ggplot2 generalized pairs plot*

Description

Make a matrix of plots with a given data set

Usage

```
ggpairs(data, mapping = NULL, columns = 1:ncol(data), title = "",
  upper = list(), lower = list(), diag = list(), params = NULL, ...,
  axisLabels = "show", columnLabels = colnames(data[, columns]),
  showStrips = NULL, legends = FALSE, verbose = FALSE)
```

Arguments

data	data set using. Can have both numerical and categorical data.
mapping	aesthetic mapping (besides x and y). See aes(). If mapping is numeric, columns will be set to the mapping value and mapping will be set to NULL.
columns	which columns are used to make plots. Defaults to all columns.
title	title for the graph
upper	see Details
lower	see Details
diag	see Details
params	deprecated. Please see wrap_fn_with_param_arg
...	other parameters being supplied to geom's aes, such as color
axisLabels	either "show" to display axisLabels, "internal" for labels in the diagonal plots, or "none" for no axis labels
columnLabels	label names to be displayed. Defaults to names of columns being used.

showStrips	boolean to determine if each plot's strips should be displayed. NULL will default to the top and right side plots only. TRUE or FALSE will turn all strips on or off respectively.
legends	boolean to determine the printing of the legend in each plot. Not recommended.
verbose	boolean to determine the printing of "Plot #1, Plot #2..."

Details

upper and lower are lists that may contain the variables 'continuous', 'combo', 'discrete', and 'na'. Each element of the list may be a function or a string. If a string is supplied, it must implement one of the following options:

continuous exactly one of ('points', 'smooth', 'density', 'cor', 'blank'). This option is used for continuous X and Y data.

combo exactly one of ('box', 'dot', 'facethist', 'facetdensity', 'denstrip', 'blank'). This option is used for either continuous X and categorical Y data or categorical X and continuous Y data.

discrete exactly one of ('facetbar', 'ratio', 'blank'). This option is used for categorical X and Y data.

na exactly one of ('na', 'blank'). This option is used when all X data is NA, all Y data is NA, or either all X or Y data is NA.

diag is a list that may only contain the variables 'continuous', 'discrete', and 'na'. Each element of the diag list is a string implementing the following options:

continuous exactly one of ('densityDiag', 'barDiag', 'blankDiag'). This option is used for continuous X data.

discrete exactly one of ('barDiag', 'blankDiag'). This option is used for categorical X and Y data.

na exactly one of ('naDiag', 'blankDiag'). This option is used when all X data is NA.

If 'blank' is ever chosen as an option, then ggpairs will produce an empty plot.

If a function is supplied to an upper, lower, or diag, it should implement the function api of `function(data, mapping, ...){#make ggplot2 plot}`. If a specific function needs its parameters set, `wrap()` the function with its parameters.

Value

ggpair object that if called, will print

Author(s)

Barret Schloerke <schloerke@gmail.com>, Jason Crowley <crowley.jason.s@gmail.com>, Di Cook <dicook@iastate.edu>, Heike Hofmann <hofmann@iastate.edu>, Hadley Wickham <h.wickham@gmail.com>

References

John W Emerson, Walton A Green, Barret Schloerke, Jason Crowley, Dianne Cook, Heike Hofmann, Hadley Wickham. The Generalized Pairs Plot. Journal of Computational and Graphical Statistics, vol. 22, no. 1, pp. 79-91, 2012.

See Also

wrap

Examples

```
# plotting is reduced to the first couple of examples.
# Feel free to print the ggpair objects created in the examples

data(tips, package = "reshape")
pm <- ggpairs(tips[, 1:3])
# pm
pm <- ggpairs(tips, 1:3, columnLabels = c("Total Bill", "Tip", "Sex"))
# pm
pm <- ggpairs(tips, upper = "blank")
# pm


# Custom Example
pm <- ggpairs(
  tips[, c(1, 3, 4, 2)],
  upper = list(continuous = "density", combo = "box"),
  lower = list(continuous = "points", combo = "dot")
)
# pm

# Use sample of the diamonds data
data(diamonds, package="ggplot2")
diamonds.samp <- diamonds[sample(1:dim(diamonds)[1], 200), ]

# Custom Example
pm <- ggpairs(
 diamonds.samp[, 1:5],
 mapping = ggplot2::aes(color = cut),
 upper = list(continuous = wrap("density", alpha = 0.5), combo = "box"),
 lower = list(continuous = wrap("points", alpha = 0.3), combo = wrap("dot", alpha = 0.4)),
 title = "Diamonds"
)
# pm

# Only Variable Labels on the diagonal (no axis labels)
pm <- ggpairs(tips[, 1:3], axisLabels="internal")
# pm
# Only Variable Labels on the outside (no axis labels)
pm <- ggpairs(tips[, 1:3], axisLabels="none")
# pm

# Custom Examples
custom_car <- ggpairs(mtcars[, c("mpg", "wt", "cyl")], upper = "blank", title = "Custom Example")
# ggplot example taken from example(geom_text)
  plot <- ggplot2::ggplot(mtcars, ggplot2::aes(x=wt, y=mpg, label=rownames(mtcars)))
  plot <- plot +
    ggplot2::geom_text(ggplot2::aes(colour=factor(cyl)), size = 3) +
```

```
    ggplot2::scale_colour_discrete(l=40)
custom_car[1, 2] <- plot
personal_plot <- ggally_text(
  "ggpairs allows you\nto put in your\nown plot.\nLike that one.\n <---"
)
custom_car[1, 3] <- personal_plot
# custom_car
```

---

ggparcoord *ggparcoord - A ggplot2 Parallel Coordinate Plot*

Description

A function for plotting static parallel coordinate plots, utilizing the ggplot2 graphics package.

Usage

```
ggparcoord(data, columns = 1:ncol(data), groupColumn = NULL,
  scale = "std", scaleSummary = "mean", centerObsID = 1,
  missing = "exclude", order = columns, showPoints = FALSE,
  splineFactor = FALSE, alphaLines = 1, boxplot = FALSE,
  shadeBox = NULL, mapping = NULL, title = "")
```

Arguments

data	the dataset to plot
columns	a vector of variables (either names or indices) to be axes in the plot
groupColumn	a single variable to group (color) by
scale	method used to scale the variables (see Details)
scaleSummary	if scale=="center", summary statistic to univariately center each variable by
centerObsID	if scale=="centerObs", row number of case plot should univariately be centered on
missing	method used to handle missing values (see Details)
order	method used to order the axes (see Details)
showPoints	logical operator indicating whether points should be plotted or not
splineFactor	logical or numeric operator indicating whether spline interpolation should be used. Numeric values will multiplied by the number of columns, TRUE will default to cubic interpolation, AsIs to set the knot count directly and 0, FALSE, or non-numeric values will not use spline interpolation.
alphaLines	value of alpha scaler for the lines of the parcoord plot or a column name of the data
boxplot	logical operator indicating whether or not boxplots should underlay the distribution of each variable

shadeBox	color of underlaying box which extends from the min to the max for each variable (no box is plotted if shadeBox == NULL)
mapping	aes string to pass to ggplot object
title	character string denoting the title of the plot

Details

scale is a character string that denotes how to scale the variables in the parallel coordinate plot. Options:

- std: univariately, subtract mean and divide by standard deviation
- robust: univariately, subtract median and divide by median absolute deviation
- uniminmax: univariately, scale so the minimum of the variable is zero, and the maximum is one
- globalminmax: no scaling is done; the range of the graphs is defined by the global minimum and the global maximum
- center: use uniminmax to standardize vertical height, then center each variable at a value specified by the scaleSummary param
- centerObs: use uniminmax to standardize vertical height, then center each variable at the value of the observation specified by the centerObsID param

missing is a character string that denotes how to handle missing missing values. Options:

- exclude: remove all cases with missing values
- mean: set missing values to the mean of the variable
- median: set missing values to the median of the variable
- min10: set missing values to 10% below the minimum of the variable
- random: set missing values to value of randomly chosen observation on that variable

order is either a vector of indices or a character string that denotes how to order the axes (variables) of the parallel coordinate plot. Options:

- (default): order by the vector denoted by columns
- (given vector): order by the vector specified
- anyClass: order variables by their separation between any one class and the rest (as opposed to their overall variation between classes). This is accomplished by calculating the F-statistic for each class vs. the rest, for each axis variable. The axis variables are then ordered (decreasing) by their maximum of k F-statistics, where k is the number of classes.
- allClass: order variables by their overall F statistic (decreasing) from an ANOVA with groupColumn as the explanatory variable (note: it is required to specify a groupColumn with this ordering method). Basically, this method orders the variables by their variation between classes (most to least).
- skewness: order variables by their sample skewness (most skewed to least skewed)
- Outlying: order by the scagnostic measure, Outlying, as calculated by the package scagnostics. Other scagnostic measures available to order by are Skewed, Clumpy, Sparse, Striated, Convex, Skinny, Stringy, and Monotonic. Note: To use these methods of ordering, you must have the scagnostics package loaded.

Value

ggplot object that if called, will print

Author(s)

Jason Crowley <crowley.jason.s@gmail.com>, Barret Schloerke <schloerke@gmail.com>, Di Cook <dicook@iastate.edu>, Heike Hofmann <hofmann@iastate.edu>, Hadley Wickham <h.wickham@gmail.com>

Examples

```
# use sample of the diamonds data for illustrative purposes
data(diamonds, package="ggplot2")
diamonds.samp <- diamonds[sample(1:dim(diamonds)[1], 100), ]

# basic parallel coordinate plot, using default settings
# ggparcoord(data = diamonds.samp, columns = c(1, 5:10))

# this time, color by diamond cut
gpd <- ggparcoord(data = diamonds.samp, columns = c(1, 5:10), groupColumn = 2)
# gpd

# underlay univariate boxplots, add title, use uniminmax scaling
gpd <- ggparcoord(data = diamonds.samp, columns = c(1, 5:10), groupColumn = 2,
  scale = "uniminmax", boxplot = TRUE, title = "Parallel Coord. Plot of Diamonds Data")
# gpd

# utilize ggplot2 aes to switch to thicker lines
gpd <- ggparcoord(data = diamonds.samp, columns = c(1, 5:10), groupColumn = 2,
  title="Parallel Coord. Plot of Diamonds Data", mapping = ggplot2::aes(size = 1))
# gpd

# basic parallel coord plot of the msleep data, using 'random' imputation and
# coloring by diet (can also use variable names in the columns and groupColumn
# arguments)
data(msleep, package="ggplot2")
gpd <- ggparcoord(data = msleep, columns = 6:11, groupColumn = "vore", missing =
  "random", scale = "uniminmax")
# gpd

# center each variable by its median, using the default missing value handler,
# 'exclude'
gpd <- ggparcoord(data = msleep, columns = 6:11, groupColumn = "vore", scale =
  "center", scaleSummary = "median")
# gpd

# with the iris data, order the axes by overall class (Species) separation using
# the anyClass option
gpd <- ggparcoord(data = iris, columns = 1:4, groupColumn = 5, order = "anyClass")
# gpd

# add points to the plot, add a title, and use an alpha scalar to make the lines
# transparent
```

```
gpd <- ggparcoord(data = iris, columns = 1:4, groupColumn = 5, order = "anyClass",
  showPoints = TRUE, title = "Parallel Coordinate Plot for the Iris Data",
  alphaLines = 0.3)
# gpd

# color according to a column
iris2 <- iris
iris2$alphaLevel <- c("setosa" = 0.2, "versicolor" = 0.3, "virginica" = 0)[iris2$Species]
gpd <- ggparcoord(data = iris2, columns = 1:4, groupColumn = 5, order = "anyClass",
  showPoints = TRUE, title = "Parallel Coordinate Plot for the Iris Data",
  alphaLines = "alphaLevel")
# gpd

## Use splines on values, rather than lines (all produce the same result)
columns <- c(1, 5:10)
gpd <- ggparcoord(diamonds.samp, columns, groupColumn = 2, splineFactor = TRUE)
# gpd
gpd <- ggparcoord(diamonds.samp, columns, groupColumn = 2, splineFactor = 3)
# gpd
splineFactor <- length(columns) * 3
gpd <- ggparcoord(diamonds.samp, columns, groupColumn = 2, splineFactor = I(splineFactor))
# gpd
```

---

ggscatmat *ggscatmat - a traditional scatterplot matrix for purely quantitative variables*

Description

This function makes a scatterplot matrix for quantitative variables with density plots on the diagonal and correlation printed in the upper triangle.

Usage

```
ggscatmat(data, columns = 1:ncol(data), color = NULL, alpha = 1)
```

Arguments

data	a data matrix. Should contain numerical (continuous) data.
columns	an option to choose the column to be used in the raw dataset. Defaults to 1:ncol(data).
color	an option to group the dataset by the factor variable and color them by different colors. Defaults to NULL.
alpha	an option to set the transparency in scatterplots for large data. Defaults to 1.

Author(s)

Mengjia Ni, Di Cook <dicook@monash.edu>

Examples

```
data(flea)
ggscatmat(flea, columns = 2:4)
ggscatmat(flea, columns = 2:4, color = "species")
```

---

ggsurv *Plot* survfit *objects using* ggplot2

Description

This function produces Kaplan-Meier plots using ggplot2. As a first argument it needs a survfit object, created by the survival package. Default settings differ for single stratum and multiple strata objects.

Usage

```
ggsurv(s, CI = "def", plot.cens = TRUE, surv.col = "gg.def",
  cens.col = "gg.def", lty.est = 1, lty.ci = 2, cens.shape = 3,
  back.white = FALSE, xlab = "Time", ylab = "Survival", main = "")
```

Arguments

s	an object of class survfit
CI	should a confidence interval be plotted? Defaults to TRUE for single stratum objects and FALSE for multiple strata objects.
plot.cens	mark the censored observations?
surv.col	colour of the survival estimate. Defaults to black for one stratum, and to the default ggplot2 colours for multiple strata. Length of vector with colour names should be either 1 or equal to the number of strata.
cens.col	colour of the points that mark censored observations.
lty.est	linetype of the survival curve(s). Vector length should be either 1 or equal to the number of strata.
lty.ci	linetype of the bounds that mark the 95% CI.
cens.shape	shape of the points that mark censored observations.
back.white	if TRUE the background will not be the default grey of ggplot2 but will be white with borders around the plot.
xlab	the label of the x-axis.
ylab	the label of the y-axis.
main	the plot label.

Value

An object of class ggplot

Author(s)

Edwin Thoen <edwinthoen@gmail.com>

Examples

```
if (require(survival) && require(scales)) {
  data(lung, package = "survival")
  sf.lung <- survival::survfit(Surv(time, status) ~ 1, data = lung)
  ggsurv(sf.lung)

  # Multiple strata examples
  sf.sex <- survival::survfit(Surv(time, status) ~ sex, data = lung)
  pl.sex <- ggsurv(sf.sex)
  pl.sex

  # Adjusting the legend of the ggsurv fit
  pl.sex +
    ggplot2::guides(linetype = FALSE) +
    ggplot2::scale_colour_discrete(
      name   = 'Sex',
      breaks = c(1,2),
      labels = c('Male', 'Female')
    )

  # We can still adjust the plot after fitting
  data(kidney, package = "survival")
  sf.kid <- survival::survfit(Surv(time, status) ~ disease, data = kidney)
  pl.kid <- ggsurv(sf.kid, plot.cens = FALSE)
  pl.kid

  # Zoom in to first 80 days
  pl.kid <- pl.kid + ggplot2::coord_cartesian(xlim = c(0, 80), ylim = c(0.45, 1))
  pl.kid

  # Add the diseases names to the plot and remove legend
  col <- scales::hue_pal(
    h         = c(0, 360) + 15,
    c         = 100,
    l         = 65,
    h.start   = 0,
    direction = 1
  )(4)
  pl.kid +
    ggplot2::annotate(
      "text",
      label  = c('AN', 'GN', 'Other', 'PKD'),
      x      = c(50, 20, 50, 71),
      y      = c(0.47, 0.55, 0.67, 0.8),
      size   = 5,
      colour = col
    ) +
```

```
    ggplot2::guides(color = FALSE, linetype = FALSE)
}
```

`glyphplot` *Glyph plot class*

Description

Glyph plot class

Usage

```
glyphplot(data, width, height, polar, x_major, y_major)

is.glyphplot(x)

## S3 method for class 'glyphplot'
x[...]

## S3 method for class 'glyphplot'
print(x, ...)
```

Arguments

`data`	A data frame containing variables named in `x_major`, `x_minor`, `y_major` and `y_minor`.
`height, width`	The height and width of each glyph. Defaults to 95% of the `resolution` of the data. Specify the width absolutely by supplying a numeric vector of length 1, or relative to the
`polar`	A logical of length 1, specifying whether the glyphs should be drawn in polar coordinates. Defaults to `FALSE`.
`x_major, y_major`	The name of the variable (as a string) for the major x and y axes. Together, the
`x`	glyphplot to be printed
`...`	ignored

Author(s)

Di Cook <dicook@monash.edu>, Heike Hofmann, Hadley Wickham

glyphs *Create the data needed to generate a glyph plot.*

Description

Create the data needed to generate a glyph plot.

Usage

```
glyphs(data, x_major, x_minor, y_major, y_minor, polar = FALSE,
  height = ggplot2::rel(0.95), width = ggplot2::rel(0.95),
  y_scale = identity, x_scale = identity)
```

Arguments

`data`	A data frame containing variables named in `x_major`, `x_minor`, `y_major` and `y_minor`.
`x_major, x_minor, y_major, y_minor`	The name of the variable (as a string) for the major and minor x and y axes. Together, each unique
`polar`	A logical of length 1, specifying whether the glyphs should be drawn in polar coordinates. Defaults to `FALSE`.
`height, width`	The height and width of each glyph. Defaults to 95% of the `resolution` of the data. Specify the width absolutely by supplying a numeric vector of length 1, or relative to the
`y_scale, x_scale`	The scaling function to be applied to each set of minor values within a grid cell. Defaults to `identity` so that no scaling is performed.

Author(s)

Di Cook <dicook@monash.edu>, Heike Hofmann, Hadley Wickham

Examples

```
data(nasa)
nasaLate <- nasa[
  nasa$date >= as.POSIXct("1998-01-01") &
  nasa$lat >= 20 &
  nasa$lat <= 40 &
  nasa$long >= -80 &
  nasa$long <= -60
, ]
temp.gly <- glyphs(nasaLate, "long", "day", "lat", "surftemp", height=2.5)
ggplot2::ggplot(temp.gly, ggplot2::aes(gx, gy, group = gid)) +
  add_ref_lines(temp.gly, color = "grey90") +
  add_ref_boxes(temp.gly, color = "grey90") +
  ggplot2::geom_path() +
```

```
ggplot2::theme_bw() +
ggplot2::labs(x = "", y = "")
```

---

happy *Data related to happiness from the General Social Survey, 1972-2006.*

Description

This data extract is taken from Hadley Wickham's `productplots` package. The original description follows, with minor edits.

Usage

```
data(happy)
```

Format

A data frame with 51020 rows and 10 variables

Details

The data is a small sample of variables related to happiness from the General Social Survey (GSS). The GSS is a yearly cross-sectional survey of Americans, run from 1972. We combine data for 25 years to yield 51,020 observations, and of the over 5,000 variables, we select nine related to happiness:

- age. age in years: 18–89.
- degree. highest education: lt high school, high school, junior college, bachelor, graduate.
- finrela. relative financial status: far above, above average, average, below average, far below.
- happy. happiness: very happy, pretty happy, not too happy.
- health. health: excellent, good, fair, poor.
- marital. marital status: married, never married, divorced, widowed, separated.
- sex. sex: female, male.
- wtsall. probability weight. 0.43–6.43.

References

Smith, Tom W., Peter V. Marsden, Michael Hout, Jibum Kim. *General Social Surveys, 1972-2006*. [machine-readable data file]. Principal Investigator, Tom W. Smith; Co-Principal Investigators, Peter V. Marsden and Michael Hout, NORC ed. Chicago: National Opinion Research Center, producer, 2005; Storrs, CT: The Roper Center for Public Opinion Research, University of Connecticut, distributor. 1 data file (57,061 logical records) and 1 codebook (3,422 pp).

lowertriangle *lowertriangle - rearrange dataset as the preparation of ggscatmat function*

Description

function for making the melted dataset used to plot the lowertriangle scatterplots.

Usage

```
lowertriangle(data, columns = 1:ncol(data), color = NULL)
```

Arguments

data	a data matrix. Should contain numerical (continuous) data.
columns	an option to choose the column to be used in the raw dataset. Defaults to 1:ncol(data)
color	an option to choose a factor variable to be grouped with. Defaults to (NULL)

Author(s)

Mengjia Ni, Di Cook <dicook@monash.edu>

Examples

```
data(flea)
head(lowertriangle(flea, columns= 2:4))
head(lowertriangle(flea))
head(lowertriangle(flea, color="species"))
```

nasa *Data from the Data Expo JSM 2006.*

Description

This data was provided by NASA for the competition.

Usage

```
data(nasa)
```

Format

A data frame with 41472 rows and 17 variables

Details

The data shows 6 years of monthly measurements of a 24x24 spatial grid from Central America:

- time integer specifying temporal order of measurements
- x, y, lat, long spatial location of measurements.
- cloudhigh, cloudlow, cloudmid, ozone, pressure, surftemp, temperature are the various satellite measurements.
- date, day, month, year specifying the time of measurements.
- id unique ide for each spatial position.

References

Murrell, P. (2010) The 2006 Data Expo of the American Statistical Association. Computational Statistics, 25:551-554.

---

print.ggmatrix *Print ggpair object*

Description

Specialized method to print the ggpair object-

Usage

```
## S3 method for class 'ggmatrix'
print(x, leftWidthProportion = 0.2,
  bottomHeightProportion = 0.1, spacingProportion = 0.03,
  gridNewPage = TRUE, ...)
```

Arguments

x	ggpair object to be plotted
leftWidthProportion	proportion of a plot area devoted to left axis labels
bottomHeightProportion	proportion of a plot area devoted to bottom axis labels
spacingProportion	proportion of a plot area devoted to the space between plots
gridNewPage	boolean that determines if a `grid.newpage()` should be executed before printing. Defaults to TRUE
...	ignored

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
data(tips, package = "reshape")
pMat <- ggpairs(tips, c(1,3,2), color = "sex")
pMat # calls print(pMat), which calls print.ggmatrix(pMat)

## defaults; (prints strips on top and right edges of matrix)
# print(pMat, left = 0.2, spacing = 0.03, bottom = 0.1)

## give the left axis labels area a proportion of 3 plot size
# print(pMat, leftWidthProportion = 3)

## give the bottom axis labels area a proportion of 1 plot size
# print(pMat, bottomHeightProportion = 1)

## give the spacing between plots a proportion of 1 plot size
# print(pMat, spacing = 1)
```

---

putPlot *Put Plot*

Description

Function to place your own plot in the layout.

Usage

```
putPlot(x, value, i, j)

## S3 replacement method for class 'ggmatrix'
x[i, j, ...] <- value
```

Arguments

x	ggally object to be altered
value	ggplot object to be placed
i	row from the top
j	column from the left
...	ignored

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```
custom_car <- ggpairs(mtcars[, c("mpg", "wt", "cyl")], upper = "blank", title = "Custom Example")
# ggplot example taken from example(geom_text)
  plot <- ggplot2::ggplot(mtcars, ggplot2::aes(x=wt, y=mpg, label=rownames(mtcars)))
  plot <- plot +
    ggplot2::geom_text(ggplot2::aes(colour=factor(cyl)), size = 3) +
    ggplot2::scale_colour_discrete(l=40)
custom_car[1, 2] <- plot
personal_plot <- ggally_text(
  "ggpairs allows you\nto put in your\nown plot.\nLike that one.\n <---"
)
custom_car[1, 3] <- personal_plot
# custom_car
```

`rescale01` *Rescaling functions*

Description

Rescaling functions

Usage

```
range01(x)

max1(x)

mean0(x)

min0(x)

rescale01(x, xlim = NULL)

rescale11(x, xlim = NULL)
```

Arguments

`x`	numeric vector
`xlim`	value used in `range`

scag_order *Find order of variables*

Description

Find order of variables based on a specified scagnostic measure by maximizing the index values of that measure along the path.

Usage

```
scag_order(scag, vars, measure)
```

Arguments

scag	scagnostics object
vars	character vector of the variables to be ordered
measure	scagnostics measure to order according to

Value

character vector of variable ordered according to the given scagnostic measure

Author(s)

Barret Schloerke

scatmat *scatmat - plot the lowertriangle plots and density plots of the scatter plot matrix.*

Description

function for making scatterplots in the lower triangle and diagonal density plots.

Usage

```
scatmat(data, columns = 1:ncol(data), color = NULL, alpha = 1)
```

Arguments

data	a data matrix. Should contain numerical (continuous) data.
columns	an option to choose the column to be used in the raw dataset. Defaults to 1:ncol(data)
color	an option to group the dataset by the factor variable and color them by different colors. Defaults to NULL
alpha	an option to set the transparency in scatterplots for large data. Defaults to 1.

Author(s)

Mengjia Ni, Di Cook <dicook@monash.edu>

Examples

```
data(flea)
scatmat(flea, columns=2:4)
scatmat(flea, columns= 2:4, color="species")
```

---

singleClassOrder *Order axis variables*

Description

Order axis variables by separation between one class and the rest (most separation to least).

Usage

```
singleClassOrder(classVar, axisVars, specClass = NULL)
```

Arguments

classVar	class variable (vector from original dataset)
axisVars	variables to be plotted as axes (data frame)
specClass	character string matching to level of classVar; instead of looking for separation between any class and the rest, will only look for separation between this class and the rest

Value

character vector of names of axisVars ordered such that the first variable has the most separation between one of the classes and the rest, and the last variable has the least (as measured by F-statistics from an ANOVA)

Author(s)

Jason Crowley <crowley.jason.s@gmail.com>

skewness *Sample skewness*

Description

Calculate the sample skewness of a vector while ignoring missing values.

Usage

```
skewness(x)
```

Arguments

x	numeric vector

Value

sample skewness of x

Author(s)

Jason Crowley <crowley.jason.s@gmail.com>

str.ggmatrix *ggmatrix structure*

Description

View the condensed version of the ggmatrix object. The attribute "class" is ALWAYS altered to "_class" to avoid recursion.

Usage

```
## S3 method for class 'ggmatrix'
str(object, ..., raw = FALSE)
```

Arguments

object	ggmatrix object to be viewed
...	passed on to the default str method
raw	boolean to determine if the plots should be converted to text or kept as original objects

twitter_spambots *Twitter spambots*

Description

A network of spambots found on Twitter as part of a data mining project.

Usage

```
data(twitter_spambots)
```

Format

An object of class `network` with 120 edges and 94 vertices.

Details

Each node of the network is identified by the Twitter screen name of the account and further carries five vertex attributes:

- location user's location, as provided by the user
- lat latitude, based on the user's location
- lon longitude, based on the user's location
- followers number of Twitter accounts that follow this account
- friends number of Twitter accounts followed by the account

Author(s)

Amos Elberg

uppertriangle *uppertriangle - rearrange dataset as the preparation of ggscatmat function*

Description

function for making the dataset used to plot the uppertriangle plots.

Usage

```
uppertriangle(data, columns = 1:ncol(data), color = NULL)
```

Arguments

data	a data matrix. Should contain numerical (continuous) data.
columns	an option to choose the column to be used in the raw dataset. Defaults to 1:ncol(data)
color	an option to choose a factor variable to be grouped with. Defaults to (NULL)

Author(s)

Mengjia Ni, Di Cook <dicook@monash.edu>

Examples

```
data(flea)
head(uppertriangle(flea, columns=2:4))
head(uppertriangle(flea))
head(uppertriangle(flea, color="species"))
```

---

wrap *Wrap a function with parameters*

Description

Wraps a function with the given parameters. This allows for very specific parameter arguements to be applied to each specific function.

Usage

```
wrap(funcVal, ..., funcArgName = substitute(funcVal))

wrap_fn_with_params(funcVal, ..., funcArgName = substitute(funcVal))

wrap_fn_with_param_arg(funcVal, params = NULL,
  funcArgName = substitute(funcVal))

wrapp(funcVal, params = NULL, funcArgName = substitute(funcVal))
```

Arguments

funcVal	function that the params will be applied to. The function should follow the api of function(data, mapping, ...){}
...	named parameters to be supplied to wrap_fn_with_param_arg
funcArgName	name of function to be displayed
params	named vector of parameters to be applied to the funcVal

Details

`wrap == wrap_fn_with_params`

`wrapp == wrap_fn_with_param_arg`

Value

a `function(data, mapping, ...){}` that will wrap the original function with the parameters applied as arguements

Examples

```
fn <- function(data, mapping, val = 2) {
  print(val)
}
fn(NULL, NULL) # 2
wrapped_fn <- wrap_fn_with_param_arg(fn, params = c(val = 5))
wrapped_fn(NULL, NULL) # 5
```

Index