Package ‘GGally’

May 18, 2018

Version 1.4.0
License GPL (>= 2.0)
Title Extension to `ggplot2`
Type Package
LazyLoad yes
LazyData true
BugReports https://github.com/ggobi/ggally/issues
Description The R package 'ggplot2' is a plotting system based on the grammar of graphics. 'GGally' extends 'ggplot2' by adding several functions to reduce the complexity of combining geometric objects with transformed data. Some of these functions include a pairwise plot matrix, a two group pairwise plot matrix, a parallel coordinates plot, a survival plot, and several functions to plot networks.
Depends R (>= 3.1), ggplot2 (> 2.2.0)
Imports grDevices, grid, gtable (>= 0.2.0), plyr (>= 1.8.3), progress, RColorBrewer, reshape (>= 0.8.5), utils, rlang
Suggests broom (>= 0.4.0), chemometrics, geosphere (>= 1.5-1), igraph (>= 1.0.1), intergraph (>= 2.0-2), maps (>= 3.1.0), mapproj, network (>= 1.12.0), scagnostics, scales (>= 0.4.0), sna (>= 2.3-2), survival, packagedocs (>= 0.4.0), rmarkdown, roxygen2, testthat, crosstalk
RoxygenNote 6.0.1
VignetteBuilder packagedocs
SystemRequirements openssl
NeedsCompilation no
Author Barret Schloerke [aut, cre] (author for ggpairs, ggduo, ggnostic, ggts, ggfacet, and ggally_*). Contributor for all functions.), Jason Crowley [aut] (ggparcoord), Di Cook [aut, ths] (ggscatmat, gglyp),
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),
Joseph Larmarange [aut] (ggcoef)

Maintainer Barret Schloerke <schloerke@gmail.com>
Repository CRAN
Date/Publication 2018-05-17 23:31:19 UTC

R topics documented:

+ .gg ................................. 4
add_ref_boxes ........................ 5
add_ref_lines ........................ 5
australia_PISA2012 ..................... 6
brew_colors .......................... 7
broomify ............................. 8
eval_data_col ........................ 8
find_plot_type ....................... 9
flea .................................. 9
fn_switch ............................ 10
getPlot .............................. 11
ggally_barDiag ....................... 12
ggally_blank ........................ 12
ggally_box ........................... 13
ggally_cor ........................... 14
ggally_density ....................... 15
ggally_densityDiag ................... 16
ggally_denstrip ...................... 16
ggally_diagAxis ...................... 17
ggally_dot ........................... 18
ggally_dot_and_box ................... 19
ggally_facetbar ...................... 19
ggally_facetdensity .................. 20
ggally_facetdensitystrip ............ 21
ggally_facethist ..................... 21
ggally_na ............................ 22
ggally_nostic_cooksd ................. 23
ggally_nostic_hat .................... 24
ggally_nostic_line ................... 25
ggally_nostic_resid ................. 26
ggally_nostic_se_fit ................. 27
ggally_nostic_sigma ................. 28
ggally_nostic_std_resid ............. 29
ggally_points ....................... 29
R topics documented:

- ggally_ratio .......................................................... 30
- ggally_smooth .......................................................... 31
- ggally_text ............................................................. 32
- ggcoef ................................................................. 33
- ggcord ................................................................. 34
- ggduo .............................................................. 36
- ggfacet .............................................................. 41
- gglegend ............................................................ 42
- ggmatrix ............................................................. 43
- ggmatrix_gtable ....................................................... 45
- ggmatrix_progress .................................................... 46
- gnet ................................................................. 47
- ggnet2 .............................................................. 50
- ggnetworkmap .......................................................... 55
- ggnostic ............................................................. 59
- ggpairs .............................................................. 61
- ggparcoord ........................................................... 65
- ggscaetmat ............................................................ 69
- ggsurv .............................................................. 70
- ggt ................................................................. 72
- glyphplot ............................................................ 72
- glyphs ............................................................... 73
- grab_legend ........................................................... 74
- happy ............................................................... 75
- is_horizontal .......................................................... 76
- lowertriangle .......................................................... 77
- mapping_string ........................................................ 77
- mapping_swap_x_y ..................................................... 78
- model_response_variables ........................................ 78
- nasa ................................................................. 79
- pigs ................................................................. 80
- print.ggmatrix ......................................................... 81
- print_if_interactive ................................................... 81
- psychademic ........................................................... 82
- putPlot .............................................................. 83
- rescale01 ............................................................ 84
- scag_order ............................................................ 84
- scatmat ............................................................. 85
- singleClassOrder ...................................................... 86
- skewness ............................................................. 86
- str.ggmatrix .......................................................... 87
- twitter_spambots ...................................................... 87
- uppertriangle ......................................................... 88
- vl_ggmatrix_theme ..................................................... 89
- wrap_fn_with_param_arg ............................................. 89

Index  ............................................................... 92
Description

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

Usage

```r
## 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

```r
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 a list of information
extra <- list(ggplot2::theme_bw(), ggplot2::labs(caption = "My caption!"))
pm + extra
```
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`
Programme for International Student Assessment (PISA) 2012 Data for Australia

Description

About PISA

Usage

data(australia_PISA2012)

Format

A data frame with 8247 rows and 32 variables

Details

The Programme for International Student Assessment (PISA) is a triennial international survey which aims to evaluate education systems worldwide by testing the skills and knowledge of 15-year-old students. To date, students representing more than 70 economies have participated in the assessment.

While 65 economies took part in the 2012 study, this data set only contains information from the country of Australia.

• gender: Factor w/ 2 levels "female","male": 1 1 2 2 2 1 1 1 1 1 2 1...
• age: Factor w/ 4 levels "4","5","6","7": 2 2 2 4 3 1 2 2 2 2...
• homework: num 5 5 9 3 2 3 4 3 5 1...
• desk: num 1 0 1 1 1 1 1 1 1 1 1...
• room: num 1 1 1 1 1 1 1 1 1 1 1...
• study: num 1 1 1 1 1 1 1 1 1 1 1...
• computer: num 1 1 1 1 1 1 1 1 1 1 1...
• software: num 1 1 1 1 1 1 1 1 1 1 1...
• internet: num 1 1 1 1 1 1 1 1 1 1 1...
• literature: num 0 0 1 0 1 1 1 1 0...
• poetry: num 0 0 1 0 1 1 0 1 1 1...
• art: num 1 0 1 0 1 1 0 1 1 1...
• textbook: num 1 1 1 1 1 0 1 1 1 1...
• dictionary: num 1 1 1 1 1 1 1 1 1 1...
• dishwasher: num 1 1 1 0 1 1 1 1 1...
• PV1MATH: num 562 565 602 520 613...
• PV2MATH: num 569 557 594 507 567...
brew_colors

- PV3MATH : num 555 553 552 501 585 ...
- PV4MATH : num 579 538 526 521 596 ...
- PV5MATH : num 548 573 619 547 603 ...
- PV1READ : num 582 617 650 554 605 ...
- PV2READ : num 571 572 608 560 557 ...
- PV3READ : num 602 560 594 517 627 ...
- PV4READ : num 572 564 575 564 597 ...
- PV5READ : num 585 565 620 572 598 ...
- PV1SCIE : num 583 627 668 574 639 ...
- PV2SCIE : num 579 600 665 612 635 ...
- PV3SCIE : num 593 574 620 571 666 ...
- PV4SCIE : num 567 582 592 598 700 ...
- PV5SCIE : num 587 625 656 662 670 ...
- SENWGT_STU : num 0.133 0.133 0.141 0.141 0.141 ...
- possessions: num 10 8 12 9 11 11 10 12 12 11 ...

Source


brew_colors

RCOLORBREWER Set1 colors

Description

RCOLORBREWER Set1 colors

Usage

brew_colors(col)

Arguments

col standard color name used to retrieve hex color value
broomify  

**Broomify a model**

**Description**

broom::augment a model and add broom::glance and broom::tidy output as attributes. X and Y variables are also added.

**Usage**

broomify(model, lmStars = TRUE)

**Arguments**

- **model**  
  model to be sent to broom::augment, broom::glance, and broom::tidy
- **lmStars**  
  boolean that determines if stars are added to labels

**Value**

broom::augmented data frame with the broom::glance data.frame and broom::tidy data.frame as 'broom_glance' and 'broom_tidy' attributes respectively. var_x and var_y variables are also added as attributes

**Examples**

data(mtcars)
model <- stats::lm(mpg ~ wt + qsec + am, data = mtcars)
broomified_model <- broomify(model)
str(broomified_model)

eval_data_col  

**Evaluate data column**

**Description**

Evaluate data column

**Usage**

eval_data_col(data, aes_col)

**Arguments**

- **data**  
  data set to evaluate the data with
- **aes_col**  
  Single value from an ggplot2::aes(...) object
Value

Aes mapping with the x and y values switched

Examples

```r
mapping <- ggplot2::aes(Petal.Length) eval_data_col(iris, mapping$x)
```

---

**find_plot_type**

*Find Plot Types*

---

**Description**

Retrieves the type of plot for the specific columns

**Usage**

```r
find_plot_type(col1Name, col2Name, type1, type2, isAllNa, allowDiag)
```

**Arguments**

- `col1Name`: x column name
- `col2Name`: y column name
- `type1`: x column type
- `type2`: y column type
- `isAllNa`: `is.na(data)`
- `allowDiag`: allow for diag values to be returned

**Author(s)**

Barret Schloerke <schloerke@gmail.com>

---

**flea**

*Historical data used for classification examples.***

**Description**

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

**Usage**

```r
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


---

**fn_switch**  
*Function switch*

**Description**  
Function that allows you to call different functions based upon an aesthetic variable value.

**Usage**

```
fn_switch(types, mapping_val = "y")
```

**Arguments**

- **types**  
  list of functions that follow the ggmatrix function standard: `function(data, mapping, ...){ #make ggplot2 object}`.  
  One key should be a 'default' key for a default switch case.

- **mapping_val**  
  mapping value to switch on. Defaults to the 'y' variable of the aesthetics list.

**Examples**

```
gnostic_continuous_fn <- fn_switch(list(  
default = ggally_points,  
.fitted = ggally_points,  
.se.fit = ggally_nostic_se_fit,  
.resid = ggally_nostic_resid,  
.hat = ggally_nostic_hat,  
.sigma = ggally_nostic_sigma,  
.cooksd = ggally_nostic_cooksd,  
.std.resid = ggally_nostic_std_resid)
```
getPlot

))

  ggNnosticcombo_fn <- fn_switch(list(
    default = ggally_box_no_facet,
    fitted = ggally_box_no_facet,
    se.fit = ggally_nostic_se_fit,
    resid = ggally_nostic_resid,
    hat = ggally_nostic_hat,
    sigma = ggally_nostic_sigma,
    cooksd = ggally_nostic_cooksd,
    std.resid = ggally_nostic_std_resid
  ))

getPlot

Description

Retrieves the ggplot object at the desired location.

Usage

getPlot(pm, i, j)

## S3 method for class 'ggmatrix'

pm[i, j, ...]

Arguments

pm    ggmatrix 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**
```r
ggally_barDiag(data, mapping, ..., rescale = FALSE)
```

**Arguments**
- `data` data set using
- `mapping` aesthetics being used
- `...` other arguments are sent to `geom_bar`
- `rescale` boolean to decide whether or not to rescale the count output. Only applies to numeric data

**Author(s)**
Barret Schloerke <schloerke@gmail.com>

**Examples**
```r
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**
```r
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>

Description

Make a box plot with a given data set. ggally_box_no_facet will be a single panel plot, while ggally_box will be a faceted plot

Usage

ggally_box(data, mapping, ...)

ggally_box_no_facet(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
)
Description

Estimate correlation from the given data.

Usage

```r
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 supplied 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

```r
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
)
```
Description

Make a scatter density plot from a given data.

Usage

```r
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

```r
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**

```r
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**

```r
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**

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

**Arguments**

- `data`: data set using
- `mapping`: aesthetics being used.
- `...`: other arguments being sent to `stat_bin`
ggally_diagAxis

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))
Description

Add jittering with the box plot. `ggally_dot_no_facet` will be a single panel plot, while `ggally_dot` will be a faceted plot.

Usage

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

ggally_dot_no_facet(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

```r
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_dot_and_box**  
Plots either Box Plot or Dot Plots

**Description**

Place box plots or dot plots on the graph

**Usage**

`ggally_dot_and_box(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**

```r
data(tips, package = "reshape")
ggally_dot_and_box(
tips,
mapping = ggplot2::aes(x = total_bill, y = sex, color = sex),
boxPlot = TRUE
)
ggally_dot_and_box(
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, ...)`
ggally_facetdensity

Arguments

- `data`: data set using
- `mapping`: aesthetics being used
- `...`: other arguments are sent to geom_bar

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```r
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

```r
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

```r
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**

```r
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**

```r
e example(ggally_facetdensity)
e example(ggally_denstrip)
```

**ggally_facethist**

*Plots the Histograms by Faceting*

**Description**

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

**Usage**

```r
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_nostic_cooksd**  
**ggnostic - Cook's distance**

**Description**

A function to display `stats::cooks.distance`.

**Usage**

```r
ggally_nostic_cooksd(data, mapping, ..., linePosition = pf(0.5, 
  length(attr(data, "var_x")), nrow(data) - length(attr(data, "var_x"))),
  lineColor = brew_colors("grey"), lineType = 2)
```

**Arguments**

- `data`, `mapping`, `...`, `lineColor`, `lineType`
  - Parameters supplied to `ggally_nostic_line`
  - `linePosition`: `4 / n` is the general cutoff point for Cook's Distance

**Details**

A line is added at `F_p, n - p(0.5)` to display the general cutoff point for Cook's Distance.


**Value**

`ggplot2` plot object

**See Also**

`stats::cooks.distance`

**Examples**

```r
dt <- broomify(stats::lm(mpg ~ wt + qsec + am, data = mtcars))
ggally_nostic_cooksd(dt, ggplot2::aes(wt, .cooksD))
```
Description

A function to display stats::influence's hat information against a given explanatory variable.

Usage

```r
ggally_nostic_hat(data, mapping, ..., linePosition = 2 * sum(eval_data_col(data, mapping$y))/nrow(data),
lineColor = brew_colors("grey"), lineSize = 0.5, lineAlpha = 1,
lineType = 2, avgLinePosition = sum(eval_data_col(data, mapping$y))/nrow(data), avgLineColor = brew_colors("grey"),
avgLineSize = lineSize, avgLineAlpha = lineAlpha, avgLineType = 1)
```

Arguments

data, mapping, ...

supplied directly to `ggally_nostic_line`

linePosition, lineColor, lineSize, lineAlpha, lineType

parameters supplied to `ggplot2::geom_line` for the cutoff line

avgLinePosition, avgLineColor, avgLineSize, avgLineAlpha, avgLineType

parameters supplied to `ggplot2::geom_line` for the average line

Details

As stated in `stats::influence` documentation:

hat: a vector containing the diagonal of the 'hat' matrix.

The diagonal elements of the 'hat' matrix describe the influence each response value has on the fitted value for that same observation.

A suggested "cutoff" line is added to the plot at a height of 2 * p / n and an expected line at a height of p / n. If either linePosition or avgLinePosition is NULL, the respective line will not be drawn.

Value

ggplot2 plot object

See Also

`stats::influence`

Examples

dt <- broomify(stats::lm(mpg ~ wt + qsec + am, data = mtcars))
ggally_nostic_hat(dt, ggplot2::aes(wt, .hat))
Description

If a non-null linePosition value is given, a line will be drawn before the given continuous_geom or combo_geom is added to the plot.

Usage

ggally_nostic_line(data, mapping, ..., linePosition = NULL, lineColor = "red", lineSize = 0.5, lineAlpha = 1, lineType = 1, continuous_geom = ggplot2::geom_point, combo_geom = ggplot2::geom_boxplot, mapColorToFill = TRUE)

Arguments

data, mapping supplied directly to ggplot2::ggplot(data, mapping)
... parameters supplied to continuous_geom or combo_geom
linePosition, lineColor, lineSize, lineAlpha, lineType parameters supplied to ggplot2::geom_line
continuous_geom ggplot2 geom that is executed after the line is (possibly) added and if the x data is continuous
combo_geom ggplot2 geom that is executed after the line is (possibly) added and if the x data is discrete
mapColorToFill boolean to determine if combo plots should cut the color mapping to the fill mapping

Details

Functions with a color in their name have different default color behavior.

Value

ggplot2 plot object
**Description**

If non-null pVal and sigma values are given, confidence interval lines will be added to the plot at the specified pVal percentiles of a N(0, sigma) distribution.

**Usage**

```r
ggally_nostic_resid(data, mapping, ..., linePosition = 0, 
  lineColor = brew_colors("grey"), lineSize = 0.5, lineAlpha = 1, 
  linType = 1, lineConfColor = brew_colors("grey"), 
  lineConfSize = lineSize, lineConfAlpha = lineAlpha, lineConfType = 2, 
  pVal = c(0.025, 0.975), sigma = attr(data, "broom.glance")$sigma, 
  se = TRUE, method = "auto")
```

**Arguments**

- `data`, `mapping`, ...
  - parameters supplied to `ggally_nostic_line`
- `linePosition`, `lineColor`, `lineSize`, `lineAlpha`, `linType`
  - parameters supplied to `ggplot2::geom_line`
- `lineConfColor`, `lineConfSize`, `lineConfAlpha`, `lineConfType`
  - parameters supplied to the confidence interval lines
- `pVal`  
  - percentiles of a N(0, sigma) distribution to be drawn
- `sigma`  
  - sigma value for the pVal percentiles
- `se`  
  - boolean to determine if the confidence intervals should be displayed
- `method`  
  - parameter supplied to `ggplot2::geom_smooth`. Defaults to "auto"

**Value**

ggplot2 plot object

**See Also**

- `stats::residuals`

**Examples**

```r
dt <- broomify(stats::lm(mpg ~ wt + qsec + am, data = mtcars))
ggally_nostic_resid(dt, ggplot2::aes(wt, .resid))
```
Description

A function to display stats::predict's standard errors

Usage

ggally_nostic_se_fit(data, mapping, ..., lineColor = brew_colors("grey"),
  linePosition = NULL)

Arguments

data, mapping, ...,
  lineColor
    parameters supplied to ggally_nostic_line
  linePosition
    base comparison for a perfect fit

Details

As stated in stats::predict documentation:

If the logical 'se.fit' is 'TRUE', standard errors of the predictions are calculated. If the numeric argument 'scale' is set (with optional "df"), it is used as the residual standard deviation in the computation of the standard errors, otherwise this is extracted from the model fit.

Since the se.fit is TRUE and scale is unset by default, the standard errors are extracted from the model fit.

A base line of 0 is added to give reference to a perfect fit.

Value

ggplot2 plot object

See Also

stats::influence

Examples

dt <- broomify(stats::lm(mpg ~ wt + qsec + am, data = mtcars))
ggally_nostic_se_fit(dt, ggplot2::aes(wt, .se.fit))
Description

A function to display stats::influence’s sigma value.

Usage

ggally_nostic_sigma(data, mapping, ..., lineColor = brew_colors("grey"),
   linePosition = attr(data, "broom_glance")$sigma)

Arguments

data, mapping, ..., lineColor
parameters supplied to ggally_nostic_line

toolsPosition line that is drawn in the background of the plot. Defaults to the overall model’s

sigma value.

Details

As stated in stats::influence documentation:

sigma: a vector whose i-th element contains the estimate of the residual standard deviation obtained
when the i-th case is dropped from the regression. (The approximations needed for GLMs can result
in this being 'NaN'.)

A line is added to display the overall model’s sigma value. This gives a baseline for comparison

Value

ggplot2 plot object

See Also

stats::influence

Examples

dt <- broomify(stats::lm(mpg ~ wt + qsec + am, data = mtcars))
ggally_nostic_sigma(dt, ggplot2::aes(wt, .sigma))
**ggally_nostic_std_resid**

*ggnostic - standardized residuals*

---

**Description**

If non-null pVal and sigma values are given, confidence interval lines will be added to the plot at the specified pVal locations of a N(0, 1) distribution.

**Usage**

```r
ggally_nostic_std_resid(data, mapping, ..., sigma = 1)
```

**Arguments**

- `data`, `mapping`, `...`
  - parameters supplied to `ggally_nostic_resid`

- `sigma`
  - sigma value for the pVal percentiles. Set to 1 for standardized residuals

**Value**

ggplot2 plot object

**See Also**

- `stats::rstandard`

**Examples**

```r
dt <- broomify(stats::lm(mpg ~ wt + qsec + am, data = mtcars))
ggally_nostic_std_resid(dt, ggplot2::aes(x = wt, y = .std.resid))
```

---

**ggally_points**

*Plots the Scatter Plot*

---

**Description**

Make a scatter plot with a given data set.

**Usage**

```r
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**

```r
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 plot

**Description**

Plots the mosaic plot by using fluctuation.

**Usage**

```r
ggally_ratio(data, mapping = do.call(ggplot2::aes_string,  
as.list(colnames(data)[1:2])), ... , floor = 0, ceiling = NULL)
```

**Arguments**

- **data**: data set using
- **mapping**: aesthetics being used. Only x and y will used and both are required
- **...**: passed to `geom_tile(...)`
- **floor**: don’t display cells smaller than this value
- **ceiling**: max value to scale frequencies. If any frequency is larger than the ceiling, the fill color is displayed darker than other rectangles
ggally_smooth

Author(s)
Barret Schloerke <schloerke@gmail.com>

Examples

```r
data(tips, package = "reshape")
ggally_ratio(tips, ggplot2::aes(sex, day))
ggally_ratio(tips, ggplot2::aes(sex, day)) + ggplot2::coord_equal()
  # only plot tiles greater or equal to 20 and scale to a max of 50
  ggally_ratio(
    tips, ggplot2::aes(sex, day),
    floor = 20, ceiling = 50
  ) + ggplot2::theme(aspect.ratio = 4/2)
```

Description

Add a smoothed condition mean with a given scatter plot.

Usage

```r
ggally_smooth(data, mapping, ..., method = "lm", se = TRUE, shrink = TRUE)

ggally_smooth_loess(data, mapping, ...)

ggally_smooth_lm(data, mapping, ...)
```

Arguments

data data set using
mapping aesthetics being used
... other arguments to add to geom_point
method, se parameters supplied to `geom_smooth`
shrink boolean to determine if y range is reduced to range of points or points and error ribbon

Details

Y limits are reduced to match original Y range with the goal of keeping the Y axis the same across plots.

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"))

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 Two", mapping = ggplot2::aes(size = 15), color = I("red"))
ggcoef - Plot Model Coefficients with broom and ggplot2

Description
Plot the coefficients of a model with broom and ggplot2.

Usage
```r
ggcoef(x, mapping = aes_string(y = "term", x = "estimate"), conf.int = TRUE, conf.level = 0.95, exponentiate = FALSE, exclude_intercept = FALSE, vline = TRUE, vline_intercept = "auto", vline_color = "gray50", vline_linetype = "dotted", vline_size = 1, errorbar_color = "gray25", errorbar_height = 0, errorbar_linetype = "solid", errorbar_size = 0.5, sort = c("none", "ascending", "decending"), ...)
```

Arguments
- `x` a model object to be tidied with tidy or a data frame (see Details)
- `mapping` default aesthetic mapping
- `conf.int` display confidence intervals as error bars?
- `conf.level` level of confidence intervals (passed to tidy if x is not a data frame)
- `exponentiate` if TRUE, x-axis will be logarithmic (also passed to tidy if x is not a data frame)
- `exclude_intercept` should the intercept be excluded from the plot?
- `vline` print a vertical line?
- `vline_intercept` xintercept for the vertical line. "auto" for x = 0 (or x = 1 if exponentiate is TRUE)
- `vline_color` color of the vertical line
- `vline_linetype` line type of the vertical line
- `vline_size` size of the vertical line
- `errorbar_color` color of the error bars
- `errorbar_height` height of the error bars
- `errorbar_linetype` line type of the error bars
- `errorbar_size` size of the error bars
- `sort` "none" (default) do not sort, "ascending" sort by increasing coefficient value, or "decending" sort by decreasing coefficient value
- `...` additional arguments sent to geom_point
Examples

```r
library(broom)
reg <- lm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width, data = iris)
ggcoef(reg)
d <- as.data.frame(Titanic)
reg2 <- glm(Survived ~ Sex + Age + Class, family = binomial, data = d, weights = d$Freq)
ggcoef(reg2, exponentiate = TRUE)
ggcoef(
  reg2, exponentiate = TRUE, exclude_intercept = TRUE,
  errorbar_height = .2, color = "blue", sort = "ascending"
)
```

---

**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 in the 'arm' package. Please visit [http://github.com/briatte/ggcorr](http://github.com/briatte/ggcorr) for the latest version of ggcorr, and see the vignette at [https://briatte.github.io/ggcorr/](https://briatte.github.io/ggcorr/) for many examples of how to use it.

Usage

```r
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 = c(-1, 1), drop = is.null(limits) ||
       identical(limits, FALSE), 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.
nbbreaks 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 "#E6E6E6" (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 nbbreaks 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 bounding of color scaling for correlations, set limits = NULL or FALSE to remove

drop if using nbbreaks, 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 \texttt{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 \texttt{arm} package.

Examples

\begin{verbatim}
# Basketball statistics provided by Nathan Yau at Flowing Data.

# 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")
)
\end{verbatim}
**Description**

Make a matrix of plots with a given data set with two different column sets

**Usage**

```r
ggduo(data, mapping = NULL, columnsX = 1:ncol(data), columnsY = 1:ncol(data), title = NULL, types = list(continuous = "smooth_loess", comboVertical = "box_no_facet", comboHorizontal = "facethist", discrete = "ratio"), axisLabels = c("show", "none"), columnLabelsX = colnames(data[columnsX]), columnLabelsY = colnames(data[columnsY]), labeller = "label_value", switch = NULL, xlab = NULL, ylab = NULL, showStrips = NULL, legend = NULL, cardinality_threshold = 15, progress = NULL, legends = stop("deprecated"))
```

**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.
- `columnsX`, `columnsY` which columns are used to make plots. Defaults to all columns.
- `title, xlab, ylab` title, x label, and y label for the graph
- `types` see Details
- `axisLabels` either "show" to display axisLabels or "none" for no axis labels
- `columnLabelsX, columnLabelsY` label names to be displayed. Defaults to names of columns being used.
- `labeller` labeller for facets. See `labellers`. Common values are "label_value" (default) and "label_parsed".
- `switch` switch parameter for `facet_grid`. See `ggplot2::facet_grid`. By default, the labels are displayed on the top and right of the plot. If "x", the top labels will be displayed to the bottom. If "y", the right-hand side labels will be displayed to the left. Can also be set to "both".
- `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.
- `legend` May be the two objects described below or the default NULL value. The legend position can be moved by using `ggplot2`’s theme element `theme(legend.position = "bottom")` 
  - **a numeric vector of length 2** provides the location of the plot to use the legend for the plot matrix’s legend. Such as `legend = c(3,5)` which will use the legend from the plot in the third row and fifth column
  - **a single numeric value** provides the location of a plot according to the display order. Such as `legend = 3` in a plot matrix with 2 rows and 5 columns displayed by column will return the plot in position c(1,2)
a object from grab_legend() a predetermined plot legend that will be displayed directly

cardinality_threshold
maximum number of levels allowed in a character / factor column. Set this value to NULL to not check factor columns. Defaults to 15

progress NULL (default) for a progress bar in interactive sessions with more than 15 plots, TRUE for a progress bar, FALSE for no progress bar, or a function that accepts at least a plot matrix and returns a new progress::progress_bar. See ggmatrix_progress.

legends deprecated

Details
types is a list 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', 'smooth_loess', 'density', 'cor', 'blank'). This option is used for continuous X and Y data.
comboHorizontal exactly one of ('box', 'box_no_facet', 'dot', 'dot_no_facet', 'facethist', 'facetdensity', 'denstrip', 'blank'). This option is used for either continuous X and categorical Y data or categorical X and continuous Y data.
comboVertical exactly one of ('box', 'box_no_facet', 'dot', 'dot_no_facet', '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.

If 'blank' is ever chosen as an option, then ggduo will produce an empty plot.
If a function is supplied as an option, it should implement the function api of function(data, mapping, ...){# make ggplot}
If a specific function needs its parameters set, wrap(fn, param1 = val1, param2 = val2) the function with its parameters.

Examples

# small function to display plots only if it's interactive
p_ <- GGally::print_if_interactive

data(baseball, package = "plyr")

# Keep players from 1990-1995 with at least one at bat
# Add how many singles a player hit
# (must do in two steps as X1b is used in calculations)
dt <- transform(
  subset(baseball, year >= 1990 & year <= 1995 & ab > 0),
  X1b = h - X2b - X3b - hr
)
# Add
# the player’s batting average,
# the player’s slugging percentage,
# and the player’s on base percentage
# Make factor a year, as each season is discrete
dt <- transform(
  dt,
  batting_avg = h / ab,
  slug = (X1b + 2*X2b + 3*X3b + 4*hr) / ab,
  on_base = (h + bb + hbp) / (ab + bb + hbp),
  year = as.factor(year)
)

pm <- ggduo(
  dt,
  c("year", "g", "ab", "lg"),
  c("batting_avg", "slug", "on_base"),
  mapping = ggplot2::aes(color = lg)
)

# Prints, but
# there is severe over plotting in the continuous plots
# the labels could be better
# want to add more hitting information
p_(pm)

pm <- ggduo(
  dt,
  c("year", "g", "ab", "lg"),
  c("batting_avg", "slug", "on_base"),
  columnLabelsX = c("year", "player game count", "player at bat count", "league"),
  columnLabelsY = c("batting avg", "slug ", "on base ",
  title = "Baseball Hitting Stats from 1990-1995",
  mapping = ggplot2::aes(color = lg),
  types = list(
    # change the shape and add some transparency to the points
    continuous = wrap("smooth_loess", alpha = 0.50, shape = "+")
  ),
  showStrips = FALSE
);

p_(pm)

# Example derived from:
## R Data Analysis Examples | Canonical Correlation Analysis. UCLA: Institute for Digital
## Research and Education.
## from http://www.stats.idre.ucla.edu/r/dae/canonical-correlation-analysis
## "Example 1. A researcher has collected data on three psychological variables, four
## academic variables (standardized test scores) and gender for 600 college freshman.
She is interested in how the set of psychological variables relates to the academic variables and gender. In particular, the researcher is interested in how many dimensions (canonical variables) are necessary to understand the association between the two sets of variables.

```r
# add color according to sex
pm <- ggduo(
  psychademic,
  mapping = ggplot2::aes(color = sex),
  rev(psych_variables), academic_variables,
  types = list(continuous = loess_with_cor),
  showStrips = FALSE,
  legend = c(5,2)
)
suppressWarnings(p_(pm))
```

# add color according to sex
ggfacet

pm <- ggduo(
  psychademic,
  mapping = ggplot2::aes(color = motivation),
  rev(psych_variables), academic_variables,
  types = list(continuous = loess_with_cor),
  showStrips = FALSE,
  legend = c(5,2)
) +
  ggplot2::theme(legend.position = "bottom")
suppressWarnings(p_(pm))

Description

ggfacet - single ggplot2 plot matrix with facet_grid

Usage

ggfacet(data, mapping = NULL, columnsX = 1:ncol(data),
  columnsY = 1:ncol(data), fn = ggally_points, ...,
  columnLabelsX = names(data[columnsX]),
  columnLabelsY = names(data[columnsY]), xlab = NULL, ylab = NULL,
  title = NULL, scales = "free")

Arguments

data data.frame that contains all columns to be displayed. This data will be melted
  before being passed into the function fn

mapping aesthetic mapping (besides x and y). See aes()

columnsX columns to be displayed in the plot matrix

columnsY rows to be displayed in the plot matrix

fn function to be executed. Similar to ggpairs and ggduo, the function may either
  be a string identifier or a real function that wrap understands.

... extra arguments passed directly to fn

columnLabelsX, columnLabelsY column and row labels to display in the plot matrix

xlab, ylab, title plot matrix labels

scales parameter supplied to ggplot2::facet_grid. Default behavior is "free"
Examples

# Small function to display plots only if it's interactive
p_ <- ggally::print_if_interactive
if (requireNamespace("chemometrics", quietly = TRUE)) {
  data(NIR, package = "chemometrics")
  NIR_sub <- data.frame(NIR$GlcEtOH, NIR$xNIR[,1:3])
  str(NIR_sub)
  x_cols <- c("X1115.0", "X1120.0", "X1125.0")
  y_cols <- c("Glucose", "Ethanol")

  # using ggduo directly
  p <- ggduo(NIR_sub, x_cols, y_cols, types = list(continuous = "points"))
  p_(p)

  # using ggfacet
  p <- ggfacet(NIR_sub, x_cols, y_cols)
  p_(p)

  # add a smoother
  p <- ggfacet(NIR_sub, x_cols, y_cols, fn = 'smooth_loess')
  p_(p)
  # same output
  p <- ggfacet(NIR_sub, x_cols, y_cols, fn = ggally_smooth_loess)
  p_(p)

  # Change scales to be the same in for every row and for every column
  p <- ggfacet(NIR_sub, x_cols, y_cols, scales = "fixed")
  p_(p)
}

### gglegend

Plot only legend of plot function

**Description**

Plot only legend of plot function

**Usage**

`gglegend(fn)`

**Arguments**

- `fn` this value is passed directly to an empty `wrap` call. Please see `?wrap` for more details.

**Value**

a function that when called with arguments will produce the legend of the plotting function supplied.
Examples

# display regular plot
ggally_points(iris, ggplot2::aes(Sepal.Length, Sepal.Width, color = Species))

# Make a function that will only print the legend
points_legend <- gglegend(ggally_points)
points_legend(iris, ggplot2::aes(Sepal.Length, Sepal.Width, color = Species))

# produce the sample legend plot, but supply a string that 'wrap' understands
same_points_legend <- gglegend("points")
identical(
  attr(attr(points_legend, "fn"), "original_fn"),
  attr(attr(same_points_legend, "fn"), "original_fn")
)

# Complicated examples
custom_legend <- wrap(gglegend("points"), size = 6)
custom_legend(iris, ggplot2::aes(Sepal.Length, Sepal.Width, color = Species))

# Use within ggpairs
pm <- ggpairs(
  iris, 1:2,
  mapping = ggplot2::aes(color = Species),
  upper = list(continuous = gglegend("points"))
)
# pm

# Place a legend in a specific location
pm <- ggpairs(iris, 1:2, mapping = ggplot2::aes(color = Species))
# Make the legend
pm[1,2] <- points_legend(iris, ggplot2::aes(Sepal.Width, Sepal.Length, color = Species))
pm

ggmatrix - A ggplot2 Matrix

Description

Make a generic matrix of ggplot2 plots.

Usage

ggmatrix(plots, nrow, ncol, xAxisLabels = NULL, yAxisLabels = NULL,
  title = NULL, xlab = NULL, ylab = NULL, byrow = TRUE,
  showStripes = NULL, showAxisPlotLabels = TRUE,
  showXAxisPlotLabels = TRUE, showYAxisPlotLabels = TRUE, labeller = NULL,
  switch = NULL, xProportions = NULL, yProportions = NULL,
  progress = NULL, data = NULL, gg = NULL, legend = NULL)
Arguments

plots list of plots to be put into matrix
nrow, ncol number of rows and columns
xAxisLabels, yAxisLabels strip titles for the x and y axis respectively. Set to NULL to not be displayed
title, xlab, ylab title, x label, and y label for the graph. Set 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.
labeller labeller for facets. See labellers. Common values are "label_value" (default) and "label_parsed".
switch switch parameter for facet_grid. See ggplot2::facet_grid. By default, the labels are displayed on the top and right of the plot. If "x", the top labels will be displayed to the bottom. If "y", the right-hand side labels will be displayed to the left. Can also be set to "both"
xProportions, yProportions Value to change how much area is given for each plot. Either NULL (default), numeric value matching respective length, or grid::unit object with matching respective length
progress NULL (default) for a progress bar in interactive sessions with more than 15 plots, TRUE for a progress bar, FALSE for no progress bar, or a function that accepts at least a plot matrix and returns a new progress::progress_bar. See ggmatrix_progress.
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
legend May be the two objects described below or the default NULL value. The legend position can be moved by using ggplot2’s theme element pm + theme(legend.position = "bottom") a numeric vector of length 2 provides the location of the plot to use the legend for the plot matrix’s legend. Such as legend = c(3,5) which will use the legend from the plot in the third row and fifth column a single numeric value provides the location of a plot according to the display order. Such as legend = 3 in a plot matrix with 2 rows and 5 columns displayed by column will return the plot in position c(1,2) a object from grab_legend() a predetermined plot legend that will be displayed directly
Memory usage

Now that the print.ggmatrix method uses a large gtable object, rather than print each plot independently, memory usage may be of concern. From small tests, memory usage flutters around object.size(data) * 0.3 * length(plots). So, for a 80Mb random noise dataset with 100 plots, about 2.4 Gb of memory needed to print. For the 3.46 Mb diamonds dataset with 100 plots, about 100 Mb of memory was needed to print. The benefits of using the ggplot2 format greatly outweigh the price of about 20

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```r
# small function to display plots only if it's interactive
p_ <- GGally::print_if_interactive

plotList <- list()
for (i in 1:6) {
  plotList[[i]] <- ggally_text(paste("Plot ", i, sep = ")
}

pm <- ggmatrix(
  plotList,
  2, 3,
  c("A", "B", "C"),
  c("D", "E"),
  byrow = TRUE
)
p_(pm)

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

---

**ggmatrix_gtable**

Print ggmatrix object

**Description**

Specialized method to print the ggmatrix object-
Usage

```r
ggmatrix_gtable(pm, ..., progress = NULL,
    progress_format = formals(ggmatrix_progress)$format)
```

Arguments

- `pm`: ggmatrix object to be plotted
- `...`: ignored
- `progress`, `progress_format`
  
  Please use the `progress` parameter in your ggmatrix-like function. See `ggmatrix_progress` for a few examples. These parameters will soon be deprecated.

Author(s)

Barret Schloerke <schloerke@gmail.com>

Examples

```r
data(tips, package = "reshape")
pm <- ggpairs(tips, c(1,3,2), mapping = ggplot2::aes_string(color = "sex"))
ggmatrix_gtable(pm)
```

Description

ggmatrix default progress bar

Usage

```r
ggmatrix_progress(format = " plot: [.plot_i,.plot_j]:.bar]:percent est:.eta ",
    clear = TRUE, show_after = 0, ...)
```

Arguments

- `format`, `clear`, `show_after`, ...
  
  parameters supplied directly to `progress::progress_bar$new()`

Value

function that accepts a plot matrix as the first argument and ... for future expansion. Internally, the plot matrix is used to determine the total number of plots for the progress bar.
Examples

```
p_ <- GGally::print_if_interactive

pm <- ggpairs(iris, 1:2, progress = ggmatrix_progress())
p_(pm)

# does not clear after finishing
pm <- ggpairs(iris, 1:2, progress = ggmatrix_progress(clear = FALSE))
p_(pm)
```

**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](http://github.com/briatte/ggnet) for the latest version of ggnet2, and [https://briatte.github.io/ggnet](https://briatte.github.io/ggnet) for a vignette that contains many examples and explanations.

### Usage

```r
ggnet(net, mode = "fruchtermanreingold", layout.par = NULL,
      layout.exp = 0, size = 9, alpha = 1, weight = "none",
      weight.legned = NA, weight.method = weight, weight.min = NA,
      weight.max = NA, weight.cut = FALSE, group = NULL, group.legned = 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 (help("tnet", package = "tnet")).

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

```r
library(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(n, 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)
```

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](http://github.com/briatte/ggnet) for the latest version of ggnet2, and [https://briatte.github.io/ggnet](https://briatte.github.io/ggnet) for a vignette that contains many examples and explanations.

Usage

```r
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 at-
tribute. 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.
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.
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).

```r
label.trim
node.alpha  
node.color  
node.label  
node.shape  
node.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
edge.label.color
edge.label.fill
edge.label.size
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 (help("tnet", package = "tnet")).

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

library(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**

- 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**

```r
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 is 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 is the 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 is 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 is 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 ggplot2 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.
Examples

# small function to display plots only if it's interactive
p_ <- GGally::print_if_interactive
invisible(lapply(c("ggplot2", "maps", "network", "sna"), base::library, character.only = TRUE))

## Example showing great circles on a simple map of the USA

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 %>% setnames("lat", "network.vertex.names(flights)", "lat")
flights %>% setnames("lon", "network.vertex.names(flights)", "long")

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

# compute degree centrality
flights %>% setnames("degree", degree(flights, gmode = "digraph")

# add random groups
flights %>% setnames("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
p <- ggnetworkmap(
  usa, flights, size = 4, great.circles = TRUE,
  node.group = mygroup, segment.color = "steelblue",
  ring.group = degree, weight = degree
)
p_(p)
## 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
p <- ggnetworkmap(world, twitter_spambots)
p_(p)

# domestic distribution
p <- ggnetworkmap(net = twitter_spambots)
p_(p)

# topology
p <- ggnetworkmap(net = twitter_spambots, arrow.size = 0.5)
p_(p)

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

p <- 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")
p_(p)

# show some vertex attributes associated with each account
p <- 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")
p_(p)


ggnostic - Plot matrix of statistical model diagnostics

Description

Ggnostic - Plot matrix of statistical model diagnostics

Usage

ggnostic(model, ..., columnsX = attr(data, "var_x"), columnsY = c(".resid", ".sigma", ".hat", ".cooks\d"), columnLabelsX = attr(data, "var_x_label"), columnLabelsY = gsub("\\\", ",", gsub("^\"\", "", columnsY)),
xlab = "explanatory variables", ylab = "diagnostics",
title = paste(deparse(model$call, width.cutoff = 500L), collapse = \\n"n"),
continuous = list(default = ggly_points, .fitted = ggly_points, .se.fit
= ggly_nostic_se_fit, .resid = ggly_nostic_resid, .hat =
ggly_nostic_hat, .sigma = ggly_nostic_sigma, .cooks\d =
ggly_nostic_cooksd, .std.resid = ggly_nostic_std_resid),
combo = list(default = ggly_box_no_facet, fitted = ggly_box_no_facet,
.se.fit = ggly_nostic_se_fit, .resid = ggly_nostic_resid, .hat =
ggly_nostic_hat, .sigma = ggly_nostic_sigma, .cooks\d =
ggly_nostic_cooksd, .std.resid = ggly_nostic_std_resid),
discrete = list(default = ggly_ratio, .fitted = ggly_ratio, .se.fit
= ggly_ratio, .resid = ggly_ratio, .hat = ggly_ratio, .sigma =
ggly_ratio, .cooks\d = ggly_ratio, .std.resid = ggly_ratio),
progress = NULL, data = broomify(model))

Arguments

model        statistical model object such as output from stats::lm or stats::glm
...
arguments passed directly to ggdoo
columnsX     columns to be displayed in the plot matrix. Defaults to the predictor columns of
            the model
columnsY     rows to be displayed in the plot matrix. Defaults to residuals, leave one out
            sigma value, diagonal of the hat matrix, and Cook's Distance. The possible
            values are the response variables in the model and the added columns provided
            by broom::augment(model). See details for more information.
columnLabelsX, columnLabelsY
            column and row labels to display in the plot matrix
xlab, ylab, title
            plot matrix labels passed directly to ggmatrix
continuous, combo, discrete
            list of functions for each y variable. See details for more information.
progress

NULL (default) for a progress bar in interactive sessions with more than 15 plots, TRUE for a progress bar, FALSE for no progress bar, or a function that accepts at least a plot matrix and returns a new progress ::progress_bar. See ggmatrix_progress.

data
data defaults to a 'broomify'ed model object. This object will contain information about the X variables, Y variables, and multiple broom outputs. See broomify(model) for more information.

'columnsY'

broom::augment() collects data from the supplied model and returns a data.frame with the following columns (taken directly from broom documentation). These columns are the only allowed values in the columnsY parameter to ggnostic.

.resid Residuals
.hat Diagonal of the hat matrix
.sigma Estimate of residual standard deviation when corresponding observation is dropped from model
.cooksd Cooks distance, cooks.distance
.fitted Fitted values of model
.se.fit Standard errors of fitted values
.std.resid Standardized residuals

response variable name The response variable in the model may be added. Such as "mpg" in the model lm(mpg ~ ., data = mtcars)

'continuous', 'combo', 'discrete' types

Similar to ggdudo and ggpairs, functions may be supplied to display the different column types. However, since the Y rows are fixed, each row has its own corresponding function in each of the plot types: continuous, combo, and discrete. Each plot type list can have keys that correspond to the broom::augment() output: ".fitted", ".resid", ".std.resid", ".sigma", ".se.fit", ".hat", ".cooksdist". An extra key, "default", is used to plot the response variables of the model if they are included. Having a function for each diagnostic allows for very fine control over the diagnostics plot matrix. The functions for each type list are wrapped into a switch function that calls the function corresponding to the y variable being plotted. These switch functions are then passed directly to the types parameter in ggdudo.

Examples

# small function to display plots only if it's interactive
p_ <- GGally::print_if_interactive
data(mtcars)

# use mtcars dataset and alter the 'am' column to display actual name values
mtc <- mtcars
mtc$am <- c("0" = "automatic", "1" = "manual")[as.character(mtc$am)]

# step the complete model down to a smaller model
mod <- stats::step(stats::lm(mpg ~ ., data = mtc), trace = FALSE)
## ggpairs

### ggpairs - A ggplot2 generalized pairs plot

**Description**

Make a matrix of plots with a given data set

**Usage**

```r
ggpairs(data, mapping = NULL, columns = 1:ncol(data), title = NULL,
upper = list(continuous = "cor", combo = "box_no_facet", discrete =
"facetbar", na = "na"), lower = list(continuous = "points", combo =
"facethist", discrete = "facetbar", na = "na"), diag = list(continuous =
"densityDiag", discrete = "barDiag", na = "naDiag"), params = NULL, ...,
xlab = NULL, ylab = NULL, axisLabels = c("show", "internal", "none"),
columnLabels = colnames(data[columns]), labeller = "label_value",
switch = NULL, showStrips = NULL, legend = NULL,
cardinality_threshold = 15, progress = NULL,
legends = stop("deprecated"))
```

**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. Default to all columns.

title, xlab, ylab: title, x label, and y label for the graph

upper: see Details

lower: see Details

diag: see Details

params: deprecated. Please see wrap_fn_with_param_arg

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. Default to names of columns being used.

labeller: labeller for facets. See labellers. Common values are "label_value" (default) and "label_parsed".

switch: switch parameter for facet_grid. See ggplot2::facet_grid. By default, the labels are displayed on the top and right of the plot. If "x", the top labels will be displayed to the bottom. If "y", the right-hand side labels will be displayed to the left. Can also be set to "both"

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.

legend: May be the two objects described below or the default NULL value. The legend position can be moved by using ggplot2’s theme element `theme(legend.position = "bottom")`

- a numeric vector of length 2 provides the location of the plot to use the legend for the plot matrix’s legend. Such as `legend = c(3, 5)` which will use the legend from the plot in the third row and fifth column
- a single numeric value provides the location of a plot according to the display order. Such as `legend = 3` in a plot matrix with 2 rows and 5 columns displayed by column will return the plot in position `c(1, 2)
- a object from grab_legend() a predetermined plot legend that will be displayed directly

cardinality_threshold: maximum number of levels allowed in a character / factor column. Set this value to NULL to not check factor columns. Default to 15

progress: NULL (default) for a progress bar in interactive sessions with more than 15 plots, TRUE for a progress bar, FALSE for no progress bar, or a function that accepts at least a plot matrix and returns a new progress::progress_bar. See `ggmatrix_progress`

legends: deprecated

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', 'smooth_loess', 'density', 'cor', 'blank'). This option is used for continuous X and Y data.

combo exactly one of ('box', 'box_no_facet', 'dot', 'dot_no_facet', '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 as an option to upper, lower, or diag, it should implement the function api of function(data, mapping, ...)(#make ggpplot2 plot). If a specific function needs its parameters set, wrap(fn, param1 = val1, param2 = val2) the function with its parameters.

Value
ggmatrix 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

See Also
wrap v1_ggmatrix_theme

Examples

# small function to display plots only if it's interactive
p_ <- GGally::print_if_interactive

## Quick example, with and without colour
```r
data(flea)
ggpairs(flea, columns = 2:4)
pm <- ggpairs(flea, columns = 2:4, ggplot2::aes(colour=species))
p_(pm)

# Note: colour should be categorical, else you will need to reset
# the upper triangle to use points instead of trying to compute corr

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

## Plot Types
# Change default plot behavior
pm <- ggpairs(
  tips[, c(1, 3, 4, 2)],
  upper = list(continuous = "density", combo = "box_no_facet"),
  lower = list(continuous = "points", combo = "dot_no_facet")
)
p_(pm)

# Supply Raw Functions (may be user defined functions!)
pm <- ggpairs(
  tips[, c(1, 3, 4, 2)],
  upper = list(continuous = ggally_density, combo = ggally_box_no_facet),
  lower = list(continuous = ggally_points, combo = ggally_dot_no_facet)
)
p_(pm)

## Use sample of the diamonds data
data(diamonds, package="ggplot2")
diamonds.samp <- diamonds[sample(1:dim(diamonds)[1], 1000), ]

## Different aesthetics for different plot sections and plot types
pm <- ggpairs(
  diamonds.samp[, 1:5],
  mapping = ggplot2::aes(color = cut),
  upper = list(continuous = wrap("density", alpha = 0.5), combo = "box_no_facet"),
  lower = list(continuous = wrap("points", alpha = 0.3), combo = wrap("dot_no_facet", alpha = 0.4)),
  title = "Diamonds"
)
p_(pm)

## Axis Label Variations
# Only Variable Labels on the diagonal (no axis labels)
pm <- ggpairs(tips[, 1:3], axisLabels="internal")
p_(pm)

# Only Variable Labels on the outside (no axis labels)
pm <- ggpairs(tips[, 1:3], axisLabels="none")
p_(pm)
```
## Facet Label Variations

### Default:

```r
def_x <- rnorm(100)
def_y <- def_x + rnorm(100, 0, 0.1)
df <- data.frame(x = def_x, y = def_y, c = sqrt(def_x^2 + def_y^2))

pm <- ggpairs(
  df,
  columnLabels = c("alpha[foo]", "alpha[bar]", "sqrt(alpha[foo]^2 + alpha[bar]^2]"))
)
p(pm)
```

### Parsed labels:

```r
def,
  columnLabels = c("alpha[foo]", "alpha[bar]", "sqrt(alpha[foo]^2 + alpha[bar]^2]")
  labeller = "label_parsed"
)
p(pm)
```

## Plot Insertion Example

```r
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\nwn plot.\nLike that one.\n---"
)
custom_car[1, 3] <- personal_plot
p(custom_car)
```

## Remove binwidth warning from ggplot2

### Displays warning about picking a better binwidth

```r
pm <- ggpairs(tips, 2:3)
p(pm)
```

### No warning displayed

```r
pm <- ggpairs(tips, 2:3, lower = list(combo = wrap("facethist", binwidth = 0.5)))
p(pm)
```

### No warning displayed with user supplied function

```r
pm <- ggpairs(tips, 2:3, lower = list(combo = wrap(ggally_facethist, binwidth = 0.5)))
p(pm)
```

---

**ggparcoord**

---

**ggparcoord - A ggplot2 Parallel Coordinate Plot**

**Description**

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

```r
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 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**

```r
# small function to display plots only if it's interactive
p_ <- GGally::print_if_interactive

# 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
p <- ggparcoord(data = diamonds.samp, columns = c(1, 5:10))

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

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

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

# 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")
p <- ggparcoord(data = msleep, columns = 6:11, groupColumn = "vore", missing =
    "random", scale = "uniminmax")

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

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

# add points to the plot, add a title, and use an alpha scalar to make the lines
# transparent
p <- ggparcoord(data = iris, columns = 1:4, groupColumn = 5, order = "anyClass",
    showPoints = TRUE, title = "Parallel Coordinate Plot for the Iris Data",
    alphaLines = 0.3)

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

```r
ggscatmat(data, columns = 1:ncol(data), color = NULL, alpha = 1,
corMethod = "pearson")
```

### 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.
- **corMethod**: method argument supplied to `cor`

### Author(s)

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

### Examples

```r
data(flea)
ggscatmat(flea, columns = 2:4)
ggscatmat(flea, columns = 2:4, color = "species")
```
Survival curves with 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, size.est = 0.5, size.ci = size.est, cens.size = 2, cens.shape = 3, back.white = FALSE, xlab = "Time", ylab = "Survival", main = "", order.legend = TRUE)

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.

size.est
line width of the survival curve

size.ci
line width of the 95% CI

cens.size
point size of the censoring points

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.

order.legend
boolean to determine if the legend display should be ordered by final survival time
Value

An object of class `ggplot`

Author(s)

Edwin Thoen <edwinthoen@gmail.com>

Examples

```r
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 + ggplot2::coord_cartesian(xlim = c(0, 80), ylim = c(0.45, 1))

  # Add the diseases names to the plot and remove legend
  pl.kid +
    ggplot2::annotate(
      "text",
      label = c("PKD", "Other", "GN", "AN"),
      x = c(90, 125, 5, 60),
      y = c(0.8, 0.65, 0.55, 0.30),
      size = 5,
      colour = scales::hue_pal(
        h = c(0, 360) + 15,
        c = 100,
        l = 65,
        h.start = 0,
        direction = 1
    )
```
ggts  Multiple Time Series

Description

GGally implementation of ts.plot. Wraps around the ggduo function and removes the column strips

Usage

```
ggts(..., columnLabelsX = NULL, xlab = "time")
```

Arguments

- `...` supplied directly to `ggduo`
- `columnLabelsX` remove top strips for the X axis by default
- `xlab` defaults to "time"

Value

ggmatrix object

Examples

```
ggts(pigs, "time", c("gilts", "profit", "s_per_herdsz", "production", "herdsz"))
```

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, ...)
```
glyphs

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

```r
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

```r
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 = "")
```

grab_legend

---

Grab the legend and print it as a plot

Description

Grab the legend and print it as a plot

Usage

```r
grab_legend(p)
```

### S3 method for class 'legend_guide_box'

```r
print(x, ..., plotNew = FALSE)
```

Arguments

- `p` ggplot2 plot object
- `x` legend object that has been grabbed from a ggplot2 object
- `...` ignored
- `plotNew` boolean to determine if the 'grid.newpage()' command and a new blank rectangle should be printed
Examples

```r
library(ggplot2)
histPlot <- qplot(
  x = Sepal.Length,
  data = iris,
  fill = Species,
  geom = "histogram",
  binwidth = 1/4
)
(right <- histPlot)
(bottom <- histPlot + theme(legend.position = "bottom"))
(top <- histPlot + theme(legend.position = "top"))
(left <- histPlot + theme(legend.position = "left"))

grab_legend(right)
grab_legend(bottom)
grab_legend(top)
grab_legend(left)
```

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


is_horizontal

Check if plot is horizontal

Description

Check if plot is horizontal

Usage

is_horizontal(data, mapping, val = "y")

is_character_column(data, mapping, val = "y")

Arguments

data   data used in ggplot2 plot
mapping ggplot2 aes() mapping
val     key to retrieve from mapping

Value

Boolean determining if the data is a character-like data

Examples

is_horizontal(iris, ggplot2::aes(Sepal.Length, Species)) # TRUE
is_horizontal(iris, ggplot2::aes(Sepal.Length, Species), "x") # FALSE
is_horizontal(iris, ggplot2::aes(Sepal.Length, Sepal.Width)) # FALSE
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"))

mapping_string

Aes name

Description

Aes name

Usage

mapping_string(aes_col)

Arguments

aes_col Single value from ggplot2::aes(...)
Value
character string

Examples
mapping <- ggplot2::aes(Petal.Length)
mapping_string(mapping$x)

mapping_swap_x_y  
Swap x and y mapping

Description
Swap x and y mapping

Usage
mapping_swap_x_y(mapping)

Arguments
mapping  
output of ggplot2::aes(...)

Value
Aes mapping with the x and y values switched

Examples
mapping <- ggplot2::aes(Petal.Length, Sepal.Width)
mapping
mapping_swap_x_y(mapping)

model_response_variables
Model term names

Description
Retrieve either the response variable names, the beta variable names, or beta variable names. If the model is an object of class 'lm', by default, the beta variable names will include anova significance stars.
Usage

model_response_variables(model, data = broom::augment(model))

model_beta_variables(model, data = broom::augment(model))

model_beta_label(model, data = broom::augment(model), lmStars = TRUE)

Arguments

model model in question
data equivalent to broom::augment(model)
lmStars boolean that determines if stars are added to labels

Value

character vector of names

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 id for each spatial position.

References

This data contains about the United Kingdom Pig Production from the book 'Data' by Andrews and Herzberg. The original data can be on Statlib: http://lib.stat.cmu.edu/datasets/Andrews/T62.1

Usage
data(pigs)

Format
A data frame with 48 rows and 8 variables

Details
The time variable has been added from a combination of year and quarter

- time year + (quarter - 1) / 4
- year year of production
- quarter quarter of the year of production
- gilts number of sows giving birth for the first time
- profit profit ratio of price to an index of feed price
- s_per_herdsz ratio of the number of breeding pigs slaughtered to the total breeding herd size
- production number of pigs slaughtered that were reared for meat
- herdsz breeding herd size

References
print.ggmatrix  

Print ggmatrix object

Description

Print method taken from ggplot2::print.ggplot and altered for a ggmatrix object

Usage

```r
## S3 method for class 'ggmatrix'
print(x, newpage = is.null(vp), vp = NULL, ...)
```

Arguments

- `x`: plot to display
- `newpage`: draw new (empty) page first?
- `vp`: viewport to draw plot in
- `...`: arguments passed onto `ggmatrix_gtable`

Author(s)

Barret Schloerke

Examples

```r
data(tips, package = "reshape")
pMat <- ggpairs(tips, c(1,3,2), mapping = ggplot2::aes_string(color = "sex"))
pMat # calls print(pMat), which calls print.ggmatrix(pMat)
```

print_if_interactive  

Print if not CRAN

Description

Small function to print a plot if the R session is interactive or in a Travis build

Usage

```r
print_if_interactive(p)
```

Arguments

- `p`: plot to be displayed
UCLA canonical correlation analysis data

Description
This data contains 600 observations on eight variables

Usage
data(psychademic)

Format
A data frame with 600 rows and 8 variables

Details
- locus_of_control - psychological
- self_concept - psychological
- motivation - psychological. Converted to four character groups
- read - academic
- write - academic
- math - academic
- science - academic
- female - academic. Dropped from original source
- sex - academic. Added as a character version of female column

References
putPlot

Put Plot

Description

Function to place your own plot in the layout.

Usage

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

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

Arguments

- `pm`: `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\nnoto put in your\own plot.\nLike that one.\n"----")
custom_car[1, 3] <- personal_plot
# custom_car

# remove plots after creating a plot matrix
custom_car[2,1] <- NULL
custom_car[3,1] <- "blank" # the same as storing null
custom_car[3,2] <- NULL
```

**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

scatter matrix. scatmat - plot the lower triangle 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
str.ggmatrix

Value
sample skewness of x

Author(s)
Jason Crowley <crowley.jason.s@gmail.com>

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>ggmatrix object to be viewed</td>
</tr>
<tr>
<td>...</td>
<td>passed on to the default str method</td>
</tr>
<tr>
<td>raw</td>
<td>boolean to determine if the plots should be converted to text or kept as original objects</td>
</tr>
</tbody>
</table>

description
Twitter spambots

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, corMethod = "pearson")

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)
corMethod method argument supplied to cor

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"))
v1_ggmatrix_theme

Modify a ggmatrix object by adding an ggplot2 object to all plots

Description

Modify a ggmatrix object by adding an ggplot2 object to all plots

Usage

v1_ggmatrix_theme()

Examples

ggpairs(iris, 1:2) + v1_ggmatrix_theme()
  # move the column names to the left and bottom
  ggpairs(iris, 1:2, switch = "both") + v1_ggmatrix_theme()

wrap_fn_with_param_arg

Wrap a function with different parameter values

Description

Wraps a function with the supplied parameters to force different default behavior. This is useful for functions that are supplied to ggpairs. It allows you to change the behavior of one function, rather than creating multiple functions with different parameter settings.

Usage

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

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

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

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

Arguments

funcVal  function that the params will be applied to. The function should follow the api of function(data, mapping, ...){}. funcVal is allowed to be a string of one of the ggally_NAME functions, such as "points" for ggally_points or "facetdensity" for ggally_facetdensity.
**wrap_fn_with_param_arg**

- **params**: named vector or list of parameters to be applied to the funcVal
- **funcArgName**: name of function to be displayed
- **...**: named parameters to be supplied to `wrap_fn_with_param_arg`

**Details**

`wrap` is identical to `wrap_fn_with_params`. These functions take the new parameters as arguments.

`wrapp` is identical to `wrap_fn_with_param_arg`. These functions take the new parameters as a single list.

The `params` and `fn` attributes are there for debugging purposes. If either attribute is altered, the function must be re-wrapped to have the changes take effect.

**Value**

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

**Examples**

```r
# small function to display plots only if it's interactive
p_ <- ggally::print_if_interactive

# example function that prints 'val'
fn <- function(data, mapping, val = 2) {
  print(val)
}
fn(data = NULL, mapping = NULL) # 2

# wrap function to change default value 'val' to 5 instead of 2
wrapped_fn1 <- wrap(fn, val = 5)
wrapped_fn1(data = NULL, mapping = NULL) # 5
# you may still supply regular values
wrapped_fn1(data = NULL, mapping = NULL, val = 3) # 3

# wrap function to change 'val' to 5 using the arg list
wrapped_fn2 <- wrap_fn_with_param_arg(fn, params = list(val = 5))
wrapped_fn2(data = NULL, mapping = NULL) # 5

# change parameter settings in ggpairs for a particular function
## Goal output:
regularPlot <- ggally_points(i
  iris,
  ggplot2::aes(Sepal.Length, Sepal.Width),
  size = 5, color = "$red"
)
p_(regularPlot)

# Wrap ggally_points to have parameter values size = 5 and color = 'red'
w_ggally_points <- wrap(ggally_points, size = 5, color = "red")
wrappedPlot <- w_ggally_points(i
  iris,
  ggplot2::aes(Sepal.Length, Sepal.Width),
  size = 5, color = "$red"
)
p_(regularPlot)
```
wrap_fn_with_param_arg

    ggplot2::aes(Sepal.Length, Sepal.Width)
  )
  p_(wrappedPlot)

# Double check the aes parameters are the same for the geom_point layer
identical(regularPlot$layers[[1]]$aes_params, wrappedPlot$layers[[1]]$aes_params)

# Use a wrapped function in ggpairs
pm <- ggpairs(iris, 1:3, lower = list(continuous = wrap(ggally_points, size = 5, color = "red")))
  p_(pm)

pm <- ggpairs(iris, 1:3, lower = list(continuous = w_ggally_points))
  p_(pm)
Index

+Topic **datasets**
  australia_PISA2012, 6
  flea, 9
  happy, 75
  nasa, 79
  pigs, 80
  psychademic, 82
  twitter_spambots, 87
+Topic **hplot**
  getPlot, 11
  ggally_barDiag, 12
  ggally_blank, 12
  ggally_box, 13
  ggally_cor, 14
  ggally_density, 15
  ggally_densityDiag, 16
  ggally_densstrip, 16
  ggally_dot, 18
  ggally_dot_and_box, 19
  ggally_facetbar, 19
  ggally_facetdensity, 20
  ggally_facetdensitystrip, 21
  ggally_facethist, 21
  ggally_na, 22
  ggally_points, 29
  ggally_ratio, 30
  ggally_smooth, 31
  ggally_text, 32
  ggmatrix, 43
  ggpairs, 61
  putPlot, 83
+gg, 4, 4
[.ggmatrix(getPlot), 11
[.glyphplot (glyphplot), 72
[<-.ggmatrix (putPlot), 83

add_ref_boxes, 5
add_ref_lines, 5
aes, 8, 37, 41, 61, 77, 78
arrow, 48, 49, 53

AsIs, 66
asNetwork, 47, 51, 56
augment, 8, 59, 60
australia_PISA2012, 6
brew_colors, 7
brewer_pal, 51
broomify, 8, 60
cooks.distance, 23, 60
cor, 34, 36, 69, 88
cut, 35, 48, 52
degree, 48, 51
dgeset.constructors, 47, 51, 56
eval_data_col, 8
expand_range, 48, 51
facet_grid, 37, 41, 44, 62
find_plot_type, 9
flea, 9
fn_switch, 10
geom_line, 5, 24–26
geom_point, 33
geom_rect, 5
geom_smooth, 26, 31
geom_text, 35, 49, 54
geom_tile, 30
getPlot, 11
ggally_barDiag, 12
ggally_blank, 12
ggally_blankDiag (ggally_blank), 12
ggally_box, 13
ggally_box_no_facet (ggally_box), 13
ggally_cor, 14
ggally_density, 15
ggally_densityDiag, 16
ggally_densstrip, 16
ggally_diagAxis, 17
INDEX

ggally_dot, 18
ggally_dot_and_box, 19
ggally_dot_no_facet (ggally_dot), 18
ggallyfacetbar, 19
ggally_facetdensity, 20
ggally_facetdensitystrip, 21
ggally_facethist, 21
ggally_nad, 22
ggally_nadDiag (ggally_nad), 22
ggally_nostic_cooksd, 23
ggally_nostic_hat, 24
ggally_nostic_line, 23, 24, 25, 26–28
ggally_nostic_resid, 26, 29
ggally_nostic_se_fit, 27
ggally_nostic_sigma, 28
ggally_nostic_std_resid, 29
ggally_points, 29
ggally_ratio, 30
ggally_smooth, 31
ggally_smooth_lm (ggally_smooth), 31
ggally_smooth_loess (ggally_smooth), 31
ggally_text, 32
ggcoef, 33
ggcorr, 34
ggduo, 36, 41, 59, 60, 72
ggfacet, 41
gglegend, 42
ggmatrix, 43, 59
ggmatrix_gtable, 45, 81
ggmatrix_progress, 38, 44, 46, 60, 62
ggnet, 47, 50, 54
ggnet2, 47, 49, 50
ggnetnetworkmap, 55
ggnostic, 59
ggpairs, 41, 60, 61
ggparcoord, 65
ggplot, 25
ggsccatmat, 69
ggsurv, 70
ggs, 72
.glance, 8
glm, 59
glyphplot, 72
glyphs, 73
gplot, 49, 54
gplot.layout, 47, 51
grab.legend, 38, 44, 62, 74
happy, 75
identity, 73
igraph, 47, 51, 56
influence, 24, 27, 28
intergraph, 47, 51, 56
is.glyphplot (glyphplot), 72
is_character_column (is_horizontal), 76
is_horizontal, 76
labellers, 37, 44, 62
lm, 59
lowertriangle, 77
mapping_string, 77
mapping_swap_x_y, 78
max1 (rescale01), 84
mean0 (rescale01), 84
min0 (rescale01), 84
model_beta_label
(model_response_variables), 78
model_beta_variables
(model_response_variables), 78
model_response_variables, 78
nasa, 79
network, 47, 49, 51, 54, 56
pigs, 80
plot.network, 49, 54
predict, 27
print.ggmatrix, 81
print.glyphplot (glyphplot), 72
print.legend_guide_box (grab.legend), 74
print_if_interactive, 81
progress.bar, 38, 44, 46, 60, 62
psychademic, 82
putPlot, 83
quantile, 48, 52
range01 (rescale01), 84
RColorBrewer, 51
rescale01, 84
rescale1 (rescale01), 84
residuals, 26
resolution, 73
rstandard, 29
scag_order, 84
scatmat, 85
singleClassOrder, 86
skewness, 86
sna, 47, 49, 51, 54
str.gggmatrix, 87
substr, 49, 53
theme, 4, 35, 49, 54
tidy, 8, 33
twitter_spambots, 87

unit, 44
uppertriangle, 88
v1_ggmatrix_theme, 89

wrap, 38, 41, 42, 63
wrap (wrap_fn_with_param_arg), 89
wrap_fn_with_param_arg, 62, 89
wrap_fn_with_params
    (wrap_fn_with_param_arg), 89
wrapp (wrap_fn_with_param_arg), 89