Package ‘ggformula’

January 13, 2021

Title  Formula Interface to the Grammar of Graphics

Description  Provides a formula interface to 'ggplot2' graphics.

Type  Package

Version  0.10.1

License  MIT + file LICENSE

LazyData  TRUE

LazyLoad  TRUE

Depends  R (>= 3.1), ggplot2 (>= 3.3), ggstance (>= 0.3.4), scales, ggridges

Imports  mosaicCore (>= 0.7.0), rlang, magrittr, tibble, stringr, ggforce, grid, labelled

Suggests  tidyr, mosaicData, dplyr, lattice, mosaic, palmerpenguins, testthat, vdiffr, knitr, rmarkdown, lubridate, survival, broom, maps, maptools, rgeos, sf, purrr, ggthemes, covr, ggplot2movies

VignetteBuilder  knitr

RoxygenNote  7.1.1

Encoding  UTF-8

URL  https://github.com/ProjectMOSAIC/ggformula

BugReports  https://github.com/ProjectMOSAIC/ggformula/issues


NeedsCompilation  no

Author  Daniel Kaplan [aut], Randall Pruim [aut, cre]

Maintainer  Randall Pruim <rpruim@calvin.edu>

Repository  CRAN

Date/Publication  2021-01-13 10:00:09 UTC
R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>discrete_breaks</td>
<td>3</td>
</tr>
<tr>
<td>gf_abline</td>
<td>4</td>
</tr>
<tr>
<td>gf_area</td>
<td>7</td>
</tr>
<tr>
<td>gf_ash</td>
<td>10</td>
</tr>
<tr>
<td>gf_bar</td>
<td>13</td>
</tr>
<tr>
<td>gf_barh</td>
<td>20</td>
</tr>
<tr>
<td>gf_bin2d</td>
<td>23</td>
</tr>
<tr>
<td>gf_blank</td>
<td>26</td>
</tr>
<tr>
<td>gf_boxplot</td>
<td>28</td>
</tr>
<tr>
<td>gf_boxplot    h</td>
<td>32</td>
</tr>
<tr>
<td>gf_col</td>
<td>36</td>
</tr>
<tr>
<td>gf_contour</td>
<td>39</td>
</tr>
<tr>
<td>gf_count</td>
<td>42</td>
</tr>
<tr>
<td>gf_crossbar</td>
<td>44</td>
</tr>
<tr>
<td>gf_curve</td>
<td>47</td>
</tr>
<tr>
<td>gf_density</td>
<td>50</td>
</tr>
<tr>
<td>gf_density_2d</td>
<td>54</td>
</tr>
<tr>
<td>gf_dist</td>
<td>59</td>
</tr>
<tr>
<td>gf_dotplot</td>
<td>60</td>
</tr>
<tr>
<td>gf_ecdf</td>
<td>63</td>
</tr>
<tr>
<td>gf_ellipse</td>
<td>65</td>
</tr>
<tr>
<td>gf_empty</td>
<td>67</td>
</tr>
<tr>
<td>gf_errorbar</td>
<td>68</td>
</tr>
<tr>
<td>gf_errorbarh</td>
<td>70</td>
</tr>
<tr>
<td>gf_facet_wrap</td>
<td>73</td>
</tr>
<tr>
<td>gf_fitdist</td>
<td>74</td>
</tr>
<tr>
<td>gf_freqpoly</td>
<td>77</td>
</tr>
<tr>
<td>gf_function</td>
<td>80</td>
</tr>
<tr>
<td>gf_function_2d</td>
<td>81</td>
</tr>
<tr>
<td>gf_hex</td>
<td>83</td>
</tr>
<tr>
<td>gf_histogram</td>
<td>86</td>
</tr>
<tr>
<td>gf_jitter</td>
<td>91</td>
</tr>
<tr>
<td>gf_labs</td>
<td>93</td>
</tr>
<tr>
<td>gf_line</td>
<td>95</td>
</tr>
<tr>
<td>gf_linerange</td>
<td>98</td>
</tr>
<tr>
<td>gf_plot</td>
<td>104</td>
</tr>
<tr>
<td>gf_point</td>
<td>105</td>
</tr>
<tr>
<td>gf_polygon</td>
<td>108</td>
</tr>
<tr>
<td>gf_qq</td>
<td>110</td>
</tr>
<tr>
<td>gf_quantile</td>
<td>113</td>
</tr>
<tr>
<td>gf_raster</td>
<td>116</td>
</tr>
<tr>
<td>gf_rect</td>
<td>119</td>
</tr>
<tr>
<td>gf_relabel</td>
<td>121</td>
</tr>
<tr>
<td>gf_ribbon</td>
<td>122</td>
</tr>
<tr>
<td>gf_ridgeline</td>
<td>124</td>
</tr>
<tr>
<td>gf_rug</td>
<td>130</td>
</tr>
</tbody>
</table>
## discrete_breaks

**Discrete Breaks**

- Creates a function that can be passed to scales for creating discrete breaks at multiples of resolution.

### Usage

```
discrete_breaks(resolution = 1)
```

### Arguments

- **resolution**: Resolution of the breaks

### Value

A function that can be passed to scales functions as the breaks argument.

### Examples

```
x <- rbinom(100, 100, 0.4)
p <- gf_bar(~ x)
p %>% gf_refine(scale_x_continuous(breaks = discrete_breaks(2)))
```
**gf_abline**

Reference lines – horizontal, vertical, and diagonal.

**Description**

These functions create layers that display lines described in various ways. Unlike most of the plotting functions in ggformula, these functions do not take a formula as input for describing positional attributes of the plot.

**Usage**

```r
gf_abline(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  slope,
  intercept,
  color,
  size,
  linetype,
  alpha,
  xlab,
  ylab,
  title,
  subtitle,
  show.legend = NA,
  show.help = NULL,
  inherit = FALSE,
  environment = parent.frame()
)
```

```r
gf_hline(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  yintercept,
  color,
  size,
  linetype,
  alpha,
  xlab,
  ylab,
  title,
  subtitle,
```
gf_abline(caption,
  show.legend = NA,
  show.help = NULL,
  inherit = FALSE,
  environment = parent.frame()
)

gf_vline(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...
  xintercept,
  color,
  size,
  linetype,
  alpha,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  show.legend = NA,
  show.help = NULL,
  inherit = FALSE,
  environment = parent.frame()
)

gf_coefline(object = NULL, coef = NULL, model = NULL, ...)

Arguments

object
  When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

gformula
  Must be NULL.

data
  The data to be displayed in this layer. There are three options:
  If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
  A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
  A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).

... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
slope Parameters that control the position of the line. If these are set, data, mapping and show.legend are overridden.

intercept Parameters that control the position of the line. If these are set, data, mapping and show.legend are overridden.

color A color or a formula used for mapping color.

size A numeric size or a formula used for mapping size.

linetype A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

alpha Opacity (0 = invisible, 1 = opaque).

xlab Label for x-axis. See also gf_labs().

ylab Label for y-axis. See also gf_labs().

title Title, sub-title, and caption for the plot. See also gf_labs().

subtitle Title, sub-title, and caption for the plot. See also gf_labs().

caption Title, sub-title, and caption for the plot. See also gf_labs().

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

show.help If TRUE, display some minimal help.

inherit A logical indicating whether default attributes are inherited.

environment An environment in which to look for variables not found in data.

yintercept Parameters that control the position of the line. If these are set, data, mapping and show.legend are overridden.

xintercept Parameters that control the position of the line. If these are set, data, mapping and show.legend are overridden.

coeff A numeric vector of coefficients.

model A model from which to extract coefficients.

See Also

ggplot2::geom_abline(), ggplot2::geom_vline(), ggplot2::geom_hline()

Examples

mtcars2 <- df_stats(wt ~ cyl, data = mtcars, median_wt = median)
gf_point(wt ~ hp, size = ~wt, color = ~cyl, data = mtcars) %>%
gf_abline(slope = ~0, intercept = ~median_wt, color = ~cyl, data = mtcars2)

gf_point(wt ~ hp, size = ~wt, color = ~cyl, data = mtcars) %>%
gf_abline(slope = 0, intercept = 3, color = "green")

# avoid warnings by using formulas:

gf_point(wt ~ hp, size = ~wt, color = ~cyl, data = mtcars) %>%
gf_abline(slope = ~0, intercept = ~3, color = "green")
gf_point(wt ~ hp, size = ~wt, color = ~cyl, data = mtcars) %>%
gf_hline(yintercept = ~median_wt, color = ~cyl, data = mtcars2)

gf_point(mpg ~ hp, color = ~cyl, size = ~wt, data = mtcars) %>%
gf_abline(color = "red", slope = ~ - 0.10, intercept = ~ 35)

gf_point(mpg ~ hp, color = ~cyl, size = ~wt, data = mtcars) %>%
gf_abline(
  color = "red", slope = ~slope, intercept = ~intercept,
  data = data.frame(slope = -0.10, intercept = 33:35)
)

# We can set the color of the guidelines while mapping color in other layers
gf_point(mpg ~ hp, color = ~cyl, size = ~wt, data = mtcars) %>%
gf_hline(color = "navy", yintercept = ~ c(20, 25), data = NA) %>%
gf_vline(color = "brown", xintercept = ~ c(200, 300), data = NA)

# If we want to map the color of the guidelines, it must work with the
# scale of the other colors in the plot.
gf_point(mpg ~ hp, size = ~wt, data = mtcars, alpha = 0.3) %>%
gf_hline(color = ~"horizontal", yintercept = ~ c(20, 25), data = NA) %>%
gf_vline(color = ~"vertical", xintercept = ~ c(100, 200, 300), data = NA)

gf_point(mpg ~ hp, size = ~wt, color = ~ factor(cyl), data = mtcars, alpha = 0.3) %>%
gf_hline(color = "orange", yintercept = ~ c(20), data = NA) %>%
gf_vline(color = ~ c("4", "6", "8"), xintercept = ~ c(80, 120, 250), data = NA)

gf_point(mpg ~ hp, size = ~wt, color = ~ factor(cyl), data = mtcars, alpha = 0.3) %>%
gf_hline(color = "orange", yintercept = ~ c(20)) %>%
gf_vline(color = c("green", "red", "blue"), xintercept = ~ c(80, 120, 250),
data = NA)

# reversing the layers requires using inherit = FALSE
gf_hline(color = "orange", yintercept = ~ c(20)) %>%
gf_vline(color = ~ c("4", "6", "8"), xintercept = ~ c(80, 120, 250), data = NA) %>%
gf_point(mpg ~ hp,
  size = ~wt, color = ~ factor(cyl), data = mtcars, alpha = 0.3,
  inherit = FALSE
)

---

**gf_area**

*Formula interface to geom_area()*

**Description**

For each x value, geom.ribbon() displays a y interval defined by ymin and ymax. geom.area() is
 a special case of geom.ribbon(), where the ymin is fixed to 0 and y is used instead of ymax.
Usage

gf_area(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha, 
  color, 
  fill, 
  group, 
  linetype, 
  size, 
  xlab, 
  ylab, 
  title, 
  subtitle, 
  caption, 
  geom = "area", 
  stat = "identity", 
  position = "identity", 
  show.legend = NA, 
  show.help = NULL, 
  inherit = TRUE, 
  environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape y ~ x. Faceting can be achieved by including | in the formula.
data The data to be displayed in this layer. There are three options:
  If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
  A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
  A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
...
  Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha Opacity (0 = invisible, 1 = opaque).
color A color or a formula used for mapping color.

fill A color for filling, or a formula used for mapping fill.

group Used for grouping.

linetype A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

size A numeric size or a formula used for mapping size.

xlab Label for x-axis. See also `gf_labs()`.

ylab Label for y-axis. See also `gf_labs()`.

title Title, sub-title, and caption for the plot. See also `gf_labs()`.

subtitle Title, sub-title, and caption for the plot. See also `gf_labs()`.

caption Title, sub-title, and caption for the plot. See also `gf_labs()`.

geom A character string naming the geom used to make the layer.

stat The statistical transformation to use on the data for this layer, as a string.

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

show.help If TRUE, display some minimal help.

inherit A logical indicating whether default attributes are inherited.

environment An environment in which to look for variables not found in data.

See Also
`ggplot2::geom_area()`

Examples

```r
if (require(dplyr) && require(mosaicData)) {
  Temps <- Weather %>%
    filter(city == "Chicago", year == 2016, month <= 4)
  gf_linerange(low_temp + high_temp ~ date, color = ~high_temp, data = Temps)
  gf_ribbon(low_temp + high_temp ~ date, data = Temps, color = "navy", alpha = 0.3)
  gf_area(high_temp ~ date, data = Temps, color = "navy", alpha = 0.3)
  gf_ribbon(low_temp + high_temp ~ date, data = Weather, alpha = 0.3) %>%
    gf_facet_grid(city ~ .)
  gf_linerange(low_temp + high_temp ~ date, data = Weather) %>%
    gf_facet_grid(city ~ .) %>%
    gf_refine(scale_colour_gradientn(colors = rev(rainbow(5))))
}
```
gf_ash

Average Shifted Histograms

Description

An ASH plot is the average over all histograms of a fixed bin width. geom_ash() and gf_ash() provide ways to create ASH plots using ggplot2 or ggformula.

Usage

gf_ash(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha, 
  color, 
  group, 
  linetype, 
  size, 
  xlab, 
  ylab, 
  title, 
  subtitle, 
  caption, 
  geom = "line", 
  stat = "ash", 
  position = "identity", 
  show.legend = NA, 
  show.help = NULL, 
  inherit = TRUE, 
  environment = parent.frame()
)

stat_ash(
  mapping = NULL, 
  data = NULL, 
  geom = "line", 
  position = "identity", 
  na.rm = FALSE, 
  show.legend = NA, 
  inherit.aes = TRUE, 
  binwidth = NULL, 
  adjust = 1, 
  ...
)
geom_ash(
  mapping = NULL,
  data = NULL,
  stat = "ash",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  binwidth = NULL,
  adjust = 1,
  ...
)

Arguments

object

When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

gformula

A formula with shape ~x or y ~ x. y may be stat(density) or stat(count) or stat(ndensity) or stat(ncount). Faceting can be achieved by including | in the formula.

data

A data frame with the variables to be plotted.

... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.

alpha

Opacity (0 = invisible, 1 = opaque).

color

A color or a formula used for mapping color.

group

Used for grouping.

linetype

A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

size

A numeric size or a formula used for mapping size.

xlab

Label for x-axis. See also gf_labs().

ylab

Label for y-axis. See also gf_labs().

title

Title, sub-title, and caption for the plot. See also gf_labs().

subtitle

Title, sub-title, and caption for the plot. See also gf_labs().

caption

Title, sub-title, and caption for the plot. See also gf_labs().

geom

A character string naming the geom used to make the layer.

stat

A character string naming the stat used to make the layer.

position

Either a character string naming the position function used for the layer or a position object returned from a call to a position function.

show.legend

A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.
show.help If TRUE, display some minimal help.
inherit A logical indicating whether default attributes are inherited.
environment An environment in which to look for variables not found in data.
mapping set of aesthetic mappings created by aes() or aes_().
na.rm If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
inherit.aes A logical indicating whether default aesthetics are inherited.
binwidth the width of the histogram bins. If NULL (the default) the binwidth will be chosen so that approximately 10 bins cover the data. adjust can be used to to increase or decrease binwidth.
adjust a numeric adjustment to binwidth. Primarily useful when binwidth is not specified. Increasing adjust makes the plot smoother.

Value
a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

geom_histogram(), link(gf_histogram()).

Examples

data(penguins, package = "palmerpenguins")
gf_ash(~bill_length_mm, color = ~species, data = penguins)
gf_ash(~bill_length_mm, color = ~species, data = penguins, adjust = 2)
gf_ash(~bill_length_mm, color = ~species, data = penguins, binwidth = 1)
gf_ash(~bill_length_mm, color = ~species, data = penguins, binwidth = 1, adjust = 2)
ggplot(faithful, aes(x = eruptions)) +
  geom_histogram(aes(y = stat(density)),
    fill = "lightskyblue", colour = "gray50", alpha = 0.2
  ) +
Description

There are two types of bar charts: `geom_bar()` and `geom_col()`. `geom_bar()` makes the height of the bar proportional to the number of cases in each group (or if the weight aesthetic is supplied, the sum of the weights). If you want the heights of the bars to represent values in the data, use `geom_col()` instead. `geom_bar()` uses `stat_count()` by default: it counts the number of cases at each x position. `geom_col()` uses `stat_identity()`: it leaves the data as is.

Usage

```r
gf_bar(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...
  alpha,
  color,
  fill,
  group,
  linetype,
  size,
  width = NULL,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "bar",
  stat = "count",
  position = "stack",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

gf_counts(
  object = NULL,
  gformula = NULL,
```

data = NULL,
..., alpha, color, fill, group, linetype, size, width = NULL, xlab, ylab, title, subtitle, caption, geom = "bar", stat = "count", position = "stack", show.legend = NA, show.help = NULL, inherit = TRUE, environment = parent.frame()
)
gf_props(
    object = NULL, gformula = NULL, data = NULL,
    ..., alpha, color, fill, group, linetype, size, xlab, ylab = "proportion", title, subtitle, caption, geom = "bar", stat = "count", position = "stack", show.legend = NA, show.help = NULL, inherit = TRUE, environment = parent.frame(),
denom = ~PANEL
)
gf_bar

gf_percents(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., alpha, color, fill, group, linetype, size, xlab, ylab = "percent", title, subtitle, caption, geom = "bar", stat = "count", position = "stack", show.legend = NA, show.help = NULL, inherit = TRUE, environment = parent.frame(), denom = ~PANEL
)

gf_countsh(
  object = NULL, gformula = NULL, data = NULL, ..., alpha, color, fill, group, linetype, size, width = NULL, xlab, ylab, title, subtitle, caption, geom = "barh", stat = "counth", position = "stackv", show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

gf_colh(
  object = NULL,
gformula = NULL,
data = NULL,
  ..., 
alpha, 
color, 
fill, 
group, 
linetype, 
size, 
width = NULL, 
xlab, 
ylab, 
title, 
subtitle, 
caption, 
geom = "colh", 
stat = "identity", 
position = "stackv", 
show.legend = NA, 
show.help = NULL, 
inherit = TRUE, 
environment = parent.frame()
)

gf_propsh(
  object = NULL,
gformula = NULL,
data = NULL,
  ..., 
alpha, 
color, 
fill, 
group, 
linetype, 
size, 
xlab = "proportion", 
ylab, 
title, 
subtitle, 
caption, 
geom = "barh", 

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

gformula A formula, typically with shape \( \sim x \). \( y \sim x \) is also possible, but typically using one of \( \text{gf\_col()} \), \( \text{gf\_props()} \), or \( \text{gf\_percents()} \) is preferable to using this formula shape.) Faceting can be achieved by including | in the formula.

data The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to \( \text{ggplot()} \).
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \text{fortify()} for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x,10)`).

Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with `attribute = value`, (b) ggplot2 aesthetics to be mapped with `attribute = ~ expression`, or (c) attributes of the layer as a whole, which are set with `attribute = value`.

**alpha**  
Opacity (0 = invisible, 1 = opaque).

**color**  
A color or a formula used for mapping color.

**fill**  
A color for filling, or a formula used for mapping fill.

**group**  
Used for grouping.

**linetype**  
A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

**size**  
A numeric size or a formula used for mapping size.

**width**  
Width of the bars.

**xlab**  
Label for x-axis. See also `gf_labs()`.

**ylab**  
Label for y-axis. See also `gf_labs()`.

**title**  
Title, sub-title, and caption for the plot. See also `gf_labs()`.

**subtitle**  
Title, sub-title, and caption for the plot. See also `gf_labs()`.

**caption**  
Title, sub-title, and caption for the plot. See also `gf_labs()`.

**geom**  
Override the default connection between `geom_bar()` and `stat_count()`.

**stat**  
Override the default connection between `geom_bar()` and `stat_count()`.

**position**  
Position adjustment, either as a string, or the result of a call to a position adjustment function.

**show.legend**  
logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

**show.help**  
If `TRUE`, display some minimal help.

**inherit**  
A logical indicating whether default attributes are inherited.

**environment**  
An environment in which to look for variables not found in data.

**denom**  
A formula, the right hand side of which describes the denominators used for computing proportions and percents. These are computed after the stat has been applied to the data and should refer to variables available at that point. See the examples.

**Value**

type: gg object
Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in `gformula`. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

`ggplot2::geom_bar()`

Examples

```r
gf_bar(~substance, data = mosaicData::HELPrct)
gf_bar(~substance, data = mosaicData::HELPrct, fill = ~sex)
gf_bar(~substance,
   data = mosaicData::HELPrct, fill = ~sex,
   position = position_dodge())
# gf_counts() is another name for gf_bar()
gf_counts(~substance,
   data = mosaicData::HELPrct, fill = ~sex,
   position = position_dodge())
# gf_probs() and gf_percents() use proportions or percentages instead of counts
# use denom to control which denominators are used.
gf_probs(~substance,
   data = mosaicData::HELPrct, fill = ~sex,
   position = position_dodge())
gf_probs(substance ~ .,
   data = mosaicData::HELPrct, fill = ~sex,
   position = position_dodge(),
   orientation = 'y')
gf_props(substance ~ .,
   data = mosaicData::HELPrct, fill = ~sex,
   position = position_dodgev(),
   orientation = 'y')
gf_percents(~substance,
   data = mosaicData::HELPrct, fill = ~sex,
   position = position_dodge())
```
gf_barh

Formula interface to geom_barh()

Description

Horizontal version of geom_bar().

Usage

gf_barh(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...
)
Arguments

- **object**: When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

- **gformula**: A formula, typically with shape ~ x. (y ~ x is also possible, but typically using one of `gf_col()`, `gf_props()`, or `gf_percent()` is preferable to using this formula shape.) Faceting can be achieved by including | in the formula.

- **data**: The data to be displayed in this layer. There are three options:
  - If NULL, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a `formula` (e.g., `~ head(.x, 10)`).

- **...**: Additional arguments. Typically these are (a) `ggplot2` aesthetics to be set with `attribute = value`, (b) `ggplot2` aesthetics to be mapped with `attribute = ~ expression`, or (c) attributes of the layer as a whole, which are set with `attribute = value`.

- **alpha**: Opacity (0 = invisible, 1 = opaque).

- **color**: A color or a formula used for mapping color.

- **fill**: A color for filling, or a formula used for mapping fill.

- **group**: Used for grouping.
linetype
A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

size
A numeric size or a formula used for mapping size.

width
Width of the bars.

xlab
Label for x-axis. See also gf_labs().

ylab
Label for y-axis. See also gf_labs().

title
Title, sub-title, and caption for the plot. See also gf_labs().

subtitle
Title, sub-title, and caption for the plot. See also gf_labs().
caption
Title, sub-title, and caption for the plot. See also gf_labs().

geom
A character string naming the geom used to make the layer.

stat
Override the default connection between geom_bar() and stat_count().

position
Position adjustment, either as a string, or the result of a call to a position adjustment function.

show.legend
logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

show.help
If TRUE, display some minimal help.

inherit
A logical indicating whether default attributes are inherited.

environment
An environment in which to look for variables not found in data.

Value
a gg object

Horizontal Geoms
There are two ways to obtain "horizontal" geoms: (1) The ggstance package provides a set of "horizontal" geoms and positions; (2) Thee ggplot2 now provides an orientation argument for "native" horizontal geoms and positions. ggformula supports both.

Specifying plot attributes
Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.

Evaluation
Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.
gf_bin2d

Formula interface to geom_bin2d()

Description

geom_bin2d() uses ggplot2::stat_bin2d() to bin the data before using gf_tile() to display the results.

Usage

gf_bin2d(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
fill,
group,
linetype,
size,
xlab,
ylab,
title,
subtitle,
caption,
geom = "tile",
stat = "bin2d",
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

Arguments

object
When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula
A formula with shape y ~ x. Faceting can be achieved by including | in the formula.
data
A data frame with the variables to be plotted.
...
Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha
Opacity (0 = invisible, 1 = opaque).
color
A color or a formula used for mapping color.
fill
A color for filling, or a formula used for mapping fill.
group
Used for grouping.
linetype
A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
size
A numeric size or a formula used for mapping size.
xlab
Label for x-axis. See also gf_labs().
ylab
Label for y-axis. See also gf_labs().
title
Title, sub-title, and caption for the plot. See also gf_labs().
subtitle
Title, sub-title, and caption for the plot. See also gf_labs().
caption
Title, sub-title, and caption for the plot. See also gf_labs().
geom
A character string naming the geom used to make the layer.
stat
A character string naming the stat used to make the layer.
Positional attributes (a.k.a, aesthetics) are specified using the formula in `gformula`. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

`ggplot2::geom_bin2d()`, `gf_tile()`

Examples

```R
gf_bin2d(eruptions ~ waiting, data = faithful, bins = 15) %>%
gf_refine(scale_fill_viridis_c(begin = 0.1, end = 0.9))
```
gf_blank

Formula interface to geom_blank()

Description

The blank geom draws nothing, but can be a useful way of ensuring common scales between different plots. See expand_limits() for more details.

Usage

gf_blank(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "blank",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

gf_frame(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "blank",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape y ~ x. Faceting can be achieved by including | in the formula.
data A data frame with the variables to be plotted.
... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
xlab Label for x-axis. See also gf_labs().
ylab Label for y-axis. See also gf_labs().
title Title, sub-title, and caption for the plot. See also gf_labs().
subtitle Title, sub-title, and caption for the plot. See also gf_labs().
caption Title, sub-title, and caption for the plot. See also gf_labs().
geom A character string naming the geom used to make the layer.
stat A character string naming the stat used to make the layer.
position Either a character string naming the position function used for the layer or a position object returned from a call to a position function.
show.legend A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.
show.help If TRUE, display some minimal help.
inherit A logical indicating whether default attributes are inherited.
environment An environment in which to look for variables not found in data.

Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.
gf_boxplot

See Also
ggplot2::geom_blank()

Examples

gf_point((c(0, 1)) ~ (c(0, 5)))
gf_frame((c(0, 1)) ~ (c(0, 5)))
gf_blank((c(0, 1)) ~ (c(0, 5)))

# gf_blank() can be used to expand the view
gf_point((c(0, 1)) ~ (c(0, 5))) %>%
gf_blank((c(0, 3)) ~ (c(-2, 7)))

gf_boxplot

Formula interface to geom_boxplot()

Description

The boxplot compactly displays the distribution of a continuous variable. It visualises five summary statistics (the median, two hinges and two whiskers), and all "outlying" points individually.

Usage

gf_boxplot(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha,
  color,
  fill,
  group,
  linetype,
  size,
  coef,
  outlier.color = NULL,
  outlier.fill = NULL,
  outlier.shape = 19,
  outlier.size = 1.5,
  outlier.stroke = 0.5,
  outlier.alpha = NULL,
  notch = FALSE,
  notchwidth = 0.5,
  varwidth = FALSE,
  xlab,
  ylab,
  title,
Arguments

**object**  When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

**gformula**  A formula with shape \( y \sim x \). Faceting can be achieved by including | in the formula.

**data**  The data to be displayed in this layer. There are three options:

- If NULL, the default, the data is inherited from the plot data as specified in the call to **ggplot()**.
- A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See **fortify()** for which variables will be created.
- A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. \(~ \text{head}(., 10)\)).

**...**  Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.

**alpha**  Opacity (0 = invisible, 1 = opaque).

**color**  A color or a formula used for mapping color.

**fill**  A color for filling, or a formula used for mapping fill.

**group**  Used for grouping.

**linetype**  A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

**size**  A numeric size or a formula used for mapping size.

**coef**  Length of the whiskers as multiple of IQR. Defaults to 1.5.

**outlier.color**  Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.

In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.

Sometimes it can be useful to hide the outliers, for example when overlaying the raw data points on top of the boxplot. Hiding the outliers can be achieved by setting **outlier.shape** = NA. Importantly, this does not remove the outliers,
it only hides them, so the range calculated for the y-axis will be the same with outliers shown and outliers hidden.

**outlier.fill**

Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.

In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.

Sometimes it can be useful to hide the outliers, for example when overlaying the raw data points on top of the boxplot. Hiding the outliers can be achieved by setting `outlier.shape = NA`. Importantly, this does not remove the outliers, it only hides them, so the range calculated for the y-axis will be the same with outliers shown and outliers hidden.

**outlier.shape**

Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.

In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.

Sometimes it can be useful to hide the outliers, for example when overlaying the raw data points on top of the boxplot. Hiding the outliers can be achieved by setting `outlier.shape = NA`. Importantly, this does not remove the outliers, it only hides them, so the range calculated for the y-axis will be the same with outliers shown and outliers hidden.

**outlier.size**

Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.

In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.

Sometimes it can be useful to hide the outliers, for example when overlaying the raw data points on top of the boxplot. Hiding the outliers can be achieved by setting `outlier.shape = NA`. Importantly, this does not remove the outliers, it only hides them, so the range calculated for the y-axis will be the same with outliers shown and outliers hidden.

**outlier.stroke**

Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.

In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.

Sometimes it can be useful to hide the outliers, for example when overlaying the raw data points on top of the boxplot. Hiding the outliers can be achieved by setting `outlier.shape = NA`. Importantly, this does not remove the outliers, it only hides them, so the range calculated for the y-axis will be the same with outliers shown and outliers hidden.

**outlier.alpha**

Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.

In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.

Sometimes it can be useful to hide the outliers, for example when overlaying the raw data points on top of the boxplot. Hiding the outliers can be achieved by setting `outlier.shape = NA`. Importantly, this does not remove the outliers, it only hides them, so the range calculated for the y-axis will be the same with outliers shown and outliers hidden.
it only hides them, so the range calculated for the y-axis will be the same with
outliers shown and outliers hidden.

notch If FALSE (default) make a standard box plot. If TRUE, make a notched box plot.
Notches are used to compare groups; if the notches of two boxes do not overlap,
this suggests that the medians are significantly different.

notchwidth For a notched box plot, width of the notch relative to the body (defaults to
notchwidth = 0.5).

varwidth If FALSE (default) make a standard box plot. If TRUE, boxes are drawn with
widths proportional to the square-roots of the number of observations in the
groups (possibly weighted, using the weight aesthetic).

dxlab Label for x-axis. See also gf_labs().
ylab Label for y-axis. See also gf_labs().
title Title, sub-title, and caption for the plot. See also gf_labs().
subtitle Title, sub-title, and caption for the plot. See also gf_labs().
caption Title, sub-title, and caption for the plot. See also gf_labs().
geom Use to override the default connection between geom_boxplot and stat_boxplot.
stat Use to override the default connection between geom_boxplot and stat_boxplot.
position Position adjustment, either as a string, or the result of a call to a position adjust-
ment function.
show.legend logical. Should this layer be included in the legends? NA, the default, includes if
any aesthetics are mapped. FALSE never includes, and TRUE always includes. It
can also be a named logical vector to finely select the aesthetics to display.
show.help If TRUE, display some minimal help.
inherit A logical indicating whether default attributes are inherited.
environment An environment in which to look for variables not found in data.

Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and
mapping of additional attributes can be done through the use of additional arguments. Attributes
can be set can be set using arguments of the form attribute = value or mapped using arguments
of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid().
This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel
more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the
right thing when formulas are created on the fly, but might not be the right thing if formulas created
in one environment are used to create plots in another.
gf_boxploth

References

See Also
ggplot2::geom_boxplot(), fivenum(), df_stats()

Examples
gf_boxplot(age ~ substance, data = mosaicData::HELPrct)
gf_boxplot(age ~ substance, data = mosaicData::HELPrct, varwidth = TRUE)
gf_boxplot(age ~ substance, data = mosaicData::HELPrct, color = ~sex)
gf_boxplot(age ~ substance,
    data = mosaicData::HELPrct,
    color = ~sex, outlier.color = "gray50"
)
# longer whiskers
gf_boxplot(age ~ substance,
    data = mosaicData::HELPrct,
    color = ~sex, coef = 2
)

# Note: width for boxplots is full width of box.
# For jittering, it is the half-width.
gf_boxplot(age ~ substance | sex,
    data = mosaicData::HELPrct,
    coef = 5, width = 0.4
)
# move boxplots away a bit by adjusting dodge
gf_boxplot(age ~ substance,
    data = mosaicData::HELPrct,
    color = ~sex, position = position_dodge(width = 0.9)
)

gf_boxploth

Formula interface to geom_boxploth()

Description
Horizontal version of geom_boxplot().

Usage
gf_boxploth(  
    object = NULL,
    gformula = NULL,
    data = NULL,
gf_boxploth

..., alpha, color, fill, group, linetype, size, coef, outlier.color = NULL, outlier.fill = NULL, outlier.shape = 19, outlier.size = 1.5, outlier.stroke = 0.5, outlier.alpha = NULL, notch = FALSE, notchwidth = 0.5, varwidth = FALSE, xlab, ylab, title, subtitle, caption, geom = "boxploth", stat = "boxploth", position = "dodgev", show.legend = NA, show.help = NULL, inherit = TRUE, environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape y ~ x. Faceting can be achieved by including | in the formula.
data The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
...

... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute =
alpha  Opacity (0 = invisible, 1 = opaque).
color   A color or a formula used for mapping color.
fill    A color for filling, or a formula used for mapping fill.
group   Used for grouping.
linetype A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
size    A numeric size or a formula used for mapping size.
coef    Length of the whiskers as multiple of IQR. Defaults to 1.5.
outlier.color Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.
          In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.
outlier.fill Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.
          In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.
          Sometimes it can be useful to hide the outliers, for example when overlaying the raw data points on top of the boxplot. Hiding the outliers can be achieved by setting outlier.shape = NA. Importantly, this does not remove the outliers, it only hides them, so the range calculated for the y-axis will be the same with outliers shown and outliers hidden.
outlier.shape Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.
          In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.
outlier.size Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.
          In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.
outlier.stroke Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.
          In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.
outlier.alpha Default aesthetics for outliers. Set to NULL to inherit from the aesthetics used for the box.
          In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.
          Sometimes it can be useful to hide the outliers, for example when overlaying the raw data points on top of the boxplot. Hiding the outliers can be achieved by setting outlier.shape = NA. Importantly, this does not remove the outliers, it only hides them, so the range calculated for the y-axis will be the same with outliers shown and outliers hidden.
gf_boxplot

notch  If FALSE (default) make a standard box plot. If TRUE, make a notched box plot. Notches are used to compare groups; if the notches of two boxes do not overlap, this suggests that the medians are significantly different.

notchwidth  For a notched box plot, width of the notch relative to the body (defaults to notchwidth = 0.5).

varwidth  If FALSE (default) make a standard box plot. If TRUE, boxes are drawn with widths proportional to the square-roots of the number of observations in the groups (possibly weighted, using the weight aesthetic).

xlab  Label for x-axis. See also gf_labs().

ylab  Label for y-axis. See also gf_labs().

title  Title, sub-title, and caption for the plot. See also gf_labs().

subtitle  Title, sub-title, and caption for the plot. See also gf_labs().

caption  Title, sub-title, and caption for the plot. See also gf_labs().

geom  A character string naming the geom used to make the layer.

stat  Use to override the default connection between geom_boxplot and stat_boxplot.

position  Position adjustment, either as a string, or the result of a call to a position adjustment function.

show.legend  logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

show.help  If TRUE, display some minimal help.

inherit  A logical indicating whether default attributes are inherited.

environment  An environment in which to look for variables not found in data.

Value

a gg object

Horizontal Geoms

There are two ways to obtain "horizontal" geoms: (1) The ggstance package provides a set of "horizontal" geoms and positions; (2) The ggplot2 now provides an orientation argument for "native" horizontal geoms and positions. ggformula supports both.

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.
Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

`ggstance::geom_boxploth()`, `fivenum()`, `df_stats()`

Examples

```r
gf_boxploth(sex ~ age, data = mosaicData::HELPrct, varwidth = TRUE)
gf_boxploth(sex ~ age, data = mosaicData::HELPrct, varwidth = TRUE, orientation = 'y')
gf_boxploth(substance ~ age, data = mosaicData::HELPrct, color = ~sex)
  # move boxplots away a bit by adjusting dodge
gf_boxploth(substance ~ age,
  data = mosaicData::HELPrct, color = ~sex,
  position = position_dodgev(height = 0.9)
)
  # gf_boxplot guesses horizontal because substance is categorical
  gf_boxplot(substance ~ age,
    data = mosaicData::HELPrct, color = ~sex,
    position = position_dodge(width = 0.9)
  )
gf_boxploth(substance ~ age, data = mosaicData::HELPrct, color = ~sex, outlier.color = "gray50")
  # longer whiskers
  gf_boxploth(substance ~ age, data = mosaicData::HELPrct, color = ~sex, coef = 2)
  # Note: height for boxplots is full width of box.
  # For jittering, it is the half-height.
  gf_boxploth(substance ~ age | sex, data = mosaicData::HELPrct, coef = 5, height = 0.4) %>%
    gf_jitter(height = 0.2, alpha = 0.3)

  # combining boxplots and histograms
  gf_histogram(~eruptions, data = faithful) %>%
    gf_boxploth(0 ~ eruptions, alpha = 0, width = 2)
  gf_histogram(~eruptions, data = faithful) %>%
    gf_boxploth(-2 ~ eruptions, alpha = 0, width = 2)
  gf_histogram(~eruptions, data = faithful) %>%
    gf_boxploth(32 ~ eruptions, alpha = 0, width = 2)
```

---

**gf_col**  
Formula interface to `geom_col()`

Description

There are two types of bar charts: `geom_bar()` and `geom_col()`. `geom_bar()` makes the height of the bar proportional to the number of cases in each group (or if the weight aesthetic is supplied, the sum of the weights). If you want the heights of the bars to represent values in the data, use `geom_col()` instead. `geom_bar()` uses `stat_count()` by default: it counts the number of cases at each x position. `geom_col()` uses `stat_identity()`: it leaves the data as is.
Usage

gf_col(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha, 
  color, 
  fill, 
  group, 
  linetype, 
  size, 
  xlab, 
  ylab, 
  title, 
  subtitle, 
  caption, 
  geom = "col", 
  stat = "identity", 
  position = "stack", 
  show.legend = NA, 
  show.help = NULL, 
  inherit = TRUE, 
  environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape y ~ x. Faceting can be achieved by including | in the formula.
data A data frame with the variables to be plotted.
... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha Opacity (0 = invisible, 1 = opaque).
color A color or a formula used for mapping color.
fill A color for filling, or a formula used for mapping fill.
group Used for grouping.
linetype A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
size A numeric size or a formula used for mapping size.
xlab Label for x-axis. See also gf_labs().
ylab  Label for y-axis. See also \textit{gf\_labs()}.

title  Title, sub-title, and caption for the plot. See also \textit{gf\_labs()}.

subtitle Title, sub-title, and caption for the plot. See also \textit{gf\_labs()}.

caption Title, sub-title, and caption for the plot. See also \textit{gf\_labs()}.

geom  A character string naming the geom used to make the layer.

stat  A character string naming the stat used to make the layer.

position Either a character string naming the position function used for the layer or a position object returned from a call to a position function.

show.legend  A logical indicating whether this layer should be included in the legends. \texttt{NA}, the default, includes layer in the legends if any of the attributes of the layer are mapped.

show.help  If \texttt{TRUE}, display some minimal help.

inherit  A logical indicating whether default attributes are inherited.

environment An environment in which to look for variables not found in data.

Value

a \texttt{gg} object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in \texttt{gformula}. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form \texttt{attribute = value} or mapped using arguments of the form \texttt{attribute = \sim expression}.

In formulas of the form \texttt{A \mid B}, \texttt{B} will be used to form facets using \texttt{facet\_wrap()} or \texttt{facet\_grid()}. This provides an alternative to \texttt{gf\_facet\_wrap()} and \texttt{gf\_facet\_grid()} that is terser and may feel more familiar to users of \texttt{lattice}.

Evaluation

Evaluation of the \texttt{ggplot2} code occurs in the environment of \texttt{gformula}. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

\texttt{ggplot2::geom\_col()}

Examples

\begin{verbatim}
SomeData <- data.frame(
  group = LETTERS[1:3],
  count = c(20, 25, 18)
)
gf_col(count \sim group, data = SomeData)
\end{verbatim}
# A Pareto chart

```r
if (require(dplyr) && require(mosaicData)) {
  HELPct %>%
  group_by(substance) %>%
  summarise(count = n()) %>%
  ungroup() %>%
  dplyr::arrange(-count) %>%
  mutate(
    cumcount = cumsum(count),
    substance = reorder(substance, -count)
  ) %>%
  gf_col(count ~ substance, fill = "skyblue") %>%
  gf_point(cumcount ~ substance) %>%
  gf_line(cumcount ~ substance, group = 1) %>%
  gf_refine(
    scale_y_continuous(sec.axis = sec_axis(~ . / nrow(HELPct)))
  )
}
```

---

**gf_contour**

**Formula interface to geom_contour() and geom_contour_filled()**

**Description**

**ggplot2** can not draw true 3D surfaces, but you can use `geom_contour()`, `geom_contour_filled()`, and `geom_tile()` to visualise 3D surfaces in 2D. To specify a valid surface, the data must contain x, y, and z coordinates, and each unique combination of x and y can appear exactly once. Contouring tends to work best when x and y form a (roughly) evenly spaced grid. If your data is not evenly spaced, you may want to interpolate to a grid before visualising, see `geom_density_2d()`.

**Usage**

```r
gf_contour(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  xlab, 
  ylab, 
  title, 
  subtitle, 
  caption, 
  geom = "contour", 
  stat = "contour", 
  position = "identity", 
  show.legend = NA, 
  show.help = NULL, 
)```
gf_contour

```
inherit = TRUE,
environment = parent.frame()
```

gf_contour_filled(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...

  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "contour_filled",
  stat = "contour_filled",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

**Arguments**

- **object** When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

- **gformula** A formula with shape \(y \sim x\). Faceting can be achieved by including \( | \) in the formula.

- **data** The data to be displayed in this layer. There are three options:
  - If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x, 10)`).

- **...** Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with `attribute = value`, (b) ggplot2 aesthetics to be mapped with `attribute = ~ expression`, or (c) attributes of the layer as a whole, which are set with `attribute = value`.

- **xlab** Label for x-axis. See also `gf_labs()`.

- **ylab** Label for y-axis. See also `gf_labs()`.

- **title** Title, sub-title, and caption for the plot. See also `gf_labs()`.

- **subtitle** Title, sub-title, and caption for the plot. See also `gf_labs()`.
caption
Title, sub-title, and caption for the plot. See also `gf_labs()`.

geom
The geometric object to use display the data

stat
The statistical transformation to use on the data for this layer, as a string.

position
Position adjustment, either as a string, or the result of a call to a position adjustment function.

show.legend
logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

show.help
If TRUE, display some minimal help.

inherit
A logical indicating whether default attributes are inherited.

environment
An environment in which to look for variables not found in `data`.

Value
a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in `gformula`. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

`ggplot2::geom_contour(), gf_density_2d()`

Examples

gf_density_2d(eruptions ~ waiting, data = faithful, alpha = 0.5, color = "navy") %>%
gf_contour(density ~ waiting + eruptions, data = faithful, bins = 10, color = "red")
gf_contour_filled(density ~ waiting + eruptions, data = faithful, bins = 10,
  show.legend = FALSE) %>%
gf_jitter(eruptions ~ waiting, data = faithful, color = "white", alpha = 0.5,
  inherit = FALSE)
**gf_count**  
*Formula interface to geom_count()*

**Description**

This is a variant `geom_point()` that counts the number of observations at each location, then maps the count to point area. It useful when you have discrete data and overplotting.

**Usage**

```r
gf_count(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha,
  color,
  fill,
  group,
  shape,
  size,
  stroke,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "point",
  stat = "sum",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
```

**Arguments**

- `object` When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
- `gformula` A formula with shape `y ~ x`. Faceting can be achieved by including `|` in the formula.
- `data` A data frame with the variables to be plotted.
- `...` Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with `attribute = value`, (b) ggplot2 aesthetics to be mapped with `attribute ~ expression`, or (c) attributes of the layer as a whole, which are set with `attribute = value`.
alpha Opacity (0 = invisible, 1 = opaque).
color A color or a formula used for mapping color.
fill A color for filling, or a formula used for mapping fill.
group Used for grouping.
shape An integer or letter shape or a formula used for mapping shape.
size A numeric size or a formula used for mapping size.
stroke A numeric size of the border or a formula used to map stroke.
xlab Label for x-axis. See also gf_labs().
ylab Label for y-axis. See also gf_labs().
title Title, sub-title, and caption for the plot. See also gf_labs().
subtitle Title, sub-title, and caption for the plot. See also gf_labs().
caption Title, sub-title, and caption for the plot. See also gf_labs().
geom A character string naming the geom used to make the layer.
stat A character string naming the stat used to make the layer.
position Either a character string naming the position function used for the layer or a position object returned from a call to a position function.
show.legend A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.
show.help If TRUE, display some minimal help.
inherit A logical indicating whether default attributes are inherited.
environment An environment in which to look for variables not found in data.

Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.
gf_crossbar

See Also

ggplot2::geom_count()

Examples

# Best used in conjunction with scale_size_area which ensures that
# counts of zero would be given size 0. This doesn’t make much difference
# here because the smallest count is already close to 0.

gf_count(hwy ~ cty, data = mpg, alpha = 0.3) %>%
  gf_refine(scale_size_area())

---

gf_crossbar

Formula interface to geom_crossbar()

Description

Various ways of representing a vertical interval defined by x, ymin and ymax. Each case draws a single graphical object.

Usage

gf_crossbar(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha,
  color,
  group,
  linetype,
  size,
  fatten = 2.5,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "crossbar",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
gf_crossbarh(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...
  alpha,
  color,
  group,
  linetype,
  size,
  fatten = 2.5,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "crossbarh",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape y + ymin + ymax ~ x. Faceting can be achieved by including | in the formula.
data The data to be displayed in this layer. There are three options:
  If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
  A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
  A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
...
  Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha Opacity (0 = invisible, 1 = opaque).
color A color or a formula used for mapping color.
group Used for grouping.
linetype A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping 
linetype.
size A numeric size or a formula used for mapping size.
fatten A multiplicative factor used to increase the size of the middle bar in `geom_crossbar()` 
and the middle point in `geom_pointrange()`.
xlab Label for x-axis. See also `gf_labs()`.
ylab Label for y-axis. See also `gf_labs()`.
title Title, sub-title, and caption for the plot. See also `gf_labs()`.
subtitle Title, sub-title, and caption for the plot. See also `gf_labs()`.
caption Title, sub-title, and caption for the plot. See also `gf_labs()`.
geom A character string naming the geom used to make the layer.
stat The statistical transformation to use on the data for this layer, as a string.
position Position adjustment, either as a string, or the result of a call to a position adjust-
dment function.
show.legend logical. Should this layer be included in the legends? NA, the default, includes if 
any aesthetics are mapped. FALSE never includes, and TRUE always includes. It 
can also be a named logical vector to finely select the aesthetics to display.
show.help If TRUE, display some minimal help.
inherit A logical indicating whether default attributes are inherited.
environment An environment in which to look for variables not found in data.

Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in `gformula`. Setting and 
mapping of additional attributes can be done through the use of additional arguments. Attributes 
can be set can be set using arguments of the form `attribute = value` or mapped using arguments 
of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. 
This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel 
more familiar to users of `lattice`.

Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the 
right thing when formulas are created on the fly, but might not be the right thing if formulas created 
in one environment are used to create plots in another.

See Also

`ggplot2::geom_crossbar()`
Examples

```r
if (require(mosaicData) && require(dplyr)) {
  HELP2 <- HELPrct %>%
    group_by(substance, sex) %>%
    summarise(
      mean.age = mean(age),
      median.age = median(age),
      max.age = max(age),
      min.age = min(age),
      sd.age = sd(age),
      lo = mean.age - sd.age,
      hi = mean.age + sd.age
    )

  gf_jitter(age ~ substance, data = HELPrct,
    alpha = 0.7, width = 0.2, height = 0, color = "skyblue") %>%
  gf_pointrange(mean.age + lo + hi ~ substance, data = HELP2) %>%
  gf_facet_grid(~sex)

  gf_jitter(age ~ substance, data = HELPrct,
    alpha = 0.7, width = 0.2, height = 0, color = "skyblue") %>%
  gf_errorbar(lo + hi ~ substance, data = HELP2, inherit = FALSE) %>%
  gf_facet_grid(~sex)

  gf_jitter(age ~ substance, data = HELPrct,
    alpha = 0.7, width = 0.2, height = 0, color = "skyblue") %>%
  gf_crossbar(mean.age + lo + hi ~ substance, data = HELP2,
    fill = "transparent") %>%
  gf_facet_grid(~sex)

  gf_jitter(substance ~ age, data = HELPrct,
    alpha = 0.7, height = 0.2, width = 0, color = "skyblue") %>%
  gf_crossbarh(substance ~ mean.age + lo + hi, data = HELP2,
    fill = "transparent", color = "red") %>%
  gf_facet_grid(~sex)
}
```

gf_curve

Formula interface to geom_curve()

Description

`geom_segment()` draws a straight line between points (x, y) and (xend, yend). `geom_curve` draws a curved line. See the underlying drawing function `grid::curveGrob()` for the parameters that control the curve.
Usage

gf_curve(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...
  alpha,
  color,
  group,
  linetype,
  size,
  curvature = 0.5,
  angle = 90,
  ncp = 5,
  arrow = NULL,
  lineend = "butt",
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "curve",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

Arguments

object     When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

gformula  A formula with shape \( y + yend \sim x + xend \).

data       The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.

A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. \( \sim \text{head(.x,10)} \)).

...       Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute =
~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.

alpha

Opacity (0 = invisible, 1 = opaque).

color

A color or a formula used for mapping color.

group

Used for grouping.

linetype

A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

size

A numeric size or a formula used for mapping size.

curvature

A numeric value giving the amount of curvature. Negative values produce left-hand curves, positive values produce right-hand curves, and zero produces a straight line.

angle

A numeric value between 0 and 180, giving an amount to skew the control points of the curve. Values less than 90 skew the curve towards the start point and values greater than 90 skew the curve towards the end point.

ncp

The number of control points used to draw the curve. More control points creates a smoother curve.

arrow

specification for arrow heads, as created by arrow().

lineend

Line end style (round, butt, square).

xlab

Label for x-axis. See also gf_labs().

ylab

Label for y-axis. See also gf_labs().

title

Title, sub-title, and caption for the plot. See also gf_labs().

subtitle

Title, sub-title, and caption for the plot. See also gf_labs().

caption

Title, sub-title, and caption for the plot. See also gf_labs().

geom

A character string naming the geom used to make the layer.

stat

The statistical transformation to use on the data for this layer, as a string.

position

Position adjustment, either as a string, or the result of a call to a position adjustment function.

show.legend

logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

show.help

If TRUE, display some minimal help.

inherit

A logical indicating whether default attributes are inherited.

environment

An environment in which to look for variables not found in data.

Value

a gg object
Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in `gformula`. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

`ggplot2::geom_curve()`

Examples

```r
D <- data.frame(x1 = 2.62, x2 = 3.57, y1 = 21.0, y2 = 15.0)
gf_point(mpg ~ wt, data = mtcars) %>%
gf_curve(y1 + y2 ~ x1 + x2, data = D, color = "navy") %>%
gf_segment(y1 + y2 ~ x1 + x2, data = D, color = "red")
```

Description

Computes and draws a kernel density estimate, which is a smoothed version of the histogram and is a useful alternative when the data come from an underlying smooth distribution. The only difference between `gf_dens()` and `gf_density()` is the default geom used to show the density curve: `gf_density()` uses an area geom (which can be filled). `gf_dens()` using a line geom (which cannot be filled).

Usage

```r
gf_density(  
  object = NULL,  
  gformula = NULL,  
  data = NULL,  
  ...,  
  alpha = 0.5,  
  color,  
  fill,  
```
gf_density

  group,
  linetype,
  size,
  kernel = "gaussian",
  n = 512,
  trim = FALSE,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "area",
  stat = "density",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

gf_dens(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha = 0.5,
  color,
  fill = NA,
  group,
  linetype,
  size,
  kernel = "gaussian",
  n = 512,
  trim = FALSE,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "line",
  stat = "density",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
gf_density

gf_dens2(
    object = NULL,
gformula = NULL,
data = NULL,
    ..., 
    alpha = 0.5,
color,
    fill = NA,
group,
    linetype,
    size,
    kernel = "gaussian",
n = 512,
    trim = FALSE,
xlab,
ylab,
title,
    subtitle,
caption,
geom = "density_line",
    stat = "density",
    position = "identity",
    show.legend = NA,
    show.help = NULL,
    inherit = TRUE,
environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

gformula A formula with shape ~ x. Faceting can be achieved by including | in the formula.

data The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).

... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha

Opacity (0 = invisible, 1 = opaque).

color

A color or a formula used for mapping color.

fill

A color for filling, or a formula used for mapping fill.

group

Used for grouping.

linetype

A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

size

A numeric size or a formula used for mapping size.

kernel

Kernel. See list of available kernels in \texttt{density()}.

n

number of equally spaced points at which the density is to be estimated, should be a power of two, see \texttt{density()} for details.

trim

If FALSE, the default, each density is computed on the full range of the data. If TRUE, each density is computed over the range of that group: this typically means the estimated x values will not line-up, and hence you won’t be able to stack density values. This parameter only matters if you are displaying multiple densities in one plot or if you are manually adjusting the scale limits.

xlab

Label for x-axis. See also \texttt{gf_labs()}.

ylab

Label for y-axis. See also \texttt{gf_labs()}.

title

Title, sub-title, and caption for the plot. See also \texttt{gf_labs()}.

subtitle

Title, sub-title, and caption for the plot. See also \texttt{gf_labs()}.

caption

Title, sub-title, and caption for the plot. See also \texttt{gf_labs()}.

geom

Use to override the default connection between \texttt{geom_density} and \texttt{stat_density}.

stat

Use to override the default connection between \texttt{geom_density} and \texttt{stat_density}.

position

Position adjustment, either as a string, or the result of a call to a position adjustment function.

show.legend

logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

show.help

If TRUE, display some minimal help.

inherit

A logical indicating whether default attributes are inherited.

environment

An environment in which to look for variables not found in data.

Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form \texttt{attribute = value} or mapped using arguments of the form \texttt{attribute = ~ expression}.

In formulas of the form \texttt{A | B}, \texttt{B} will be used to form facets using \texttt{facet_wrap()} or \texttt{facet_grid()}. This provides an alternative to \texttt{gf_facet_wrap()} and \texttt{gf_facet_grid()} that is terser and may feel more familiar to users of \texttt{lattice}. 
Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

gf_ash(), ggplot2::geom_density()

Examples

gf_dens()
data(penguins, package = "palmerpenguins")
gf_density(~bill_length_mm, fill = ~species, data = penguins)
gf_dens(~bill_length_mm, color = ~species, data = penguins)
gf_dens2(~bill_length_mm, color = ~species, fill = ~species, data = penguins)
gf_freqpoly(~bill_length_mm, color = ~species, data = penguins, bins = 15)
  # Chaining in the data
data(penguins, package = "palmerpenguins")
penguins %>% gf_dens(~bill_length_mm, color = ~species)
  # horizontal orientation
penguins %>% gf_dens(bill_length_mm ~ ., color = ~species)

gf_density_2d

Formula interface to geom_density_2d() and geom_density_2d_filled()

Description

Perform a 2D kernel density estimation using MASS::kde2d() and display the results with contours. This can be useful for dealing with overplotting. This is a 2D version of geom_density(). geom_density_2d() draws contour lines, and geom_density_2d_filled() draws filled contour bands.

Usage

gf_density_2d(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha, 
  color, 
  group, 
  linetype, 
  size, 
  contour = TRUE, 
  n = 100, 
)
gf_density_2d

h = NULL,
lineend = "butt",
linejoin = "round",
linemitre = 1,
xlab,
ylab,
title,
subtitle,
caption,
geom = "density_2d",
stat = "density_2d",
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

gf_density_2d_filled(
object = NULL,
gformula = NULL,
data = NULL,

...,  
alpha,  
color,  
group,  
linetype,  
size,  
contour = TRUE,  
n = 100,  
h = NULL,  
lineend = "butt",  
linejoin = "round",  
linemitre = 1,
xlab,
ylab,
title,
subtitle,
caption,
geom = "density_2d_filled",
stat = "density_2d_filled",
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)
gf_density2d(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  group,
  linetype,
  size,
  contour = TRUE,
  n = 100,
  h = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 1,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "density2d",
  stat = "density2d",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

gf_density2d_filled(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  group,
  linetype,
  size,
  contour = TRUE,
  n = 100,
  h = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 1,
  xlab,
  ylab,
title,
subtitle,
caption,
geom = "density2d_filled",
stat = "density_2d_filled",
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape y ~ x. Faceting can be achieved by including | in the formula.
data The data to be displayed in this layer. There are three options:
  If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
  A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
  A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
...
  Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value,
  (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alphas Opacity (0 = invisible, 1 = opaque).
color A color or a formula used for mapping color.
group Used for grouping.
linetype A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
size A numeric size or a formula used for mapping size.
contour If TRUE, contour the results of the 2d density estimation.
n Number of grid points in each direction.
h Bandwidth (vector of length two). If NULL, estimated using MASS::bandwidth.nrd().
lineend Line end style (round, butt, square).
linejoin Line join style (round, mitre, bevel).
linemtine Line mitre limit (number greater than 1).
xlab Label for x-axis. See also gf_labs().
ylab  Label for y-axis. See also `gf_labs()`.
title  Title, sub-title, and caption for the plot. See also `gf_labs()`.
subtitle  Title, sub-title, and caption for the plot. See also `gf_labs()`.
caption  Title, sub-title, and caption for the plot. See also `gf_labs()`.
geom  Use to override the default connection between `geom_density_2d` and `stat_density_2d`.
stat  Use to override the default connection between `geom_density_2d` and `stat_density_2d`.
position  Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.legend  logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
show.help  If TRUE, display some minimal help.
inherit  A logical indicating whether default attributes are inherited.
environment  An environment in which to look for variables not found in data.

Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in `gformula`. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

`ggplot2::geom_density_2d()`

Examples

```r
gf_jitter(avg_drinks ~ age,
   alpha = 0.2, data = mosaicData::HELPrct,
   width = 0.4, height = 0.4
)
```

```r
%>%
gf_density_2d(avg_drinks ~ age, data = mosaicData::HELPrct)
gf_density_2d_filled(avg_drinks ~ age, data = mosaicData::HELPrct, show.legend = FALSE)
```
gf_dist

gf_jitter(avg_drinks ~ age,
  alpha = 0.3, data = mosaicData::HELPct,
  width = 0.4, height = 0.4,
  color = "white"
)

gf_jitter(avg_drinks ~ age,
  alpha = 0.2, data = mosaicData::HELPct,
  width = 0.4, height = 0.4
)

gf_density2d(avg_drinks ~ age, data = mosaicData::HELPct, show.legend = FALSE)

gf_density2d_filled(avg_drinks ~ age, data = mosaicData::HELPct, show.legend = FALSE)

gf_jitter(avg_drinks ~ age,
  alpha = 0.4, data = mosaicData::HELPct,
  width = 0.4, height = 0.4,
  color = "white"
)

gf_dist

Plot distributions

Description

Create a layer displaying a probability distribution.

Usage

gf_dist(
  object = ggplot(),
  dist,
  ...,
  xlim = NULL,
  kind = c("density", "cdf", "qq", "qqstep", "histogram"),
  resolution = 5000L,
  params = NULL
)

Arguments

object a gg object.

dist A character string providing the name of a distribution. Any distribution for
  which the functions with names formed by prepending "d", "p", or "q" to dist
  exist can be used.

... additional arguments passed both to the distribution functions and to the layer.
Note: Possible ambiguities using params or by preceding plot argument with plot_.

xlim A numeric vector of length 2 providing lower and upper bounds for the portion
  of the distribution that will be displayed. The default is to attempt to determine
  reasonable bounds using quantiles of the distribution.
gf_dotplot

kind: One of "density", "cdf", "qq", "qqstep", or "histogram" describing what kind of plot to create.
resolution: An integer specifying the number of points to use for creating the plot.
params: A list of parameters for the distribution.

Examples

gf_dhistogram(~ rnorm(100), bins = 20) %>%
gf_dist("norm", color = "red")
  
  # shading tails -- but see pdist() for this
  gf_dist("norm", fill = ~ (abs(x) <= 2), geom = "area")
  gf_dist("norm", color = "red", kind = "cdf")
  gf_dist("norm", fill = "red", kind = "histogram")
  gf_dist("norm", color = "red", kind = "qqstep", resolution = 25) %>%
  gf_dist("norm", color = "black", kind = "qq", resolution = 25, size = 2, alpha = 0.5)
  
  # size is used as parameter for binomial distribution
  gf_dist("binom", size = 20, prob = 0.25)
  
  # If we want to adjust size argument for plots, we have two choices:
  gf_dist("binom", size = 20, prob = 0.25, plot_size = 2)
  gf_dist("binom", params = list(size = 20, prob = 0.25), size = 2)

---
gf_dotplot

**Formula interface to geom_dotplot()**

Description

Scatterplots in ggformula.

Usage

gf_dotplot(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  fill,
  group,
  binwidth = NULL,
  binaxis = "x",
  method = "dotdensity",
  binpositions = "bygroup",
  stackdir = "up",
  stackratio = 1,
  dotsize = 1,
  stackgroups = FALSE,
gf_dotplot

origin = NULL,
right = TRUE,
width = 0.9,
drop = FALSE,
xlab,
ylab,
title,
subtitle,
caption,
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape ~ x. Faceting can be achieved by including | in the formula.
data A data frame with the variables to be plotted.
... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha Opacity (0 = invisible, 1 = opaque).
color A color or a formula used for mapping color.
fill A color for filling, or a formula used for mapping fill.
group Used for grouping.
binwidth When method is "dotdensity", this specifies maximum bin width. When method is "histodot", this specifies bin width. Defaults to 1/30 of the range of the data
binaxis The axis to bin along, "x" (default) or "y"
method "dotdensity" (default) for dot-density binning, or "histodot" for fixed bin widths (like stat_bin)
binpositions When method is "dotdensity", "bygroup" (default) determines positions of the bins for each group separately. "all" determines positions of the bins with all the data taken together; this is used for aligning dot stacks across multiple groups.
stackdir which direction to stack the dots. "up" (default), "down", "center", "centerwhole" (centered, but with dots aligned)
stackratio how close to stack the dots. Default is 1, where dots just touch. Use smaller values for closer, overlapping dots.
dotsize The diameter of the dots relative to binwidth, default 1.
stackgroups should dots be stacked across groups? This has the effect that position = "stack" should have, but can’t (because this geom has some odd properties).

origin When method is "histodot", origin of first bin

right When method is "histodot", should intervals be closed on the right (a, b], or not [a, b)

width When binaxis is "y", the spacing of the dot stacks for dodging.

drop If TRUE, remove all bins with zero counts

xlab Label for x-axis. See also gf_labs().

ylab Label for y-axis. See also gf_labs().

title Title, sub-title, and caption for the plot. See also gf_labs().

subtitle Title, sub-title, and caption for the plot. See also gf_labs().

caption Title, sub-title, and caption for the plot. See also gf_labs().

position Either a character string naming the position function used for the layer or a position object returned from a call to a position function.

show.legend A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.

show.help If TRUE, display some minimal help.

inherit A logical indicating whether default attributes are inherited.

environment An environment in which to look for variables not found in data.

Details

There are two basic approaches: dot-density and histodot. With dot-density binning, the bin positions are determined by the data and binwidth, which is the maximum width of each bin. See Wilkinson (1999) for details on the dot-density binning algorithm. With histodot binning, the bins have fixed positions and fixed widths, much like a histogram.

When binning along the x axis and stacking along the y axis, the numbers on y axis are not meaningful, due to technical limitations of ggplot2. You can hide the y axis, as in one of the examples, or manually scale it to match the number of dots.

Value

a gg object

Warning

Dotplots in ggplot2 (and hence in ggformula) often require some fiddling because the default y-axis is meaningless and the ideal size of the dots depends on the aspect ratio of the plot.
Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

References


See Also

ggplot2::geom_dotplot()

data(penguins, package = "palmerpenguins")
gf_dotplot(~bill_length_mm, fill = ~species, data = penguins)

gf_ecdf

Description

The empirical cumulative distribution function (ECDF) provides an alternative visualization of distribution. Compared to other visualizations that rely on density (like histograms or density plots) the ECDF doesn’t require any tuning parameters and handles both continuous and categorical variables. The downside is that it requires more training to accurately interpret, and the underlying visual tasks are somewhat more challenging.

Usage

gf_ecdf(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  group,
Arguments

object  When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

gformula  A formula with shape y ~ x. Faceting can be achieved by including | in the formula.

data  The data to be displayed in this layer. There are three options:

- If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
- A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
- A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).

...  Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

group  Used for grouping.

pad  If TRUE, pad the ecdf with additional points (-Inf, 0) and (Inf, 1)
n  if NULL, do not interpolate. If not NULL, this is the number of points to interpolate with.

xlab  Label for x-axis. See also gf_labs().

ylab  Label for y-axis. See also gf_labs().

title  Title, sub-title, and caption for the plot. See also gf_labs().

subtitle  Title, sub-title, and caption for the plot. See also gf_labs().

caption  Title, sub-title, and caption for the plot. See also gf_labs().

geom  The geometric object to use display the data
The statistical transformation to use on the data for this layer, as a string.

Position adjustment, either as a string, or the result of a call to a position adjustment function.

logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

If TRUE, display some minimal help.

A logical indicating whether default attributes are inherited.

An environment in which to look for variables not found in data.

Examples

```r
Data <- data.frame(
  x = c(rnorm(100, 0, 1), rnorm(100, 0, 3), rt(100, df = 3)),
  g = gl(3, 100, labels = c("N(0, 1)", "N(0, 3)", "T(df = 3)"))
)
gf_ecdf(~ x, data = Data)
# Don't go to positive/negative infinity
gf_ecdf(~ x, data = Data, pad = FALSE)

# Multiple ECDFs
gf_ecdf(~ x, data = Data, color = ~ g)
```

Description

Formula interface to `ggplot2::stat_ellipse()`.

Usage

```r
gf_ellipse(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha,
  color,
  group,
  type = "t",
  level = 0.95,
  segments = 51,
  xlab,
  ylab,
  title,
```
Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape \( y \sim x \). Faceting can be achieved by including \( | \) in the formula.
data A data frame with the variables to be plotted.
... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with `attribute = value`, (b) ggplot2 aesthetics to be mapped with `attribute = ~ expression`, or (c) attributes of the layer as a whole, which are set with `attribute = value`.
alpha Opacity \((0 = \text{invisible}, 1 = \text{opaque})\).
color A color or a formula used for mapping color.
group Used for grouping.
type The type of ellipse. The default "t" assumes a multivariate t-distribution, and "norm" assumes a multivariate normal distribution. "euclid" draws a circle with the radius equal to level, representing the euclidean distance from the center. This ellipse probably won't appear circular unless `coord_fixed()` is applied.
level The level at which to draw an ellipse, or, if `type="euclid"`, the radius of the circle to be drawn.
segments The number of segments to be used in drawing the ellipse.
xlab Label for x-axis. See also `gf_labs()`.
ylab Label for y-axis. See also `gf_labs()`.
title Title, sub-title, and caption for the plot. See also `gf_labs()`.
subtitle Title, sub-title, and caption for the plot. See also `gf_labs()`.
caption Title, sub-title, and caption for the plot. See also `gf_labs()`.
geom Geom for drawing ellipse. Note: "polygon" allows fill; "path" does not; on the other hand, "path" allows alpha to be applied to the border, while "polygon" applies alpha only to the interior.
stat A character string naming the stat used to make the layer.
position Either a character string naming the position function used for the layer or a position object returned from a call to a position function.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>show.legend</td>
<td>A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.</td>
</tr>
<tr>
<td>show.help</td>
<td>If TRUE, display some minimal help.</td>
</tr>
<tr>
<td>inherit</td>
<td>A logical indicating whether default attributes are inherited.</td>
</tr>
<tr>
<td>environment</td>
<td>An environment in which to look for variables not found in data.</td>
</tr>
</tbody>
</table>

**See Also**

`ggplot2::stat_ellipse()`

**Examples**

```r
gf_ellipse()
gf_point(eruptions ~ waiting, data = faithful) %>%
  gf_ellipse(alpha = 0.5)

gf_point(eruptions ~ waiting, data = faithful, color = ~ (eruptions > 3)) %>%
  gf_ellipse(alpha = 0.5)

gf_point(eruptions ~ waiting, data = faithful, color = ~ (eruptions > 3)) %>%
  gf_ellipse(type = "norm", linetype = ~ "norm") %>%
  gf_ellipse(type = "t", linetype = ~ "t")

gf_point(eruptions ~ waiting, data = faithful, color = ~ (eruptions > 3)) %>%
  gf_ellipse(type = "norm", linetype = ~ "norm") %>%
  gf_ellipse(type = "eulid", linetype = ~ "eulid", level = 3) %>%
  gf_refine(coord_fixed())

# Use geom = "polygon" to enable fill
gf_point(eruptions ~ waiting, data = faithful, fill = ~ (eruptions > 3)) %>%
  gf_ellipse(geom = "polygon", alpha = 0.3, color = "black")

gf_point(eruptions ~ waiting, data = faithful, fill = ~ (eruptions > 3)) %>%
  gf_ellipse(geom = "polygon", alpha = 0.3) %>%
  gf_ellipse(alpha = 0.3, color = "black")

gf_ellipse(eruptions ~ waiting, data = faithful, show.legend = FALSE, alpha = 0.3, fill = ~ (eruptions > 3), geom = "polygon") %>%
  gf_ellipse(level = 0.68, geom = "polygon", alpha = 0.3) %>%
  gf_point(data = faithful, color = ~ (eruptions > 3), show.legend = FALSE)
```

---

**gf_empty**

Create an "empty" plot

**Description**

This is primarily useful as a way to start a sequence of piped plot layers.
Usage

gf_empty(environment = parent.frame())

Arguments

environment: An environment passed to `ggplot2::ggplot()`

Value

A plot with now layers.

Examples

gf_empty()
data(penguins, package = "palmerpenguins")
gf_empty() %>%
gf_point(bill_length_mm ~ bill_depth_mm, data = penguins, color = ~species)

---

**gf_errorbar**

*Formula interface to geom_errorbar()*

Description

For each x value, `geom_ribbon()` displays a y interval defined by `ymin` and `ymax`. `geom_area()` is a special case of `geom_ribbon()`, where the `ymin` is fixed to 0 and `y` is used instead of `ymax`.

Usage

gf_errorbar(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha,
  color,
  group,
  linetype,
  size,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "errorbar",
  stat = "identity",
  position = "identity",
  show.legend = NA,


```r
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
```

**Arguments**

- **object**: When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

- **gformula**: A formula with shape $y_{\min} + y_{\max} \sim x$. Faceting can be achieved by including | in the formula.

- **data**: The data to be displayed in this layer. There are three options:

  - If NULL, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x, 10)`).

- **...**: Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with `attribute = value`, (b) ggplot2 aesthetics to be mapped with `attribute = ~ expression`, or (c) attributes of the layer as a whole, which are set with `attribute = value`.

- **alpha**: Opacity (0 = invisible, 1 = opaque).

- **color**: A color or a formula used for mapping color.

- **group**: Used for grouping.

- **linetype**: A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

- **size**: A numeric size or a formula used for mapping size.

- **xlab**: Label for x-axis. See also `gf_labs()`.

- **ylab**: Label for y-axis. See also `gf_labs()`.

- **title**: Title, sub-title, and caption for the plot. See also `gf_labs()`.

- **subtitle**: Title, sub-title, and caption for the plot. See also `gf_labs()`.

- **caption**: Title, sub-title, and caption for the plot. See also `gf_labs()`.

- **geom**: A character string naming the geom used to make the layer.

- **stat**: The statistical transformation to use on the data for this layer, as a string.

- **position**: Position adjustment, either as a string, or the result of a call to a position adjustment function.

- **show.legend**: logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

- **show.help**: If TRUE, display some minimal help.

- **inherit**: A logical indicating whether default attributes are inherited.

- **environment**: An environment in which to look for variables not found in data.
See Also
  
  `ggplot2::geom_errorbar()`

Examples

```r
if (require(mosaicData) && require(dplyr)) {
  HELP2 <- HELPrct %>%
    group_by(substance, sex) %>%
    summarise(
      mean.age = mean(age),
      median.age = median(age),
      max.age = max(age),
      min.age = min(age),
      sd.age = sd(age),
      lo = mean.age - sd.age,
      hi = mean.age + sd.age
    )

  gf_jitter(age ~ substance, data = HELPrct,
    alpha = 0.5, width = 0.2, height = 0, color = "skyblue") %>%
  gf_pointrange(mean.age + lo + hi ~ substance, data = HELP2,
    inherit = FALSE) %>%
  gf_facet_grid(~sex)

  gf_jitter(age ~ substance, data = HELPrct,
    alpha = 0.5, width = 0.2, height = 0, color = "skyblue") %>%
  gf_errorbar(lo + hi ~ substance, data = HELP2, inherit = FALSE) %>%
  gf_facet_grid(~sex)

  gf_jitter(age ~ substance, data = HELPrct,
    alpha = 0.5, width = 0.2, height = 0, color = "skyblue") %>%
  gf_boxplot(age ~ substance, data = HELPrct, color = "red") %>%
  gf_crossbar(mean.age + lo + hi ~ substance, data = HELP2) %>%
  gf_facet_grid(~sex)
}
```

`gf_errorbarh`  
Formula interface to `geom_errorbarh()`

Description

A rotated version of `geom_errorbar()`.  

Usage

```r
gf_errorbarh(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...
)```

gf_errorbarh

alpha, color, group, linetype, size, xlab, ylab, title, subtitle, caption, geom = "errorbarh", stat = "identity", position = "identity", show.legend = NA, show.help = NULL, inherit = TRUE, environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape y ~ x + xmin + xmax. Faceting can be achieved by including | in the formula.
data The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
...
Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha Opacity (0 = invisible, 1 = opaque).
color A color or a formula used for mapping color.
group Used for grouping.
linetype A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
size A numeric size or a formula used for mapping size.
xlab Label for x-axis. See also gf_labs().
ylab  Label for y-axis. See also `gf_labs()`.
title  Title, sub-title, and caption for the plot. See also `gf_labs()`.
subtitle  Title, sub-title, and caption for the plot. See also `gf_labs()`.
caption  Title, sub-title, and caption for the plot. See also `gf_labs()`.
geom  A character string naming the geom used to make the layer.
stat  The statistical transformation to use on the data for this layer, as a string.
position  Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.legend  logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
show.help  If TRUE, display some minimal help.
inherit  A logical indicating whether default attributes are inherited.
environment  An environment in which to look for variables not found in data.

Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in `gformula`. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

`ggplot2::geom_errorbarh()`

Examples

```r
if (require(dplyr)) {
  HELP2 <- mosaicData::HELPrct %>%
    group_by(substance, sex) %>%
    summarise(
      mean.age = mean(age),
      median.age = median(age),
```
max.age = max(age),
min.age = min(age),
sd.age = sd(age),
lo = mean.age - sd.age,
hi = mean.age + sd.age
)

gf_jitter(substance ~ age, data = mosaicData::HELPrct,
  alpha = 0.5, height = 0.2, width = 0, color = "skyblue") %>%
gf_errorbarh(substance ~ lo + hi, data = HELP2, inherit = FALSE) %>%
gf_facet_grid(~sex)

gf_jitter(age ~ substance, data = mosaicData::HELPrct,
  alpha = 0.5, width = 0.2, height = 0, color = "skyblue") %>%
gf_errorbar(lo + hi ~ substance, data = HELP2, inherit = FALSE) %>%
gf_facet_grid(~sex)

Description

These functions provide more control over faceting than is possible using the formula interface.

Usage

gf_facet_wrap(object, ...)
gf_facet_grid(object, ...)

Arguments

object  A ggplot object
...

Additional arguments passed to facet_wrap() or facet_grid(). This typically includes an unnamed formula argument describing the facets. scales and space are additional useful arguments. See the examples.

See Also

ggplot2::facet_grid(), ggplot2::facet_wrap().

Examples

gf_histogram(~avg_drinks, data = mosaicData::HELPrct) %>%
gf_facet_grid(~substance)
gf_histogram(~avg_drinks, data = mosaicData::HELPrct) %>%
gf_facet_grid(~substance, scales = "free")
gf_histogram(~avg_drinks, data = mosaicData::HELPrct) %>%
gf_facet_grid(~substance, scales = "free", space = "free")
gf_line(births ~ date, data = mosaicData::Births, color = ~wday) %>%
gf_facet_wrap(~year, scales = "free_x", nrow = 5) %>%
gf_theme(
  axis.title.x = element_blank(),
  axis.text.x = element_blank(), axis.ticks.x = element_blank()
) %>%
gf_labs(color = "Day")

---

**gf_fitdistr**

*Plot density function based on fit to data*

**Description**

`MASS::fitdistr()` is used to fit coefficients of a specified family of distributions and the resulting density curve is displayed.

**Usage**

```
gf_fitdistr(  
  object = NULL,  
  gformula = NULL,  
  data = NULL,  
  ...,  
  dist = "dnorm",  
  start = NULL,  
  alpha,  
  color,  
  fill,  
  group,  
  linetype,  
  size,  
  xlab,  
  ylab,  
  title,  
  subtitle,  
  caption,  
  geom = "path",  
  stat = "fitdistr",  
  position = "identity",  
  show.legend = NA,  
  show.help = NULL,  
  inherit = FALSE,  
  environment = parent.frame()
)
```
Arguments

object
When chaining, this holds an object produced in the earlier portions of the chain.
Most users can safely ignore this argument. See examples.

gformula
A formula with shape \(~ x\) used to specify the data to be fit to a family of distributions.

data
A data frame containing the variable to be fitted.

...
Additional arguments

dist
A quoted name of a distribution function. See `mosaicCore::fit_distr_fun()` for more details about allowable distributions.

start
Starting value(s) for the search for MLE. (See `MASS::fitdistr`.)

alpha
Opacity (0 = invisible, 1 = opaque).

color
A color or a formula used for mapping color.

fill
A color for filling, or a formula used for mapping fill.

group
Used for grouping.

linetype
A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

size
A numeric size or a formula used for mapping size.

xlab
Label for x-axis. See also `gf_labs()`.

ylab
Label for y-axis. See also `gf_labs()`.

title
Title, sub-title, and caption for the plot. See also `gf_labs()`.

subtitle
Title, sub-title, and caption for the plot. See also `gf_labs()`.

caption
Title, sub-title, and caption for the plot. See also `gf_labs()`.

geom
A character string naming the geom used to make the layer.

stat
A character string naming the stat used to make the layer.

position
Either a character string naming the position function used for the layer or a position object returned from a call to a position function.

show.legend
A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.

show.help
If TRUE, display some minimal help.

inherit
A logical indicating whether default attributes are inherited.

environment
An environment in which to look for variables not found in data.

Value

a gg object
Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

mosaicCore::fit_distr_fun()

Examples

gf_fitdistr(~length, data = mosaicData::KidsFeet, inherit = FALSE) %>%
gf_dhistogram(~length, data = mosaicData::KidsFeet, binwidth = 0.5, alpha = 0.25)

gf_dhistogram(~length, data = mosaicData::KidsFeet, binwidth = 0.5, alpha = 0.25) %>%
gf_fitdistr()

set.seed(12345)
Dat <- data.frame(g = rgamma(500, 3, 10), f = rf(500, df1 = 3, df2 = 47))
gf_dhistogram(~g, data = Dat) %>%
gf_fitdistr(dist = "dgamma")

gf_dhistogram(~g, data = Dat) %>%
g_fun(mosaicCore::fit_distr_fun(~g, data = Dat, dist = "dgamma"))

gf_dhistogram(~f, data = Dat) %>%
gf_fitdistr(dist = "df", start = list(df1 = 2, df2 = 50))

# fitted parameters are default argument values
args(
  mosaicCore::fit_distr_fun(~f, 
    data = Dat, dist = "df", 
    start = list(df1 = 2, df2 = 50)
  )
)

args(mosaicCore::fit_distr_fun(~g, data = Dat, dist = "dgamma"))
Description

Visualise the distribution of a single continuous variable by dividing the x axis into bins and counting the number of observations in each bin. Histograms (geom_histogram()) display the counts with bars; frequency polygons (geom_freqpoly()) display the counts with lines. Frequency polygons are more suitable when you want to compare the distribution across the levels of a categorical variable.

Usage

gf_freqpoly(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha,
  color,
  group,
  linetype,
  size,
  binwidth, bins, center, boundary,
  xlab, ylab, title, subtitle, caption,
  geom = "path",
  stat = "bin",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

A formula with shape ~ x or y ~ x. Faceting can be achieved by including | in the formula.
The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).

Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.

Opacity (0 = invisible, 1 = opaque).
A color or a formula used for mapping color.
Used for grouping.
A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
A numeric size or a formula used for mapping size.
The width of the bins. Can be specified as a numeric value or as a function that calculates width from unscaled x. Here, "unscaled x" refers to the original x values in the data, before application of any scale transformation. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use the number of bins in bins, covering the range of the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data.
The bin width of a date variable is the number of days in each time; the bin width of a time variable is the number of seconds.
Number of bins. Overridden by binwidth. Defaults to 30.
bin position specifiers. Only one, center or boundary, may be specified for a single plot. center specifies the center of one of the bins. boundary specifies the boundary between two bins. Note that if either is above or below the range of the data, things will be shifted by the appropriate integer multiple of binwidth. For example, to center on integers use binwidth = 1 and center = 0, even if 0 is outside the range of the data. Alternatively, this same alignment can be specified with binwidth = 1 and boundary = 0.5, even if 0.5 is outside the range of the data.
bin position specifiers. Only one, center or boundary, may be specified for a single plot. center specifies the center of one of the bins. boundary specifies the boundary between two bins. Note that if either is above or below the range of the data, things will be shifted by the appropriate integer multiple of binwidth. For example, to center on integers use binwidth = 1 and center = 0, even if 0 is outside the range of the data. Alternatively, this same alignment can be specified with binwidth = 1 and boundary = 0.5, even if 0.5 is outside the range of the data.
**gf_freqpoly**

- **xlab**: Label for x-axis. See also `gf_labs()`.
- **ylab**: Label for y-axis. See also `gf_labs()`.
- **title**: Title, sub-title, and caption for the plot. See also `gf_labs()`.
- **subtitle**: Title, sub-title, and caption for the plot. See also `gf_labs()`.
- **caption**: Title, sub-title, and caption for the plot. See also `gf_labs()`.
- **geom**: Use to override the default connection between `geom_histogram()`/`geom_freqpoly()` and `stat_bin()`.
- **stat**: Use to override the default connection between `geom_histogram()`/`geom_freqpoly()` and `stat_bin()`.
- **position**: Position adjustment, either as a string, or the result of a call to a position adjustment function.
- **show.legend**: logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
- **show.help**: If TRUE, display some minimal help.
- **inherit**: A logical indicating whether default attributes are inherited.
- **environment**: An environment in which to look for variables not found in data.

**Value**

- a gg object

**Specifying plot attributes**

Positional attributes (a.k.a, aesthetics) are specified using the formula in `gformula`. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

**Evaluation**

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

**See Also**

`ggplot2::geom_freqpoly()`
Examples

```r
data(penguins, package = "palmerpenguins")
gf_histogram(~ bill_length_mm | species, alpha = 0.2, data = penguins, bins = 20) %>%
gf_freqpoly(~bill_length_mm, data = penguins, color = ~species, bins = 20)
gf_freqpoly(~bill_length_mm, color = ~species, data = penguins, bins = 20)
gf_dens(~bill_length_mm, data = penguins, color = "navy") %>%
gf_freqpoly(stat(density) ~ bill_length_mm,
  data = penguins,
  color = "red", bins = 20
)
```

---

gf_function

Layers displaying graphs of functions

Description

These functions provide two different interfaces for creating a layer that contains the graph of a function.

Usage

```r
gf_function(object = NULL, fun, xlim, ..., inherit = FALSE)
gf_fun(object = NULL, formula, xlim, ..., inherit = FALSE)
```

Arguments

- **object**
  When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
- **fun**
  A function.
- **xlim**
  A numeric vector providing the extent of the x-axis when creating the first layer in a plot. Ignored when creating a subsequent layer.
- **...**
  Other arguments such as `position="dodge"`.
- **inherit**
  A logical indicating whether default attributes are inherited.
- **formula**
  A formula describing a function. See examples and `mosaicCore::makeFun()`.

Examples

```r
gf_function(fun = sqrt, xlim = c(0, 10))
gf_dhistogram(~age, data = mosaicData::HELPrct, binwidth = 3, alpha = 0.6) %>%
gf_function(  
  fun = stats::dnorm,
  args = list(mean = mean(mosaicData::HELPrct$age), sd = sd(mosaicData::HELPrct$age)),
  color = "red"
)
gf_fun(5 + 3 * cos(10 * x) ~ x, xlim = c(0, 2))
# Utility bill is quadratic in month?
```
gf_function_2d

```r
f <- makeFun(lm(totalbill ~ poly(month, 2), data = mosaicData::Utilities))
gf_point(totalbill ~ month, data = mosaicData::Utilities, alpha = 0.6) %>%
gf_fun(f(m) ~ m, color = "red")
```

---

**gf_function_2d**  
Plot functions of two variables

**Description**

Plot functions of two variables as tile and/or contour plots.

**Usage**

```r
gf_function_2d(
  object = NULL,
  fun = identity,
  xlim = NULL,
  ylim = NULL,
  ..., 
  tile = TRUE,
  contour = TRUE,
  resolution = 50
)
```

```r
gf_function2d(
  object = NULL,
  fun = identity,
  xlim = NULL,
  ylim = NULL,
  ..., 
  tile = TRUE,
  contour = TRUE,
  resolution = 50
)
```

```r
gf_function_contour(
  object = NULL,
  fun = identity,
  xlim = NULL,
  ylim = NULL,
  ..., 
  resolution = 50
)
```

```r
gf_function_tile(
  object = NULL,
  fun = identity,
```
gf_function_2d

xlim = NULL,
ylim = NULL,
..., resolution = 50
)

gf_fun_2d(
  object = NULL,
  formula = NULL,
  xlim = NULL,
  ylim = NULL,
  tile = TRUE,
  contour = TRUE,
  ..., resolution = 50
)

gf_fun2d(
  object = NULL,
  formula = NULL,
  xlim = NULL,
  ylim = NULL,
  tile = TRUE,
  contour = TRUE,
  ..., resolution = 50
)

gf_fun_tile(
  object = NULL,
  formula = NULL,
  xlim = NULL,
  ylim = NULL,
  ..., resolution = 50
)

gf_fun_contour(
  object = NULL,
  formula = NULL,
  xlim = NULL,
  ylim = NULL,
  ..., resolution = 50
)

Arguments

object An R object, typically of class "gg".
fun A function of two variables to be plotted.
xlim x limits for generating points to be plotted.
ylim y limits for generating points to be plotted.
... additional arguments passed to `gf_tile()` or `gf_contour()`.
tile A logical indicating whether the tile layer should be drawn.
contour A logical indicating whether the contour layer should be drawn.
resolution A numeric vector of length 1 or 2 specifying the number of grid points at which
the function is evaluated (in each dimension).
formula A formula describing a function of two variables to be plotted. See `mosaic::makeFun()`
for details regarding the conversion from a formula to a function.

Value
A gg plot.

Examples
```
theme_set(theme_bw())
gf_function_2d(fun = function(x, y) sin(2 * x * y), xlim = c(-pi, pi), ylim = c(-pi, pi)) %>%
gf_refine(scale_fill_viridis_c())
gf_function_2d(fun = function(x, y) x + y, contour = FALSE)
gf_function_tile(fun = function(x, y) x * y) %>%
gf_function_contour(fun = function(x, y) x * y, color = "white") %>%
gf_refine(scale_fill_viridis_c())
gf_fun_tile(x * y ~ x + y, xlim = c(-3, 3), ylim = c(-2, 2)) %>%
gf_fun_contour(x * y ~ x + y, color = "white") %>%
gf_refine(scale_fill_viridis_c())
gf_labs(fill = "product")
```

gf_hex **Formula interface to geom_hex()**

Description
Line plots in ggformula. `gf_path()` differs from `gf_line()` in that points are connected in the order in which they appear in data.

Usage
```
gf_hex(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  bins,
  binwidth,
)```
alpha, color, fill, group, size, xlab, ylab, title, subtitle, caption, geom = "hex", stat = "binhex", position = "identity", show.legend = NA, show.help = NULL, inherit = TRUE, environment = parent.frame()

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape y ~ x. Faceting can be achieved by including | in the formula.
data The data to be displayed in this layer. There are three options:
  If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
  A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
  A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).

... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.

bins numeric vector giving number of bins in both vertical and horizontal directions. Set to 30 by default.

binwidth Numeric vector giving bin width in both vertical and horizontal directions. Overrides bins if both set.

alpha Opacity (0 = invisible, 1 = opaque).

color A color or a formula used for mapping color.

fill A color for filling, or a formula used for mapping fill.

group Used for grouping.
### gf_hex

- **size**: A numeric size or a formula used for mapping size.
- **xlab**: Label for x-axis. See also `gf_labs()`.
- **ylab**: Label for y-axis. See also `gf_labs()`.
- **title**: Title, sub-title, and caption for the plot. See also `gf_labs()`.
- **subtitle**: Title, sub-title, and caption for the plot. See also `gf_labs()`.
- **caption**: Title, sub-title, and caption for the plot. See also `gf_labs()`.
- **geom**: Override the default connection between `geom_hex` and `stat_binhex`.
- **stat**: Override the default connection between `geom_hex` and `stat_binhex`.
- **position**: Position adjustment, either as a string, or the result of a call to a position adjustment function.
- **show.legend**: logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
- **show.help**: If TRUE, display some minimal help.
- **inherit**: A logical indicating whether default attributes are inherited.
- **environment**: An environment in which to look for variables not found in data.

### Value

a gg object

### Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in `gformula`. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

### Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

### See Also

`ggplot2::geom_hex()`

### Examples

```r
gf_hex(avg_drinks ~ age, data = mosaicData::HELPPrct, bins = 15) %>%
gf_density2d(avg_drinks ~ age, data = mosaicData::HELPPrct, color = "red", alpha = 0.5)
```
gf_histogram  

Formula interface to geom_histogram()

Description

Count and density histograms in ggformula.

Usage

```r
gf_histogram(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,  
  bins = 25,
  binwidth,
  alpha = 0.5,
  color,
  fill,
  group,
  linetype,
  size,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "bar",
  stat = "bin",
  position = "stack",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
```

gf_dhistogram(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,  
  bins = 25,
  binwidth,
  alpha = 0.5,
  color,
  fill,
  group,
)
gf_histogram

    linetype,
    size,
    xlab,
    ylab,
    title,
    subtitle,
    caption,
    geom = "bar",
    stat = "bin",
    position = "stack",
    show.legend = NA,
    show.help = NULL,
    inherit = TRUE,
    environment = parent.frame()
  )

gf_histogramh(
    object = NULL,
    gformula = NULL,
    data = NULL,
    ...
    bins = 25,
    binwidth,
    alpha = 0.5,
    color,
    fill,
    group,
    linetype,
    size,
    xlab,
    ylab,
    title,
    subtitle,
    caption,
    geom = "barh",
    stat = "binh",
    position = "stackv",
    show.legend = NA,
    show.help = NULL,
    inherit = TRUE,
    environment = parent.frame()
  )

gf_dhistogramh(
    object = NULL,
    gformula = NULL,
    data = NULL,
    ...,
gf_histogram

```
bins = 25,
binwidth,
alpha = 0.5,
color,
fill,
group,
linetype,
size,
xlab,
ylab,
title,
subtitle,
caption,
geom = \"barh\",
stat = \"binh\",
position = \"stackv\",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
```

### Arguments

- **object**
  When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

- **gformula**
  A formula with shape \(~ x\) (or \(~ y \sim x\), but this shape is not generally needed).

- **data**
  The data to be displayed in this layer. There are three options:
  - If **NULL**, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A **data.frame**, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a **data.frame**, and will be used as the layer data. A function can be created from a **formula** (e.g. \(~ \text{head}(x,10)\)).

- **...**
  Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with **attribute = value**, (b) ggplot2 aesthetics to be mapped with **attribute = ~ expression**, or (c) attributes of the layer as a whole, which are set with **attribute = value**.

- **bins**
  Number of bins. Overridden by `binwidth`. Defaults to 30.

- **binwidth**
  The width of the bins. Can be specified as a numeric value or as a function that calculates width from unscaled x. Here, "unscaled x" refers to the original x values in the data, before application of any scale transformation. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use the number of bins in `bins`, covering the range of
the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data.

The bin width of a date variable is the number of days in each time; the bin width of a time variable is the number of seconds.

alpha Opacity (0 = invisible, 1 = opaque).

color A color or a formula used for mapping color.

fill A color for filling, or a formula used for mapping fill.

group Used for grouping.

linetype A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

size A numeric size or a formula used for mapping size.

xlab Label for x-axis. See also gf_labs().

ylab Label for y-axis. See also gf_labs().

title Title, sub-title, and caption for the plot. See also gf_labs().

subtitle Title, sub-title, and caption for the plot. See also gf_labs().

caption Title, sub-title, and caption for the plot. See also gf_labs().

geom Use to override the default connection between geom_histogram()/geom_freqpoly() and stat_bin().

stat Use to override the default connection between geom_histogram()/geom_freqpoly() and stat_bin().

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

show.help If TRUE, display some minimal help.

inherit A logical indicating whether default attributes are inherited.

environment An environment in which to look for variables not found in data.

Value

a gg object

Horizontal Geoms

There are two ways to obtain "horizontal" geoms: (1) The ggstance package provides a set of "horizontal" geoms and positions; (2) Thee ggplot2 now provides an orientation argument for "native" horizontal geoms and positions. ggformula supports both.
Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

`ggplot2::geom_histogram()`

Examples

```r
x <- rnorm(1000)
gf_histogram(~x, bins = 30)
gf_dhistogram(~x, bins = 30)
gf_dhistogram(~x, binwidth = 0.5, center = 0, color = "black")
gf_dhistogram(~x, binwidth = 0.5, boundary = 0, color = "black")
gf_dhistogram(~x, bins = 30) %>%
gf_fitdistr(dist = "dnorm") # see help for gf_fitdistr() for more info.
gf_histogram(~x, fill = ~ (abs(x) <= 2), boundary = 2, binwidth = 0.25)
data(penguins, package = "palmerpenguins")
gf_histogram(~ bill_length_mm | species, data = penguins, binwidth = 0.25)
gf_histogram(~age,
data = mosaicData::HELPrct, binwidth = 5,
fill = "skyblue", color = "black")
) # bins can be adjusted left/right using center or boundary
gf_histogram(~age,
data = mosaicData::HELPrct,
binwidth = 5, fill = "skyblue", color = "black", center = 42.5
)
gf_histogram(~age,
data = mosaicData::HELPrct,
binwidth = 5, fill = "skyblue", color = "black", boundary = 40
)
gf_histogram(age ~ .,
data = mosaicData::HELPrct,
binwidth = 5, fill = "skyblue", color = "black", boundary = 40
)
```
gf_jitter

gf_histogramh(~x, bins = 30)
gf_histogram(x ~ ., bins = 30)
gf_histogramh(x ~ ., bins = 30)
gf_histogramh(x ~ stat(density), bins = 30)
gf_dhistogramh(~x, bins = 30)
gf_dhistogram(x ~ ., bins = 30)
gf_dhistogramh(x ~ ., bins = 30)

---

gf_jitter

*Formula interface to geom_jitter()*

**Description**

Jittered scatter plots in ggformula.

**Usage**

```r
gf_jitter(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha, color, size, shape, fill, width, height,
  group, stroke, xlab, ylab, title, subtitle, caption,
  geom = "point",
  stat = "identity",
  position = "jitter",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
```
Arguments

- **object**: When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
- **gformula**: A formula with shape \( y \sim x \). Faceting can be achieved by including \( | \) in the formula.
- **data**: A data frame with the variables to be plotted.
- **...**: Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with `attribute = value`, (b) ggplot2 aesthetics to be mapped with `attribute = ~ expression`, or (c) attributes of the layer as a whole, which are set with `attribute = value`.
- **alpha**: Opacity (0 = invisible, 1 = opaque).
- **color**: A color or a formula used for mapping color.
- **size**: A numeric size or a formula used for mapping size.
- **shape**: An integer or letter shape or a formula used for mapping shape.
- **fill**: A color for filling, or a formula used for mapping fill.
- **width**: Amount of horizontal jitter.
- **height**: Amount of vertical jitter.
- **group**: Used for grouping.
- **stroke**: A numeric size of the border or a formula used to map stroke.
- **xlab**: Label for x-axis. See also `gf_labs()`.
- **ylab**: Label for y-axis. See also `gf_labs()`.
- **title**: Title, sub-title, and caption for the plot. See also `gf_labs()`.
- **subtitle**: Title, sub-title, and caption for the plot. See also `gf_labs()`.
- **caption**: Title, sub-title, and caption for the plot. See also `gf_labs()`.
- **geom**: A character string naming the geom used to make the layer.
- **stat**: A character string naming the stat used to make the layer.
- **position**: Either a character string naming the position function used for the layer or a position object returned from a call to a position function.
- **show.legend**: A logical indicating whether this layer should be included in the legends. `NA`, the default, includes layer in the legends if any of the attributes of the layer are mapped.
- **show.help**: If `TRUE`, display some minimal help.
- **inherit**: A logical indicating whether default attributes are inherited.
- **environment**: An environment in which to look for variables not found in `data`.

Value

A gg object
Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

`ggplot2::geom_jitter()`, `gf_point()`

Examples

gf_jitter()
# without jitter
gf_point(age ~ sex, alpha = 0.25, data = mosaicData::HELPrct)
# jitter only horizontally

gf_jitter(age ~ sex, alpha = 0.25, data = mosaicData::HELPrct, width = 0.2, height = 0)
# alternative way to get jitter

gf_point(age ~ sex,  
  alpha = 0.25, data = mosaicData::HELPrct,  
  position = "jitter", width = 0.2, height = 0  
)

---

gf_labs

Non-layer functions for gf plots

Description

These functions modify things like labels, limits, scales, etc. for plots ggplot2 plots. They are wrappers around functions in ggplot2 that allow for chaining syntax.

Usage

gf_labs(object, ...)

gf_lims(object, ...)

gf_refine(object, ...)
Arguments

- object: a gg object
- ... additional arguments passed through to the similarly named function in `ggplot2`.

Details

`gf_refine()` provides a mechanism to replace `+` with the chaining operator from `magrittr`. Each of its ... arguments is added in turn to the base plot in `object`. The other functions are thin wrappers around specific `ggplot2` refinement functions and pass their ... arguments through to the similarly named `ggplot2` functions.

Value

a modified gg object

Examples

```r
gf_dens(~cesd, color = ~substance, size = 1.5, data = mosaicData::HELPrct) %>%
gf_labs(
  title = "Center for Epidemiologic Studies Depression measure",
  subtitle = "(at baseline)",
  color = "Abused substance: ",
  x = "CESD score",
  y = "",
  caption = "Source: HELPrct"
) %>%
gf_theme(theme_classic()) %>%
gf_theme(
  axis.text.y = element_blank(),
  legend.position = "top",
  plot.title = element_text(hjust = 0.5, color = "navy"),
  plot.subtitle = element_text(hjust = 0.5, color = "navy", size = 12)
)

gf_point(eruptions ~ waiting, data = faithful, alpha = 0.5)

gf_point(eruptions ~ waiting, data = faithful, alpha = 0.5) %>%
gf_lims(x = c(65, NA), y = c(3, NA))

# modify scales using `gf_refine()`
data(penguins, package = "palmerpenguins")
gf_jitter(bill_length_mm ~ bill_depth_mm, color = ~species, data = penguins) %>%
gf_refine(scale_color_brewer(type = "qual", palette = 3)) %>%
gf_theme(theme_bw())

gf_jitter(bill_length_mm ~ bill_depth_mm, color = ~species, data = penguins) %>%
gf_refine(scale_color_manual(values = c("red", "navy", "limegreen"))) %>%
gf_theme(theme_bw())
```
**gf_line**

Formula interface to `geom_line()` and `geom_path()`

**Description**

Line plots in ggformula. `gf_path()` differs from `gf_line()` in that points are connected in the order in which they appear in data.

**Usage**

```r
gf_line(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  fill,
  group,
  linetype,
  size,
  lineend,
  linejoin,
  linemitre,
  arrow,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "line",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
```

```r
gf_path(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  group,
```
linetype,  
size,  
lineend = "butt",  
linejoin = "round",  
linemitre = 1,  
arrow = NULL,  
xlab,  
ylab,  
title,  
subtitle,  
caption,  
geom = "path",  
stat = "identity",  
position = "identity",  
show.legend = NA,  
show.help = NULL,  
inherit = TRUE,  
environment = parent.frame()
)

**Arguments**

- **object**
  When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

- **gformula**
  A formula with shape y ~ x. Faceting can be achieved by including | in the formula.

- **data**
  A data frame with the variables to be plotted.

- **...**
  Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.

- **alpha**
  Opacity (0 = invisible, 1 = opaque).

- **color**
  A color or a formula used for mapping color.

- **fill**
  A color for filling, or a formula used for mapping fill.

- **group**
  Used for grouping.

- **linetype**
  A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

- **size**
  A numeric size or a formula used for mapping size.

- **lineend**
  Line end style (round, butt, square).

- **linejoin**
  Line join style (round, mitre, bevel).

- **linemitre**
  Line mitre limit (number greater than 1).

- **arrow**
  Arrow specification, as created by `grid::arrow()`.

- **xlab**
  Label for x-axis. See also `gf_labs()`.

- **ylab**
  Label for y-axis. See also `gf_labs()`.
Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in `gformula`. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

`ggplot2::geom_line()`, `gf_point()`

Examples

```r
gf_line()
gf_point(age ~ sex, alpha = 0.25, data = mosaicData::HELPrc)
gf_point(births ~ date, color = ~wday, data = mosaicData::Births78)
# lines make the exceptions stand out more prominently
gf_line(births ~ date, color = ~wday, data = mosaicData::Births78)
gf_path()
if (require(dplyr)) {
```
data.frame(t = seq(1, 10 * pi, length.out = 400)) %>%
mutate(x = t * cos(t), y = t * sin(t)) %>%
gf_path(y ~ x, color = ~t)
}

gf_linerange

Formula interface to geom_linerange() and geom_pointrange()

Description

Various ways of representing a vertical interval defined by x, ymin and ymax. Each case draws a single graphical object.

Usage

gf_linerange(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha,
  color,
  group,
  linetype,
  size,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "linerange",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

gf_pointrange(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha,
  color,
  group,
gf_linerange

linetype,
size,
fatten = 2,
xlab,
ylab,
title,
subtitle,
caption,
geom = "pointrange",
stat = "identity",
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

gf_summary(
    object = NULL,
gformula = NULL,
data = NULL,
...,
alpha,
color,
group,
linetype,
size,
fun.y = NULL,
fun.ymax = NULL,
fun.ymin = NULL,
fun.args = list(),
fatten = 2,
xlab,
ylab,
title,
subtitle,
caption,
geom = "pointrange",
stat = "summary",
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

gf_linerangeh(
    object = NULL,
gf_linerange

gformula = NULL,
data = NULL,
..., 
alpha, 
color, 
group, 
linetype, 
size, 
xlab, 
ylab, 
title, 
subtitle, 
caption, 
geom = "linerangeh", 
stat = "identity", 
position = "identity", 
show.legend = NA, 
show.help = NULL, 
inherit = TRUE, 
environment = parent.frame()
)

gf_pointrangeh(
    object = NULL, 
gformula = NULL, 
data = NULL, 
..., 
alpha, 
color, 
group, 
linetype, 
size, 
xlab, 
ylab, 
title, 
subtitle, 
caption, 
geom = "pointrangeh", 
stat = "identity", 
position = "identity", 
show.legend = NA, 
show.help = NULL, 
inherit = TRUE, 
environment = parent.frame()
)
Arguments

object
When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

gformula
A formula with shape \( y_{\text{min}} + y_{\text{max}} \sim x \). Faceting can be achieved by including | in the formula.

data
The data to be displayed in this layer. There are three options:

- If NULL, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}
- A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{fortify()}
- A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. \( \sim \text{head}(x,10) \)).

Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with \texttt{attribute = value}, (b) ggplot2 aesthetics to be mapped with \texttt{attribute = \sim expression}, or (c) attributes of the layer as a whole, which are set with \texttt{attribute = value}.

alpha
Opacity (0 = invisible, 1 = opaque).

color
A color or a formula used for mapping color.

group
Used for grouping.

linetype
A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

size
A numeric size or a formula used for mapping size.

xlab
Label for x-axis. See also \texttt{gf_labs()}. 

ylab
Label for y-axis. See also \texttt{gf_labs()}. 

title
Title, sub-title, and caption for the plot. See also \texttt{gf_labs()}. 

subtitle
Title, sub-title, and caption for the plot. See also \texttt{gf_labs()}. 

caption
Title, sub-title, and caption for the plot. See also \texttt{gf_labs()}. 

geom
A character string naming the geom used to make the layer.

stat
The statistical transformation to use on the data for this layer, as a string.

position
Position adjustment, either as a string, or the result of a call to a position adjustment function.

show.legend
logical. Should this layer be included in the legends? \texttt{NA}, the default, includes if any aesthetics are mapped. \texttt{FALSE} never includes, and \texttt{TRUE} always includes. It can also be a named logical vector to finely select the aesthetics to display.

show.help
If \texttt{TRUE}, display some minimal help.

inherit
A logical indicating whether default attributes are inherited.

environment
An environment in which to look for variables not found in data.

fatten
A multiplicative factor used to increase the size of the middle bar in \texttt{geom_crossbar()} and the middle point in \texttt{geom_pointrange()}. 


fun.y Deprecated, use the versions specified above instead.
fun.ymax Deprecated, use the versions specified above instead.
fun.ymin Deprecated, use the versions specified above instead.
fun.args Optional additional arguments passed on to the functions.

See Also

ggplot2::geom_linerange()
ggplot2::geom_pointrange()
ggplot2::geom_pointrange(), ggplot2::stat_summary()

Examples

gf_linerange()

gf_ribbon(low_temp + high_temp ~ date,
  data = mosaicData::Weather,
  fill = ~city, alpha = 0.4
) %>%
gf_theme(theme = theme_minimal())
gf_linerange(
  low_temp + high_temp ~ date | city ~ .,
  data = mosaicData::Weather,
  color = ~ ((low_temp + high_temp) / 2)
) %>%
gf_refine(scale_colour_gradientn(colors = rev(rainbow(5)))) %>%
gf_labs(color = "mid-temp")

gf_ribbon(low_temp + high_temp ~ date | city ~ ., data = mosaicData::Weather)

# Chaining in the data
mosaicData::Weather %>%
gf_ribbon(low_temp + high_temp ~ date, alpha = 0.4) %>%
gf_facet_grid(city ~ .)
if (require(mosaicData) && require(dplyr)) {
  HELP2 <- HELPrct %>%
    group_by(substance, sex) %>%
    summarise(
      age = NA,
      mean.age = mean(age),
      median.age = median(age),
      max.age = max(age),
      min.age = min(age),
      sd.age = sd(age),
      lo = mean.age - sd.age,
      hi = mean.age + sd.age
    )

  gf_jitter(age ~ substance, data = HELPrct,
    alpha = 0.5, width = 0.2, height = 0, color = "skyblue") %>%
gf_pointrange(mean.age + lo + hi ~ substance, data = HELP2) %>%
gf_facet_grid(~sex)
gf_jitter(age ~ substance, data = HELPrct, 
  alpha = 0.5, width = 0.2, height = 0, color = "skyblue") %>%
gf_errorbar(lo + hi ~ substance, data = HELP2, inherit = FALSE) %>%
gf_facet_grid(~sex)

# width is defined differently for gf_boxplot() and gf_jitter()
# * for gf_boxplot() it is the full width of the box.
# * for gf_jitter() it is half that -- the maximum amount added or subtracted.
gf_boxplot(age ~ substance, data = HELPrct, width = 0.4) %>%
gf_jitter(width = 0.4, height = 0, color = "skyblue", alpha = 0.5)

gf_boxplot(age ~ substance, data = HELPrct, width = 0.4) %>%
gf_jitter(width = 0.2, height = 0, color = "skyblue", alpha = 0.5)
}
p <- gf_jitter(mpg ~ cyl, data = mtcars, height = 0, width = 0.15); p
p %>% gf_summary(fun.data = "mean_cl_boot", color = "red", size = 2)

# You can supply individual functions to summarise the value at
# each x:
p %>% gf_summary(fun.y = "median", color = "red", size = 2, geom = "point")
p %>% gf_summary(fun.y = "mean", color = "red", size = 2, geom = "point") %>%
gf_summary(fun.y = mean, geom = "line")
p %>%
gf_summary(fun.y = mean, fun.ymin = min, fun.ymax = max, color = "red")
p %>%
gf_summary(fun.ymin = min, fun.ymax = max, color = "red", geom = "linerange")
gf_bar(~ cut, data = diamonds)
gf_col(price ~ cut, data = diamonds, stat = "summary_bin", fun.y = "mean")

# Don’t use gf_lims() to zoom into a summary plot - this throws the
# data away
p <- gf_summary(mpg ~ cyl, data = mtcars, fun.y = "mean", geom = "point")
p %>% gf_lims(y = c(15, 30))

# Instead use coord_cartesian()
p %>% gf_refine(coord_cartesian(ylim = c(15, 30)))
# A set of useful summary functions is provided from the Hmisc package.
p <- gf_jitter(mpg ~ cyl, data = mtcars, width = 0.15, height = 0); p
p %>% gf_summary(fun.data = mean_cl_boot, color = "red")
p %>% gf_summary(fun.data = mean_cl_boot, color = "red", geom = "crossbar")
p %>% gf_summary(fun.data = mean sdl, group = ~ cyl, color = "red", geom = "crossbar", width = 0.3)
p %>% gf_summary(group = ~ cyl, color = "red", geom = "crossbar", width = 0.3, fun.data = mean sdl, fun.args = list(mult = 1))
p %>% gf_summary(fun.data = median_hilow, group = ~ cyl, color = "red", geom = "crossbar", width = 0.3)

# An example with highly skewed distributions:
if (require("ggplot2movies")) {

set.seed(596)
Mov <- movies[sample(nrow(movies), 1000), ]
m2 <- gf_jitter(votes ~ factor(round(rating)), data = Mov, width = 0.15, height = 0, alpha = 0.3)
m2 <- m2 %>%
  gf_summary(fun.data = "mean_cl_boot", geom = "crossbar",
  colour = "red", width = 0.3) %>%
  gf_labs(x = "rating")
m2
# Notice how the overplotting skews off visual perception of the mean
# supplementing the raw data with summary statistics is _very_ important

# Next, we'll look at votes on a log scale.

# Transforming the scale means the data are transformed
# first, after which statistics are computed:
m2 %>%
  gf_refine(scale_y_log10())
# Transforming the coordinate system occurs after the
# statistic has been computed. This means we’re calculating the summary on the raw data
# and stretching the geoms onto the log scale. Compare the widths of the
# standard errors.
m2 %>%
  gf_refine(coord_trans(y="log10"))
}
gf_linerangeh(date ~ low_temp + high_temp | ~city,
  data = mosaicData::Weather,
  color = ~avg_temp )
  %>%
  gf_refine(scale_color_viridis_c(begin = 0.1, end = 0.9, option = "C"))
gf_linerange(date ~ low_temp + high_temp | ~city,
  data = mosaicData::Weather,
  color = ~avg_temp,
  orientation = 'y'
  )
  %>%
  gf_refine(scale_color_viridis_c(begin = 0.1, end = 0.9, option = "C"))
gf_pointrangeh(date ~ avg_temp + low_temp + high_temp | ~city,
  data = Weather,
  color = ~avg_temp
  )
  %>%
  gf_refine(scale_color_viridis_c(begin = 0.1, end = 0.9, option = "C"))

---

gf_plot

Formula interface to ggplot()

Description

Create a new ggplot and (optionally) set default dataset aesthetics mapping.

Usage

gf_plot(...)
Arguments

... arguments that can include data (a data frame or something that can be `ggplot2::fortify()`ed to become one) and aesthetics specified using the following formula notation:

\[
\text{aesthetic} = \sim \text{expression}
\]

See examples.

Value

a gg object

Examples

```r
gf_plot(mtcars, x = ~ wt, y = ~ mpg, color = ~ factor(cyl)) %>%
gf_density_2d() %>%
gf_point()
```

Description

Scatterplots in ggformula.

Usage

```r
gf_point(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  size,
  shape,
  fill,
  group,
  stroke,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "point",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
)```
Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape y ~ x. Faceting can be achieved by including | in the formula.
data A data frame with the variables to be plotted.
... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, (c) attributes of the layer as a whole, which are set with attribute = value, or (d) arguments for the geom, stat, or position function.
alpha Opacity (0 = invisible, 1 = opaque).
color A color or a formula used for mapping color.
size A numeric size or a formula used for mapping size.
shape An integer or letter shape or a formula used for mapping shape.
fill A color for filling, or a formula used for mapping fill.
group Used for grouping.
stroke A numeric size of the border or a formula used to map stroke.
xlab Label for x-axis. See also gf_labs().
ylab Label for y-axis. See also gf_labs().
title, subtitle, caption Title, sub-title, and caption for the plot. See also gf_labs().
geom A character string naming the geom used to make the layer.
stat A character string naming the stat used to make the layer.
position Either a character string naming the position function used for the layer or a position object returned from a call to a position function.
show.legend A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.
show.help If TRUE, display some minimal help.
inherit A logical indicating whether default attributes are inherited.
environment An environment in which to look for variables not found in data.

Value

a gg object
Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

ggplot2::geom_point(), gf_line(), gf_jitter()

Examples

gf_point()
gf_point((10 * ((1:25) %% 10)) ~ ((1:25) %/% 10),
  shape = 1:25,
  fill = "skyblue", color = "navy", size = 4, stroke = 1, data = NA)
gf_point(mpg ~ hp, color = ~cyl, size = ~wt, data = mtcars)
# faceting -- two ways
gf_point(mpg ~ hp, data = mtcars) %>%
  gf_facet_wrap(~am)
gf_point(mpg ~ hp | am, group = ~cyl, data = mtcars)
gf_point(mpg ~ hp | ~am, group = ~cyl, data = mtcars)
gf_point(mpg ~ hp | am ~ ., group = ~cyl, data = mtcars)
# Chaining in the data
mtcars %>% gf_point(mpg ~ wt)

# short cuts for main labels in the plot
gf_point(births ~ date,
  color = ~wday, data = mosaicData::Births78,
  xlab = "Date", ylab = "Number of Live Births",
  title = "Interesting Patterns in the Number of Births",
  subtitle = "(United States, 1978)",
  caption = "Source: mosaicData::Births78")
gf_polygon  

Formula interface to geom_polygon()

Description

Scatterplots in ggformula.

Usage

\[
gf_polygon(  
  object = NULL,  
  gformula = NULL,  
  data = NULL,  
  ...,  
  alpha,  
  color,  
  size,  
  shape,  
  fill,  
  group,  
  stroke,  
  xlab,  
  ylab,  
  title,  
  subtitle,  
  caption,  
  geom = "polygon",  
  stat = "identity",  
  position = "identity",  
  show.legend = NA,  
  show.help = NULL,  
  inherit = TRUE,  
  environment = parent.frame())  
\]

Arguments

- **object**: When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
- **gformula**: A formula with shape \( y \sim x \). Faceting can be achieved by including | in the formula.
- **data**: A data frame with the variables to be plotted.
- **...**: Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, (c) attributes of the layer as a whole, which are set with attribute = value, or (d) arguments for the geom, stat, or position function.
alpha Opacity (0 = invisible, 1 = opaque).

color A color or a formula used for mapping color.

size A numeric size or a formula used for mapping size.

shape An integer or letter shape or a formula used for mapping shape.

fill A color for filling, or a formula used for mapping fill.

group Used for grouping.

stroke A numeric size of the border or a formula used to map stroke.

xlab Label for x-axis. See also gf_labs().

ylab Label for y-axis. See also gf_labs().

title Title, sub-title, and caption for the plot. See also gf_labs().

subtitle Title, sub-title, and caption for the plot. See also gf_labs().

caption Title, sub-title, and caption for the plot. See also gf_labs().

geom A character string naming the geom used to make the layer.

stat A character string naming the stat used to make the layer.

position Either a character string naming the position function used for the layer or a position object returned from a call to a position function.

show.legend A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.

show.help If TRUE, display some minimal help.

inherit A logical indicating whether default attributes are inherited.

environment An environment in which to look for variables not found in data.

Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.
gf_qq

Formula interface to geom_qq()

Description

gf_qq() an gf_qqstep() both create quantile-quantile plots. They differ in how they display the
qq-plot. gf_qq() uses points and gf_qqstep() plots a step function through these points.

Usage

gf_qq(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  group,
  distribution = stats::qnorm,
  dparams = list(),
  xlab,
  ylab,
  title,
  subtitle,
  caption,
gf_qq

  geom = "point",
  stat = "qq",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

gf_qqline(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  group,
  distribution = stats::qnorm,
  dparams = list(),
  linetype = "dashed",
  alpha = 0.7,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "line",
  stat = "qqline",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

gf_qqstep(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  group,
  distribution = stats::qnorm,
  dparams = list(),
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "step",
  stat = "qq",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape ~ sample. Facets can be added using |.
data The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(x,10)).
...
Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
group Used for grouping.
distribution Distribution function to use, if x not specified
dparams Additional parameters passed on to distribution function.
xlab Label for x-axis. See also gf_labs().
ylab Label for y-axis. See also gf_labs().
title Title, sub-title, and caption for the plot. See also gf_labs().
subtitle Title, sub-title, and caption for the plot. See also gf_labs().
caption Title, sub-title, and caption for the plot. See also gf_labs().
geom Use to override the default connection between geom_histogram()/geom_freqpoly() and stat_bin().
stat Use to override the default connection between geom_histogram()/geom_freqpoly() and stat_bin().
position Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
show.help If TRUE, display some minimal help.
**gf_quantile**

- **inherit** A logical indicating whether default attributes are inherited.
- **environment** An environment in which to look for variables not found in data.
- **linetype** A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
- **alpha** Opacity (0 = invisible, 1 = opaque).

**Value**

A gg object

**Specifying plot attributes**

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

**Evaluation**

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

**See Also**

`ggplot2::geom_qq()`

**Examples**

```r
gf_qq(~ rnorm(100))
data(penguins, package = "palmerpenguins")
gf_qq(~ bill_length_mm | species, data = penguins) %>% gf_qqline()
gf_qq(~ bill_length_mm | species, data = penguins) %>% gf_qqline(tail = 0.10)
gf_qq(~ bill_length_mm, color = ~species, data = penguins) %>%
gf_qqstep(~ bill_length_mm, color = ~species, data = penguins)
```

---

**gf_quantile**

*Formula interface to geom_quantile()*

**Description**

This fits a quantile regression to the data and draws the fitted quantiles with lines. This is as a continuous analogue to `geom_boxplot()`.
Usage

gf_quantile(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha, 
  color, 
  group, 
  linetype, 
  size, 
  weight, 
  lineend = "butt", 
  linejoin = "round", 
  linemitre = 1, 
  quantiles, 
  formula, 
  method, 
  method.args, 
  xlab, 
  ylab, 
  title, 
  subtitle, 
  caption, 
  geom = "quantile", 
  stat = "quantile", 
  position = "identity", 
  show.legend = NA, 
  show.help = NULL, 
  inherit = TRUE, 
  environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape y ~ x. Faceting can be achieved by including | in the formula.
data The data to be displayed in this layer. There are three options:
  If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
  A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
  A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function
can be created from a formula (e.g. `~ head(.x,10))

Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with `attribute = value`, (b) ggplot2 aesthetics to be mapped with `attribute = ~ expression`, or (c) attributes of the layer as a whole, which are set with `attribute = value`.

- **alpha**: Opacity (0 = invisible, 1 = opaque).
- **color**: A color or a formula used for mapping color.
- **group**: Used for grouping.
- **linetype**: A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
- **size**: A numeric size or a formula used for mapping size.
- **weight**: Useful for summarized data, weight provides a count of the number of values with the given combination of x and y values.
- **lineend**: Line end style (round, butt, square).
- **linejoin**: Line join style (round, mitre, bevel).
- **linemitre**: Line mitre limit (number greater than 1).
- **quantiles**: Conditional quantiles of y to calculate and display.
- **formula**: Formula relating y variables to x variables.
- **method**: Quantile regression method to use. Available options are "rq" (for `quantreg::rq()`) and "rqss" (for `quantreg::rqss()`).
- **method.args**: List of additional arguments passed on to the modelling function defined by `method`.
- **xlab**: Label for x-axis. See also `gf_labs()`.
- **ylab**: Label for y-axis. See also `gf_labs()`.
- **title**: Title, sub-title, and caption for the plot. See also `gf_labs()`.
- **subtitle**: Title, sub-title, and caption for the plot. See also `gf_labs()`.
- **caption**: Title, sub-title, and caption for the plot. See also `gf_labs()`.
- **geom**: Use to override the default connection between `geom_quantile` and `stat_quantile`.
- **stat**: Use to override the default connection between `geom_quantile` and `stat_quantile`.
- **position**: Position adjustment, either as a string, or the result of a call to a position adjustment function.
- **show.legend**: Logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
- **show.help**: If TRUE, display some minimal help.
- **inherit**: A logical indicating whether default attributes are inherited.
- **environment**: An environment in which to look for variables not found in data.

**Value**

A gg object
Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, B will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

`ggplot2::geom_quantile()`

Examples

```r
gf_point((1 / hwy) ~ displ, data = mpg) %>%
gf_quantile((1 / hwy) ~ displ)
```

---

**gf_raster**

*Formula interface to geom_raster()*

Description

Formula interface to geom_raster()

Usage

```r
gf_raster(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha,
  color,
  fill,
  group,
  linetype,
  size,
  hjust = 0.5,
  vjust = 0.5,
  interpolate = FALSE,
)```
Arguments

object
When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

gformula
A formula with shape y ~ x or fill ~ x + y

data
A data frame with the variables to be plotted.

... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.

alpha
Opacity (0 = invisible, 1 = opaque).

color
A color or a formula used for mapping color.

fill
A color for filling, or a formula used for mapping fill.

group
Used for grouping.

linetype A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

size
A numeric size or a formula used for mapping size.

hjust
horizontal and vertical justification of the grob. Each justification value should be a number between 0 and 1. Defaults to 0.5 for both, centering each pixel over its data location.

vjust
horizontal and vertical justification of the grob. Each justification value should be a number between 0 and 1. Defaults to 0.5 for both, centering each pixel over its data location.

interpolate
If TRUE interpolate linearly, if FALSE (the default) don’t interpolate.

xlab
Label for x-axis. See also gf_labs().

ylab
Label for y-axis. See also gf_labs().

subtitle
Title, sub-title, and caption for the plot. See also gf_labs().

caption
Title, sub-title, and caption for the plot. See also gf_labs().
geom
A character string naming the geom used to make the layer.

stat
A character string naming the stat used to make the layer.

position
Either a character string naming the position function used for the layer or a
position object returned from a call to a position function.

show.legend
A logical indicating whether this layer should be included in the legends. NA,
the default, includes layer in the legends if any of the attributes of the layer are
mapped.

show.help
If TRUE, display some minimal help.

inherit
A logical indicating whether default attributes are inherited.

environment
An environment in which to look for variables not found in data.

Value
a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and
mapping of additional attributes can be done through the use of additional arguments. Attributes
can be set can be set using arguments of the form attribute = value or mapped using arguments
of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid().
This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel
more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the
right thing when formulas are created on the fly, but might not be the right thing if formulas created
in one environment are used to create plots in another.

See Also

ggplot2::geom_raster()

Examples

# Justification controls where the cells are anchored
D <- expand.grid(x = 0:5, y = 0:5)
D$z <- runif(nrow(D))
# centered squares
gf_raster(z ~ x + y, data = D)
gf_raster(y ~ x, fill = ~z, data = D)
# zero padding
gf_raster(z ~ x + y, data = D, hjust = 0, vjust = 0)
gf_rect

Formula interface to geom_rect()

Description

Line plots in ggformula. gf_path() differs from gf_line() in that points are connected in the order in which they appear in data.

Usage

gf_rect(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha,
  color,
  fill,
  group,
  linetype,
  size,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "rect",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape ymin + ymax ~ xmin + xmax. Faceting can be achieved by including | in the formula.
data A data frame with the variables to be plotted.
... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha        Opacity (0 = invisible, 1 = opaque).
color         A color or a formula used for mapping color.
fill          A color for filling, or a formula used for mapping fill.
group         Used for grouping.
linetype      A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping
              linetype.
size          A numeric size or a formula used for mapping size.
xlab          Label for x-axis. See also gf_labs().
ylab          Label for y-axis. See also gf_labs().
title         Title, sub-title, and caption for the plot. See also gf_labs().
subtitle      Title, sub-title, and caption for the plot. See also gf_labs().
caption       Title, sub-title, and caption for the plot. See also gf_labs().
geom          A character string naming the geom used to make the layer.
stat          A character string naming the stat used to make the layer.
position      Either a character string naming the position function used for the layer or a
              position object returned from a call to a position function.
show.legend   A logical indicating whether this layer should be included in the legends. NA,
              the default, includes layer in the legends if any of the attributes of the layer are
              mapped.
show.help      If TRUE, display some minimal help.
inherit       A logical indicating whether default attributes are inherited.
environment   An environment in which to look for variables not found in data.

Value

A gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting
and mapping of additional attributes can be done through the use of additional arguments. Attributes
can be set can be set using arguments of the form attribute = value or mapped using arguments
of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid().
This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel
more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the
right thing when formulas are created on the fly, but might not be the right thing if formulas created
in one environment are used to create plots in another.
gf_relabel

Modify plot labeling

Description

Some packages like expss provide mechanisms for providing longer labels to R objects. These labels can be used when labeling plots and tables, for example, without requiring long or awkward variable names. This is an experimental feature and currently only supports expss or any other system that stores a label in the label attribute of a vector.

Usage

gf_relabel(plot, labels = get_variable_labels(plot$data), ...)

## S3 method for class 'gf_ggplot'
print(x, labels = get_variable_labels(x$data), ...)

Arguments

- **plot**: A ggplot.
- **labels**: A named list of labels.
- **...**: Additional named labels. See examples.
- **x**: A ggplot.

Value

A plot with potentially modified labels.
Examples

```r
# labeling using a list
labels <- list(width = "width of foot (cm)", length = "length of foot (cm)",
               domhand = "dominant hand")
gf_point(length ~ width, color = ~domhand, data = mosaicData::KidsFeet) %>%
gf_relabel(labels)

# labeling using ...
gf_point(length ~ width, color = ~domhand, data = mosaicData::KidsFeet) %>%
gf_relabel(
  width = "width of foot (cm)",
  length = "length of foot (cm)",
  domhand = "dominant hand")

# Alternatively, we can store labels with data.
KF <- mosaicData::KidsFeet %>%
    set_variable_labels(
      length = "foot length (cm)",
      width = "foot width (cm)"
    )
gf_point(length ~ width, data = KF)
gf_density2d(length ~ width, data = KF)
get_variable_labels(KF)
```

---

gf_ribbon

**Formula interface to geom_ribbon()**

**Description**

For each x value, `geom_ribbon()` displays a y interval defined by `ymin` and `ymax`. `geom_area()` is a special case of `geom_ribbon()`, where the `ymin` is fixed to 0 and `y` is used instead of `ymax`.

**Usage**

```r
gf_ribbon(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha = 0.3,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
```
geom = "ribbon",
stat = "identity",
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

Arguments

object
When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula
A formula with shape ymin + ymax ~ x. Faceting can be achieved by including | in the formula.
data
The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot()
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).

...
Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha
Opacity (0 = invisible, 1 = opaque).
xlab
Label for x-axis. See also gf_labs().
ylab
Label for y-axis. See also gf_labs().
title
Title, sub-title, and caption for the plot. See also gf_labs().
subtitle
Title, sub-title, and caption for the plot. See also gf_labs().
caption
Title, sub-title, and caption for the plot. See also gf_labs().
geom
A character string naming the geom used to make the layer.
stat
The statistical transformation to use on the data for this layer, as a string.
position
Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.legend
logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
show.help
If TRUE, display some minimal help.
inherit
A logical indicating whether default attributes are inherited.
environment
An environment in which to look for variables not found in data.
gf_ridgeline

Formula interface to ggridges plots

Description

Formula interface to ggridges plots

Usage

gf_ridgeline(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  height,
  scale = 1,
  min_height = 0,
  color,
  fill,
  alpha,
  group,
  linetype,
  size,
  point_size,
  point_shape,
  point_colour,
  point_fill,
)
gf_ridgeline

point_alpha,
point_stroke,
xlab,
ylab,
title,
subtitle,
caption,
geom = "ridgeline",
stat = "identity",
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

gf_density_ridges(
  object = NULL,
gformula = NULL,
data = NULL,
...,
height,
scale = 1,
rel_min_height = 0,
color,
fill,
alpha,
group,
linetype,
size,
point_size,
point_shape,
point_colour,
point_fill,
point_alpha,
point_stroke,
panel_scaling = TRUE,
xlab,
ylab,
title,
subtitle,
caption,
geom = "density_ridges",
stat = "density_ridges",
position = "points_sina",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

gf_density_ridges2(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ....
  height,
  scale = 1,
  rel_min_height = 0,
  color,
  fill,
  alpha,
  group,
  linetype,
  size,
  point_size,
  point_shape,
  point_colour,
  point_fill,
  point_alpha,
  point_stroke,
  panel_scaling = TRUE,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "density_ridges2",
  stat = "density_ridges",
  position = "points_sina",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

gf_density_ridgeline_gradient(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ....
  height,
  color,
  fill,
  alpha,
  group,
Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula
A formula with shape ~ x. Faceting can be achieved by including | in the formula.

data
The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).

... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.

height
The height of each ridgeline at the respective x value. Automatically calculated and provided by ggridges::stat_density_ridges() if the default stat is not changed.

scale
A scaling factor to scale the height of the ridgelines relative to the spacing between them. A value of 1 indicates that the maximum point of any ridgeline touches the baseline right above, assuming even spacing between baselines.

min_height
A height cutoff on the drawn ridgelines. All values that fall below this cutoff will be removed. The main purpose of this cutoff is to remove long tails right at the baseline level, but other uses are possible. The cutoff is applied before any height scaling is applied via the scale aesthetic. Default is 0, so negative values are removed.

color
A color or a formula used for mapping color.

fill
A color for filling, or a formula used for mapping fill.

alpha
Opacity (0 = invisible, 1 = opaque).

group
Used for grouping.

linetype
A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

size
A numeric size or a formula used for mapping size.

point_shape, point_colour, point_size, point_fill, point_alpha, point_stroke
As in ggridges::geom_ridgeline().

xlab
Label for x-axis. See also gf_labs().

ylab
Label for y-axis. See also gf_labs().

title
Title, sub-title, and caption for the plot. See also gf_labs().

subtitle
Title, sub-title, and caption for the plot. See also gf_labs().

caption
Title, sub-title, and caption for the plot. See also gf_labs().

geom
Use to override the default connection between geom_density and stat_density.

stat
Use to override the default connection between geom_density and stat_density.
**gf_ridgeline**

position  
Position adjustment, either as a string, or the result of a call to a position adjustment function.

show.legend  
logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

show.help  
If TRUE, display some minimal help.

inherit  
A logical indicating whether default attributes are inherited.

environment  
An environment in which to look for variables not found in data.

rel_min_height  
Lines with heights below this cutoff will be removed. The cutoff is measured relative to the overall maximum, so rel_min_height = 0.01 would remove everything. Default is 0, so nothing is removed.

panel_scaling  
If TRUE, the default, relative scaling is calculated separately for each panel. If FALSE, relative scaling is calculated globally.

gradient_lwd  
A parameter to needed to remove rendering artifacts inside the rendered gradients. Should ideally be 0, but often needs to be around 0.5 or higher.

Details

Note that the `ggridges::stat_density_ridges()` makes joint density estimation across all datasets. This may not generate the desired result when using faceted plots. As an alternative, you can set `stat = "density"` to use `ggplot2::stat_density()`. In this case, it is required to add the aesthetic mapping `height = stat(density)` (see examples).

See Also

`ggridges::geom_density_ridges()`  
`ggridges::geom_ridgeline()`  
`ggridges::geom_density_ridges_gradient()`

Examples

data.frame(
  x = rep(1:5, 3), y = c(rep(0, 5), rep(1, 5), rep(3, 5)),
  height = c(0, 1, 3, 4, 0, 1, 2, 3, 5, 4, 0, 5, 4, 4, 1)
)  
gf_ridgeline(y ~ x, height = ~ height, group = ~y, fill = "lightblue", alpha = 0.7)
diamonds  
gf_density_ridges(cut ~ price,  
  scale = 2, fill = ~ cut, alpha = 0.6, show.legend = FALSE)  
gf_theme(theme_ridges())  
gf_refine(  
  scale_y_discrete(expand = c(0.01, 0)),  
  scale_x_continuous(expand = c(0.01, 0))
)  
diamonds  
gf_density_ridges(clarity ~ price | cut,  
  scale = 2, fill = ~ clarity, alpha = 0.6, show.legend = FALSE)  
gf_theme(theme_ridges())
gf_rug

Formula interface to geom_rug()
Description

gf_rugx() and gf_rugy() are versions that only add a rug to x- or y- axis. By default, these functions do not inherit from the formula in the original layer (because doing so would often result in rugs on both axes), so the formula is required.

Usage

gf_rug(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  sides = "bl",
  alpha,
  color,
  group,
  linetype,
  size,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "rug",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

gf_rugx(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  sides = "b",
  alpha,
  color,
  group,
  linetype,
  size,
  height = 0,
  xlab,
  ylab,
  title,
  subtitle,
caption,
geom = "rug",
stat = "identity",
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

gf_rugy(
  object = NULL,
gformula = NULL,
data = NULL,
...
  sides = "1",
  alpha,
  color,
  group,
  linetype,
  size,
  width = 0,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "rug",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape y ~ x (gf_rug()) or ~ x (gf_rugx()) or ~ y (gf_rugy()).
data The data to be displayed in this layer. There are three options:
  If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
  A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).

Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.

sides A string that controls which sides of the plot the rugs appear on. It can be set to a string containing any of "trbl", for top, right, bottom, and left.

alpha Opacity (0 = invisible, 1 = opaque).

color A color or a formula used for mapping color.

group Used for grouping.

linetype A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

size A numeric size or a formula used for mapping size.

xlab Label for x-axis. See also gf_labs().

ylab Label for y-axis. See also gf_labs().

title Title, sub-title, and caption for the plot. See also gf_labs().

subtitle Title, sub-title, and caption for the plot. See also gf_labs().

caption Title, sub-title, and caption for the plot. See also gf_labs().

geom A character string naming the geom used to make the layer.

stat The statistical transformation to use on the data for this layer, as a string.

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

show.help If TRUE, display some minimal help.

inherit A logical indicating whether default attributes are inherited.

environment An environment in which to look for variables not found in data.

height amount of vertical jittering when position is jittered.

width amount of horizontal jittering when position is jittered.

Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression.
In formulas of the form A | B, B will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

**Evaluation**

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

**See Also**

`ggplot2::geom_rug()`

**Examples**

data(penguins, package = "palmerpenguins")
gf_point(bill_length_mm ~ bill_depth_mm, data = penguins) %>%
gf_rug(bill_length_mm ~ bill_depth_mm)

# There are several ways to control x- and y-rugs separately
gf_point(bill_length_mm ~ bill_depth_mm, data = penguins) %>%
gf_rugx(~bill_depth_mm, data = penguins, color = "red", inherit = FALSE) %>%
gf_rugy(bill_length_mm ~ ., data = penguins, color = "green", inherit = FALSE)

gf_point(bill_length_mm ~ bill_depth_mm, data = penguins) %>%
gf_rug (~ bill_depth_mm, data = penguins, color = "green", sides = "l")

gf_point(bill_length_mm ~ bill_depth_mm, data = penguins) %>%
gf_rugx(~bill_depth_mm, data = penguins, color = "red", sides = ~ "b") %>%
gf_rugy(bill_length_mm ~ ., data = penguins, color = "green", sides = ~ "l")

# Jitter requires both an x and a y, but we can turn off one or the other with sides

gf_jitter(bill_length_mm ~ bill_depth_mm, data = penguins) %>%
gf_rug(color = "red", sides = ~ "b", position = ~ "jitter")

gf_histogram(~eruptions, data = faithful) %>%
gf_rug(~eruptions, data = faithful, color = "red") %>%
gf_rug(~eruptions, data = faithful, color = "green", sides = ~ "l")

# We can take advantage of inheritance to shorten the code

gf_histogram(~eruptions, data = faithful) %>%
gf_rug(color = ~ "red") %>%
gf_rug(color = ~ "green", sides = ~ "l")

# Need to turn off inheritance when using gf_dhistogram:
gf_dhistogram(~eruptions, data = faithful) %>%
gf_rug(~eruptions, data = faithful, color = ~ "red", inherit = FALSE)

# Using jitter with gf_histogram() requires manually setting the y value.
gf_segment

gf_dhistogram(~bill_depth_mm, data = penguins) %>%
gf_rug(0 ~ bill_depth_mm, data = penguins, color = "green", sides = "b", position = "jitter")

# the choice of y value can affect how the plot looks.
gf_dhistogram(~bill_depth_mm, data = penguins) %>%
gf_rug(0.5 ~ bill_depth_mm, data = penguins, color = "green", sides = "b", position = "jitter")

---

**gf_segment**

*Formula interface to geom_segment()*

**Description**

`geom_segment()` draws a straight line between points (x, y) and (xend, yend). `geom_curve` draws a curved line. See the underlying drawing function `grid::curveGrob()` for the parameters that control the curve.

**Usage**

```r
gf_segment(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  group,
  linetype,
  size,
  arrow = NULL,
  lineend = "butt",
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "segment",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
```

**Arguments**

- **object**
  - When chaining, this holds an object produced in the earlier portions of the chain.
  - Most users can safely ignore this argument. See details and examples.
gf_segment

A formula with shape \( y + y_{end} \sim x + x_{end} \).

data

The data to be displayed in this layer. There are three options:

If NULL, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. \( \sim \text{head}(x, 10) \)).

Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = \( \sim \) expression, or (c) attributes of the layer as a whole, which are set with attribute = value.

alpha

Opacity (0 = invisible, 1 = opaque).

color

A color or a formula used for mapping color.

group

Used for grouping.

linetype

A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

size

A numeric size or a formula used for mapping size.

arrow

specification for arrow heads, as created by arrow().

lineend

Line end style (round, butt, square).

xlab

Label for x-axis. See also `gf_labs()`.

ylab

Label for y-axis. See also `gf_labs()`.

title

Title, sub-title, and caption for the plot. See also `gf_labs()`.

subtitle

Title, sub-title, and caption for the plot. See also `gf_labs()`.

caption

Title, sub-title, and caption for the plot. See also `gf_labs()`.

geom

A character string naming the geom used to make the layer.

stat

The statistical transformation to use on the data for this layer, as a string.

position

Position adjustment, either as a string, or the result of a call to a position adjustment function.

show.legend

logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

show.help

If TRUE, display some minimal help.

inherit

A logical indicating whether default attributes are inherited.

environment

An environment in which to look for variables not found in data.

Value

a gg object
Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

`ggplot2::geom_segment()`

Examples

```r
D <- data.frame(x1 = 2.62, x2 = 3.57, y1 = 21.0, y2 = 15.0)
gf_point(mpg ~ wt, data = mtcars) %>%
gf_curve(y1 + y2 ~ x1 + x2, data = D, color = "navy") %>%
gf_segment(y1 + y2 ~ x1 + x2, data = D, color = "red")
```

---

gf_sf  

Mapping with shape files

Description

Mapping with shape files

Usage

```r
gf_sf(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  fill,
  group,
  linetype,
  size,
  geometry,
```
gf_sf

```r
xlab,
ylab,
title,
subtitle,
caption,
stat = "sf",
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
```

**Arguments**

- **object**
  When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

- **gformula**
  A formula with shape \( y \sim x \). Faceting can be achieved by including | in the formula.

- **data**
  A data frame with the variables to be plotted.

- **...**
  Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, (c) attributes of the layer as a whole, which are set with attribute = value, or (d) arguments for the geom, stat, or position function.

- **alpha**
  Opacity (0 = invisible, 1 = opaque).

- **color**
  A color or a formula used for mapping color.

- **fill**
  A color for filling, or a formula used for mapping fill.

- **group**
  Used for grouping.

- **linetype**
  A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.

- **size**
  A numeric size or a formula used for mapping size.

- **geometry**
  A column of class sf containing simple features data. (Another option is that data may contain a column named geometry.) geometry is never inherited.

- **xlab**
  Label for x-axis. See also `gf_labs()`.

- **ylab**
  Label for y-axis. See also `gf_labs()`.

- **title**
  Title, sub-title, and caption for the plot. See also `gf_labs()`.

- **subtitle**
  Title, sub-title, and caption for the plot. See also `gf_labs()`.

- **caption**
  Title, sub-title, and caption for the plot. See also `gf_labs()`.

- **stat**
  A character string naming the stat used to make the layer.

- **position**
  Either a character string naming the position function used for the layer or a position object returned from a call to a position function.

- **show.legend**
  A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.
gf_sf

show.help If TRUE, display some minimal help.
inherit A logical indicating whether default attributes are inherited.
environment An environment in which to look for variables not found in data.

Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

ggplot2::geom_line(), gf_point()

Examples

```r
## Not run:
if (require(maps) && require(maptools) && require(sf) && require(rgeos))
US <- sf::st_as_sf(maps::map("state", plot = FALSE, fill = TRUE))
gf_sf(fill = ~ factor(nchar(ID)), data = US) %>%
gf_refine(coord_sf())

# We can specify shape data and external data separately using geometry
MI <- sf::st_as_sf(maps::map("county", "michigan", plot = FALSE, fill = TRUE))
MGeom <- MI$geom
gf_sf(
  fill = ~ log10(population), data = MIpop %>% dplyr::arrange(county),
  geometry = ~MGeom, color = "white"
) %>%
gf_refine(coord_sf(), theme_bw())

# alternatively we can merge external data and shape data into one data frame.
MI %>%
dplyr::mutate(county = gsub("michigan,"", ID)) %>%
dplyr::left_join(MIpop %>% dplyr::mutate(county = tolower(county))) %>%
```
gf_sina

Formula interface to geom_sina()

Description

The sina plot is a data visualization chart suitable for plotting any single variable in a multiclass dataset. It is an enhanced jitter strip chart, where the width of the jitter is controlled by the density distribution of the data within each class.

Usage

```r
gf_sina(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  size,
  fill,
  group,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "point",
  stat = "sina",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
```

Arguments

- `object` When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape \( y \sim x \). Faceting can be achieved by including | in the formula.

data The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}.
A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{fortify()} for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a \texttt{data.frame}, and will be used as the layer data. A function can be created from a formula (e.g. \( \sim \text{head}(.x, 10) \)).

... Other arguments passed on to \texttt{layer()}. These are often aesthetics, used to set an aesthetic to a fixed value, like \texttt{colour = "red"} or \texttt{size = 3}. They may also be parameters to the paired geom/stat.

alpha Opacity (0 = invisible, 1 = opaque).
color A color or a formula used for mapping color.
size A numeric size or a formula used for mapping size.
fill A color for filling, or a formula used for mapping fill.
group Used for grouping.
xlab Label for x-axis. See also \texttt{gf_labs()}.
ylab Label for y-axis. See also \texttt{gf_labs()}.
title Title, sub-title, and caption for the plot. See also \texttt{gf_labs()}.
subtitle Title, sub-title, and caption for the plot. See also \texttt{gf_labs()}.
caption Title, sub-title, and caption for the plot. See also \texttt{gf_labs()}.
geom The geometric object to use display the data.
stat The statistical transformation to use on the data for this layer, as a string.
position Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.legend logical. Should this layer be included in the legends? \texttt{NA}, the default, includes if any aesthetics are mapped. \texttt{FALSE} never includes, and \texttt{TRUE} always includes. It can also be a named logical vector to finely select the aesthetics to display.
show.help If \texttt{TRUE}, display some minimal help.
inherit A logical indicating whether default attributes are inherited.
environment An environment in which to look for variables not found in data.

Value

a gg object
Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

`ggforce::geom_sina()`

Examples

```r
gf_sina(age ~ substance, data = mosaicData::HELPct)
```

---

**gf_smooth**  
*Formula interface to geom_smooth()*

**Description**

LOESS and linear model smoothers in `ggformula`.

**Usage**

```r
gf_smooth(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  method = "auto",
  formula = y ~ x,
  se = FALSE,
  method.args,
  n = 80,
  span = 0.75,
  fullrange = FALSE,
  level = 0.95,
  xlab,
```
gf_smooth

ylab,
title,
subtitle,
caption,
geom = "smooth",
stat = "smooth",
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

gf_lm(
  object = NULL,
gformula = NULL,
data = NULL,
...,
alpha = 0.3,
lm.args = list(),
interval = "none",
level = 0.95,
fullrange = TRUE,
xlab,
ylab,
title,
subtitle,
caption,
geom = "lm",
stat = "lm",
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

Arguments

object

When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

gformula

A formula with shape \( y \sim x \). Faceting can be achieved by including | in the formula.

data

A data frame with the variables to be plotted.

...  

Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
gf_smooth

method

Smoothing method (function) to use, accepts either NULL or a character vector, e.g. "lm", "glm", "gam", "loess" or a function, e.g. MASS::rlm or mgcv::gam, stats::lm, or stats::loess. "auto" is also accepted for backwards compatibility. It is equivalent to NULL.

For method = NULL the smoothing method is chosen based on the size of the largest group (across all panels). stats::loess() is used for less than 1,000 observations; otherwise mgcv::gam() is used with formula = y ~ s(x, bs = "cs") with method = "REML". Somewhat anecdotally, loess gives a better appearance, but is $O(N^2)$ in memory, so does not work for larger datasets.

If you have fewer than 1,000 observations but want to use the same gam() model that method = NULL would use, then set method = "gam", formula = y ~ s(x, bs = "cs").

formula

Formula to use in smoothing function, eg. y ~ x, y ~ poly(x,2), y ~ log(x). NULL by default, in which case method = NULL implies formula = y ~ x when there are fewer than 1,000 observations and formula = y ~ s(x, bs = "cs") otherwise.

se

Display confidence interval around smooth? (TRUE by default, see level to control.)

method.args

List of additional arguments passed on to the modelling function defined by method.

n

Number of points at which to evaluate smoother.

span

Controls the amount of smoothing for the default loess smoother. Smaller numbers produce wigglier lines, larger numbers produce smoother lines.

fullrange

Should the fit span the full range of the plot, or just the data?

level

Level of confidence interval to use (0.95 by default).

xlab

Label for x-axis. See also gf_labs().

ylab

Label for y-axis. See also gf_labs().

title

Title, sub-title, and caption for the plot. See also gf_labs().

subtitle

Title, sub-title, and caption for the plot. See also gf_labs().

caption

Title, sub-title, and caption for the plot. See also gf_labs().

geom

A character string naming the geom used to make the layer.

stat

A character string naming the stat used to make the layer.

position

Either a character string naming the position function used for the layer or a position object returned from a call to a position function.

show.legend

A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.

show.help

If TRUE, display some minimal help.

inherit

A logical indicating whether default attributes are inherited.

environment

An environment in which to look for variables not found in data.

alpha

Opacity (0 = invisible, 1 = opaque).

lm.args

A list of arguments to stats::lm().

interval

One of "none", "confidence" or "prediction".
Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in `gformula`. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

`ggplot2::geom_smooth()`, `gf_spline()`

Examples

gf_smooth()

```r
gf_lm()
gf_smooth(births ~ date, color = ~wday, data = mosaicData::Births78)
gf_smooth(births ~ date,
    color = ~wday, data = mosaicData::Births78,
    fullrange = TRUE
)
gf_smooth(births ~ date,
    color = ~wday, data = mosaicData::Births78,
    show.legend = FALSE, se = FALSE
)
gf_smooth(births ~ date,
    color = ~wday, data = mosaicData::Births78,
    show.legend = FALSE, se = TRUE
)
gf_lm(length ~ width,
    data = mosaicData::KidsFeet,
    color = ~biggerfoot, alpha = 0.2
  )
gf_point()
gf_lm(length ~ width,
    data = mosaicData::KidsFeet,
    color = ~biggerfoot, fullrange = FALSE, alpha = 0.2
  )
gf_point()
gf_lm(length ~ width,
```
```

gf_spline(formula = y ~ poly(x, 2), linetype = "dashed") %>%
gf_point()

gf_lm(length ~ width, color = ~sex, data = mosaicData::KidsFeet, formula = log(y) ~ x, backtrans = exp)
) %>%
gf_point()

gf_lm(hwy ~ displ, data = mpg, formula = log(y) ~ poly(x, 3), backtrans = exp, interval = "prediction", fill = "skyblue")
) %>%
gf_point()
```

---

**gf_spline**

Formula interface to geom_spline()

**Description**

Fitting splines in ggformula.

**Usage**

```
gf_spline(
  object = NULL, 
  gformula = NULL, 
  data = NULL, 
  ..., 
  alpha, 
  color, 
  group, 
  linetype, 
  size, 
  weight, 
  df, 
  spar, 
  tol, 
  xlab, 
  ylab, 
  title, 
  subtitle, 
  caption,
```

```
gf_spline

geom = "line",
stat = "spline",
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape y ~ x. Faceting can be achieved by including | in the formula.
data A data frame with the variables to be plotted.
... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha Opacity (0 = invisible, 1 = opaque).
color A color or a formula used for mapping color.
group Used for grouping.
linetype A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
size A numeric size or a formula used for mapping size.
weight An optional vector of weights. See smooth.spline().
df desired equivalent vector of degrees of freedom. See smooth.spline() for details.
spar A smoothing parameter, typically in (0,1]. See smooth.spline() for details.
tol A tolerance for sameness or uniqueness of the x values. The values are binned into bins of size tol and values which fall into the same bin are regarded as the same. Must be strictly positive (and finite). When NULL, IQR(x) * 10e-6 is used.
xlab Label for x-axis. See also gf_labs().
ylab Label for y-axis. See also gf_labs().
title Title, sub-title, and caption for the plot. See also gf_labs().
subtitle Title, sub-title, and caption for the plot. See also gf_labs().
caption Title, sub-title, and caption for the plot. See also gf_labs().
geom A character string naming the geom used to make the layer.
stat A character string naming the stat used to make the layer.
position Either a character string naming the position function used for the layer or a position object returned from a call to a position function.
show.legend: A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.

show.help: If TRUE, display some minimal help.

inherit: A logical indicating whether default attributes are inherited.

environment: An environment in which to look for variables not found in data.

Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

geom_spline(), gf_smooth(), gf_lm()

Examples

gf_spline(births ~ date, color = ~wday, data = mosaicData::Births78)
gf_spline(births ~ date, color = ~wday, data = mosaicData::Births78, df = 20)
gf_spline(births ~ date, color = ~wday, data = mosaicData::Births78, df = 4)

gf_spoke

Formula interface to geom_spoke()

Description

This is a polar parameterisation of geom_segment. It is useful when you have variables that describe direction and distance.
Usage

gf_spoke(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  angle, 
  radius, 
  alpha, 
  color, 
  group, 
  linetype, 
  size, 
  xlab, 
  ylab, 
  title, 
  subtitle, 
  caption, 
  geom = "spoke", 
  stat = "identity", 
  position = "identity", 
  show.legend = NA, 
  show.help = NULL, 
  inherit = TRUE, 
  environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.

gformula A formula with shape y ~ x. Faceting can be achieved by including | in the formula.

data The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).

... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.

angle The angle at which segment leaves the point (x,y).
radius  The length of the segment.
alpha  Opacity (0 = invisible, 1 = opaque).
color  A color or a formula used for mapping color.
group  Used for grouping.
linetype  A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
size  A numeric size or a formula used for mapping size.
xlab  Label for x-axis. See also gf_labs().
ylab  Label for y-axis. See also gf_labs().
title  Title, sub-title, and caption for the plot. See also gf_labs().
subtitle  Title, sub-title, and caption for the plot. See also gf_labs().
caption  Title, sub-title, and caption for the plot. See also gf_labs().
geom  A character string naming the geom used to make the layer.
stat  The statistical transformation to use on the data for this layer, as a string.
position  Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.legend  logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
show.help  If TRUE, display some minimal help.
inherit  A logical indicating whether default attributes are inherited.
environment  An environment in which to look for variables not found in data.

Value
  a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.
gf_step

See Also

ggplot2::geom_spoke()

Examples

SomeData <- expand.grid(x = 1:10, y = 1:10)
SomeData$angle <- runif(100, 0, 2 * pi)
SomeData$speed <- runif(100, 0, sqrt(0.1 * SomeData$x))

gf_point(y ~ x, data = SomeData) %>%
gf_spoke(y ~ x, angle = ~angle, radius = 0.5)

gf_point(y ~ x, data = SomeData) %>%
gf_spoke(y ~ x, angle = ~angle, radius = ~speed)

gf_step

Formula interface to geom_step()

Description

gem_path() connects the observations in the order in which they appear in the data. geom_line() connects them in order of the variable on the x axis. geom_step() creates a stairstep plot, highlighting exactly when changes occur. The group aesthetic determines which cases are connected together.

Usage

gf_step(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  group,
  linetype,
  size,
  direction = "hv",
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "step",
  stat = "identity",
  position = "identity",
  show.legend = NA,
)
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape \( y \sim x \). Faceting can be achieved by including | in the formula.
data The data to be displayed in this layer. There are three options:
        If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
        A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
        A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. \(~\) head(.x,10)).

... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.

alpha Opacity (0 = invisible, 1 = opaque).
color A color or a formula used for mapping color.
group Used for grouping.
linetype A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
size A numeric size or a formula used for mapping size.
direction direction of stairs: 'vh' for vertical then horizontal, 'hv' for horizontal then vertical, or 'mid' for step half-way between adjacent x-values.
xlab Label for x-axis. See also gf_labs().
ylab Label for y-axis. See also gf_labs().
title Title, sub-title, and caption for the plot. See also gf_labs().
subtitle Title, sub-title, and caption for the plot. See also gf_labs().
caption Title, sub-title, and caption for the plot. See also gf_labs().
geom A character string naming the geom used to make the layer.
stat The statistical transformation to use on the data for this layer, as a string.
position Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
show.help If TRUE, display some minimal help.
inherit A logical indicating whether default attributes are inherited.
environment An environment in which to look for variables not found in data.

Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

ggplot2::geom_step()

Examples

gf_step(births ~ date, data = mosaicData::Births78, color = ~wday)

# Roll your own Kaplan-Meier plot

if (require(survival) && require(broom)) {
  # fit a survival model
  surv_fit <- survfit(coxph(Surv(time, status) ~ age + sex, lung))
  surv_fit
  # use broom::tidy() to create a tidy data frame for plotting
  surv_df <- tidy(surv_fit)
  head(surv_df)
  # now create a plot
  surv_df %>%
    gf_step(estimate ~ time) %>%
    gf_ribbon(conf.low + conf.high ~ time, alpha = 0.2)
}
gf_text

Formula interface to geom_text() and geom_label()

Description

Text geoms are useful for labeling plots. They can be used by themselves as scatterplots or in combination with other geoms, for example, for labeling points or for annotating the height of bars. geom_text() adds only text to the plot. geom_label() draws a rectangle behind the text, making it easier to read.

Usage

```r
gf_text(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  label,
  alpha,
  angle,
  color,
  family,
  fontface,
  group,
  hjust,
  lineheight,
  size,
  vjust,
  parse = FALSE,
  nudge_x = 0,
  nudge_y = 0,
  check_overlap = FALSE,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "text",
  stat = "identity",
  position = "nudge",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

gf_label()
```
object = NULL,
gformula = NULL,
data = NULL,
...,
label, 
alpha, 
age, 
color, 
family, 
fontface, 
group, 
hjust, 
vjust, 
lineheight, 
size, 
parse, 
nudge_x = 0, 
nudge_y = 0, 
label.padding = unit(0.25, "lines"), 
label.r = unit(0.15, "lines"), 
label.size = 0.25, 
xlab, 
ylab, 
title, 
subtitle, 
caption, 
stat = "identity", 
position = "nudge", 
show.legend = NA, 
show.help = NULL, 
inherit = TRUE, 
environment = parent.frame()
)

Arguments

object
When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula
A formula with shape y ~ x. Faceting can be achieved by including | in the formula.
data
The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot()
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function
can be created from a formula (e.g. \( \sim \) head(.x,10)).

... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = \( \sim \) expression, or (c) attributes of the layer as a whole, which are set with attribute = value.

**label** The text to be displayed.

**alpha** Opacity (0 = invisible, 1 = opaque).

**angle** An angle for rotating the text.

**color** A color or a formula used for mapping color.

**family** A font family.

**fontface** One of "plain", "bold", "italic", or "bold italic".

**group** Used for grouping.

**hjust, vjust** Numbers between 0 and 1 indicating how to justify text relative the the specified location.

**lineheight** Line height.

**size** A numeric size or a formula used for mapping size.

**parse** If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath.

**nudge_x** Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with position.

**nudge_y** Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with position.

**check_overlap** If TRUE, text that overlaps previous text in the same layer will not be plotted. check_overlap happens at draw time and in the order of the data. Therefore data should be arranged by the label column before calling geom_label() or geom_text().

**xlab** Label for x-axis. See also gf_labs().

**ylab** Label for y-axis. See also gf_labs().

**title** Title, sub-title, and caption for the plot. See also gf_labs().

**subtitle** Title, sub-title, and caption for the plot. See also gf_labs().

**caption** Title, sub-title, and caption for the plot. See also gf_labs().

**geom** A character string naming the geom used to make the layer.

**stat** The statistical transformation to use on the data for this layer, as a string.

**position** Position adjustment, either as a string, or the result of a call to a position adjustment function. Cannot be jointly specified with nudge_x or nudge_y.

**show.legend** logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

**show.help** If TRUE, display some minimal help.
gf_text

inherit  A logical indicating whether default attributes are inherited.

environment  An environment in which to look for variables not found in data.

label.padding  Amount of padding around label. Defaults to 0.25 lines.

label.r  Radius of rounded corners. Defaults to 0.15 lines.

label.size  Size of label border, in mm.

Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in gformula. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment of gformula. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

See Also

ggplot2::geom_text()

Examples

data(penguins, package = "palmerpenguins")
gf_text(bill_length_mm ~ bill_depth_mm,
    data = penguins,
    label = ~species, color = ~species, size = 2, angle = 30
  )

if (require(dplyr)) {
  data(penguins, package = "palmerpenguins")
penguins_means <-
    penguins %>%
    group_by(species) %>%
    summarise(bill_length_mm = mean(bill_length_mm), bill_depth_mm = mean(bill_depth_mm))
gf_point(bill_length_mm ~ bill_depth_mm, data = penguins, color = ~species) %>%
gf_tile

```r
gf_label(bill_length_mm ~ bill_depth_mm,
    data = penguins_means,
    label = ~species, color = ~species, size = 2, alpha = 0.7
)
```

---

**gf_theme**  
*Themes for ggformula*

**Description**
Themes for ggformula

**Usage**

```r
gf_theme(object, theme, ...)
```

**Arguments**

- `object`: a gg object
- `theme`: a ggplot2 theme function like `theme_minimal()`.
- `...`: If `theme` is missing, then these additional arguments are theme elements of the sort handled by `ggplot2::theme()`.

**Value**

a modified gg object

---

**gf_tile**  
*Formula interface to geom_tile()*

**Description**

`geom_rect()` and `geom_tile()` do the same thing, but are parameterised differently: `geom_rect()` uses the locations of the four corners (`xmin`, `xmax`, `ymin` and `ymax`), while `geom_tile()` uses the center of the tile and its size (`x`, `y`, `width`, `height`). `geom_raster` is a high performance special case for when all the tiles are the same size.
Usage

gf_tile(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., alpha,
  color,
  fill,
  group,
  linetype,
  size,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "tile",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

Arguments

object When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape y ~ x. Faceting can be achieved by including | in the formula.
data A data frame with the variables to be plotted.
... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha Opacity (0 = invisible, 1 = opaque).
color A color or a formula used for mapping color.
fill A color for filling, or a formula used for mapping fill.
group Used for grouping.
linetype A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
size A numeric size or a formula used for mapping size.
xlab Label for x-axis. See also gf_labs().
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| ylab       | Label for y-axis. See also `gf_labs()`.
| title      | Title, sub-title, and caption for the plot. See also `gf_labs()`.
| subtitle   | Title, sub-title, and caption for the plot. See also `gf_labs()`.
| caption    | Title, sub-title, and caption for the plot. See also `gf_labs()`.
| geom       | A character string naming the geom used to make the layer.
| stat       | A character string naming the stat used to make the layer.
| position   | Either a character string naming the position function used for the layer or a position object returned from a call to a position function.
| show.legend| A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.
| show.help  | If TRUE, display some minimal help.
| inherit    | A logical indicating whether default attributes are inherited.
| environment| An environment in which to look for variables not found in data.

**Value**

a gg object

**Specifying plot attributes**

Positional attributes (a.k.a, aesthetics) are specified using the formula in `gformula`. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Positional attributes (a.k.a, aesthetics) are specified using the formula in `gformula`. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

**Evaluation**

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.
gf_violin

See Also

ggplot2::geom_tile()

Examples

D <- expand.grid(x = 0:5, y = 0:5)
D$z <- runif(nrow(D))
gf_tile(y ~ x, fill = ~z, data = D)
gf_tile(z ~ x + y, data = D)

gf_violin

Description

A violin plot is a compact display of a continuous distribution. It is a blend of geom_boxplot() and geom_density(): a violin plot is a mirrored density plot displayed in the same way as a boxplot.

Usage

gf_violin(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., alpha,
  color,
  fill,
  group,
  linetype,
  size,
  weight,
  draw_quantiles = NULL,
  trim = TRUE,
  scale = "area",
  bw,
  adjust = 1,
  kernel = "gaussian",
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "violin",
  stat = "ydensity",
  position = "dodge",
  show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

gf_violinh(
object = NULL,
gformula = NULL,
data = NULL,
..., 
alpha, 
color, 
fill, 
group, 
linetype, 
size, 
weight, 
draw_quantiles = NULL, 
trim = TRUE, 
scale = "area", 
bw, 
adjust = 1, 
kernel = "gaussian", 
xlab, 
ylab, 
title, 
subtitle, 
caption, 
geom = "violinh", 
stat = "xdensity", 
position = "dodgev", 
show.legend = NA, 
show.help = NULL, 
inherit = TRUE, 
environment = parent.frame()
)

Arguments

object  When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula A formula with shape y ∼ x. Faceting can be achieved by including | in the formula.
data  The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be
created.
A function will be called with a single argument, the plot data. The return
value must be a data.frame, and will be used as the layer data. A function
can be created from a formula (e.g. `~ head(x, 10))

... Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with
attribute = value, (b) ggplot2 aesthetics to be mapped with attribute =
~ expression, or (c) attributes of the layer as a whole, which are set with
attribute = value.

alpha Opacity (0 = invisible, 1 = opaque).
color A color or a formula used for mapping color.
fill A color for filling, or a formula used for mapping fill.
group Used for grouping.
linetype A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping
linetype.
size A numeric size or a formula used for mapping size.
weight Useful for summarized data, weight provides a count of the number of values
with the given combination of x and y values.
draw_quantiles If not(NULL) (default), draw horizontal lines at the given quantiles of the density
estimate.
trim If TRUE (default), trim the tails of the violins to the range of the data. If FALSE,
don’t trim the tails.
scale if "area" (default), all violins have the same area (before trimming the tails).
If "count", areas are scaled proportionally to the number of observations. If
"width", all violins have the same maximum width.
bw The smoothing bandwidth to be used. If numeric, the standard deviation of
the smoothing kernel. If character, a rule to choose the bandwidth, as listed in
stats::bw.nrd().
adjust A multiplicate bandwidth adjustment. This makes it possible to adjust the band-
width while still using the a bandwidth estimator. For example, adjust = 1/2
means use half of the default bandwidth.
kernel Kernel. See list of available kernels in density().
xlab Label for x-axis. See also gf_labs().
ylab Label for y-axis. See also gf_labs().
title Title, sub-title, and caption for the plot. See also gf_labs().
subtitle Title, sub-title, and caption for the plot. See also gf_labs().
caption Title, sub-title, and caption for the plot. See also gf_labs().
geom Use to override the default connection between geom_violin and stat_ydensity.
stat Use to override the default connection between geom_violin and stat_ydensity.
position Position adjustment, either as a string, or the result of a call to a position adjust-
ment function.
show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

show.help If TRUE, display some minimal help.

inherit A logical indicating whether default attributes are inherited.

environment An environment in which to look for variables not found in data.

Value

a gg object

Specifying plot attributes

Positional attributes (a.k.a, aesthetics) are specified using the formula in `gformula`. Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`.

In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of `lattice`.

Evaluation

Evaluation of the `ggplot2` code occurs in the environment of `gformula`. This will typically do the right thing when formulas are created on the fly, but might not be the right thing if formulas created in one environment are used to create plots in another.

References


See Also

`ggplot2::geom_violin()`

Examples

gf_violin(age ~ substance, data = mosaicData::HELPrc)
gf_violin(age ~ substance, data = mosaicData::HELPrc, fill = ~sex)
gf_violinh(substance ~ age, data = mosaicData::HELPrc)
gf_violinh(substance ~ age, data = mosaicData::HELPrc, fill = ~sex)
Description

Formula interface to ggplot2

The ggformula system

The functions in ggformula provide a formula interface to ggplot2 layer functions and a system for working with pipes to create multi-layer plots and to refine plots. For plots with just one layer, the formula interface is more compact than native ggplot2 code and is consistent with modeling functions like stats::lm() that use a formula interface and with the numerical summary functions in the mosaic package.

Specifying plot attributes

Positional attributes (a.k.a. aesthetics) are typically specified using a formula (see the gformula argument). Setting and mapping of additional attributes can be done through the use of additional arguments. Attributes can be set can be set using arguments of the form attribute = value or mapped using arguments of the form attribute = ~ expression. A (sometimes partial) list of available attributes can be obtained by executing plotting functions with no arguments.

In formulas of the form A | B, B will be used to form facets using facet_wrap() or facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of lattice.

Evaluation

Evaluation of the ggplot2 code occurs in the environment specified by environment. This will typically do the right thing, but is exposed in case some non-standard behavior is desired. In earlier versions, the environment of the formula was used, but since some functions in the package do not require a formula, a separate argument is used now.

Examples

apropos("gf-")
gf_point()
layer_factory

Create a ggformula layer function

Description

Primarily intended for package developers, this function factory is used to create the layer functions in the ggformula package.

Usage

layer_factory(
  geom = "point",
  position = "identity",
  stat = "identity",
  pre = { },
  aes_form = y ~ x,
  extras = alist(),
  note = NULL,
  aesthetics = aes(),
  inherit.aes = TRUE,
  check.aes = TRUE,
  data = NULL,
  layer_fun = quo(ggplot2::layer),
  ...
)

Arguments

geom The geom to use for the layer (may be specified as a string).
position The position function to use for the layer (may be specified as a string).
stat The stat function to use for the layer (may be specified as a string).
pre code to run as a "pre-process".
aes_form A single formula or a list of formulas specifying how attributes are inferred from the formula. Use NULL if the function may be used without a formula.
extras An alist of additional arguments (potentially with defaults)
note A note to add to the quick help.
aesthetics Additional aesthetics (typically created using ggplot2::aes()) set rather than inferred from formula. gf_dhistogram() uses this to set the y aesthetic to stat(density), for example.
inherit.aes A logical indicating whether aesthetics should be inherited from prior layers or a vector of character names of aesthetics to inherit.
check.aes A logical indicating whether a warning should be emitted when aesthetics provided don’t match what is expected.
data A data frame or NULL or NA.
layer_fun

The function used to create the layer or a quosure that evaluates to such a function.

Additional arguments.

Value

A function.

---

**MIpop**

*Population of Michigan counties*

**Description**

Population of Michigan counties

**Usage**

data(MIpop)

**Format**

A data frame with populations of Michigan counties.

- **rank** Population rank.
- **county** County name.
- **population** Population (2010 census).

---

**percs_by_group**

*Compute groupwise proportions and percents*

**Description**

Transform a vector of counts and a vector of groups into a vector of proportions or percentages within groups.

**Usage**

- percs_by_group(x, group)
- props_by_group(x, group)

**Arguments**

- **x** A vector of counts
- **group** A vector to determine groups.
**Examples**

```r
x <- c(20, 30, 30, 70)
g1 <- c("A", "A", "B", "B")
g2 <- c("A", "B", "A", "B")
props_by_group(x, g1)
percs_by_group(x, g1)
props_by_group(x, g2)
```

---

**StatAsh**

*ggproto classes for ggplot2*

**Description**

These are typically accessed through their associated `geom_*`, `stat_*` or `gf_*` functions.

**Usage**

- `StatAsh`
- `StatSpline`
- `StatQqline`
- `StatLm`
- `GeomLm`
- `StatAsh`
- `StatFitdistr`

**See Also**

- `stat_ash()`
- `gf_ash()`
- `stat_spline()`
- `gf_spline()`
- `stat_qq()`
- `gf_qq()`
- `stat_lm()`
- `gf_lm()`
- `geom_lm()`
Description

This stat computes points for plotting a distribution function. Fitting is done using \texttt{MASS::fitdistr()} when analytic solutions are not available.

Usage

\begin{verbatim}
stat_fitdistr(
  mapping = NULL,
  data = NULL,
  geom = "path",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  dist = "dnorm",
  start = NULL,
  ...
)
\end{verbatim}

Arguments

- \texttt{mapping} Aesthetics created using \texttt{aes()} or \texttt{aes_string()}.
- \texttt{data} A data frame.
- \texttt{geom} A character string naming the geom used to make the layer.
- \texttt{position} Either a character string naming the position function used for the layer or a position object returned from a call to a position function.
- \texttt{na.rm} If \texttt{TRUE}, do not emit a warning about missing data.
- \texttt{show.legend} A logical. Should this layer be included in the legends? \texttt{NA}, the default, includes if any aesthetics are mapped. \texttt{FALSE} never includes, and \texttt{TRUE} always includes.
- \texttt{inherit.aes} If \texttt{FALSE}, overrides the default aesthetics, rather than combining with them.
- \texttt{dist} A character string indicating the distribution to fit. Examples include "dnorm", "dgamma", etc.
- \texttt{start} A list of starting values used by \texttt{MASS::fitdistr()} when numerically approximating the maximum likelihood estimate.
- \texttt{...} Additional arguments.
Value

A gg object

stat_lm

Linear Model Displays

Description

Adds linear model fits to plots. geom_lm() and stat_lm() are essentially equivalent. Use geom_lm() unless you want a non-standard geom.

Usage

stat_lm(
  mapping = NULL,
  data = NULL,
  geom = "lm",
  position = "identity",
  interval = c("none", "prediction", "confidence"),
  level = 0.95,
  formula = y ~ x,
  lm.args = list(),
  backtrans = identity,
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_lm(
  mapping = NULL,
  data = NULL,
  stat = "lm",
  position = "identity",
  interval = c("none", "prediction", "confidence"),
  level = 0.95,
  formula = y ~ x,
  lm.args = list(),
  backtrans = identity,
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
Arguments

- `mapping`: Set of aesthetic mappings created by `aes()` or `aes_()`. If specified and `inherit.aes` = `TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

- `data`: The data to be displayed in this layer. There are three options:
  - If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a `formula` (e.g. `~ head(.x, 10)`).

- `geom`, `stat`: Use to override the default connection between `geom_lm` and `stat_lm`.

- `position`: Position adjustment, either as a string, or the result of a call to a position adjustment function.

- `interval`: One of "none", "confidence" or "prediction".

- `level`: The level used for confidence or prediction intervals

- `formula`: A formula describing the model in terms of `y` (response) and `x` (predictor).

- `lm.args`: A list of arguments supplied to `lm()` when performing the fit.

- `backtrans`: A function that transforms the response back to the original scale when the formula includes a transformation on `y`.

- `...`: Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

- `na.rm`: If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

- `show.legend`: Logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

- `inherit.aes`: If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.

Details

Stat calculation is performed by the (currently undocumented) `predictdf`. Pointwise confidence or prediction bands are calculated using the `predict()` method.

See Also

- `lm()` for details on linear model fitting.
Examples

```r
ggplot(data = mosaicData::KidsFeet, aes(y = length, x = width, color = sex)) +
  geom_lm() +
  geom_point()

ggplot(data = mosaicData::KidsFeet, aes(y = length, x = width, color = sex)) +
  geom_lm(interval = "prediction", color = "skyblue") +
  geom_lm(interval = "confidence") +
  geom_point() +
  facet_wrap(~sex)

# non-standard display
ggplot(data = mosaicData::KidsFeet, aes(y = length, x = width, color = sex)) +
  stat_lm(aes(fill = sex),
          color = NA, interval = "confidence", geom = "ribbon",
          alpha = 0.2
    ) +
  geom_point() +
  facet_wrap(~sex)

ggplot(mpg, aes(displ, hwy)) +
  geom_lm(formula = log(y) ~ poly(x, 3), backtrans = exp,
          interval = "prediction", fill = "skyblue"
    ) +
  geom_lm(formula = log(y) ~ poly(x, 3), backtrans = exp, interval = "confidence",
          color = "red"
    ) +
  geom_point()
```

stat_qqline

A Stat for Adding Reference Lines to QQ-Plots

Description

This stat computes quantiles of the sample and theoretical distribution for the purpose of providing reference lines for QQ-plots.

Usage

```r
stat_qqline(
  mapping = NULL,
  data = NULL,
  geom = "line",
  position = "identity",
  ...,
  distribution = stats::qnorm,
  dparams = list(),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```
Arguments

- **mapping**: An aesthetic mapping produced with `aes()` or `aes_string()`.
- **data**: A data frame.
- **geom**: A geom.
- **position**: A position object.
- **distribution**: A quantile function.
- **dparams**: A list of arguments for `distribution`.
- **na.rm**: A logical indicating whether a warning should be issued when missing values are removed before plotting.
- **show.legend**: A logical indicating whether legends should be included for this layer. If `NA`, legends will be include for each aesthetic that is mapped.
- **inherit.aes**: A logical indicating whether aesthetics should be inherited. When `FALSE`, the supplied mapping will be the only aesthetics used.

Examples

```r
data(penguins, package = "palmerpenguins")
ggplot(data = penguins, aes(sample = bill_length_mm)) +
  geom_qq() +
  stat_qqline(alpha = 0.7, color = "red", linetype = "dashed") +
  facet_wrap(~species)
```

Description

Similar to `geom_smooth`, this adds spline fits to plots.

Usage

```r
stat_spline(
  mapping = NULL,
  data = NULL,
  geom = "line",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  weight = NULL,
  df = NULL,
  spar = NULL,
  cv = FALSE,
```
stat_spline

all.knots = FALSE,
nknots = stats::.nknots.smspl,
df.offset = 0,
penalty = 1,
control.spar = list(),
tol = NULL,

geom_spline

mapping = NULL,
data = NULL,
stat = "spline",
position = "identity",
na.rm = FALSE,
show.legend = NA,
inherit.aes = TRUE,
weight = NULL,
df = NULL,
spar = NULL,
cv = FALSE,
all.knots = FALSE,
nknots = stats::.nknots.smspl,
df.offset = 0,
penalty = 1,
control.spar = list(),
tol = NULL,

Arguments

**mapping**  
An aesthetic mapping produced with `aes()` or `aes_string()`.

**data**  
A data frame.

**geom**  
A geom.

**position**  
A position object.

**na.rm**  
A logical indicating whether a warning should be issued when missing values are removed before plotting.

**show.legend**  
A logical indicating whether legends should be included for this layer. If NA, legends will be included for each aesthetic that is mapped.

**inherit.aes**  
A logical indicating whether aesthetics should be inherited. When FALSE, the supplied mapping will be the only aesthetics used.

**weight**  
An optional vector of weights. See `smooth.spline()`.

**df**  
desired equivalent degrees of freedom. See `smooth.spline()` for details.

**spar**  
A smoothing parameter, typically in (0,1]. See `smooth.spline()` for details.

**cv**  
A logical. See `smooth.spline()` for details.
all.knots  A logical. See smooth.spline() for details.
nknots    An integer or function giving the number of knots to use when all.knots = FALSE. See smooth.spline() for details.
df.offset A numerical value used to increase the degrees of freedom when using GVC. See smooth.spline() for details.
penalty   the coefficient of the penalty for degrees of freedom in the GVC criterion. See smooth.spline() for details.
control.spar An optional list used to control root finding when the parameter spar is computed. See smooth.spline() for details.
tol       A tolerance for sameness or uniqueness of the x values. The values are binned into bins of size tol and values which fall into the same bin are regarded as the same. Must be strictly positive (and finite). When NULL, IQR(x) * 10e-6 is used.
...       Additional arguments
stat      A stat.

Examples

if (require(mosaicData)) {
  ggplot(Births) + geom_spline(aes(x = date, y = births, colour = wday))
  ggplot(Births) + geom_spline(aes(x = date, y = births, colour = wday), nknots = 10)
}

---

var_label

Set and extract labels from a labeled object

Description

Some packages like expss provide mechanisms for providing longer labels to R objects. These labels can be used when labeling plots and tables, for example, without requiring long or awkward variable names. This is an experimental feature and currently only supports expss or any other system that stores a label in the label attribute of a vector.

Usage

var_label(x, unlist = FALSE)

var_label(x) <- value

get_variable_labels(x, unlist = FALSE)

var_label(x, unlist = FALSE)

set_variable_labels(.data, ..., .labels = NA, .strict = TRUE)
Arguments

x an object
unlist for data frames, return a named vector instead of a list
value a character string or NULL to remove the label For data frames, it could also be a
named list or a character vector of same length as the number of columns in x.
.data a data frame
... name-value pairs of variable labels (see examples)
.labels variable labels to be applied to the data.frame, using the same syntax as value
in var_label(df) <-value.
.strict should an error be returned if some labels doesn’t correspond to a column of x?

Details

For data frames, if value is a named list, only elements whose name will match a column of the
data frame will be taken into account. If value is a character vector, labels should in the same order
as the columns of the data.frame.

Value

set_variable_labels() will return an updated copy of .data.

Note

These functions are imported from the \{labelled\} package.

Examples

KF <-
mosaicData::KidsFeet %>%
set_variable_labels(
  length = 'foot length (cm)',
  width = 'foot width (cm)',
  birthmonth = 'birth month',
  birthyear = 'birth year',
  biggerfoot = 'bigger foot',
  domhand = 'dominant hand'
)
KF %>%
gf_point(length ~ width, color = ~ domhand)
get_variable_labels(KF)
Index

* datasets
  StatAsh, 168
  aes(), 12, 171, 173, 174
  aes_(), 12, 171
  aes_string(), 173, 174
  borders(), 171
  density(), 53, 163
  df_stats(), 32, 36
  discrete_breaks, 3
  expand_limits(), 26
  facet_wrap(), 12, 19, 22, 25, 27, 31, 35, 38, 41, 43, 46, 50, 53, 58, 63, 72, 73, 76, 79, 85, 90, 93, 97, 107, 109, 113, 116, 118, 120, 134, 137, 139, 142, 145, 148, 150, 153, 157, 160, 164, 165
  fivenum(), 32, 36
  fortify(), 5, 8, 17, 21, 29, 33, 40, 45, 48, 52, 57, 64, 69, 71, 78, 84, 88, 101, 112, 114, 123, 128, 132, 136, 141, 149, 152, 155, 162, 171
  geom_ash(gf_ash), 10
  geom_bar, 20
  geom_boxplot, 32
  geom_boxplot(), 113, 161
  geom_density(), 54, 161
  geom_density_2d(), 39
  geom_errorbar(), 70
  geom_histogram(), 12
  geom_lm(stat_lm), 170
  geom_lm(), 168
  geom_point(), 42
  geom_smooth, 173
  geom_spline(stat_spline), 173
  geom_spline(), 148
  geom_tile(), 39
  geomLm(StatAsh), 168
  get_variable_labels(var_label), 175
  gf_abline, 4
  gf_area, 7
  gf_ash, 10
  gf_ash(), 54, 168, 169
  gf_bar, 13
  gf_barh, 20
  gf_bin2d, 23
  gf_blank, 26
  gf_boxplot, 28
  gf_boxplot2, 32
  gf_coefline(gf_abline), 4
  gf_col, 36
  gf_col(), 17, 21
  gf_colh(gf_bar), 13
  gf_contour, 39
  gf_contour(), 83
  gf_contour_filled(gf_contour), 39
  gf_count, 42
  gf_counts(gf_bar), 13
  gf_countsh(gf_bar), 13
  gf_crossbar, 44
  gf_crossbarh(gf_crossbar), 44
  gf_curve, 47
  gf_density(gf_density), 50
  gf_density2d(gf_density), 50
  gf_density, 50
  gf_density2d(gf_density_2d), 54
  gf_density2d_filled(gf_density_2d), 54
  gf_density_2d, 54
  gf_density_2d(), 41
gf_density_ridgeline_gradient (gf_ridgeline), 124

gf_density_ridges2 (gf_ridgeline), 124

gf_density_ridges (gf_ridgeline), 124

gf_density_ridges_gradient (gf_ridgeline), 124

gf_dhistogramh (gf_histogram), 86

gf_dhistogram (gf_histogram), 86

gf_dist, 59

gf_dotplot, 60

gf_ecdf, 63

gf_ellipse, 65

gf_empty, 67

gf_errorbar, 68

gf_errorbarh, 70

gf_facet_grid (gf_facet_wrap), 73


gf_facet_wrap, 73


gf_fitdistr, 74

gf_frame (gf_blank), 26

gf_frepoly, 77

gf_fun (gf_function), 80

gf_fun2d (gf_function_2d), 81

gf_function2d (gf_function_2d), 81

gf_fun_contour (gf_function_2d), 81

gf_function_contour (gf_function_2d), 81

gf_function_tile (gf_function_2d), 81

gf_function, 80

gf_function_2d (gf_function_2d), 81

gf_hex, 83

gf_histogram, 86

gf_histogramh (gf_histogram), 86

gf_hline (gf_abline), 4

gf_jitter, 91

gf_jitter (), 107, 110

gf_label (gf_text), 154

gf_labs, 93


gf_lim, 95

gf_line, 107, 110

gf_line (), 107, 110

gf_linerange, 98

gf_linerangeh (gf_linerange), 98

gf_lm (gf_smooth), 142

gf_lm (), 148, 168, 169

gf_path (gf_line), 95

gf_percent, 13

gf_percent (), 17, 21

gf_percentsh (gf_bar), 13

gf_pl, 104

gf_point, 105

gf_point (), 93, 97, 139

gf_pointrange, 98

gf_pointrangeh (gf_pointrange), 98

gf_polygon, 108

gf_props (gf_bar), 13

gf_props (), 17, 21

gf_props_h (gf_bar), 13

gf_qq, 110

gf_qq (), 168

gf_qq_line (gf_qq), 110

gf_qq_step (gf_qq), 110

gf_quantile, 113

gf_raster, 116

gf_rect, 119

gf_refine (gf_labs), 93

gf_rescale, 121

gf_ridgeline, 124

gf_ridgeline, 124

gf_rug, 130

gf_rug (gf_rug), 130

gf_rug (gf_rug), 130

gf_segment, 135

gf_sf, 137

gf_sina, 140

gf_smooth, 142
gf_smooth(), 148
gf_spline(), 146
gf_spline(), 145, 168
gf_spoke, 148
gf_step, 151
gf_summary (gf_linerange), 98
gf_text, 154
gf_tile, 158
gf_violin, 161
gf_violin (gf_violin), 161
ggforce::geom_sina(), 142
ggplot(), 5, 8, 17, 21, 29, 33, 40, 45, 48, 52, 57, 64, 69, 71, 78, 84, 88, 101, 112, 114, 123, 128, 132, 136, 141, 149, 152, 155, 162, 171
ggplot2::aes(), 166
ggplot2::facet_grid(), 73
ggplot2::facet_wrap(), 73
ggplot2::fortify(), 105
ggplot2::geom_abline(), 6
ggplot2::geom_area(), 9
ggplot2::geom_bar(), 19
ggplot2::geom_bin2d(), 25
ggplot2::geom_blank(), 28
ggplot2::geom_boxplot(), 32
ggplot2::geom_col(), 38
ggplot2::geom_contour(), 41
ggplot2::geom_count(), 44
ggplot2::geom_crossbar(), 46
ggplot2::geom_curve(), 50
ggplot2::geom_density(), 54
ggplot2::geom_density_2d(), 58
ggplot2::geom_dotplot(), 63
ggplot2::geom_errorbar(), 70
ggplot2::geom_errorbarh(), 72
ggplot2::geom_freqpoly(), 79
ggplot2::geom_hex(), 85
ggplot2::geom_histogram(), 90
ggplot2::geom_hline(), 6
ggplot2::geom_jitter(), 93
ggplot2::geom_line(), 97, 139
ggplot2::geom_linerange(), 102
ggplot2::geom_point(), 107, 110
ggplot2::geom_pointrange(), 102
ggplot2::geom_qq(), 113
ggplot2::geom_quantile(), 116
ggplot2::geom_raster(), 118
ggplot2::geom_rect(), 121
ggplot2::geom.ribbon(), 124
ggplot2::geom_rug(), 134
ggplot2::geom_segment(), 137
ggplot2::geom_smooth(), 145
ggplot2::geom_spoke(), 151
ggplot2::geom_step(), 153
ggplot2::geom_text(), 157
ggplot2::geom_tile(), 161
ggplot2::geom_violin(), 164
ggplot2::geom_vline(), 6
ggplot2::ggplot(), 68
ggplot2::stat_bin2d(), 23
ggplot2::stat_density(), 129
ggplot2::stat_ellipse(), 65, 67
ggplot2::stat_summary(), 102
ggplot2::theme(), 158
ggridges::geom_density_ridges(), 129
ggridges::geom_density_ridges_gradient(), 129
ggridges::geom_density_ridges_gradient(), 129
layer(), 64, 141, 171
layer_factory, 166
lm(), 171
MASS::bandwidth.nrd(), 57
MASS::fitdistr, 75
MASS::kde2d(), 54
mgcv::gam(), 144
MIpop, 167
mosaic::makeFun(), 83
mosaicCore::fit_distr_fun(), 75, 76
mosaicCore::makeFun(), 80
percs_by_group, 167
predict(), 171
print.ggplot (gf_relabel), 121
props_by_group (percs_by_group), 167
quantreg::rq(), 115
quantreg::rqss(), 115

set_variable_labels(var_label), 175
smooth.spline(), 147, 174, 175
stat_ash(gf_ash), 10
stat_ash(), 168, 169
stat_fitdistr, 169
stat_lm, 170
stat_lm(), 168
stat_qq(), 168
stat_qqline, 172
stat_spline, 173
stat_spline(), 168
StatAsh, 168
StatFitdistr(StatAsh), 168
StatLm(StatAsh), 168
StatQqline(StatAsh), 168
stats::bw.nrd(), 163
stats::lm(), 144, 165
stats::loess(), 144
StatSpline(StatAsh), 168

theme_minimal(), 158

var_label, 175
var_label<- (var_label), 175