Package ‘ggformula’

November 9, 2023

Title  Formula Interface to the Grammar of Graphics
Description  Provides a formula interface to 'ggplot2' graphics.
Type  Package
Version  0.12.0
License  MIT + file LICENSE
LazyData  TRUE
LazyLoad  TRUE
Depends  R (>= 4.1), ggplot2 (>= 3.3), scales, ggridges
Imports  mosaicCore (>= 0.7.0), rlang, tibble, stringr, grid, labelled
Suggests  tidyr, mosaicData, dplyr, lattice, mosaic, palmerpenguins,
         testthat, vdiffr, knitr, rmarkdown, lubridate, survival, broom,
         maps, sf, purrr, ggthemes, covr, ggplot2movies, interp,
         quantreg
VignetteBuilder  knitr
RoxygenNote  7.2.3
Encoding  UTF-8
URL  https://github.com/ProjectMOSAIC/ggformula
BugReports  https://github.com/ProjectMOSAIC/ggformula/issues
         ‘layer_factory.R’ ‘gf_function2d.R’ ‘gf_functions.R’
NeedsCompilation  no
Author  Daniel Kaplan [aut],
        Randall Pruim [aut, cre]
Maintainer  Randall Pruim <rpruim@calvin.edu>
Repository  CRAN
Date/Publication  2023-11-09 12:30:07 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>get_variable_labels</td>
<td>4</td>
</tr>
<tr>
<td>gf_abline</td>
<td>5</td>
</tr>
<tr>
<td>gf_area</td>
<td>8</td>
</tr>
<tr>
<td>gf_ash</td>
<td>10</td>
</tr>
<tr>
<td>gf_bar</td>
<td>13</td>
</tr>
<tr>
<td>gf_barch</td>
<td>19</td>
</tr>
<tr>
<td>gf_bin2d</td>
<td>20</td>
</tr>
<tr>
<td>gf_blank</td>
<td>22</td>
</tr>
<tr>
<td>gf_boxplot</td>
<td>24</td>
</tr>
<tr>
<td>gf_col</td>
<td>28</td>
</tr>
<tr>
<td>gf_contour</td>
<td>30</td>
</tr>
<tr>
<td>gf_count</td>
<td>33</td>
</tr>
<tr>
<td>gf_crossbar</td>
<td>35</td>
</tr>
<tr>
<td>gf_curve</td>
<td>38</td>
</tr>
<tr>
<td>gf_density</td>
<td>41</td>
</tr>
<tr>
<td>gf_density_2d</td>
<td>45</td>
</tr>
<tr>
<td>gf_dist</td>
<td>50</td>
</tr>
<tr>
<td>gf_dotplot</td>
<td>51</td>
</tr>
<tr>
<td>gf_ecdf</td>
<td>54</td>
</tr>
<tr>
<td>gf_ellipse</td>
<td>56</td>
</tr>
<tr>
<td>gf_empty</td>
<td>58</td>
</tr>
<tr>
<td>gf_errorbar</td>
<td>59</td>
</tr>
<tr>
<td>gf_facet_wrap</td>
<td>61</td>
</tr>
<tr>
<td>gf_fitdistr</td>
<td>62</td>
</tr>
<tr>
<td>gf_freqpoly</td>
<td>65</td>
</tr>
<tr>
<td>gf_function</td>
<td>68</td>
</tr>
<tr>
<td>gf_function_2d</td>
<td>69</td>
</tr>
<tr>
<td>gf_hex</td>
<td>71</td>
</tr>
<tr>
<td>gf_histogram</td>
<td>74</td>
</tr>
<tr>
<td>gf_litter</td>
<td>78</td>
</tr>
<tr>
<td>gf_labs</td>
<td>80</td>
</tr>
<tr>
<td>gf_line</td>
<td>81</td>
</tr>
<tr>
<td>gf_linerange</td>
<td>84</td>
</tr>
<tr>
<td>gf_plot</td>
<td>90</td>
</tr>
<tr>
<td>gf_point</td>
<td>91</td>
</tr>
<tr>
<td>gf_polygon</td>
<td>93</td>
</tr>
<tr>
<td>gf_qq</td>
<td>96</td>
</tr>
<tr>
<td>gf_quantile</td>
<td>99</td>
</tr>
<tr>
<td>gf_raster</td>
<td>102</td>
</tr>
<tr>
<td>gf_rect</td>
<td>104</td>
</tr>
<tr>
<td>gf_relabel</td>
<td>106</td>
</tr>
<tr>
<td>gf_ribbon</td>
<td>107</td>
</tr>
<tr>
<td>gf_ridgeline</td>
<td>109</td>
</tr>
<tr>
<td>gf_rug</td>
<td>116</td>
</tr>
<tr>
<td>gf_segment</td>
<td>120</td>
</tr>
</tbody>
</table>
**discrete_breaks**

Discrete Breaks

Description

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

Usage

```r
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

```r
x <- rbinom(100, 100, 0.4)
p <- gf_bar(~ x)
p |> gf_refine(scale_x_continuous(breaks = discrete_breaks()))
p |> gf_refine(scale_x_continuous(breaks = discrete_breaks(5)))
p |> gf_refine(scale_x_continuous(breaks = discrete_breaks(2)))
```
get_variable_labels  

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

get_variable_labels(...)

Arguments

... passed to labelled::var_label()

Details

get_variable_labels() is a synonym of labelled::var_label().

See Also

labelled::var_label(), labelled::set_variable_labels()

Examples

KF <-
mosaicData::KidsFeet |> 
set_variable_labels(
  length = 'foot length (cm)',
  width  = 'foot width (cm)',
  birthmonth = 'birth month',
  bithyear  = 'birth year',
  biggerfoot = 'bigger foot',
  domhand = 'dominant hand'
)
KF |> 
gf_point(length ~ width, color = ~ domhand)
get_variable_labels(KF)
**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,
  linetype,
  linewidth,
  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,
  linetype,
  linewidth,
  alpha,
  xlab,
  ylab,
  title,
  subtitle,
```
caption,
show.legend = NA,
show.help = NULL,
inherit = FALSE,
environment = parent.frame()
)

gf_vline(
  object = NULL,
gformula = NULL,
data = NULL,
  ...,
xintercept,
color,
linetype,
linewidth,
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.
color
    A color or a formula used for mapping color.
linetype
    A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping
    linetype.
linewidth
    A numerical line width or a formula used for mapping linewidth.
alpha
    Opacity (0 = invisible, 1 = opaque).
mlab
    Label for x-axis. See also gf_labs().
mlab
    Label for y-axis. See also gf_labs().
title, subtitle, 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.
xintercept, yintercept, slope, intercept
    Parameters that control the position of the line. If these are set, data, mapping
    and show.legend are overridden.
coef
    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(
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,
**gf_area**

```r
fill,
group,
linetype,
linewidth,
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.

- **linewidth**
  A numerical line width or a formula used for mapping linewidth.

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

**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**.

**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, color = ~high_temp, data = Weather) |
  gf_facet_grid(city ~ .) |
  gf_refine(scale_colour_gradientn(colors = rev(rainbow(5))))
}
```

**gf_ash**

- `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`: The geometric object to use to display the data, either as a ggproto Geom subclass or as a string naming the geom stripped of the `geom_` prefix (e.g. "point" rather than "geom_point")
- `stat`: The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the `stat_` prefix (e.g. "count" rather than "stat_count")
- `position`: Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use `position_jitter`), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
- `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()`
Usage

gf_ash(
  object = NULL,
  gfformula = NULL,
  data = NULL,
  ..., 
  alpha, 
  color, 
  group, 
  linetype, 
  linewidth, 
  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, 
  ...
)

gem_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 \( \sim x \) or \( y \sim x \). \( y \) may be \text{stat(density)} or \text{stat(count)} or \text{stat(ndensity)} or \text{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 = \( \sim \) 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.

linewidth
A numerical line width or a formula used for mapping linewidth.

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

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

title, subtitle, caption
Title, sub-title, and caption for the plot. See also \text{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.

mapping
set of aesthetic mappings created by \text{aes()} or \text{aes_()}.

na.rm
If \texttt{FALSE} (the default), removes missing values with a warning. If \texttt{TRUE} silently removes missing values.

inherit.aes
A logical indicating whether default aesthetics are inherited.

binwidth
the width of the histogram bins. If \text{NULL} (the default) the binwidth will be chosen so that approximately 10 bins cover the data. \text{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**

```r
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
  ) +
  geom_ash(colour = "red") +
  geom_ash(colour = "forestgreen", adjust = 2) +
  geom_ash(colour = "navy", adjust = 1 / 2) +
  theme_minimal()
```

---

**gf_bar**

*Formula interface to geom_bar()*

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

gf_counts(
  object = NULL, 
  gformula = NULL, 
  data = NULL, 
  ..., 
  alpha, 
  color, 
  fill, 
  group, 
  linetype, 
  linewidth, 
  width = NULL, 
  xlab, 
  ylab, 
  title, 
  subtitle, 
  caption, 
  geom = "bar", 
  stat = "count", 
  position = "stack", 
  show.legend = NA,
gf_bar

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

)

gf_props(
    object = NULL,
    gformula = NULL,
    data = NULL,
    ...
    alpha,
    color,
    fill,
    group,
    linetype,
    linewidth,
    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_percents(
    object = NULL,
    gformula = NULL,
    data = NULL,
    ...
    alpha,
    color,
    fill,
    group,
    linetype,
    linewidth,
    xlab,
    ylab = "percent",
    title,
    subtitle,
    caption,
    geom = "bar",
    }
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 \texttt{gf_col()}, \texttt{gf_props()}, or \texttt{gf_percent()}. (Using this formula 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 \texttt{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 \texttt{formula} (e.g. \texttt{~ head(.x, 10)}).

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

linewidth
A numerical line width or a formula used for mapping linewidth.

width
Width of the bars.
xlab
Label for x-axis. See also \texttt{gf_labs()}.
ylab
Label for y-axis. See also \texttt{gf_labs()}.
title, subtitle, caption
Title, sub-title, and caption for the plot. See also \texttt{gf_labs()}.

geom, stat
Override the default connection between \texttt{geom_bar()} and \texttt{stat_count()}.

position
Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use \texttt{position_jitter}), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
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 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_bar()

Examples

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_props() and gf_percent() use proportions or percentages instead of counts
# use denom to control which denominators are used.
gf_props(~substance,
  data = mosaicData::HELPrct, fill = ~sex,
  position = position_dodge()
)
gf_props(substance ~ .,
  data = mosaicData::HELPrct, fill = ~sex,
  position = position_dodge(),
  orientation = 'y'
)
gf_props(substance ~ .,
  data = mosaicData::HELPrct, fill = ~sex,
  position = "dodge"
)

gf_percents(~substance,
  data = mosaicData::HELPrct, fill = ~sex,
  position = position_dodge()
)
gf_percents(~substance,
  data = mosaicData::HELPrct, fill = ~sex,
  position = position_dodge(),
  denom = ~x
)
gf_percents(~substance,
  data = mosaicData::HELPrct, fill = ~sex,
  position = position_dodge(),
  denom = ~fill
)
gf_percents(~substance | sex,
  data = mosaicData::HELPrct, fill = ~homeless,
  position = position_dodge()
)
gf_percents(~substance | sex,
  data = mosaicData::HELPrct,
  fill = ~homeless,
  denom = ~fill,
  position = position_dodge()
)
gf_percents(~substance | sex,
  data = mosaicData::HELPrct,
  fill = ~homeless,
  denom = ~interaction(fill, PANEL),
  position = position_dodge()
)
if (require(scales)) {
gf_percents(~substance,
  data = mosaicData::HELPrct, fill = ~sex,
  position = position_dodge(),
  denom = ~ x,
  ) |>  
  gf_refine(scale_y_continuous(labels = scales::percent))
}
Description

These functions were wrappers around functions from ggstance from an era before ggplot2 supported horizontally oriented geoms. ggstance has not been updated to comply with the current version of ggplot2, and since the functionality is now available by other means, these functions have been deprecated.

Usage

gf_barh(...)
gf_countsh(...)
gf_colh(...)
gf_propsh(...)
gf_percentsh(...)
gf_boxploth(...)
gf_linerangeh(...)
gf_pointrangeh(...)
gf_crossbarh(...)
gf_violinh(...)
gf_errorbarh(...)

Arguments

... additional arguments

Examples

gf_violin(carat ~ color, data = diamonds)
gf_violin(carat ~ color, data = diamonds) |>  
gf_refine(coord_flip())
gf_violin(color ~ carat, data = diamonds)
gf_density(~ carat, data = diamonds)
gf_density(carat ~ ., data = diamonds)
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,
  linewidth,
  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.
linewidth A numerical line width or a formula used for mapping linewidth.
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.

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 \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.

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_bin2d()}, \texttt{gf_tile()}

Examples

\begin{verbatim}
gf_bin2d(eruptions ~ waiting, data = faithful, bins = 15) |>
  gf_refine(scale_fill_viridis_c(begin = 0.1, end = 0.9))
\end{verbatim}

\begin{verbatim}
gf_blank
\end{verbatim}

\textit{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 \texttt{expand_limits()} for more details.

Usage

\begin{verbatim}
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()  
)
\end{verbatim}

\begin{verbatim}
gf_frame(  
)
\end{verbatim}
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, 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_blank()`

Examples

```r
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)))
```

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

```r
gf_boxplot(
  object = NULL,
  gformula = NULL,
  data = NULL,
)```
..., alpha, color, fill, group, linetype, linewidth, 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 = "boxplot", stat = "boxplot", position = "dodge", 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 \((0 = \text{invisible}, 1 = \text{opaque})\).

- **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.

- **linewidth**
  - A numerical line width or a formula used for mapping linewidth.

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

- **outlier.color, outlier.fill, outlier.shape, outlier.size, outlier.stroke, 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.

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

- **geom, stat**
  - Use to override the default connection between geom_boxplot() and stat_boxplot().

- **position**
  - Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

- **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_boxplot

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_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
)
n> gf_jitter(width = 0.2, alpha = 0.3)
# move boxplots away a bit by adjusting dodge
gf_boxplot(age ~ substance,
data = mosaicData::HELPrct,
color = ~sex, position = position_dodge(width = 0.9))

gf_col

GF 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, 
linewidth, 
  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 \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.
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.
linewidth
A numerical line width or a formula used for mapping linewidth.
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_col()`

Examples

```r
SomeData <- data.frame(
  group = LETTERS[1:3],
  count = c(20, 25, 18)
)
gf_col(count ~ group, data = SomeData)

# A Pareto chart

if (require(dplyr) && require(mosaicData)) {
  HELPrct |> |
    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(HELPrct)))
    )
}
```

`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.

These functions require regular data, where the x and y coordinates form an equally spaced grid, and each combination of x and y appears once. Missing values of z are allowed, but contouring will only work for grid points where all four corners are non-missing. If you have irregular data, you’ll need to first interpolate on to a grid before visualising, using `interp::interp()`, `akima::bilinear()`, or similar.
gf_contour

Usage

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,
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 ~ 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().
gf_contour

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

geom  
The geometric object to use to display the data, either as a ggproto Geom subclass or as a string naming the geom stripped of the `geom_` prefix (e.g. "point" rather than "geom_point")

stat  
The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the `stat_` prefix (e.g. "count" rather than "stat_count")

position  
Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use `position_jitter`), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

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`. 
**gf_count** 33

**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 = faithfuld, bins = 10, color = "red")
gf_contour_filled(density ~ waiting + eruptions, data = faithfuld, 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**

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

Examples

```r
# 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())
```

### Description

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

### Usage

```r
gf_crossbar(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha,
  color,
  group,
  linetype,
)```
linewidth,
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()
)

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.
linewidth A numerical line width or a formula used for mapping linewidth.
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, 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
   The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count").

position
   Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

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
   if (require(mosaicData) && require(dplyr)) {
     HELP2 <- HELPrct |> summarize(.by = c(substance, sex),
                                  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_crossbar(substance ~ mean.age + lo + hi, data = HELP2,
  fill = "transparent", color = "red") |>
gf_facet_grid(~sex)
}

---

**gf_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,
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. \(~\text{head}(.x, 10)\)).

... 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.

linewidth
A numerical line width or a formula used for mapping linewidth.
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 grid::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, 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 The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")

position Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

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_density**

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,
  group,
  linetype,
  linewidth,
  kernel = "gaussian",
  n = 512,
  trim = FALSE,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "area",
  stat = "density",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
)```
gf_density

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

gf_dens(
  object = NULL,
  gformula = NULL,
  data = NULL,
...
  alpha = 0.5,
  color,
  fill = NA,
  group,
  linetype,
  linewidth,
  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_dens2(
  object = NULL,
  gformula = NULL,
  data = NULL,
...
  alpha = 0.5,
  color,
  fill = NA,
  group,
  linetype,
  linewidth,
  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)).
...
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.
linewidth A numerical line width or a formula used for mapping linewidth.
kernel Kernel. See list of available kernels in density().
n number of equally spaced points at which the density is to be estimated, should be a power of two, see 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.
**gf_density**

- **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, stat**: Use to override the default connection between `geom_density()` and `stat_density()`.
- **position**: Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use `position_jitter()`), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
- **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**

`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)

### 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, 
  linewidth, 
  contour = TRUE, 
  n = 100, 
  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, 
  linewidth, 
  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, 
  linewidth, 
  contour = TRUE, 
  n = 100, 
  h = NULL, 
  lineend = "butt", 
  linejoin = "round", 
  linemitre = 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.
gf_density_2d

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.
linewidth
A numerical line width or a formula used for mapping linewidth.
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).
linemitre
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, subtitle, caption
Title, sub-title, and caption for the plot. See also gf_labs().
geom, stat
Use to override the default connection between geom_density_2d() and stat_density_2d().
position
Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
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
) |>  
gf_density_2d(avg_drinks ~ age, data = mosaicData::HELPrct)
gf_density_2d_filled(avg_drinks ~ age, data = mosaicData::HELPrct, show.legend = FALSE) |>  
gf_jitter(avg_drinks ~ age,
  alpha = 0.3, data = mosaicData::HELPrct,
  width = 0.4, height = 0.4,
  color = "white"
)  
gf_jitter(avg_drinks ~ age,
  alpha = 0.2, data = mosaicData::HELPrct,
  width = 0.4, height = 0.4
) |>  
gf_density2d(avg_drinks ~ age, data = mosaicData::HELPrct)
gf_density2d_filled(avg_drinks ~ age, data = mosaicData::HELPrct, show.legend = FALSE) |>  
gf_jitter(avg_drinks ~ age,
  alpha = 0.4, data = mosaicData::HELPrct,
  width = 0.4, height = 0.4,
  color = "white"
)
```
**gf_dist**  
*Plot distributions*

**Description**
Create a layer displaying a probability distribution.

**Usage**
```r
gf_dist(
  object = ggplot(),
  dist,
  ...,  
  xlim = NULL,
  kind = c("density", "cdf", "qq", "qqstep", "histogram"),
  resolution = 5000L,
  eps = 1e-06,
  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.
- `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.
- `eps` a (small) numeric value. When other defaults are not available, the distribution is processed from the eps to 1 - eps quantiles.
- `params` a list of parameters for the distribution.

**Examples**
```r
gf_dhistogram(~ rnorm(100), bins = 20) |>
gf_dist("norm", color = "red")

# shading tails -- but see pdist() for this
```
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, 
  origin = NULL, 
  right = TRUE, 
  width = 0.9, 
  drop = FALSE, 
  xlab, 
  ylab, 
  title, 
  subtitle, 
  caption, 
  position = "identity", 
  show.legend = NA,
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.
binner
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", "centerhole" (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().
gf_dotplot

title, subtitle, 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.
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

```r
gf_ecdf(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  group, 
  pad, 
  n = NULL, 
  xlab, 
  ylab, 
  title, 
  subtitle, 
  caption, 
  geom = "step", 
  stat = "ecdf", 
  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)).

... 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, subtitle, caption Title, sub-title, and caption for the plot. See also gf_labs().
geom The geometric object to use to display the data, either as a ggproto Geom subclass or as a string naming the geom stripped of the geom_ prefix (e.g., "point" rather than "geom_point")

stat The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g., "count" rather than "stat_count")

position Position adjustment, either as a string naming the adjustment (e.g., "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

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.
Examples

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)

gf_ellipse

Description

Formula interface to ggplot2::stat_ellipse().

Usage

gf_ellipse(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha,
  color,
  group,
  type = "t",
  level = 0.95,
  segments = 51,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "path",
  stat = "ellipse",
  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.
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, subtitle, 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.
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.

See Also

ggplot2::stat_ellipse()
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 no 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,  
  linewidth,  
  xlab,  
  ylab,  
  title,  
  subtitle,  
  caption,  
  geom = "errorbar",  
  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).

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.

linewidth A numerical line width or a formula used for mapping linewidth.

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 The geometric object to use to display the data, either as a ggproto Geom sub-
class or as a string naming the geom stripped of the geom_ prefix (e.g. "point"
rather than "geom_point")

stat The statistical transformation to use on the data for this layer, either as a ggproto
Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g.
"count" rather than "stat_count")

position Position adjustment, either as a string naming the adjustment (e.g. "jitter" to
use `position_jitter`), or the result of a call to a position adjustment function.
Use the latter if you need to change the settings of the adjustment.

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_facet_wrap

**Add facets to a plot**

**Description**

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

**Usage**

```r
gf_facet_wrap(object, ...)
```

```r
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.
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,
  linewidth,
  size,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
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.
linewidth A numerical line width or a formula used for mapping linewidth.
size size aesthetic for dots in pmf plots.
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

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(  
f = rf(500, df1 = 3, df2 = 47),  
g = rgamma(500, 3, 10)
)
gf_dhistogram(~g, data = Dat) |>  
gf_fitdistr(dist = "dgamma", linewidth = 1.4)

gf_dhistogram(~g, data = Dat) |>  
gf_fitdistr(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)
)  
)
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

```r
gf_freqpoly(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  group,
  linetype,
  linewidth,
  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 \( \sim x \) or \( y \sim x \). 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. \( \sim \text{head}(.x, 10) \)).

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

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 numerical line width or a formula used for mapping linewidth.
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.
Label for x-axis. See also `gf_labs()`.
Label for y-axis. See also `gf_labs()`.
Title, sub-title, and caption for the plot. See also `gf_labs()`.
**gf_freqpoly**

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

**position**  Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

**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**

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(after_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

gf_function(object = NULL, fun, data = NULL, ..., inherit = FALSE)
gf_fun(object = NULL, formula, data = NULL, ..., 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.
- **data**: A data frame with the variables to be plotted.
- **...**: Additional arguments passed as `params` to `layer()`. This includes `xlim`, a numeric vector providing the extent of the x-axis values used to evaluate `fun` for plotting. By default, `xlim` is not used for other layers.
- **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?

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")
```
Description

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

Usage

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

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

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

gf_function_tile(
  object = NULL,
  fun = identity,
  xlim = NULL,
  ylim = NULL,
  ..., resolution = 50
)
gf_function_2d

)  
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An R object, typically of class &quot;gg&quot;.</td>
</tr>
<tr>
<td>fun</td>
<td>A function of two variables to be plotted.</td>
</tr>
<tr>
<td>xlim</td>
<td>x limits for generating points to be plotted.</td>
</tr>
<tr>
<td>ylim</td>
<td>y limits for generating points to be plotted.</td>
</tr>
</tbody>
</table>
gf_hex

... 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")

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>linewidth</code></td>
<td>A numerical line width or a formula used for mapping linewidth.</td>
</tr>
<tr>
<td><code>xlab</code></td>
<td>Label for x-axis. See also <code>gf_labs()</code>.</td>
</tr>
<tr>
<td><code>ylab</code></td>
<td>Label for y-axis. See also <code>gf_labs()</code>.</td>
</tr>
</tbody>
</table>
| `title`, `subtitle`, `caption` | Title, sub-title, and caption for the plot. See also `gf_labs()`.
| `geom`, `stat` | Override the default connection between `geom_hex()` and `stat_binhex()`. |
| `position`     | Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use `position_jitter`), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment. |
| `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**

gf_hex(avg_drinks ~ age, data = mosaicData::HELPrc, bins = 15) |>
gf_density2d(avg_drinks ~ age, data = mosaicData::HELPrc, color = "red", alpha = 0.5)
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,
  linewidth,
  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,
  linetype,
  linewidth,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "bar",
  stat = "bin",
  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 ~ x (or y ~ x, but this shape is not generally needed).
The data to be displayed in this layer. There are three options:

If \( \text{NULL} \), 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. \( \sim \text{head}(., 10) \)).

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

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

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.

Opacity (0 = invisible, 1 = opaque).

A color or a formula used for mapping color.

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

Used for grouping.

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

A numerical line width or a formula used for mapping linewidth.

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

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

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

Use to override the default connection between \( \text{geom_histogram}() \)/\( \text{geom_freqpoly}() \) and \( \text{stat_bin}() \).

Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use \( \text{position_jitter}() \)), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

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

If \( \text{TRUE} \), display some minimal help.

A logical indicating whether default attributes are inherited.

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

Formula interface to geom_jitter()

Description

Jittered scatter plots in ggformula.

Usage

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 ~ 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, 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 \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_jitter()}, \texttt{gf_point()}

Examples

\begin{verbatim}
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)
\end{verbatim}

\section*{gf_labs}
\textit{Non-layer functions for gf plots}

Description

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

Usage

\begin{verbatim}
gf_labs(object, ...)
gf_lims(object, ...)
gf_refine(object, ...)
\end{verbatim}

Arguments

\begin{verbatim}
object a gg object
...
additional arguments passed through to the similarly named function in \texttt{ggplot2}.
\end{verbatim}

Details

\texttt{gf_refine()} provides a mechanism to replace + with the chaining/pipe operator \texttt{|>}. Each of its \texttt{\dots} arguments is added in turn to the base plot in \texttt{object}. The other functions are thin wrappers around specific \texttt{ggplot2} refinement functions and pass their \texttt{\dots} arguments through to the similarly named \texttt{ggplot2} functions.
Value

a modified gg object

Examples

gf_dens(~cesd, color = ~substance, linewidth = 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() (with a formula)

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_line(
  object = NULL,
  gformula = NULL,
data = NULL,
..., 
alpha, 
color, 
fill, 
group, 
linetype, 
linewidth, 
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()
)

gf_path( 
    object = NULL, 
gformula = NULL, 
data = NULL, 
..., 
alpha, 
color, 
group, 
linetype, 
linewidth, 
lineend = "butt", 
linejoin = "round", 
linemitre = 1, 
arrow = NULL, 
xlab, 
ylab, 
title, 
subtitle, 
caption, 
geom = "path", 
stat = "identity", 
position = "identity", 
...
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.

linewidth A numerical line width or a formula used for mapping linewidth.

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().

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 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

gf_line()
gf_point(age ~ sex, alpha = 0.25, data = mosaicData::HELPrct)
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, 
  linewidth, 
  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, 
  linetype, 
  linewidth, 
  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, 
  linewidth, 
  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()
)

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)).
gf_linerange

... 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.

linewidth A numerical line width or a formula used for mapping linewidth.

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 The geometric object to use to display the data, either as a ggproto Geom subclass or as a string naming the geom stripped of the geom_ prefix (e.g. "point" rather than "geom_point")

stat The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")

position Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

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.

size size aesthetic for points (gf_pointrange()).

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

fun.ymin, fun.y, fun.ymax [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, linewidth = 1.3)
# You can supply individual functions to summarise the value at
# each x:
p |> gf_summary(fun.y = "median", color = "red", size = 3, geom = "point")
p |> 
gf_summary(fun.y = "mean", color = "red", size = 3, geom = "point") |> 
gf_summary(fun.y = mean, geom = "line")
p |>
gf_summary(fun.y = mean, fun.ymin = min, fun.ymax = max, color = "red")
## Not run:
p |>
gf_summary(fun.ymin = min, fun.ymax = max, color = "red", geom = "linerange")
## End(Not run)

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
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.
## Not run:
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)
## End(Not run)

# 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_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:

```
expression
```

See examples.

**Value**

a gg object

**Examples**

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

Formula interface to geom_point()

Description

Scatterplots in ggformula.

Usage

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,
  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, (c) attributes of the layer as a whole, which are set with attribute = value, or (d) arguments for the geom, stat, or position function.
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 \mid 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()
gf_polygon

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

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_polygon(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., alpha, color, linewidth, shape, fill, group, stroke, xlab,
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.
linewidth
A numerical line width or a formula used for mapping linewidth.
shape, stroke
Aesthetics for polygons.
fill
A color for filling, or a formula used for mapping fill.
group
Used for grouping.
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.
**gf_polygon**

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
gf_polygon()
if (require(maps) && require(ggthemes) && require(dplyr)) {
  US <- map_data("state") |
    dplyr::mutate(name_length = nchar(region))
  States <- US |
    dplyr::group_by(region) |
    dplyr::summarise(lat = mean(range(lat)), long = mean(range(long))) |
    dplyr::mutate(name = abbreviate(region, 3))

  gf_polygon(lat ~ long,
    data = US, group = ~group,
    fill = ~name_length, color = "white"
  ) |
  gf_text(lat ~ long,
    label = ~name, data = States,
    color = "gray70", inherit = FALSE
  ) |
  gf_refine(ggthemes::theme_map())
}
```
**gf_qq**

Formula interface to `geom_qq()`

**Description**

`gf_qq()` and `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**

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

---

**gf_qqline**

`gf_qqline()` is similar to `gf_qq()`, but it draws a line rather than points. This can be useful for visually connecting the points in a quantile-quantile plot.
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, subtitle, caption
Title, sub-title, and caption for the plot. See also gf_labs().
geom, stat
Use to override the default connection between geom_histogram()/geom_freqpoly()
and stat_bin().
position
Position adjustment, either as a string naming the adjustment (e.g. "jitter" to
use position_jitter), or the result of a call to a position adjustment function.
Use the latter if you need to change the settings of the adjustment.
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.
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.
gf_quantile

See Also

ggplot2::geom_qq()

Examples

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)

df_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,  
  linewidth,  
  lineend = "butt",  
  linejoin = "round",  
  linemitre = 1,  
  quantiles,  
  formula,  
  method,  
  method.args,  
  xlab,  
  ylab,  
  title,  
  subtitle,  
  caption,  
  geom = "quantile",  
  stat = "quantile",  
  position = "identity",  
  data = NULL,  
  ...)
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.
linewidth  A numerical line width or a formula used for mapping linewidth.
lineend  Line end style (round, butt, square).
linejoin  Line join style (round, mitre, bevel).
linemiter  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()).
methode.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, sub-title, and caption for the plot. See also `gf_labs()`.

Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use `position_jitter`), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

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`.

a `gg` object

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)
```
Formula interface to geom_raster()

**Usage**

```r
gf_raster(
  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 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`. 

**Description**

Formula interface to geom_raster()
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.
linewidth A numerical line width or a formula used for mapping linewidth.
hjust, 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().
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_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

```r
gf_rect(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  fill,
  group,
  linetype,
  linewidth,
  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.

linewidth
A numerical line width or a formula used for mapping linewidth.

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_rect()

Examples

gf_rect(1 + 2 ~ 3 + 4, alpha = 0.3, color = "red")
# use data = data.frame() so we get 1 rectangle and not 1 per row of faithful
# use inherit = FALSE because we are not reusing eruptions and waiting
gf_point(eruptions ~ waiting, data = faithful) |> 
gf_rect(1.5 + 3 ~ 45 + 68, 
  fill = "red", alpha = 0.2, 
  data = data.frame(), inherit = FALSE) |> 
gf_rect(3 + 5.5 ~ 68 + 100, 
  fill = "green", alpha = 0.2, 
  data = data.frame(), inherit = FALSE)

---

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, subtitle, caption Title, sub-title, and caption for the plot. See also gf_labs().
geom The geometric object to use to display the data, either as a ggproto Geom subclass or as a string naming the geom stripped of the geom_ prefix (e.g. "point" rather than "geom_point")
stat The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
gf_ridgeline

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_ribbon()

Examples

gf_ribbon()
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 ~ .,  color = ~high_temp,  data = mosaicData::Weather)
) |> gf_refine(scale_colour_gradientn(colors = rev(rainbow(5))))
gf_ribbon(low_temp + high_temp ~ date | city ~ ., data = mosaicData::Weather)
# Chaining in the data
## Not run:
mosaicData::Weather |> gf_ribbon(low_temp + high_temp ~ date, alpha = 0.4) |> gf_facet_grid(city ~ .)
## End(Not run)

GF_RIDGELINE

GF_RIDGELINE

Description

Formula interface to ggriddles plots

Usage

gf_ridgeline(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  height,
  scale = 1,
min_height = 0,
color,
fill,
alpha,
group,
linetype,
linewidth,
point_size,
point_shape,
point_colour,
point_fill,
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,
linewidth,
point_size,
point_shape,
point_colour,
point_fill,
point_alpha,
point_stroke,
panel_scaling = TRUE,
gf_ridgeline

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,
  linewidth,
  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,
  linetype,
  linewidth,
  gradient_lwd = 0.5,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "ridgeline_gradient",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

gf_density_ridges_gradient(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  height,
  panel_scaling = TRUE,
  color,
  fill = ~stat(x),
  alpha,
  group,
  linetype,
  linewidth,
  gradient_lwd = 0.5,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "density_ridges_gradient",
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 \( \sim x \). 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()} 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) ggplot aesthetics to be set with \texttt{attribute = value}, (b) ggplot 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}.

height
The height of each ridgeline at the respective x value. Automatically calculated and provided by \texttt{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.

linewidth
A numerical line width or a formula used for mapping linewidth.
point_shape, point_colour, point_size, point_fill, point_alpha, point_stroke
As in \texttt{ggridges::geom_ridgeline}.

\texttt{xlab} \hspace{1em} Label for x-axis. See also \texttt{gf_labs}.

\texttt{ylab} \hspace{1em} Label for y-axis. See also \texttt{gf_labs}.

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

\texttt{geom}, \texttt{stat}
Use to override the default connection between \texttt{geom_density()} and \texttt{stat_density}.

\texttt{position}
Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use \texttt{position_jitter}), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

\texttt{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.

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

\texttt{inherit}
A logical indicating whether default attributes are inherited.

\texttt{environment}
An environment in which to look for variables not found in \texttt{data}.

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

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

\texttt{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 \texttt{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 \texttt{stat = "density"} to use \texttt{ggplot2::stat_density}. In this case, it is required to add the aesthetic mapping \texttt{height = after_stat(density)} (see examples).

See Also
\texttt{ggridges::geom_density_ridges}()
\texttt{ggridges::geom_ridgeline}()
\texttt{ggridges::geom_density_ridges_gradient}()

Examples
```r
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,
```
```r
diamonds |>
gf_density_ridges(clarity ~ price | cut,
  scale = 2, fill = ~ clarity, 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))
)

## Not run:
diamonds |>
gf_density_ridges2(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))
)

## End(Not run)

## Not run:
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))
)

## End(Not run)

diamonds |>
gf_density_ridges(cut ~ price | cut,
  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))
)

## Not run:
diamonds |
```

---

```
```
gf_density_ridges(clarity ~ price | cut, height = ~ after_stat(density), stat = "density",
    scale = 2, fill = ~ clarity, 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))
)

## End(Not run)
## Not run:
mosaicData::Weather |>
gf_density_ridges_gradient(month ~ high_temp | city ~ ., fill = ~stat(x),
    group = ~ month, show.legend = FALSE, rel_min_height = 0.02) |>
gf_refine(scale_fill_viridis_c(option = "B"), theme_bw())

## End(Not run)

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,
    linewidth,
    xlab,
    ylab,
    title,
    subtitle,
    caption,
    geom = "rug",
    stat = "identity",
    position = "identity",
    show.legend = NA,
gf_rug

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

linewidth
A numerical line width or a formula used for mapping linewidth.

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
The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
gf_rug

Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

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") |>
gf_rugy(bill_length_mm ~ ., data = penguins, color = "green")

gf_point(bill_length_mm ~ bill_depth_mm, data = penguins) |>
gf_rug(. ~ bill_depth_mm, data = penguins, color = "red", inherit = FALSE) |> 
gf_rug(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 = "red", sides = "b") |> 
gf_rug(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 = "green", sides = "b", position = "jitter")

# rugs work with some 1-varialbe plots as well.
gf_histogram(~eruptions, data = faithful) |> 
gf_rug(~eruptions, data = faithful, color = "red") |> 
gf_rug(~eruptions, data = faithful, color = "navy", sides = "t")

# we can take advantage of inheritance to shorten the code
gf_histogram(~eruptions, data = faithful) |> 
gf_rug(color = "red") |> 
gf_rug(color = "navy", sides = "t")

# 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_histogram(~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**

gf_segment(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha, 

---
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 \( y + \text{yend} \sim x + \text{xend} \).

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

- If \text{NULL}, the default, the data is inherited from the plot data as specified in the call to \text{ggplot()}. A \text{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 \text{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 = \text{~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.

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

linewidth
A numerical line width or a formula used for mapping linewidth.

arrow
Specification for arrow heads, as created by \text{grid::arrow()}.
Line end style (round, butt, square).

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

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

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

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

The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the `stat_` prefix (e.g. "count" rather than "stat_count")

Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use `position_jitter`), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

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`.

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

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")

---

Mapping with shape files

Description

Mapping with shape files

Usage

gf_sf(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  fill,
  group,
  linetype,
  linewidth,
  geometry,
  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 ~ x. Faceting can be achieved by including | in the formula.
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.
gf_sina

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_line()}, \texttt{gf_point()}

Examples

```r
if (requireNamespace('maps', quietly = TRUE)) {
  library(maps)
  world1 <- sf::st_as_sf(map('world', plot = FALSE, fill = TRUE))
  gf_sf(data = world1)
}

if (requireNamespace('maps', quietly = TRUE)) {
  world2 <- sf::st_transform(
    world1,
    "+proj=laea +y_0=0 +lon_0=155 +lat_0=-90 +ellps=WGS84 +no_defs"
  )
  gf_sf(data = world2)
}
```

\textit{gf_sina} \hspace{1cm} \textit{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

\texttt{gf_sina(}
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha, 
  color, 
  size, 
  fill, 
  group, 
  xlab, 
)}
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.
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 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  The geometric object to use to display the data, either as a ggproto Geom subclass or as a string naming the geom stripped of the geom_prefix (e.g. "point" rather than "geom_point")
stat
The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")

position
Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

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 by setting 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

## Not run:
library(ggforce)
gf_sina(age ~ substance, data = mosaicData::HELPrtc)

## End(Not run)
**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,
  ylab,
  title,
  subtitle,
  caption,
  geom = "smooth",
  stat = "smooth",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
```

```r
gf_lm(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ..., 
  alpha = 0.3,
  lm.args = list(),
  interval = "none",
  level = 0.95,
  fullrange = TRUE,
  xlab,
)```
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.

**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 = \( y \sim s(x, \text{bs} = \text{"cs"}) \) with method = "REML". Somewhat anecdotaly, 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 = \( y \sim s(x, \text{bs} = \text{"cs"}) \).

**formula**  
Formula to use in smoothing function, e.g. \( y \sim x, y \sim \text{poly}(x, 2), y \sim \log(x) \). NULL by default, in which case method = NULL implies formula = \( y \sim x \) when there are fewer than 1,000 observations and formula = \( y \sim s(x, \text{bs} = \text{"cs"}) \) 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. Only used
with loess, i.e. when method = "loess", or when method = NULL (the default) and there are fewer than 1,000 observations.

fullrange  If TRUE, the smoothing line gets expanded to the range of the plot, potentially beyond the data. This does not extend the line into any additional padding created by expansion.

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

gem  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.
gf_smooth

See Also

ggplot2::geom_smooth(), gf_spline()

Examples

gf_smooth()
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,
  color = ~sex, data = mosaicData::KidsFeet,
  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_lm( 
    formula = log(y) ~ poly(x, 3), backtrans = exp,
    interval = "confidence", color = "red"
  ) |> 
gf_point()
gf_spline <- data.frame(
  u = c(5,10,15,20,30,40,60,80,100),
  lot1 = c(118,58,42,35,27,25,21,19,18),
  lot2 = c(69,35,26,21,18,16,13,12,12))
gf_point(lot1 ~ u, data = clotting) |> 
gf_smooth(formula = y ~ log(x), method = "glm",
  method.args = list(family = Gamma))
gf_point(lot2 ~ u, data = clotting) |> 
gf_smooth(formula = y ~ log(x), color = "red", method = "glm",
  method.args = list(family = Gamma))

---

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,
  linewidth,
  weight,
  df,
  spar,
  tol,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  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 \( y \sim x \). 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.
group
Used for grouping.
linetype
A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
linewidth
A numerical line width or a formula used for mapping linewidth.
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.
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, 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

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

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, 
  linewidth,
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.

linewidth

A numerical line width or a formula used for mapping linewidth.

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 The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")

position Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

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.

evironment 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_spoke()`

Examples

```r
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)
```
Description

geom_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,
  linewidth,
  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 ~ 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.
- **linewidth** A numerical line width or a formula used for mapping linewidth.
- **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, 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** The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
- **position** Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use `position_jitter`), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
- **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

```r
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

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,
  angle,
  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. ~ 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.
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 to 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, 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_text(). Note that this argument is not supported by geom_label().

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       The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
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.
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
)
penguins |>
gf_point(bill_length_mm ~ bill_depth_mm, color = ~species, alpha = 0.5) |>
gf_text(bill_length_mm ~ bill_depth_mm,
   label = ~species, color = ~species,
   size = 2, angle = 0, hjust = 0, nudge_x = 0.1, nudge_y = 0.1
)
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_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
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,                  
  linewidth,
gf_tile

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.
linewidth A numerical line width or a formula used for mapping linewidth.
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_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

Formula interface to geom_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,
  linewidth,
  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()
)

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_violin

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 ($0 = \text{invisible}, 1 = \text{opaque}$).
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.
linewidth A numerical line width or a formula used for mapping linewidth.
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 multiplicative bandwidth adjustment. This makes it possible to adjust the bandwidth while still using the a bandwidth estimator. For example, adjust = 1/2 means use half of the default bandwidth.
kernl 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, subtitle, caption Title, sub-title, and caption for the plot. See also `gf_labs()`.
gf_violin

geom, stat
Use to override the default connection between geom_violin() and stat_ydensity().

position
Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

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::HELPrct)
gf_violin(age ~ substance, data = mosaicData::HELPrct, fill = ~sex)
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.
**MIpop**

*Population of Michigan counties*

**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**

**Description**
Population of Michigan counties

**Usage**

```r
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**

```r
percs_by_group(x, group)
```

```r
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()`
stat_fitdistr

A stat for fitting distributions

Description

This stat computes points for plotting a distribution function. Fitting is done using MASS::fitdistr() when analytic solutions are not available.

Usage

stat_fitdistr(
  mapping = NULL,
  data = NULL,
  geom = "path",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  dist = "dnorm",
  start = NULL,
  ...
)

Arguments

mapping Aesthetics created using aes() or aes_string().
data A data frame.
geom A character string naming the geom 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.
na.rm If TRUE, do not emit a warning about missing data.
show.legend A 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.
inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them.
dist A character string indicating the distribution to fit. Examples include "dnorm", 
dgamma", etc.
start A list of starting values used by MASS::fitdistr() when numerically approximating the maximum likelihood estimate.
... Additional arguments.
Value

A gg object

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()`. 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 naming the adjustment (e.g. "jitter" to use `position_jitter`), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.

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)
```

---

**stat_spline**  
*Geoms and stats for spline smoothing*

**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,
)```
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

```r
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)
}
```
Index

* datasets
  StatAsh, 152
aes(), 12, 155, 157, 158
aes(), 12
aes_string(), 157, 158
akima::bilinear(), 30
borders(), 155
density(), 43, 148
df_stats(), 27
discrete_breaks, 3
expand_limits(), 22
fivenum(), 27
fortify(), 6, 9, 16, 25, 32, 36, 39, 43, 48, 55, 60, 66, 72, 76, 86, 97, 100, 108, 113, 118, 121, 126, 135, 138, 141, 148, 155
geom_ash (gf_ash), 10
geom_boxplot(), 99, 147
geom_density(), 45, 147
geom_histogram(), 13
geom_lm (stat_lm), 154
geom_lm(), 152
geom_point(), 33
geom_smooth, 157
geom_spline (stat_spline), 157
geom_spline(), 134
geom_tile(), 30
GeomLm (StatAsh), 152
get_variable_labels, 4
gf_abline(), 5
gf_area, 8
gf_ash, 10
gf_ash(), 44, 152, 153
gf_bar, 13
gf_barh, 19
gf_bin2d, 20
gf_blank, 22
gf_boxplot, 24
gf_boxploth (gf_barh), 19
gf_coefline (gf_abline), 5
gf_col, 28
gf_col(), 16
gf_colh (gf_barh), 19
gf_contour, 30
gf_contour(), 71
gf_contour_filled (gf_contour), 30
gf_count, 33
gf_counts (gf_bar), 13
gf_countsh (gf_barh), 19
gf_crossbar, 35
gf_crossbarh (gf_barh), 19
gf_curve, 38
gf_dens (gf_density), 41
gf_dens2 (gf_density), 41
gf_density, 41
gf_density2d (gf_density_2d), 45
gf_density2d_filled (gf_density_2d), 45
gf_density_2d, 45
gf_density_2d(), 33
gf_density_2d_filled (gf_density_2d), 45
gf_density_ridgeline_gradient (gf_ridgeline), 109
gf_density_ridges (gf_ridgeline), 109
gf_density_ridges2 (gf_ridgeline), 109
gf_density_ridges_gradient
  (gf_ridgeline), 109
gf_dhistogram (gf_histogram), 74
gf_dhistograph (gf_histogram), 74
gf_dist, 50
gf_dotplot, 51
gf_ecdf, 54
gf_ellipse, 56
gf_empty, 58
gf_errorbar, 59
gf_errorbarh (gf_barh), 19
gf_facet_grid (gf_facet_wrap), 61
gf_facet_wrap, 61
gf_fitdistr, 62
gf_frame (gf_blank), 22
gf_freqpoly, 65
gf_fun (gf_function), 68
gf_fun2d (gf_function_2d), 69
gf_fun_2d (gf_function_2d), 69
gf_fun_contour (gf_function_2d), 69
gf_fun_tile (gf_function_2d), 69
gf_function, 68
gf_function2d (gf_function_2d), 69
gf_function_2d, 69
gf_function_contour (gf_function_2d), 69
gf_function_tile (gf_function_2d), 69
gf_hex, 71
gf_histogram, 74
gf_hline (gf_abline), 5
gf_jitter, 78
gf_jitter (), 92
gf_label (gf_text), 139
gf_labs, 80
gf_labs (), 7, 9, 10, 12, 16, 21, 23, 26, 29, 32, 34, 36, 37, 40, 44, 48, 52, 53, 55, 57, 60, 63, 66, 73, 76, 79, 83, 87, 92, 94, 98, 100, 101, 103, 105, 108, 114, 118, 122, 124, 126, 130, 133, 135, 138, 142, 145, 148
gf_lims (gf_labs), 80
gf_line, 81
gf_line (), 92
gf_linerange, 84
gf_linerangeh (gf_barh), 19
gf_lm (gf_smooth), 128
gf_lm (), 134, 152, 153
gf_path (gf_line), 81
gf_percent, 13
gf_percent ( ), 16
gf_percentsh (gf_barh), 19
gf_plot, 90
gf_point, 91
gf_point (), 80, 84, 95, 125
gf_pointrange (gf_linerange), 84
gf_pointrangeh (gf_barh), 19
gf_polygon, 93
gf_props (gf_bar), 13
gf_props ( ), 16
gf_props (gf_barh), 19
gf_qq, 96
gf_qq (), 152
gf_qqline (gf_qq), 96
gf_qqstep (gf_qq), 96
gf_quartile, 99
gf_raster, 102
gf_rect, 104
gf_refine (gf_labs), 80
gf_rel, 106
gf_rug, 107
gf_ridgeline, 109
gf_rugc (gf_rug), 116
gf_rugx (gf_rug), 116
gf_rugy (gf_rug), 116
gf_segment, 120
gf_sf, 123
gf_sina, 125
gf_smooth, 128
gf_smooth ( ), 134
gf_spline, 132
gf_spline ( ), 131, 152
gf_spoke, 134
gf_step, 137
gf_summary (gf_linerange), 84
gf_text, 139
gf_theme, 144
gf_tile, 144
gf_tile(), 20, 22, 71
gf_violin, 147
gf_violinh (gf_barh), 19
gf_vline (gf_abline), 5
ggforce::geom_sina(), 127
ggplot(), 6, 9, 16, 25, 31, 36, 39, 43, 48, 55, 60, 66, 72, 76, 86, 97, 100, 108, 113, 118, 121, 126, 135, 137, 141, 148, 155
ggplot2::aes(), 150
ggplot2::facet_grid(), 62
ggplot2::facet_wrap(), 62
ggplot2::fortify(), 90
ggplot2::geom_abline(), 7
ggplot2::geom_area(), 10
ggplot2::geom_bar(), 17
ggplot2::geom_bin2d(), 22
ggplot2::geom_blank(), 24
ggplot2::geom_boxplot(), 27
ggplot2::geom_col(), 30
ggplot2::geom_contour(), 33
ggplot2::geom_count(), 35
ggplot2::geom_crossbar(), 37
ggplot2::geom_density(), 44
ggplot2::geom_density_2d(), 49
ggplot2::geom_dotplot(), 54
ggplot2::geom_errorbar(), 60
ggplot2::geom_freqpoly(), 67
ggplot2::geom_hex(), 73
ggplot2::geom_histogram(), 77
ggplot2::geom_hline(), 7
ggplot2::geom_jitter(), 80
ggplot2::geom_line(), 84, 95, 125
ggplot2::geom_linerange(), 87
ggplot2::geom_point(), 92
ggplot2::geom_pointrange(), 87
ggplot2::geom_qq(), 99
ggplot2::geom_quantile(), 101
ggplot2::geom_raster(), 104
ggplot2::geom_rect(), 106
ggplot2::geom.ribbon(), 109
ggplot2::geom.rug(), 119
ggplot2::geom_segment(), 122
ggplot2::geom_smooth(), 131
ggplot2::geom_spoke(), 136
ggplot2::geom_step(), 139
ggplot2::geom_text(), 143
ggplot2::geom_tile(), 146
ggplot2::geom_violin(), 149
ggplot2::geom_vline(), 149
ggplot2::ggplot(), 58
ggplot2::stat_bin2d(), 20
ggplot2::stat_density(), 114
ggplot2::stat_ellipse(), 56, 57
ggplot2::stat_summary(), 87
ggplot2::theme(), 144
ggridges::geom_density_ridges(), 114
ggridges::geom_density_ridges_gradient(), 114
ggridges::geom_ridgeline(), 114
ggridges::stat_density_ridges(), 113, 114
ggstance (gf_barh), 19
grid::arrow(), 40, 83, 121
grid::curveGrob(), 38, 120
interp::interp(), 30
labelled::set_variable_labels(), 4
labelled::var_label(), 4
layer(), 55, 126, 155
layer_factory, 150
lm(), 155
MASS::bandwidth.nrd(), 48
MASS::fitdistr, 63
MASS::kde2d(), 45
mgcv::gam(), 129
MIPop, 151
mosaic::makeFun(), 71
mosaicCore::fit_distr_fun(), 63, 64
mosaicCore::makeFun(), 68
percs_by_group, 151
predict(), 155
print.gf_ggplot (gf_relabel), 106
props_by_group (percs_by_group), 151
quantreg::rq(), 100
quantreg::rqss(), 100
smooth.spline(), 133, 158, 159
stat_ash (gf_ash), 10
stat_ash(), 152, 153
stat_fitdistr, 153
stat_lm, 154
stat_lm(), 152
stat_qq(), 152
stat_qqline, 156
stat_spline, 157
stat_spline(), 152
StatAsh, 152
StatFitdistr (StatAsh), 152
StatLm (StatAsh), 152
StatQqline (StatAsh), 152
stats::bw.nrd(), 148
stats::lm(), 130
stats::loess(), 129
StatSpline (StatAsh), 152

theme_minimal(), 144